Workshop Proceedings

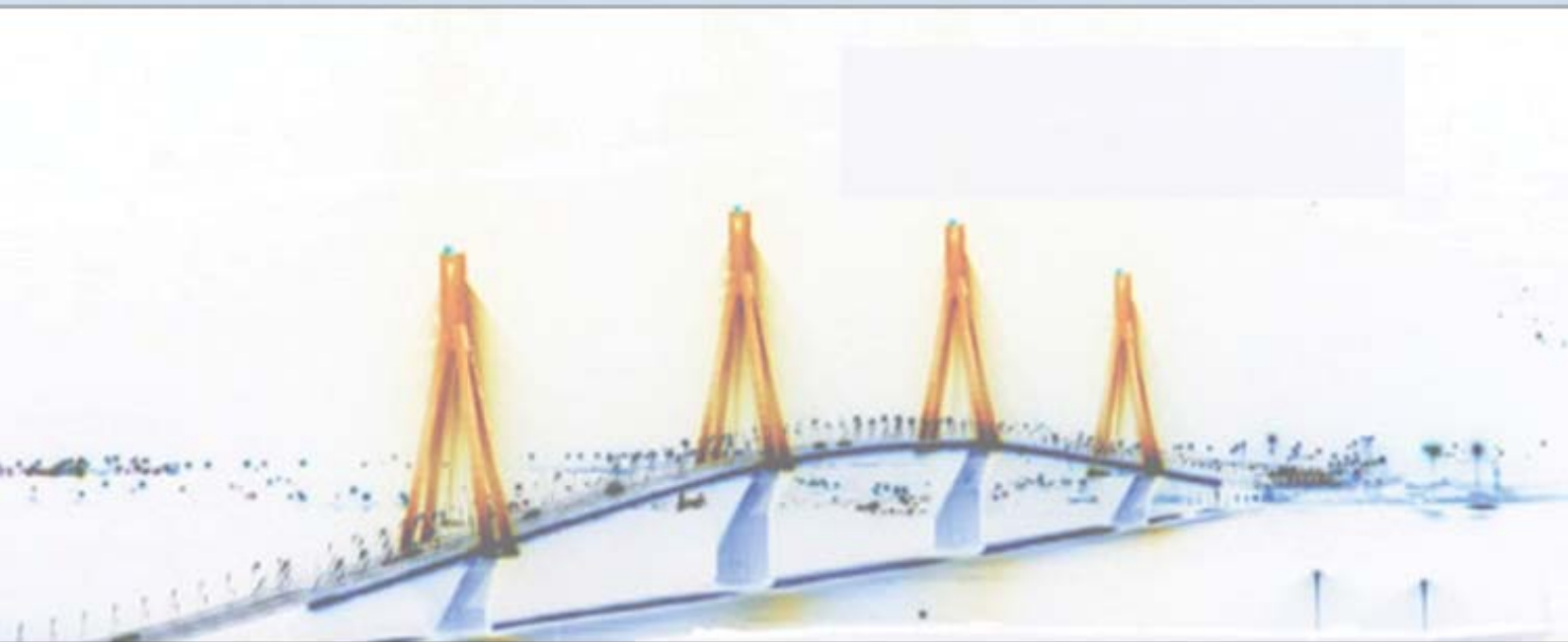
ECAI Patras 2008

18th European Conference on Artificial Intelligence

22 July 2008

ECAI'08 Workshop on Recommender Systems

Markus Zanker Alexander Felfernig Robin Burke



University of Patras

The 18th European Conference on Artificial Intelligence

Proceedings

Workshop on Recommender Systems

Tuesday July 22, 2008
Patras, Greece

Markus Zanker, Alexander Felfernig and Robin Burke (Eds.)

ISBN 978-960-6843-06-8

Conference Organization

Programme Chairs

Markus Zanker
Alexander Felfernig
Robin Burke

Programme Committee

Esma Aimeur
Matthias Bauer
Bettina Berendt
Shlomo Berkovsky
Derek Bridge
Robin Burke
Alexander Felfernig
Gerhard Friedrich
Dietmar Jannach
Markus Jessenitschnig
Joseph Konstan
Christian Kop
Gerhard Leitner
Kevin McCarthy
Bamshad Mobasher
Seung-Taek Park
Sven Radde
Francesco Ricci
John Riedl
Vincent Schickel-Zuber
Lars Schmidt-Thieme
Marius Silaghi
Barry Smyth
Erich Teppan
Hannes Werthner
Cai-Nicolas Ziegler
Markus Zanker

Additional Reviewers

Christoph Herzog
Venkatesh Ramamoorthy
Roussi Roussev

Preface

Recommender Systems (RS) suggest useful and interesting items to users in order to increase their satisfaction and online conversion rates. They contribute to the commercial success of many online ventures and are a very active area of research. The goal of this scientific event is to promote research collaborations and active discussions in all technical areas related to Recommender Systems.

The ECAI 2008 Workshop on Recommender Systems continues a series of successful Workshops on Recommendation Systems over the past decade like the Workshop on Recommender Systems at ECAI 2006 or the Joint Workshop on Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce at AAAI 2007 to name the most recent ones.

We received 15 submissions to this workshop that underwent a double-blind peer-review process. Each paper was reviewed by at least three members of the international programme committee. Following the reviewers' recommendations 7 papers have been selected for publication as full papers (6 pages), 4 as short papers (4 pages) and 3 as position statements (2 pages). They represent contributions in diverse fields such as scalable algorithms and security, social and interactivity aspects of recommender systems or applications for knowledge management.

Furthermore, Thomas Roth-Berghofer from DFKI will *reveal the magic of product recommendation* in his invited opening talk when addressing the issue of explanatory capabilities of recommender systems.

Finally, we want to thank the programme committee members, the additional reviewers and the authors for their fruitful contributions to this workshop and look forward to interesting talks and discussions in Patras.

Markus Zanker, Alexander Felfernig and Robin Burke

July 2008

Table of Contents

Session 5: Position statements.

Revealing the Magic of Product Recommendation

Thomas R. Roth-Berghofer[1]

From the user's point of view recommender systems quite often have a certain magical quality. Users look for interesting products or useful information and, often miraculously, get some suggestions alongside their browsing results. In this scenario, recommender systems take on a role we usually ascribe to colleagues and friends who help us choose one product over an other. Their advice may be based on content—the features of this product seem to have advantages over the features of that product—or the recommendations are community-based—others bought this product, too.

On the Web, e-commerce platforms have operationalised the two approaches quite successfully, with Amazon.com probably being the best known example for community-based product recommendations. But where we can ask a human about their advice, on what they based their recommendation, recommendation systems are not necessarily able to to give us this information. They do not explain their reasoning and how they came up with the suggested solution, although recently considerable research has been undertaken to remedy this situation.

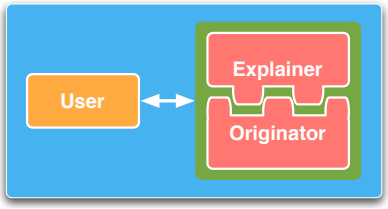


Figure 1. Participants in explanation scenario [3]

In this talk I address the problem of designing and implementing content-based recommender systems with explanation capabilities. A recommender system (like any knowledge-based system) is embedded in the following general explanation scenario [3] comprising three participants (see Figure 1): *originator*, *user*, and *explainer*.

The *originator* is the problem solving agent, e.g., a recommender engine. It provides something to be explained, for example a list of recommendations.

The *user* is the addressee of the explanation, presented by the *explainer*, which chooses the form of the explanation and organises a dialog if needed. In order to provide good explanations, originator and explainer need to be tightly integrated. Not all of the knowledge needed for explaining is available from the originator. Additional knowledge needs to be acquired for the explainer.

Explanations are strongly associated with trust and transparency. One trusts a knowledge-based system much more if it is able to explain what it is doing and, thus, can "prove" its trustworthiness to its user. Any information system, and even more so any recommender system, should be able to explain at every point in time why it prefers solution A over solution B.

Furthermore, it should also clarify the meaning of used concepts, and where an information item originally came from ("knowledge provenance"). Explanations are part of human understanding processes and part of most dialogues, and, therefore, need to be incorporated into system interactions in order to improve decision-making processes.

Case-Based Reasoning (CBR) systems are well suited for building content-based recommender systems as is demonstrated by their utilisation in e-commerce scenarios. The case base often is filled from product catalogues. Modelling and editing the similarity measures for products is a complex task for the knowledge engineer, who can be supported by explanations about the structure and content of the case base. Such support features have been implemented in the open source CBR tool myCBR[2] [4]. I will illustrate myCBR's explanation capabilities [1, 2] using an online shop scenario.

REFERENCES

[1] Daniel Bahls, *Explanation Support for the Case-Based Reasoning Tool myCBR*, Project thesis, University of Kaiserslautern, 2008.
[2] Thomas Roth-Berghofer, ed. *Künstliche Intelligenz: Anwendungen im Semantischen Web (Integriertes Seminar)*. Technische Universität Kaiserslautern, Juli 2004.
[3] Thomas R. Roth-Berghofer and Michael M. Richter, 'On explanation', *Künstliche Intelligenz*, **22**(2), 5–7, (May 2008).
[4] Armin Stahl and Thomas R. Roth-Berghofer, 'Rapid prototyping of CBR applications with the open source tool myCBR', in *Advances in Case-Based Reasoning*, eds., Ralph Bergmann and Klaus-Dieter Althoff. Springer Verlag, (2008).

[1] Knowledge Management Department, German Research Center for Artificial Intelligence DFKI GmbH, 67663 Kaiserslautern, Germany, and Knowledge-based Systems Group, Department of Computer Science, University of Kaiserslautern, email: thomas.roth-berghofer@dfki.de

[2] `http://myCBR-project.net`

A Model-Based Customer Inference Engine

Sven Radde and **Andreas Kaiser** and **Burkhard Freitag**[1]

Abstract. In complex and frequently changing product domains, customers need qualified consultation services that allow them to make well-informed purchasing decisions without requiring excessive technical knowledge about the product domain. In this paper we present an intuitive customer metamodel with an associated inference engine which can be integrated into conversational recommender systems to provide comfortable reasoning about customer needs. The inferred knowledge can be used to manage the recommendation dialogue and to map non-technical customer statements into preferences for technical attributes.

1 Introduction

To be able to offer high quality assistance in complex product domains, recommender systems have to move away from their usually strictly feature-centric recommendation approaches towards customer-oriented models. Any good natural salesperson does not primarily ask technical questions to a customer but rather tries to elicit the customer's needs and expectations about his/her new product. From there, the salesperson uses his/her own technical expertise to map these "soft" statements to technical attributes that are most likely to satisfy the customer's needs.

Furthermore, customers expect a quality of recommendation when shopping online that is comparable to visiting a store – particularly if the product domain is highly complex or changes frequently, so that even technically savvy users would require assistance in their purchasing decisions. Electronic recommender systems that surpass the common simple "configurators" in functionality are a necessity to increase acceptance of online sales in these domains.

The contribution of this paper is an industrial strength customer metamodel, coupled with a Bayesian inference engine which is automatically derived from an instance of the customer model. The Bayesian engine allows a recommender system to classify its users with respect to different stereotypes, to assess their needs and, finally, to obtain information about the most recommendable products by inferring likelihoods for the different possible technical characteristics of the products-to-be-sold. Apart from being used to obtain product recommendations, a conversational recommender system can use the inference engine to decide on a dynamic course of its dialogue. The presented customer metamodel was designed in cooperation with an industry partner, to closely resemble the natural recomendation procedures in a wide range of imaginable business domains. When using the metamodel, the mathematical complexity is hidden, which enables intuitive model maintenance by non-programmers, i.e. marketing experts.

In summary, we present: (1) an efficiently maintainable, industrial strength *customer metamodel*, (2) an application of *Bayesian networks* as an inference engine for reasoning about customer profiles in recommender systems, (3) a *generation-method* to automatically derive the inference engine from an instance of the metamodel.

The rest of this paper is organized as follows: In section 2 we detail a representative use case before we describe our customer metamodel in section 3. We give a detailed description of the inference engine and its use in a recommender system in sections 4 and 5. We briefly cover model maintenance in section 6 and review some related work in section 7, before concluding with an outlook in section 8.

2 Use Case

Today's market for new automobiles is characterized by a huge number of choices, extended by different variants per vehicle, numerous optional features and special equipment often available only in packages combined with other extras. Customers need qualified consultation to match their (often vague) preferences with these complex product models. However, when visiting the web sites of major car manufacturers, we find that these offer so-called "configurators" only, that completely lack high quality recommendation functionality.

The course of action of "natural" salespersons is notably different, according to our personal experiences and interviews with vendors: Initially, the customer is classified into a set of broad stereotypes, such as "business customer", "young", or "male". Based on this classification, the vendor determines the further course of the dialogue. Different stereotypes of customers likely have different needs and expectations about their future car and the salesperson tries to assess those based on his/her experience in the field. Those needs are then mapped into suggestions for technical features and, consequently, available products. The customer is only asked technical questions when it is necessary for the recommendation process. Note that this concentration on *soft criteria* is the key difference between a natural sales dialogue and the common technology-oriented online car configurators.

Apart from its complexity, the product domain changes frequently and often radically. Updates of the product domain may stem from anything ranging from a temporary marketing campaign to the introduction of new vehicle variants or even technical innovations which often require significant adjustments to a recommendation dialogue to accommodate pre-

[1] Institute for Information Systems and Software Technology, University of Passau, Germany, email: {radde, kaiser, freitag}@uni-passau.de

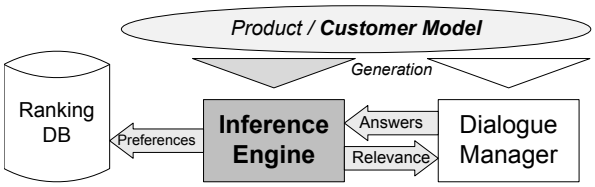


Figure 1. Architecture overview

viously unknown functionality. As an example, imagine the recent boom of GPS navigation assistants and the currently emerging approaches to location-based services where salespersons have to familiarize themselves with entirely new technologies, services and even business models. Therefore, maintainability of the domain model plays an important role, too.

We are cooperating with a local industry partner to realize a model-based conversational recommender system for this use case and similarly structured business domains. In [11], a dialogue structuring method was presented, into which the inference engine described here can be embedded. A ranking-enabled database querying technique was presented in [3]. Fig. 1 illustrates how the approach described here is integrated with those components to form a complete recommender system architecture (cf. section 5 for more detail).

The metamodel structure was designed by interviewing domain experts to closely resemble actual sales practice, recommendation processes and product catalog structures. Apart from being used in our prototypical "car domain", it was successfully instanced for another, similarly structured business domain in cooperation with a local industry partner. A detailed market study was conducted to supplement the experts' opinions and to build a solid understanding of the currently applied recommendation methodologies in that market to enable evaluations of our approach and to validate our notions of the used catalog and customer models. In the further course of our ongoing project, we will roll-out prototypes that build upon the metamodel to enable us to do field-tests to judge 1) the ability of domain experts to model their business domain and 2) the quality and precision of the produced predictions.

3 Customer Metamodel

The central part of the model is a domain-dependent description of the (prospective) customers. To represent the course of action shown in section 2, a domain expert defines a number of "interesting" *stereotypes* that are deemed relevant for an initial classification of the customer. Furthermore, a number of customer *needs* are defined that are assumed to be the driving force for the customer choosing certain products (for the product domain at hand). Finally, the model includes the technical attributes of the products to be sold, along with their possible values, as is shown in Fig. 2.

Owing to our usage of Bayesian networks in the inference engine, *stereotypes* are assigned a fixed a priori probability as part of the model. As detailed in section 4, the *stereotypes* will be represented as nodes in a Bayesian network modeling the event that a customer is of that particular stereotype.

To represent the interrelations between *stereotypes*, *needs* and *attributes*, we introduce *influences* and *matches*. These

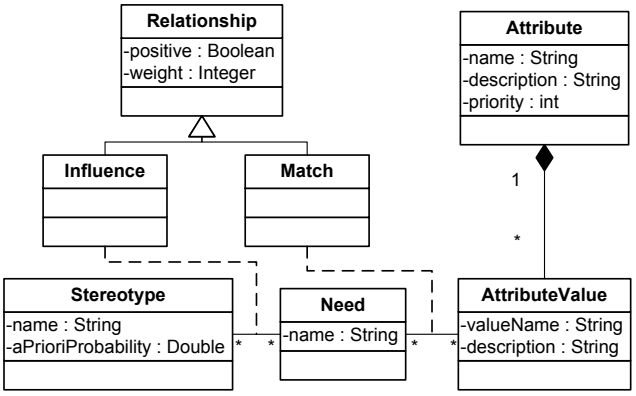


Figure 2. UML diagram for the customer metamodel

signify what kind of *influence* the fact that a customer belongs to a certain *stereotype* has on the likelihood that a customer will or will not have a particular *need* and how likely it is that a certain *attribute value* will satisfy the customer's *needs*.

Example 1 *As a brief and somewhat stripped example in the "car" domain, a domain expert might define the following* stereotypes *as relevant: "male", "young (age under 25)", "senior (age over 55)" and "business customer". Assume that the following* needs *are considered in this domain: "low price", "fast car", "comfortable car", "representative car".*

Furthermore, the domain expert defines the influences *between the* stereotypes *and the* needs *based on his marketing and domain knowledge, as shown in Table 1.*

Table 1. Positive and negative influences for example 1

positive influences		
"male"	→	"fast car"
"young"	→	"fast car"
"young"	→	"low price"
"senior"	→	"comfortable car"
"business customer"	→	"comfortable car"
"business customer"	→	"representative car"
negative influences		
"male"	→	"comfortable car"
"young"	→	"representative car"
"senior"	→	"low price"
"senior"	→	"fast car"
"business customer"	→	"low price"

Influences from *stereotypes* to *needs* have a *type* (cf. Fig. 2): A *positive influence* means that the likelihood that a customer will have a particular *need* increases if he or she is classified as belonging to the corresponding *stereotype*. In this case, being classified as not belonging to that particular *stereotype* decreases the corresponding likelihood. A *negative influence* is to be interpreted the other way round. We also introduce a numerical *weight* to be able to assign some measure of "importance" to different relationships.

In our Bayesian engine described later, the relation will be used to calculate the conditional probability tables for the nodes that represent customer *needs*, based on the a priori

probabilities of the *stereotypes* (which would be replaced by evidence once the salesperson begins the dialogue).

Now, to be able to find products that are suitable for a particular customer, we have to establish another relationship that maps our knowledge about the customer to the technical properties of a product as stored in the product database. Our notion of the interaction between a customer's needs and the technical features is that particular values of product features may help to *satisfy* the needs of a customer.To this end, we define positive and negative *matches* between customer *needs* and possible *values* of product features very much the same way as the *influences* between *stereotypes* and *needs* (cf. Fig. 2). The underlying assumption is that having certain *needs* increases (or decreases) the likelihood that a customer prefers products with particular technical attributes, which enables us to retrieve the appropriate products from the catalogue.

Example 2 *When looking at the car painting, the* need *of "low price" has an influence on the possible values of the attribute "price" (e.g., "up to €500", "€500 to 1.000", and "more than €1.000"): The influence on "up to €500" is clearly positive, as it is the cheapest painting available. For the contrary reason it is negative on the price category "more than €1.000". The middle price category is not influenced by the* need *"low price". The rest remaining unchanged, this means that the cheapest paint has the highest likelihood of satisfying the customer's* needs*, followed by the middle price paint. The most expensive paint has the smallest probability.*

Furthermore, the need *"representative car" has a positive match with "metallic paint type" and the colors "black" and "blue", while it has a negative match with colors like "red", "yellow" etc. As an example showing that the relationship does not necessarily have to be based on purely objective criteria, consider a positive match between the* need *"fast car" and the color "red".*

Apart from eliciting the necessary information from a domain expert, it is possible to try to *learn* this relationship, e.g., by conducting dedicated marketing research or by analyzing data from previous successful sales or CRM systems. Those techniques may also be used to verify and refine an initial assessment made by a domain expert.

4 Generating the Inference Engine

4.1 Network Structure

To realize the inference engine with these desired properties, our approach is to represent the user model as a Bayesian network (cf., e.g., [12]). Some of the properties of Bayesian networks come particularly handy in our use case:

- Partial knowledge is transparently integrated into the network: In some cases, we do not have explicit information (= evidence) for, e.g., all the *stereotypes* that the customer may belong to. Evidence may easily be provided at any time to correct or refine the predictions.
- Although the predictions may not be very precise when evidence is scarce, all inquiries about a posteriori probabilities can be answered any time.
- Introducing evidence for a node has influence on the probability distributions of that node's parents. This leads to refined predictions in other parts of the network as well.

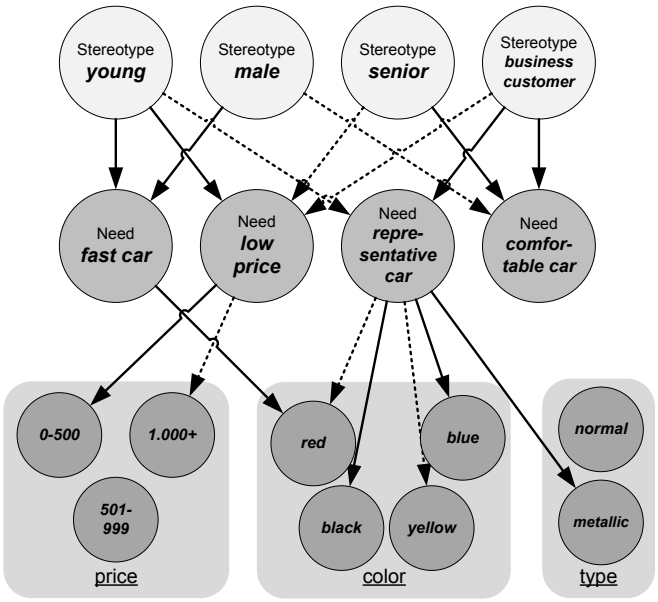


Figure 3. Bayesian network for the car painting *(solid edges denote positive influences/matches, dashed edges denote negative influences/matches)*

As established in section 3, our customer metamodel consists of *stereotypes*, *needs*, *attributes* and relations linking *stereotypes* and *needs*, as well as *needs* and *attributes* respectively. *Stereotypes* and *needs* will be represented as random variables (nodes) in our network, with a Boolean value range $\{false, true\}$. *Influences* and *matches* will form the edges of the network.

The Boolean values model the situation that a customer either belongs to a certain *stereotype* (resp. has a certain *need*) or not. Modeling each *stereotype* as a separate node which may be true independently of the others allows a very detailed classification of the customer in the form of a *"composite" stereotype*, e.g., "young female business customer" (based on the stereotypes introduced in example 1).

Product attributes in our Bayesian network are represented in a way similar to the representation of *needs* and *stereotypes*. However, attention must be paid to the fact that attributes generally do not have a Boolean value range. Instead, an attribute may have one of several values (e.g. the painting may be of several colors), as already detailed in the UML model (cf. Fig. 2). We chose to represent *each possible value* of each feature as a distinct node in our Bayesian network, again with a value range of *true* and *false*. A node in this case therefore represents the probability distribution that a particular value of an attribute is useful or desirable for the customer (i.e., the attribute value *matches* the *needs* of the customer).

Combining the model elements given in examples 1 and 2, a Bayesian network that deals with the car's paint is given in Fig. 3. For a complete, real life use case, the number of nodes at all three "layers" of the graph would grow, with the largest number of nodes to be expected for the attribute values.

An alternative representation of attributes could have used one single node per attribute that has all possible attribute values as its value range (i.e. a node "color" with "blue",

"yellow" and "red" as possible values instead of three Boolean nodes "color::blue" etc.). While at a first look this approach seems more intuitive, the Boolean modeling captures some possible situations better than the alternative:

(1) The probabilities that the individual attribute values meet the needs of the customer are independent of each other. In particular, the case that two different attribute values are definitely useful for the customer (i.e. with 100% probability) could not be modeled with a single node per attribute.

(2) Having individual probabilites per attribute value gives an automatic normalization of the value range to [0,1]. With a single node per attribute, the information that one particular value would be useful with a probability of, e.g., 20% would not mean much without knowing at least the cardinality of the value range.

4.2 Conditional Probability Tables

Stereotype nodes will be assigned an a priori probability distribution (see also Fig. 2), based on CRM information about the demographic structure of the potential customers.

Definition 1 *r_n is defined as the* random variable *corresponding to node n in our Bayesian network. As mentioned previously, all r_n are binary random variables.*

To denote the set of all random variables corresponding to a set N of nodes, we use R_N.

Example 3 *Using data provided by the company's CRM system, the product manager assigns the following probabilities for the* stereotypes *of example 1:*

n	$p(r_n = true)$	$p(r_n = false)$
"male"	*0.70*	*0.30*
"young"	*0.40*	*0.60*
"senior"	*0.20*	*0.80*
"business cust."	*0.25*	*0.75*

We now have to provide a means to derive the conditional probability tables for nodes that represent *needs* and *attribute values*. To this end, we use a method similar to the "Noisy Add" technique (cf. [5]), which we now present in more detail.

Definition 2 *Let N be the set of nodes in the Bayesian network. An* edge *is a tuple $e = (s, d, t, w)$ with $s \in N$ the source node, $d \in N$ the destination node, $t \in \{pos, neg\}$ the "*type*" and w a positive real number representing the weight.*

Note that there are only edges between stereotypes *and* needs*, and* needs *and* attribute values*, respectively, as illustrated by the "three-layered" shape of the example network in Fig. 3.*

We define E as the set of all edges in the network. To access the individual components of a single edge, we use the common dot-notation, e.g., $e.w$ to denote the weight w of edge e.

The semantics of an edge $e = (s, d, t, w)$ are as follows: If $t = pos$, the probability of r_d is increased if and only if r_s is *true*, "weighted" by w. For the case $t = neg$, the probability is decreased correspondingly. Intuitively, e.g., for a positive influence between a stereotype s and a need n, the probability that the customer feels the need n is increased iff. the customer belongs to s.

Definition 3 *For a* node $n \in N$ *we define the set E_n of* incoming edges *of n by $E_n := \{(s, n, t, w) \in E\}$ and the set P_n of* parents *of n by $P_n = \{s \mid (s, n, t, w) \in E\}$.*

Definition 4 *$V(P_n)$ is defined as a* valuation *of the random variables in R_{P_n}. We use $V_s(P_n)$ to denote a valuation of a single random variable $r_s \in R_{P_n}$.*

Using these defined valuations, the conditional probability $P(r_n = true \mid R_{P_n})$ that n occurs provided that the parent nodes of n occur is calculated using a *weighted average* of the parent random variables. For ease of notation, we first introduce the *weighted value* of an edge $e \in E$ given a valuation $V(P_{e.d})$ as:

$$wv(e, V(P_{e.d})) = \begin{cases} e.w, & iff\, V_s(P_{e.d}) = true \wedge e.t = pos \\ e.w, & iff\, V_s(P_{e.d}) = false \wedge e.t = neg \\ 0, & otherwise \end{cases}$$

Intuitively, $wv(e, V(P_{e.d}))$ returns the weight of the edge e in two cases: 1) The type of e is *pos* and $r_{e.s}$ is *true* for the valuation at hand or 2) the type of e is *neg* and $r_{e.s}$ is *false*. In all other cases, $wv(e, V(P_{e.d}))$ returns 0.

Now, to calculate the conditional probability distribution $P(r_n \mid V(P_{e.d}))$ of node n for a given valuation $wv(e, V(P_{e.d}))$ as a weighted average of the parents, we have to sum up the weighted values of all edges in E_n and divide by the sum of all weights in E_n:

$$P(r_n = true \mid R_{P_n})) = \frac{\sum_{e \in E_n} wv(e, V(P_n))}{\sum_{e \in E_n} e.w}$$
$$P(r_n = false \mid R_{P_n})) = 1 - P(r_n = true \mid R_{P_n}))$$

This equation enables us to calculate the complete conditional probability table of the random variable r_n of node n, when the calculation is executed for all possible valuations $V(P_n)$. As an example, we detail the calculation of the conditional probability for node "comfortable car" (cf. Fig. 3):

Example 4 *For ease of notation, let n be the* need *"comfortable car" and s_1, s_2, s_3 the* stereotypes *"male", "senior" and "business customer", respectively.*

Using sample weights of 1, 2, and 3, respectively, we get the following influences for n:
$E_n = \{(s_1, n, neg, 1),\ (s_2, n, pos, 2),\ (s_3, n, neg, 3)\}$

We can now calculate the conditional probability distribution of r_n for a given valuation $V(R_{P_n^E})$, e.g.:
$p(r_n = true \mid r_{s_1} = true, r_{s_2} = false, r_{s_3} = true) = \frac{0+0+3}{6} = 0.5$
$p(r_n = false \mid \ldots) = 1 - p(r_n = true \mid \ldots) = 0.5$

Intuitively, this means that a customer who is characterized as a "male business customer that is not a senior" has a 50% probability of feeling the need for a comfortable car. Table 2 shows the complete conditional probability table for all possible valuations (i.e. all possible customers).

Given the a priori probability distributions of the *stereotype* nodes, an initial assessment of the *needs* that the "stereotypical" customer will or will not have can be done. When conducting an actual recommendation process with a customer, a recommender system would now assign evidence to any or all of the *stereotypes* nodes based on its initial assessment of the customer to obtain a personalized *needs* profile. Naturally, it is possible to re-state this evidence later or, instead of providing evidence, to just modify the a priori probability distribution if a more fine-grained assessment is desired.

Table 2. Conditional prob. for the need "comfortable car"

s_1	s_2	s_3	$p(n = true \mid \ldots)$	$p(n = false \mid \ldots)$
t	t	t	$(0+2+3)/6$	$1/6$
t	t	f	$(0+2+0)/6$	$4/6$
t	f	t	$(0+0+3)/6$	$3/6$
t	f	f	$(0+0+0)/6$	1
f	t	t	$(1+2+3)/6$	0
f	t	f	$(1+2+0)/6$	$3/6$
f	f	t	$(1+0+3)/6$	$2/6$
f	f	f	$(1+0+0)/6$	$5/6$

5 Using the Inferred Knowledge

In [3] an approach to rank items based on weight-annotated boolean conditions has been presented which was implemented as an extension of the standard SQL syntax allowing for a ranked result retrieval in databases. In this case, the calculated likelihoods can be directly used as weights defining the desired ranking, thereby balancing all of the customer's preferences against each other. When seen as an application of Multi-Attribute Utility Theory (MAUT) [13], this approach defines a utility function and profits from the fact that the single-attribute evaluations are not done on the attributes themselves but rather on the conditions, implying a normalized (i.e. boolean) value range.

As an alternative to utility methods, PreferenceSQL [8] could be used to query the product database, if the calculated likelihoods are transformed into suitable terms of its pareto-optimality based preference algebra.

Apart from creating product recommendations, the primary task of a conversational recommender system as presented in [11] is to choose the next question to ask the customer in an intelligent manner. Our inference engine allows to estimate the probability that a customer will have a particular *need* n. This probability is interpreted as how relevant n is to the customer, as the following definition details.

Definition 5 *Let $p(n) \in [0,1]$ be the calculated probability that the current customer has the* need *n. Then, $rel(n) = 2 * |p(n) - 0.5|$ is the* relevance *of n to the customer. The higher the numerical value of $rel(n)$, the more relevant is n to the customer.*

The definition of $rel(n)$ is based on the assumption that a user will have stronger feelings about a particular *need* when we predict either a very high or a very low probability for that *need*. The dialogue manager of a recommender system should make sure to verify these predictions first, before attempting to clarify any *needs* that the user may have no clear opinion about (i.e. those with a predicted probability close to 0.5). Those *needs* may be examined later, when the customer has begun to build up trust in the recommender system.

Should it become necessary that detailed questions about the customer's technical preferences are required, it is necessary to define a relevance measure for *attributes* as well. We do so in a way similar to *needs*, taking into account that a question about an *attribute* will try to elicit answers about all possible values at once. The likelihoods that the various values of an attribute are useful for a customer are combined into a single relevance indicator.

Definition 6 *Given attribute a with $dom(a) = \{v_1, \ldots, v_n\}$ and the probabilities $p(v_1), \ldots, p(v_n)$ that value v_i matches the preferences of the customer, we define the* relevance *of a as $rel(a) = \frac{\sum_{v \in dom(a)} |p(v_i) - 0.5|}{n/2}$. Intuitively, the "distances" of the probabilities to the "undecided" value of 0.5 for each v_i are summed up and divided by $n/2$ returning a value in $[0,1]$. The higher the numerical value of $rel(a)$, the more relevant is a to the customer.*

Again, as with definition 5, $rel(a)$ is based on the assumption that attributes the customer has a clearly stated opinion about have a higher importance than those he or she is indifferent about. Consequently, these should be verified first and also be given a higher influence when computing the product recommendations than those for which the predicted probabilities are closer to 0.5.

Depending on the way the dialogue manager combines the relevance information for *needs* and *attributes*, using a common scale for both may be in order. Therefore, we chose to normalize $rel(n)$ and $rel(a)$ to the interval $[0,1]$.

6 Model Maintenance

The approach was specifically designed to accommodate frequent model changes as efficiently as possible. Apart from major revisions of the model, all changes, such as extending the list of considered stereotypes are rather local modifications that leave the rest of the model intact. After the model has been modified to satisfaction, the Bayesian network can be rebuilt automatically from the model.

Example 5 *Regarding the current trend towards "green" technologies, a product manager decides to include the new* need *"environmentally friendly" into the customer model. First, he or she must decide which of the considered stereotypes influence this need. Second, the same must be done to find technical attributes that match this need, e.g., fuel consumption.*

Of course, developing friendly user interfaces that efficiently *guide* a non tech-savvy domain expert (say, e.g., a marketing manager) through the model maintenance process is still a task to be done. However, the stability of the model and inference layers allows for a separation of concerns in the development process and the model was specifically designed to allow an intuitive representation of a domain expert's marketing knowledge. When properly integrated into the existing business processes of catalogue creation and maintenance, these tools have the potential for even further optimizations, as they can be re-used, e.g., for the creation of printed marketing material in a natural way.

7 Related Work

Bayesian networks are used by Ji et al. in [7] to obtain recommendations in the commodities market. In contrast to our approach, they do not rely on a domain or customer model, but focus on learning the structure of the network and all probabilities from history data. Based on evidence provided by the current customer's purchases, other commodities are recommended depending on their posteriori probabilities. This kind

of evidence is not available in our application scenario, as the customer generally will make a single purchase and leave.

Park et al. use Bayesian networks for a very detailed user representation in [10]. They use an expectation maximization algorithm to learn the conditional probability tables on their network. However, the structure of the network itself has been designed by a domain expert and is intended to remain fixed. Therefore, their approach requires extensive work when the underlying model changes.

Ardissono et al. [1, 2] present a personalized recommender system for configurable products. Their approach involves preference elicitation techniques that employ reasoning about customer profiles to tailor the dialogue to a particular customer by providing explanations and smartly chosen default values wherever possible. The customer preferences learned this way are used as constraints in the configuration problem at hand to generate the recommended product configuration, which might result in empty recommendations (i.e. the specified constraints are not satisfyable), thus requiring repair actions. Our approach does not directly take elicited preferences as constraints but rather uses them as inputs to ranking-enabled database queries, returning a list of product recommendations which is ordered according to the customer's preferences. Giving a pre-sorted list of products instead of a single "optimal" product is intended to improve the customer's freemdom of choice. By evaluating the relevance measure, our approach can also suggest personalized default answers. It does not need to rely on extensive domain knowledge or a set of business rules as is the case in [1].

An approach similar to the one proposed in this paper is presented by Jameson et al. [6]. However, their utility estimations (the "value tree") do not seem to be built on an explicit model of the currently served customer but rather on an average user of their system. Hence, the recommendations are not personalized as strongly as in our approach which allows an adaption even to atypical customers by setting the appropriate stereotypes. Also, as the value tree is a strictly hierarchical structure, it cannot capture the fact that a technical attribute may be influenced by more than a single need. Furthermore, it is not completely clear how informal statements (i.e., "I am a law student.") can be interpreted as relevant knowledge (i.e. an increased interest in politics) by the system unless a domain expert models this association directly within the Bayesian Network.

In [4] Cao and Li develop a recommender system for consumer electronics by using a fuzzy-based approach for the inference process which involves reasoning about a product's features. The approach does not include an explicit customer model and therefore is limited to reason only about the technical features of products. Therefore, its potential for a conversational recommender system that aims at complex product domains appears limited.

A domain model based on dynamic logic programming was introduced by Leite and Babini in [9]. Both customer and user model are represented using a massive set of declarative rules which allows a detailed and powerful specification of the business domain – possibly even extended by user-supplied personalized rules. However, the complex formal models appear difficult to maintain and use even by domain experts, let alone customers.

8 Conclusion

We presented an application of Bayesian networks as an inference engine for reasoning about customer profiles in conversational recommender systems, enabling intelligent dialogue-management and preference-elicitation. The inference engine is generated from instances of an industry strength customer metamodel which improves the maintainability of the recommender system in complex and frequently changing product domains.

Initial evaluation of the approach based on interviews with domain experts has been quite positive. As a continuation of our work, a full-scale field test in a real-life application domain will be conducted in cooperation with a business partner. Furthermore, although model creation and maintenance are deliberately easy and supposed to be feasible by domain experts, we investigate how techniques to learn the Bayesian network structure from available sales data can be integrated to refine and verify the experts' assumptions.

REFERENCES

[1] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, M. Meyer, G. Petrone, R. Schaefer, W. Schuetz, and M. Zanker, 'Personalizing online configuration of products and services', in *Proc. of the 15th European Conference on Artificial Intelligence (ECAI)*, (2002).

[2] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schaefer, and M. Zanker, 'A framework for the development of personalized, distributed web-based configuration systems', *AI Magazine*, **24**, 93–110, (2003).

[3] M. Beck and B. Freitag, 'Weighted boolean conditions for ranking', in *Proc. of the ICDE-08 Workshop on Ranking in Databases (DBRank'08)*, (2008).

[4] Y. Cao and Y. Li, 'An intelligent fuzzy-based recommendation system for consumer electronic products', *Expert Systems with Applications*, **33**, 230–240, (2007).

[5] P. Dagum and A. Galper, 'Additive belief-network models', in *Proc. of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, (1993).

[6] A. Jameson, R. Schaefer, J. Simons, and T. Weis, 'Adaptive provision of evaluation-oriented information: Tasks and techniques', in *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, (1995).

[7] J.Z. Ji, Z.Q. Sha, C.N. Liu, and N. Zhong, 'Online recommendation based on customer shopping model in e-commerce', in *Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence (WI)*, (2003).

[8] W. Kießling and G. Köstler, 'Preference SQL – Design, Implementation, Experiences.', in *Proc. of the 28th International Conference on Very Large Data Bases (VLDB)*, (2002).

[9] J. Leite and M. Babini, 'Dynamic knowledge based user modeling for recommender systems', in *Proc. of the ECAI-06 Workshop on Recommender Systems*, (2006).

[10] M.-H. Park, J.-H. Hong, and S.-B. Cho, 'Location-based recommendation system using bayesian user's preference model in mobile devices', in *Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC)*, (2007).

[11] S. Radde, M. Beck, and B. Freitag, 'Generating recommendation dialogues from product models', in *Proc. of the AAAI-07 Workshop on Recommender Systems in E-Commerce*, (2007).

[12] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall International Editions, 1995.

[13] D. von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*, Cambridge University Press, 1986.

Social Ranking: Finding Relevant Content in Web 2.0

Valentina Zanardi[1] and **Licia Capra**[2]

Abstract. Social (or folksonomic) tagging has become a very popular way to describe, categorise, search, discover and navigate content within Web 2.0 websites. Unlike taxonomies, which overimpose a hierarchical categorisation of content, folksonomies empower end users by enabling them to freely create and choose the categories (in this case, tags) that best describe some content. However, as tags are informally defined, continually changing, and ungoverned, social tagging has often been criticised for lowering, rather than increasing, the efficiency of searching, due to the number of synonyms, homonyms, polysemy, as well as the heterogeneity of users and the noise they introduce. In this paper, we propose a method to increase the efficiency of searches within Web 2.0 that is grounded on recommender system techniques. We measure users' similarity based on their past tag activity. We infer tags' relationships based on their association to content. We then propose a mechanism to answer a user's query that ranks (recommends) content based on the inferred semantic distance of the query to the tags associated to such content, weighted by the similarity of the querying user to the users who created those tags. We evaluate the effectiveness of this mechanism when performing searches on the CiteULike dataset.

1 INTRODUCTION

The advent of Web 2.0 has transformed users from passive consumers to active producers of content. This has tremendously increased the amount of information that is available to users (from videos on sites like YouTube and MySpace, to pictures on Flickr, to music on Last.fm, and so on). This content is no longer categorised according to pre-defined taxonomies. Rather, a new trend called *social* (or *folksonomic*) *tagging* has emerged and quickly become the most popular way to describe, categorise, search, discover and navigate content within Web 2.0 websites.

Unlike taxonomies, which overimpose a hierarchical categorisation of content, folksonomies empower end users by enabling them to *personally* and *freely* create and choose the categories (in this case, tags) that best describe a piece of information (a picture, a blog entry, a video clip, etc.). Tag clouds are then widely used to visualise a set of related tags that best describe either individual items or the content of a website as a whole, with the most frequently used tags being given more importance either in font size or color. Other visualisation techniques have been studied, in order to give more importance to tags' relationships rather than popularity [5, 11]. When users want to find content, they navigate, via hyperlinks, from a tag to a collection of items that are associated with that tag.

However, as tags are informally defined, continually changing, and ungoverned, social tagging has often been criticized for lowering, rather than increasing, the efficiency of searching [3]. This is due to the number of synonyms, homonyms, polysemy, as well as the heterogeneity of users, contexts, and the noise that they introduce.

In order to 'connect' users with content that they deem relevant with respect to their interests, efficient searching techniques have to be developed for this novel and unique domain. By efficient, we mean that the searching technique should be both *accurate* (i.e., the returned content does satisfy users' interests), and *complete* (i.e., if there is relevant content in the system, this should be found).

In this paper, we propose a technique, called *social ranking*, that aims to efficiently find, within a potentially huge dataset, content that is relevant to a user's query. In typical Web 2.0 fashion, we assume such content to have been described with an arbitrary number of tags and by an arbitrary number of users. We begin with a study of the key characteristics of a typical Web 2.0 website (Section 2). Based on these insights, we propose a mechanism to answer a user's query that is grounded on traditional recommender system techniques (Section 3): we measure users' similarity based on their past tag activity; we infer tags' relationships based on their association to content; finally, we rank (recommend) content based on the inferred distance of the query to the tags associated to such content, weighted by the similarity of the querying user to the users who created those tags. Preliminary experimental results demonstrate the good accuracy and coverage of social ranking (Section 4). We position ourselves with respect to other works in the area in Section 5, before discussing our plans for the future (Section 6).

2 DATASET ANALYSIS

In order to understand the key characteristics of the target scenario, and thus develop a query model that is grounded on its peculiarities, we started with the analysis of a typical Web 2.0 website, that is, CiteULike (http://www.citeulike.org). CiteULike is a social bookmarking website that aims to promote and develop the sharing of scientific references amongst researchers. Similarly to the cataloging of web pages within del.icio.us, and of photographs within Flickr, CiteULike enables scientists to organize their libraries with freely chosen tags which produce a folksonomy of academic interests. CiteULike runs a daily process which produces a snapshot summary of what articles have been posted by whom and with what tags. We downloaded one such archive in December 2007. The archive contained roughly 28,000 users, who had tagged 820,000 papers overall, using 240,000 distinct tags. A pre-analysis of the archive revealed the presence of a vast amount of papers and a vast amount of tags bookmarked/used by one user only. In order to make the dataset more manageable, we decided to prune it so to remove those papers and tags that had been bookmarked/used only once over the entire dataset. We were thus left with roughly 100,000 papers and 55,000 distinct tags (while keeping

[1] Dept. of Computer Science, University College London, UK, email: v.zanardi@cs.ucl.ac.uk
[2] Dept. of Computer Science, University College London, UK, email: l.capra@cs.ucl.ac.uk

all 28,000 users). We believe that this pruning of the dataset has not compromised our analysis (and subsequent performance results), as there is little one can do to improve search efficiency on papers/tags nobody else knows.

We then analysed the remaining dataset more carefully in terms of *users' activity*, *papers' popularity*, and *tags' usage*.

Users' Activity. To begin with, we studied how many papers were tagged on average by each user in the system. As expected, there is a huge variance in users' activity, with roughly 70% of the users tagging less than 10 papers (low activity), while the remaining 30% bookmarks between 10 and 50 papers (medium activity), and between 50 and 200 papers (high activity). Note that even users with the most intense activity only bookmark a tiny portion of the whole paper set, thus suggesting a very focused and scoped interest within the broader scientific community.

We also analysed the size of the vocabulary these users spoke, that is, how many different tags were ever used by each user. We found that more than 70% of users only used less than 20 different tags, another 15% of users used between 20 and 60 tags, and the remaining used between 60 and 120 different tags. Once again, the extremely narrow proportion of tags used by each user suggests that user's interest is rather scoped in this domain, so that the vocabulary spoken by each of them is just a tiny proportion of the emerging folksonomy.

Papers' Popularity. We studied papers' relevance next, that is, we quantified how many users had bookmarked the same paper. The vast majority of papers (roughly 87%) were tagged by less than 5 users (low popularity); 12% were tagged between 5 and 15 times (medium popularity), and the remaining 1% more than 15 times (high popularity). This suggests that there is a small subset of highly popular papers who have been bookmarked by a significant proportion of the community, while there is a very long tail of less popular ones.

We also looked at how many different tags were used to describe each paper. 84% of papers had less than 10 different tags associated to them (and more than 54% of them with less than 5). The remaining 16% of papers used between 10 and 30 tags. This would suggest, in accordance with the analysis of users' activity previosuly done, that only a small subset of the whole folksonomy is needed to describe papers (and thus topics) - that is, users and tags are highly clustered around papers/topics.

Tags' Usage. Finally, we studied tags' usage, that is, to what extent the emerging vocabulary is shared among users. The vast majority of tags (roughly 70%) were used by less than 20 users, and an additional 12% by between 20 and 40 users. However, a non negligible 18% were actually shared by more than 40 users. As for papers' popularity, there exists a small subset of tags that are very widely used, and a very long tail of less popular ones.

We also studied how spread was the usage of tags, that is, to how many different papers was a tag associated. Confirming previous observations, we found the vast majority of tags (in excess of 70%) to be used on a tiny proportion of papers (less than 20), another 10% to be used for between 20 and 40 papers, with the remaining being used for more than 40 papers. Despite the huge number of tags in use in the CiteULike folksonomy, tags are thus shared by small communities of users and highly clustered around papers/topics.

2.1 Insights

Based on the dataset analysis summarised above, the following insights can be drawn.

Clustering of Users: users vary a lot in terms of activity; however, even the most active users bookmark a tiny proportion of the whole paper set. This suggests that users have clearly defined interests that map to a small proportion of the whole CiteULike content. This is confirmed by tags' usage: each user masters a small subset of the whole folksonomy, and users sharing part of the folksonomy form small clusters. We formulate the hypothesis that, by looking at users' tag activity, *users' similarity* can be quantified and exploited to answer content searches more accurately.

Clustering of Tags: despite the emergence of a rather broad folksonomy, each paper only needed a small set of tags to be described. This would suggest that there is a core of shared and agreed knowledge about tags within the communities who use them, and these are recurrently used to describe the same papers. We formulate the hypothesis that, by looking at what tags were associated to what papers, *tags' similarity* (or, rather, 'relationship') can be quantified and exploited to uncover relevant content.

In the next section, we describe how we used these hypothesis to develop our content search and recommendation technique.

3 SOCIAL RANKING

Let us consider a user $\overline{u}$ who is interested in finding some content of interest (in our specific case, papers). In a typical Web 2.0 scenario, $\overline{u}$ would submit a query $q_{\overline{u}}$ which consists of query tags $t_1, t_2, \ldots, t_n$. The system answering the query would normally rank results according to the following two criteria: the higher the number of query tags associated to the resource, the higher its ranking; and, the higher the number of users u_i who tagged the resource using (some of the) query tags, the higher its ranking. Intuitively speaking, the first criterion caters for accuracy of the result, the second caters for confidence in it. The formula used could look like:

$$R(p) = \sum_{u_i} (\#t_i \text{ used by } u_i \text{ on } p \mid t_i \in q_{\overline{u}}) \ , \qquad (1)$$

that is, the ranking of paper p is computed as the number of tags t_i that users u_i who bookmarked p used and that belonged to the query set $q_{\overline{u}}$.

While this simple technique could work well to find popular content described with popular tags (i.e., with reference to our previous analysis, papers that have been tagged more than 10 times using a small subset of popular tags), the technique would likely fail to address queries that look for the very long tail of medium-to-low popularity content, as a large amount of results would be returned, all scoring low. Accuracy would not be the only problem: if the user running the query used tags that also belong to the long tail, chances are that no content would be found at all, and coverage would then become a major issue.

To address these problems, we propose *social ranking*, a technique inspired by traditional Collaborative Filtering [2]: first, we identify who are the users with similar interests to the querying user $\overline{u}$; according to our analysis, such community should be easily identified by studying users' tag activity. Content tagged by these users should be scored higher in a way that is proportional to the quantified similarity. Second, even though tags can be broadly clustered in domains of knowledge, people tend to use slightly different subsets of them within each domain (as shown by the low number of tags used by each individual and on each paper). We thus identify the tags that are similar (or, rather, related) to the query tags, thus expanding the query to this enlarged set. We believe, and our evaluation will confirm, that *users' similarity improves accuracy* of the results, while *tags' similarity (i.e., query expansion) improves coverage*.

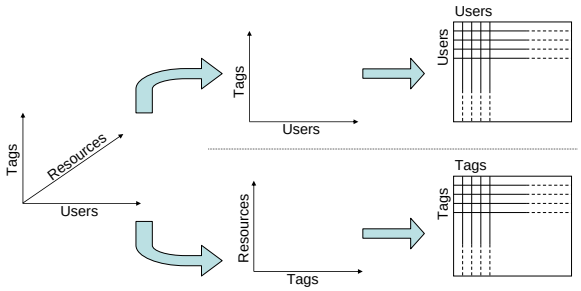


Figure 1. Transformation of the dataset

In the reminder of this Section, we illustrate how we compute users' similarity (Section 3.1), how we compute tags' similarity (Section 3.2), and how we combine these two techniques together (Section 3.3).

3.1 Users' Similarity

Social tagging typically provides a 3-dimensional relationship between users, resources and tags (users bookmark resources using a certain number of tags). Different definitions of users' similarity can be derived; here we consider a simple yet effective one: the more tags two users have used in common, the more similar they are, regardless of what resources they used it on. This definition projects our 3-dimensional space onto a 2-dimensional one, throwing away information about 'resources', and keeping only information about what tags a users has used and how often (Figure 1, top). While one may argue that, in so doing, we discard important information, we believe that, in scenarios where tags are highly clustered around topics, the information lost is not significant.

We thus describe each user u_i with a vector v_i where $v_i[j]$ counts the number of times that users u_i used tag t_j. Given two users u_i and u_j, we then quantify users' similarity $sim(u_i, u_j)$ as the cosine of the angle between their vectors:

$$sim(u_i, u_j) = cos(v_i, v_j) = \frac{v_i \cdot v_j}{||v_i|| * ||v_j||}$$

Various similarity measures can be used other than the cosine-based similarity [6]. For example, concordance-based similarity [1] could be used, so that the more tags two users share, the more similar they are (regardless of how many times they have used them). However, we believe tag frequency to be an important piece of information to determine a user's interests. Alternatively, Pearson Correlation (and its variations - e.g., weighted Pearson [19, 6]) could be used; as shown in [14], different similarity measures perform differently, both in terms of accuracy and coverage; we chose cosine-based similarity for its constantly good performance, although we plan to study the impact of other similarity measures in the future.

3.2 Tags' Similarity

We define tags' similarity as follows: the more resources have been tagged with the same pair of tags, the more similar (related) these tags are, regardless of the users who used them. This definition projects our 3-dimensional space onto a 2-dimensional one, as shown in Figure 1, bottom part. Similarly to what we said before, in scenarios where users' interests are a rather small and consistent subset of the broader range of topics in the whole website, we believe that the information thrown away during the projection is not significant.

We thus describe each tag t_i with a vector w_i where $w_i[j]$ counts the number of times that tag t_i was associated to paper p_j. Given two tags t_i and t_j, we then quantify tags' similarity $sim(t_i, t_j)$ as the cosine of the angle between their vectors:

$$sim(t_i, t_j) = cos(w_i, w_j) = \frac{w_i \cdot w_j}{||w_i|| * ||w_j||}$$

3.3 Two-Step Query Model

The query model we propose exploits the two similarity measures discussed above (on users and on tags) in the following way. When user $\overline{u}$ submits a query $q_{\overline{u}} = \{t_1, t_2, \ldots, t_n\}$ to discover content that can be described by query tags $t_1, t_2, \ldots, t_n$, two steps take place:

1. **Query Expansion**: the set of query tags q is expanded so to include all $t_i \mid t_i \in q_{\overline{u}}$ (for which $sim(t_i, t_i) = 1$), plus those tags $t_{n+1}, \ldots, t_{n+m}$ that are deemed most similar to the query tags (for which $0 < sim(t_i, t_j) \leq 1$, with $i \in [1, n]$ and $j \in [n+1, n+m]$). We call this set q^*. This set is constructed so that, for each $t_i \in q_{\overline{u}}$, its top k most similar tags are included, in a fashion similar to the top k Nearest Neighbour (kNN) strategy in recommender systems. Different choices of k will have an impact on both accuracy and coverage; we will discuss preliminary results for different values of k in Section 4.
2. **Ranking**: all resources that have been tagged with at least one tag from the extended query set are retrieved. Their ranking depends on a combination of: the relevance of the tags associated to the paper with respect to the query tags (papers tagged with $t_i, i \in [1, n]$ should count more than those tagged with $t_j, j \in [n+1, n+m]$); and, the similarity of the taggers with respect to the querying user $\overline{u}$ (papers tagged by similar users should be ranked higher, as these users are more likely to share interests with $\overline{u}$ than others, and thus are in a better position to recommend relevant content).

The ranking of a paper p would then be computed as:

$$R(p) = \sum_{u_i} \left(\sum_{\substack{t_i \\ t_j \in q^*}} sim(t_i, t_j) \right) * (sim(\overline{u}, u_i) + 1) \quad (2)$$

where, for each user u_i who tagged p, $\sum_{\substack{t_i \\ t_j \in q^*}} sim(t_i, t_j)$ quantifies how relevant the tags t_i associated by u_i to p are with respect to the tags t_j belonging to expanded query q^*; note that, in the basic case of formula 1, this simply meant counting how many tags from q user u_i associated to p. Moreover, the relevance is then magnified (i.e., papers are pushed higher up in the ranking) in a way that is proportional to user's similarity $sim(\overline{u}, u_i)$.

We call this approach social ranking as it exploits information coming from the emergent social network of users and social network of tags to rank content in a way that is meaningful to the querying user. We are now ready to evaluate this approach.

4 EVALUATION

In this section, we present preliminary results of the ongoing evaluation of social ranking. We begin with a brief description of the portion of the dataset we have been experimenting with, together with a characterization of the properties that are mostly relevant to social filtering (Section 4.1). We then describe how we have conducted

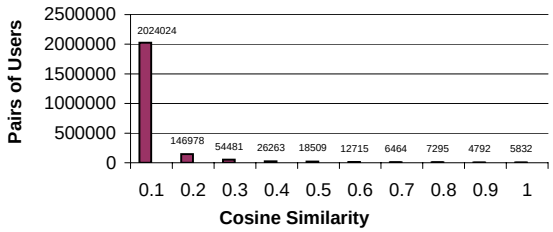


Figure 2. Distribution of users' similarity

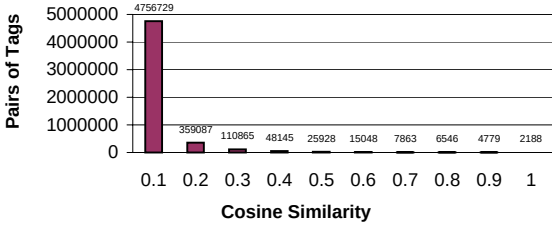


Figure 3. Distribution of tags' similarity

the experiments (Section 4.2), before analysing social ranking performance, both in terms of accuracy and of coverage (Section 4.3).

4.1 The Dataset

Based on the pre-analysis of the CiteULike dataset described in Section 2, we have performed a further cut of the dataset, in order to obtain a small yet meaningful subset to experiment with. In particular, we have considered only those tags that have been used on at least 15 different papers, and by at least 20 users. This has left us with a dataset consisting of roughly 12,000 users, 83,000 papers, and 16,000 tags. Note that the long tail phenomenon still dominates in the pruned dataset:

Long tail of users' similarity: as shown in Figure 2, the vast majority of users' pairs have very low value of similarity (below 0.1), while there exists a long tail of higher similarity pairs. This would suggest users are highly focused (and clustered) around topics, and thus only a small portion of users are indeed good recommenders to each other.

Long tail of tags' similarity: as shown in Figure 3, each tag is related to only a small subset of other tags, again suggesting that only a handful of tags are used (and thus need to be learned) to describe specific categories of content.

We believe that the results we are going to present in this section generally hold for datasets that exhibit similar characteristics.

4.2 Simulation Setup

In order to quantify accuracy and coverage of social ranking, we have been conducting the following experiment. We randomly picked up a user $\overline{u}$, randomly "hid" one of his bookmarked papers p, and then performed a query q with the tags that $\overline{u}$ had associated to p. Since p was bookmarked by $\overline{u}$ (before we hid it), $\overline{u}$ is obviously interested in it, so a recommender system should be able to return p (coverage). Note that, in our pruned dataset, it was always the case that, even after hiding $\overline{u}$'s bookmark for p, at least another bookmark made by a user u' for p existed, as we only kept in the dataset those papers that had been bookmarked by at least one user; it should thus be possible, in principle, to locate and return p. Moreover, the highest the position of p in the ranked list of returned papers, the better the accuracy of the ranking algorithm.

In all experiments, we compare the output of our social ranking algorithm (formula 2) with the simple benchmark presented in Section 3 (formula 1). Given the high variability of users' behaviour and papers' popularity in the dataset, we have been conducting 6 different sets of experiments where we varied:

- the level of activity of the querying user, distinguishing heavy taggers HT (users who tagged more than 50 papers), medium taggers MT (users who tagged between 10 and 50 papers), and low taggers LT (users who tagged less than 10 papers);
- the level of popularity of the hidden bookmark, distinguishing popular papers PP (those that had been bookmarked by at least 5 users), and unpopular ones UP (those that had been bookmarked by less than 5 users).

Our goal is to investigate the impact of these two characteristics onto the efficiency of the querying model.

4.3 Results

The first set of experiments we conducted aimed to analyse the impact of users' similarity alone on the ranking of results. We thus compared the basic query model with the advanced query model where tag expansion had been disabled. For each query, the list of returned papers is thus the same, but ranked differently. For each user in each group (heavy/medium/low taggers), three bookmarks where removed for each paper category (popular/unpopular), and their corresponding tags searched. As the number of users in each group varies, so does the total number of queries performed (from 2,200 for the small group of HT/PP, to 14,000 for the much larger group of LT/UP). Table 1 summarises the results obtained, in terms of percentage of times the advanced model does better/same/worse than the basic query model. We also report the percentage of queries for which the target paper remained uncovered.

Experiment	Better	Worse	Tied	Not Found
HT/PP	25%	28%	30%	17%
HT/UP	49%	12%	15%	24%
MT/PP	25%	15%	42%	18%
MT/UP	42%	13%	9%	36%
LT/PP	26%	13%	45%	16%
LT/UP	39%	13%	8%	40%

Table 1. Impact of Users' Similarity on the Ranking of Result

In all scenarios but the first one, the advanced query model outperforms the basic query model, and it does so more dramatically when considering unpopular papers, where the gap between the two approaches (the difference between the 'better' and 'worse' column) reaches 37%. This result confirms the importance of weighting the recommendations coming from similar users more, when looking for less 'mainstream' content. The ranking of results is slightly better (28% against 25%) when using the basic query model in the first scenario instead: when focusing on more mainstream content (i.e., the

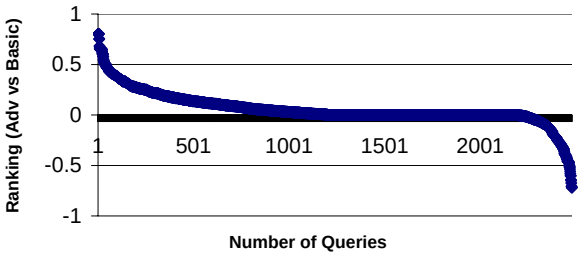


Figure 4. Improvement obtained for unpopular papers

hidden paper has been tagged many times by different users), simple searches based on exact tag matching work well enough.

In order to quantify the improvement obtained, we have computed the difference of the ranking at which a paper is found using the advanced model versus the basic model, normalised by the total number of results returned (i.e., a tiny difference in the ranking between the two techniques, for example, position 12 versus position 14, is not really appreciable by an end user). The higher the positive difference, the better the performance of the advanced model and viceversa. Figure 4 illustrates the results for heavy taggers and unpopular papers. As shown, there is a considerable set of results for which the performance of the advanced approach is not simply better than basic (positive difference), but significantly so (more than 20% of the queries ran, returned the hidden paper at a position that is between 80% and 20% higher up with the advanced model than it was with the basic model).

An important observation can be drawn when looking at the column labeled 'Not Found' in Table 1: searching techniques purely based on the matching of the query tags with those associated to papers reveal a substantial amount of uncovered resources; this percentage is approximately 16-17% for popular papers, and it increases up to 40% for unpopular ones. Low coverage is an indication that users bookmark resources differently; in order to uncover resources of interest, query tags must be expanded to include semantically related ones. We have thus conducted a second set of experiments, using the full social ranking model against the basic one, when extending each query tag with the top kNN tags. The goal of these experiemnts is to quantify the improvement obtained by social ranking on coverage, and its consequences on accuracy. We have focused on the long tail of unpopular papers, as this was the most problematic scenario, and the one where social ranking should bring the most benefits. So far we have obtained results for the heavy taggers/unpopular papers scenario for $k = 5$: the items not found are reduced from 24% to 14%; in cases where both techniques find the hidden paper, social ranking performs better in 40% of the cases, while in 31% it does worse. These inital results would confirm that it is possible, with simple techniques, to automatically learn tags' similarity and use it to boost coverage, without giving accuracy away. We are running experiments with higher values of k: preliminary results, obtained on a smaller set of queries, would indicate that coverage keeps improving up to $k = 20$, while higher values begin to have a non-negligible negative impact on accuracy. This is aligned with the pre-analysis we have conducted: each paper usually receives 10 tags or less, up to a maximum of 30 tags; expanding the query tags to larger sets thus injects too much noise within results.

5 RELATED WORK

Research in the area of social tagging has proliferated in recent years, due to the increasing popularity of such systems. Studies have been conducted both to understand tag usage and evolution (e.g., [22, 4]), and to learn and exploit their hidden semantics. In [8], a large study of social tagging on the popular del.icio.us bookmarking system is presented, aimed at characterizing users' activity, pages' popularity, and tags' distribution; the knowledge base (in this case, the whole Web) is so large and dynamic that the authors are quite pessimistic on the benefits that social bookmarking can bring to web searches. In [7], the same authors have shown how searches on del.icio.us can be improved if a navigable hierarchical taxonomy of tags is derived from tag usage, to help users broadening/narrowing the set of tags that best describe their interests. Our approach takes a different stance, and rather than offering users an organised tag navigation system, it aims to transparently improve users' searches based on emergent tags semantics and query expansion. In [18], tags are related back to a fixed ontology of concepts, thus exploiting both techniques to enhance information retrieval capabilities. Differently from this approach, our goal is to autonomically derive tags' relationships, which can then be fitted into an effective query search algorithm, without relying on a prefixed ontology. In [20], semantics that specifically relate to places and events are inferred for resources within the Flickr dataset; their approach is highly tied to location information, and thus not easily generalizable to other domains. In [24], a probabilistic generative model is proposed to describe users' annotation behavior, and to automatically derive tags emergent semantics; during searches, their approach is capable of grouping together synonymous tags, while it calls for user's intervention when highly ambiguous tags are found. Very early work, but with similar goals, is presented in [25], where a simpler technique, based on an analysis of the relationship between users, tags and resources, is proposed to disambiguate tags. Tag systems have recently revealed their susceptibility to tag spam, that is, malicious annotations generated to confuse users. The problem has been well analysed in [13], where they tried to identify misused tags, and quantify the extent to what tagging systems are robust against spam. Robust solutions to tag spamming are still being investigated.

Research has been very active also in relating tag activity to users, in order to discover their interests and consequently users' communities. Work within the Semantic Web domain has tried to classify users into categories and describe the key features of such categories [15]. More recently, users have been classified according to their explicitly stated profile [10], based on a probabilistic model which takes into account users's interest to topics [26], and based on their level of tagging activity and breadth of interests [12]. In [16], users' common interests are discovered based on patterns of frequently co-occurrent tags, using a classical association rule algorithm, which however does not take into account considerations about user's activity. All these works, including our attempt to find similar users, are based on the observation that real world networks exhibit a so-called community structure [21]; defining the set of characteristics that would enable the best fitting and natural clustering of taggers and tags is an open research question.

In this paper, we have been combining the two research streams highlighted above (i.e., automatic learning of tag semantics and users' interests) in order to improve query searches and ranking. Other research groups have been conducting research in the same area. In [23, 17], the integration of tag information within standard recommender system's algorithms has been proposed, in order to give better recommendations to users; although very promising, at

present such works do not take into account the 'activity' of users, in terms of amount of resources being tagged, and number of tags being used. We believe this information to be crucial to extract users' interests and thus improve the efficiency of searches. Tag activity has been combined with a PageRank-like algorithm, in order to improve the ranking mechanism, in situations where resources are not linked together as in a typical web graph structure [9]; their approach, called FolkRank, provides good results when querying the folksonomy for topically related elements, while it is easily subverted if less related/popular tags are being used, due to the size and sparsity of folksonomies on the web. In this work, we have tried to ameliorate the sparsity problem in folksonomy; further improvements could be achieved by clustering users within better scoped communities; we intend to explore this aspect next.

6 FUTURE DIRECTIONS

In this paper we have presented social ranking, a technique that aims to improve content searches in Web 2.0 scenarios, by exploiting users' similarity and tags' similarity. The former is used to gain confidence in the relevance of the retrieved content: the higher the similarity between the querying user and the user that has bookmarked it, the higher the chances that the paper is of relevance, thus reducing the amount of uninteresting content being presented to users. The latter is used to tackle the problem of heterogeneity, sparsity and lack of structure in folksonomy instead: by implicitly learning tags' similarity from their usage, we can increase the amount of relevant yet unpopular content being uncovered.

Ongoing work spans different directions. First, we are conducting a variety of experiments to better assess the current ranking model: we are varying the value of k during query tag expansion, the similarity function used to compute both user' and tags' similarity, and we plan to experiment with a different dataset too (namely, Last.fm). The technique we currently use to populate the user-by-tag matrix, and the tag-by-resource matrix, is rather simplistic (a basic counter of how many times a user has used a tagged, and how many times a tag has been used on a resource). More advanced techniques could be used, which could then lead to more accurate similarity results: for example, including time information, to cater for the most frequently used tags, the most recently used tags, etc.

In terms of model, our plan is to refine the techniques we use to find both similar users and similar tags. We have started analysing the impact of a variety of clustering techniques to identify communities of users; beyond similarity in the tags' usage, there exist other parameters of relevance, including level of activity (to distinguish active users who contribute to the knowledge base, from passive consumers), variety of tags used (unpopular tags may reveal more about a user's interests than popular ones), and so on. In parallel, we are studying more refined techniques to learn relationships between tags. The ultimate goal is to enrich Web 2.0 applications with accurate and robust techniques to give users what they are really looking for.

REFERENCES

[1] A. Agresti, *Analysis of Ordinal Categorical Data*, J.Wiley & Sons, 1984.

[2] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, 'Using Collaborative Filtering to Weave an Information Tapestry', *Communications of the ACM*, **35**, 61–70, (1992).

[3] Scott Golder and Bernardo A. Huberman, 'Usage patterns of collaborative tagging systems', *Journal of Information Science*, **32**(2), 198–208, (2006).

[4] Harry Halpin, Valentin Robu, and Hana Shepherd, 'The complex dynamics of collaborative tagging', in *Proceedings of the 16th International Conference on World Wide Web*, pp. 211–220, New York, NY, USA, (2007). ACM Press.

[5] Y. Hassan-Montero and V.Herrero-Solana, 'Improving tag-clouds as visual information retrieval interfaces', in *Intl. Conference on Multidisciplinary Information Sciences and Technologies*, Merida, Spain, (2006).

[6] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, 'An Algorithmic Framework for Performing Collaborative Filtering', in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230–237, (1999).

[7] Paul Heymann and Hector Garcia-Molina, 'Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems', Technical Report 2006-10, Stanford University, (April 2006).

[8] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina, 'Can Social Bookmarking Improve Web Search?', *Resource Shelf*, (2007).

[9] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, *Information Retrieval in Folksonomies: Search and Ranking*, 411–426, 2006.

[10] William H. Hsu, Joseph Lancaster, Martin S.R. Paradesi, and Tim Weninger, 'Structural Link Analysis from User Profiles and Friends Networks: A Feature Construction Approach', (March 2007).

[11] O. Kaser and D. Lemire, 'Tag-cloud drawing: Algorithms for cloud visualization, tagging and metadata for social information organizatio', in *Intl. Conference on the World Wide Web*, Alberta, Canada, (2007).

[12] Shreeharsh Kelkar, Ajita John, and Doree Seligmann, 'An Activity-based Perspective of Collaborative Tagging', (March 2007).

[13] G. Koutrika, F. A. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina, 'Combating spam in tagging systems', in *Proc.of the 3rd Intl. Workshop on Adversarial Information Retrieval on the Web*, pp. 57–64, New York, NY, USA, (2007).

[14] N. Lathia, S. Hailes, and L. Capra, 'The effect of correlation coefficients on communities of recommenders', in *Proceedings of 23rd Annual ACM Symposium on Applied Computing*, (2008).

[15] K. Faith Lawrence and M. C. Schraefel, 'Bringing Communities to the Semantic Web and the Semantic Web to Communities', in *Proceedings of the 15th International Conference on World Wide Web*, (2006).

[16] X. Li, L. Guo, and Y. E. Zhao, 'Tag-based Social Interest Discovery', in *Proc. of the 17th Intl. World Wide Web Conference*, (2008).

[17] Reyn Nakamoto, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura, 'Tag-based Contextual Collaborative Filtering', in *18th IEICE Data Engineering Workshop*, (2007).

[18] Alexandre Passant, 'Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs', in *Proceedings of International Conference on Weblogs and Social Media*, (2007).

[19] H. Polat and W. Du, 'Privacy-Preserving Collaborative Filtering using Randomized Perturbation Techniques', in *The Third IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, FL, (November 2003).

[20] T. Rattenbury, N. Good, and M. Naaman, 'Towards automatic extraction of event and place semantics from flickr tags', in *Proc.of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 103–110, New York, NY, USA, (2007).

[21] Jianhua Ruan and Weixiong Zhang, 'Identifying network communities with a high resolution', *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, **77**(1), (2008).

[22] S. Sen, S. K. Lam, Al M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, M. F. Harper, and J. Riedl, 'tagging, communities, vocabulary, evolution', in *Proc.of the 20th Conference on Computer Supported Cooperative Work*, pp. 181–190, New York, NY, USA, (2006).

[23] Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme, 'Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms', in *Proceedings of 23rd Annual ACM Symposium on Applied Computing*, pp. 16–20. ACM Press, (2008).

[24] X. Wu, L. Zhang, and Y. Yu, 'Exploring social annotations for the semantic web', in *Proc. of the 15th Intl. Conference on World Wide Web*, pp. 417–426, New York, NY, USA, (2006).

[25] C. Man Au Yeung, N. Gibbins, and N. Shadbolt, 'Mutual Contextualization in Tripartite Graphs of Folksonomies', in *Proc. of the 6th Intl. Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pp. 960–964, (2007).

[26] D. Zhou, E. Manavoglu, J. Li, L. C. Giles, and H. Zha, 'Probabilistic models for discovering e-communities', in *Proc. of the 15th Intl. Conference on World Wide Web*, pp. 173–182, New York, NY, USA, (2006).

Over- and underestimation in different product domains

Nava Tintarev and **Judith Masthoff** [1]

Abstract. This paper investigates the effects of over and underestimation on the perceived Effectiveness (helpfulness) of recommender systems. We consider four different product along two dimensions, degree of objectivity and investment. Overestimation was considered more severely than underestimation with regard to perceived Effectiveness. Overestimation was also considered more severely in high investment domains compared to low investment domains. In addition, we surveyed the effect of different gaps between initial (initial impression) and final ratings (true estimate). We found that for gaps which remained in the negative half of the scale were considered less Effective than gaps which crossed over from good to bad (or from bad to good), and gaps which remained in the positive half of the scale.

1 INTRODUCTION

Explanations of products play an important role in improving the user experience in recommender systems [10, 14, 15]. Among other things, good explanations could help users find what they want and/or persuade them to try or purchase a recommended product. Previous recommender systems with explanation facilities have been evaluated in a number of ways, reviewed and discussed in [19].

In this paper, we expand on the criterion of Effectiveness, or how helpful additional information is with regard to aiding users in making decisions about products. In this section, we define Effectiveness in more detail, describe different types of information skews that may occur in recommendations, and different product domains. In Section 2 we describe our experiment. We conclude with a summary of our results and discuss implications for related research in Section 4.

1.1 Effectiveness

In this paper, we consider the metric of Effectiveness, or decision support, with regard to recommendation information. Good decision support can in part be quantified by the metric suggested by Bilgic and Mooney [2]:

1. **(Rating1)** The user rates the product on the basis of the explanation
2. The user tries the product
3. **(Rating2)** The user re-rates the product

Effectiveness can then be measured by the discrepancy between Steps 1 and 3 (Rating1-Rating2). According to this metric, an Effective explanation is one which minimizes the gap between these two ratings. If an explanation helps users make good decisions, getting more (accurate and balanced) information or trying the product should not change their valuation of the product greatly.

[1] University of Aberdeen, Scotland, U.K., email: n.tintare@abdn.ac.uk

The difference between the two ratings may be positive (overestimation of the product) or negative (underestimation). Overestimation may result in false positives; users trying products they do not end up liking. Particularly in high investment recommendation domains such as holidays, a false positive is likely to result in a large blow to trust in the system. Underestimation may on the other hand lead to false negatives; user missing products they might have appreciated. If a user recognizes an underestimation due to previous knowledge or subsequent exposure, this may lead to a loss of trust as well. Likewise an underestimation may needlessly decrease an e-commerce site's revenue. For example, [16] argued that misperceptions which involve underestimating quality affect long term sales compared to perfect information, or even overestimation.

Our aim is to broaden the definition of Effectiveness suggested by [2]. The metric proposed by [2] does not give an indication of whether over or underestimation is preferable to users, or if this preference might be a domain dependent factor. As a consequence it also does not discuss whether skews of the same type, but with different starting points, are comparable. For example, skews can be represented in terms of the following three gap types: gaps which remain in the negative half of a Likert scale, gaps which cross over from good to bad (or from bad to good), and gaps which remain in the positive half of a scale.

1.2 Related work

1.2.1 Skews in valuations of recommendations

User valuations of recommended items can be skewed (either over- or underestimation) by a number of factors. For example if the quality of the information used to form a recommendation, or if the recommendation accuracy is otherwise compromised, this is likely to lead to poor Effectiveness. Likewise, the nature of the recommended object and presentation of the recommended items are likely to be contributing factors.

Firstly, the recommendation algorithm may be flawed. Other times, skewed recommendations are due to insufficient information, or a bias in data. [4] showed that manipulating a rating prediction can alter the user's valuation of a movie to cause either an over- or underestimation. For example, users rated movies lower than their initial rating when they saw a lower prediction for the movie, and vice versa. The study also suggests that users can be influenced to change their rating of a movie from negative to positive. [4] does not discuss whether over- or underestimation is considered more severely by users, but did find that users' valuations of movies changed more for lower predictions (underestimation) than for inflated predictions (overestimation). Also in the movie domain, [13] found that using the difference between the predicted rating (by similar users) for a given user and item, and the actual rating of the user for this item, could be used to increase recommendation accuracy. They considered the sign

of the error, and used this measure to define a prediction range which they used to improve recommendation accuracy. On average, errors based on underestimation were smaller than for overestimation, but were as such least effective for increasing accuracy.

Secondly, presentational choices for recommendations may skew a user's valuation of an item. For example, it has been argued that order of presentation [6], and the use of images [12] can have a persuasive effect on users. [6] found that users click more on highly ranked links, while [12] found that domain credible images could be used to increase credibility of websites.

Thirdly, assuming good algorithmic accuracy, additional information such as explanations can be used to either aid or hinder decision support. An explanation may contain both positive and negative information, and in that sense may have a polarity in a similar way to numerical ratings of a product. Modifying the polarity of an explanation is likely to lead to a similar skew to the one found by [4]. For example, in the study by Herlocker et al [5] participants were most likely to see a movie if they saw an explanation interface consisting of a bar chart of how similar users had rated the movie. This bar chart had one bar for "good", a second for "ok" and a third for "bad" ratings. Bilgic and Mooney [2] later showed that using this type of histogram causes users to overestimate their valuation for items when the dataset is skewed toward positive ratings.

Online reviews are another form of additional information and might sway user valuation of an item. Previous research considering the properties of helpful reviews has found a positive bias in the movie domain [18] as well as for cameras and mobile phones [7].

In our experiment, we study the effects of over- and underestimation due to additional information such as explanations. However, since the skew in the valuation of recommendations can be caused by any of these factors (e.g. limited algorithm, skewed or limited data, presentation, and additional information) the effects on evaluations of skews may be relevant to these causes as well.

1.2.2 Domains

In economics, there has been a great deal of debate about classification of products into different categories. [16] uses the distinction between experience goods, or goods that consumers learn about through experience, and "search goods" which they do not need to learn about through direct experience. Similarly, [3] distinguishes between sensory products and non-sensory products. We propose an interpretation of these categories which distinguishes between products which are easy to evaluate objectively and those which commonly require an experiential and subjective judgment.

Another common categorization in economics involves investment or cost. Often this is a complex construct. For example, [11] discusses *perceived* price in terms of the dimensions of risk and effort. This construct of risk includes financial risk but also psychological, physical, functional and social risk. The construct of effort considers purchase price, but also time that the purchase takes. [3] also discuss perceived price in terms of non-monetary effort and degree of involvement. [8] narrows down the definition of cost to the objective measure of the purchase price of an item. For simplicity, we will also use a definition of investment which only considers purchase price.

2 OVER- AND UNDERESTIMATION

In this experiment, we wanted to find out whether users are more accepting of underestimation or overestimation in general. We also investigated how the nature of a product domain can mitigate, or conversely, exacerbate faulty information.

2.1 Materials

The experiment was conducted using two questionnaires (one for overestimation and one for underestimation). The questionnaires considered four domains distributed over the dimensions of investment (low vs. high) and valuation type (objective vs. subjective) as shown in Table 1.

We defined investment in terms of price. By this definition cameras and holidays are high investment domains. Relatively to these domains, light bulbs and movies can be considered low investment domains.

We considered cameras and light bulbs as objective domains, and movies and holidays as subjective. Our definition of this dimension is based on the premise that while some domains are highly subjective, it is easier to give a quantitative judgment in others. For example, users might be able to reach a consensus as to what properties are important in a camera, and what generally constitutes good quality, while this might be harder for a movie. It might be easier to define good image resolution in a camera than define good acting in a movie. Note also that our choice of definition for this dimension does not preclude that different product features (such as resolution and shutter speed, or actors and director) may vary in terms of importance to different users in all four product domains.

Table 1. Choice of domains

	Low investment	**High investment**
Objective	Light bulb	Camera
Subjective	Movie	Holiday

2.2 Hypotheses

We expect that users will be more lenient toward underestimation, and consider it more helpful than overestimation in general. This hypothesis is based on the assumption that users would like to save money, and are wary of persuasion in commercial systems. Users may prefer being recommended only great items (and miss decent items) to buying more, and being recommended items that they will not like.

It also seems probable that users will have higher demands on accuracy in high investment domains such as movies and holidays. Likewise, users may respond more leniently to skews in subjective compared to objective domains as these are harder to gage.

We also consider that it is possible that the strength of an over- or underestimation may also depend on the starting point on a scale. Therefore, we also consider the effects of over- and estimations of the same magnitude, but with different starting points. For example, what is the effect of underestimation on perceived Effectiveness if a user's valuation of an item changes from negative to ok, and how does this compare to a change from ok to great? A user may consider an explanation least helpful when it causes them to perform an action they would not have performed if they had been given accurate information, e.g. when it changes their valuation of a product from good to bad, or from bad to good. Our hypotheses are thus:

- **H1:** Users will perceive overestimations as less Effective than underestimation.

- **H2:** Users will perceive skews as less Effective in high investment domains compared to low investment domains.
- **H3:** Users will perceive skews as less Effective in objective compared to subjective domains.
- **H4:** Users will perceive cross-over gaps which cross the line from good to bad and vice-versa as less Effective compared to other gap types.

2.3 Participants

Twenty participants (7 female, 12 male, one unknown) were recruited at the University of Aberdeen. They were all postgraduates or researchers in Computing Science. The average age was 31.95 (range 20-62).

2.4 Design

We used a mixed-design, with product domain as a within subject factor, and over- vs. underestimation as a between subject factor. Participants were assigned to one of two conditions. In the first, participants were given a questionnaire with overestimation scenarios, in the second underestimation scenarios.

In the underestimation condition participants saw *Paragraph A*:

Paragraph A: *"Assume you are on a website looking for a particular product to buy (such as a camera, holiday, light bulb, movie). Based on the information given, you form an opinion of the product, and decide* ***not*** *to buy it and to spend the money on something else. Later you talk to a friend who used the product, and your opinion changes."*

The user decides not to buy a product and spends the money on something else. This is to ensure that the choice (not to purchase) is perceived to be irreversible by the participants. Only later do they discover that the product was not as bad as they first thought.

For overestimation we considered situations in which the user initially rated the product highly, but then found the true value of the product lower after buying and trying it. *Paragraph A* is replaced with *Paragraph B* below:

Paragraph B: *"Assume you are on a website looking for a particular product to buy (such as a camera, holiday, light bulb, movie). Based on the information given, you form an opinion of the product, and decide to buy it. After using the product, your opinion changes."*

In both cases participants were asked to consider that they were viewing a new website for each scenario even for similar products. All participants considered products in all four product domains (cameras, light bulbs, movies and holidays) in randomized order. Each participant was given scenarios in which their valuation of the product changed by a magnitude of 2 on a scale from 1 (bad) to 5 (good). We varied the starting point for the initial valuation. The rating of the product can be either:

1. Positive, i.e. staying on the positive side ($3 \leftrightarrow 5$)
2. Negative, i.e. staying on the negative side ($1 \leftrightarrow 3$)
3. Cross-over, i.e. changing polarity ($2 \leftrightarrow 4$)

The order of the three starting points (positive, negative and cross-over) was randomized. The orders of the before and after values were reversed between over- and underestimation, e.g. $3 \rightarrow 5$ (underestimation) became $5 \rightarrow 3$ (overestimation). Given three different starting points and four product domains, each participant considered twelve scenarios.

For each of the twelve scenarios, participants rated how helpful they found the (presumed) information given on the website on a seven point Likert scale (1 = very bad, 7 = very good): *"How do you rate the information on this website given this experience?"*. While this *perceived* Effectiveness differs from true Effectiveness, it also differs from Persuasion. Persuasive information would give the user an initial impression (either positive or negative), but fails to consider the way the user finally rates the product once they try it. In this study the final rating is assumed to be known and true. Step 2 of the proposed metric (see Section 1.1), where the user would normally receive information about the product, is assumed to be a black box.

2.5 Results

2.5.1 Which is better?

Firstly we inquire if over- or underestimation is considered generally more helpful by users. Similarly we want to know just how harmful these skews are considered by users. As can be expected, in Table 2 we see that both over- and underestimation are considered unhelpful. Since it is arguable that the values on a Likert scale may not be equal in distance, we performed a Mann-Whitney non-parametric test which rendered a significant result ($p < 0.01$). Overestimation is considered to be less Effective than underestimation: H1 is confirmed.

Table 2. Perceived helpfulness (on a scale from 1 to 7) for over- and underestimation

	Mean	**StD**
Overestimation	2.59	1.065
Underestimation	3.08	1.212

2.5.2 Does the domain matter?

In Table 3 we offer an overview of perceived helpfulness, for all four domains.

Table 3. Mean (and Std) of perceived helpfulness (on a scale from 1 to 7) for the four domains

	Underestimation	**Overestimation**
Camera	2.87 (1.252)	2.37 (0.964)
Light bulb	3.15 (1.231)	2.63 (1.066)
Movie	3.30 (1.236)	3.00 (1.145)
Holiday	3.00 (1.145)	2.37 (0.999)

Low vs. High Investment Table 4 summarizes the perceived investment in low (light bulbs and movies) and high (cameras and holidays) investment domains. The perceived helpfulness was lower for high investment than for low investment domains (Mann-Whitney test, $p < 0.05$). A separate analysis for over- and underestimation shows a significant effect (Mann-Whitney test, $p < 0.05$ with Bonferroni correction) for overestimation, but not for underestimation. We also see that underestimation is considered as less Effective in high investment compared to low investment domains, but this trend is not statistically significant. It seems as if users are more sensitive to skews in high investment domains, but in particular with regard to overestimation. H2 is confirmed.

Table 4. Mean (and StD) of perceived helpfulness for low vs. high investment domains

	Underestimation	Overestimation
High	2.93 (1.191)	2.37 (0.974)
Low	3.23 (1.225)	2.82 (1.112)

Objective vs. Subjective In Table 5 we see that both over and underestimation are considered less Effective in objective compared to subjective domains, but the trend is not statistically significant. This hints that correct estimates may be more important in objective domains than subjective, regardless of direction of skew. User comments also confirm that some users are more forgiving of misleading information in subjective domains than objective: *"a wrong suggestion about 'subjective' evaluations of products (such as for movie or holidays) should not determine a severe bad judgment of the website."*, *"whether I like a movie (or holiday) is very subjective, and I would not blame my liking a movie less on the quality 1st description"*. The effect is however not sufficiently strong, and H3 is not confirmed.

Table 5. Mean (and StD) of perceived helpfulness for objective vs. subjective domains

	Underestimation	Overestimation
Objective	3.00 (1.239)	2.50 (1.017)
Subjective	3.15 (1.191)	2.68 (1.112)

2.5.3 Does the type of gap matter?

Table 6. Mean (and StD) of perceived helpfulness for different gap types

	Underestimation	Overestimation
Positive	3.90 (0.940)	3.02 (1.084)
Cross-over	3.03 (0.140)	2.68 (0.944)
Negative	2.31 (1.239)	2.05 (0.944)

We hypothesized that gaps which cross over between the positive and negative ends of the scale (cross-over gaps) are less helpful than the two other gap types. We found a significant effect of gap type on perceived Effectiveness in a Kruskal-Wallis test ($p < 0.05$). However, in a Mann-Whitney test we found no significant difference between cross-over gaps and the two other gap types combined. H4 is not confirmed.

Investigating the difference between gap types further, in Table 6 we see that participants found gaps on the negative end of the scale ($1 \leftrightarrow 3$) less helpful than gaps on the positive end ($3 \leftrightarrow 5$), and gaps which cross over between the positive and negative ends of the scale ($2 \leftrightarrow 4$), for data using both over and underestimation. Cross-gaps in turn were considered less helpful than positive gaps. Three Mann-Whitney tests comparing the three gap types pairwise were all found to be statistically significant ($p < 0.05$ with Bonferroni correction). Apparently, negative gaps damage perceived helpfulness the most out of the three gap types rather than cross-over gaps.

A similar series of Mann-Whitney tests were run for over and underestimation separately. All tests returned significant results (p < 0.05, with Bonferroni correction), except for the difference between positive and cross-over gaps for overestimation. That is, the difference in perceived Effectiveness between positive and cross-over gaps for overestimation is negligible.

2.6 Discussion

Our finding of user preference for underestimation compared to overestimation is in line with persuasive theory regarding expectancy violations and attitude change [17]. An audience's initial expectations will affect how persuasive they find a message. In a persuasive context, if expectations of what a source will say are disconfirmed, the message source can be judged to be less biased and more persuasive. For example, if a political candidate is expected to take a certain position with regard to an issue, but ends up advocating another position, their credibility rises.

Since it is a likely assumption that users expect a commercial recommender system to overestimate the value of an item, underestimation disconfirms this expectation and might cause users to find a recommender system less biased and more trustworthy. Two users stated expectations on an emphasis on high ratings in qualitative comments: *"I would expect the web to present items at their best and sometimes with some exaggeration."*, *"I expect there to be hype about a movie and to have to read between the lines to form a judgment for myself."*

The effect of gap type was surprising, we also were surprised to find that negative gaps were considered least helpful, and positive gaps most helpful, for *both* over and underestimation. This may reflect the way users distribute and assign ratings. The polar ratings of 1's and 5's are more uncommon and differently distributed from the other ratings, i.e. the 'distance' between 2 and 3 may be perceived as smaller than the distance between 2 and 1. So a user is much less likely to buy an item rated 1 rather than 2. Likewise, the probability of a user trying an item increases more between 4 and 5 than it does between 3 and 4. The lack of significant results for overestimation might be attributed to users' general expectation of overestimation in commercial recommender systems.

User comments also revealed some other interesting views on product categories. Two users left comments where they differentiate between holidays and the other products:
"Things like 'Holidays' matter more compared to goods, because holiday is a destination could be once in a life time thing.", *"A holiday is an experience of value that cannot be replaced or compensated for, knowledge should be accurate."*. One user found it difficult to imagine using a recommender system to buy light bulbs: *"I can't imagine going on to a web site to look for information on a light bulb!"*.

3 Reflections on the experimental setup

When considering the design of our experiment, two criticisms can be raised. In this section, we discuss what these criticisms are, and why we decided to perform the experiment in this particular way.

3.1 Why the wording for underestimation differs

In the scenario for overestimation the user changes their value judgment by experiencing the product directly. In contrast, in the underestimation scenario, the user changes their value judgment based on comments from a friend who experienced the product. So, why did we not let the user "experience" the product directly in the latter case, as this would have made the conditions more comparable? As the user did not buy the product, it was hard to devise a plausible story of how they ended up experiencing it after all. If somebody else bought it for them as a gift, the user is not likely to regret missing the item, and thus will not harbor feelings of resentment over poor information to the same degree. Experiencing the item by borrowing it from a friend is not possible for all domains (e.g. holidays).

3.2 Why the experiment is indirect

Instead of participants really experiencing the products, we only told them about their experience. What participants think they would do in such a situation may diverge from what they really would do [1]. We were however working on the basis of these assumptions:

- *Gap size matters.* Participants' perceived Effectiveness will depend on the size of the discrepancy between their first impression and their valuation after experiencing the item.
- *Gap position matters.* The influence of a skew will depend on the gap's position. For example, an under-estimation from 1 (first rating) to 3 (final valuation) may have a different effect than one from 3 to 5. Evidence for this was found in our experiment.

Given these assumptions, for a fair comparison between domains (H2, H3) we need to control for gap size and position. Practically, this would mean that participant's valuations (before and after) need to similarly distributed for all products. This would be very hard (if not impossible) to control rigorously. Even making the experiment a little more realistic, by giving participants particular information to form a first opinion, and then more information to form a final valuation, would be hard to control. Attempts in our earlier work to construct item descriptions with predictable ratings for all participants failed [9].

For a fair comparison between over- and underestimation (H1), we also need the gap size and position to be the same [2]. Suppose we knew that people *on average* like a particular item, and disliked another item. This may be hard to obtain in certain product domains, or limit us to a small subset of items where people converge on valuation. This is also likely to require a separate study to decide on suitable items. The estimated valuation allows us to know, on average, the real valuation (and in analysis, we would need to remove all subjects whose valuation differed from this average). We would still have to make the explanations such that they induce the right initial rating (namely the valuation for the liked item in the disliked item's case, and the other way around). Given that we also wanted to study gap types (H4), we would need multiple of these item pairs plus explanations per domain.

This does not mean that we will not do more direct experiments in the future. It is just that given the factors we wanted to investigate here, there were very clear benefits in doing an indirect experiment.

4 Conclusions

H1 is confirmed: overestimation is considered less helpful by users than underestimation. H2 is partially confirmed: overestimation is considered less helpful in high investment domains than in low investment domains. Underestimation in high investment domains is not considered significantly less helpful, even if there is a trend in this direction. The lack of significant result may be due to underestimation having a stronger effect on perceived Effectiveness. H3 is not confirmed, only a trend suggests that some users may be more critical in objective than subjective domains. H4 is disconfirmed: cross-gaps are not considered the least helpful by users, negative gaps are, for both over- and underestimation. For overestimation, positive gaps are not considered less helpful than cross-over gaps.

As mentioned in Section 1.2.1, recommendations can be skewed for a variety of reasons. The results of this study would be relevant for algorithmic correction as well as studies comparing different presentational interfaces. Understanding the role of factors such as gap type, domain type and over and underestimation will help better control for these factors when optimizing a recommender system for Effectiveness.

In light of our results we suggest an enhancement to the Effectiveness metric proposed by [2] and described in Section 1.1. We propose fine tuning this measure of Effectiveness by weighting it according to gap type, over/underestimation and degree of investment.

[2] We consider the gap '1 to 3' to be comparable to the gap '3 to 1' w.r.t. to position

REFERENCES

[1] Icek Ajzen and Martin Fishbein, 'Attitude-behavior relations: A theoretical analysis and review of empirical research.', *Psychological Bulletin*, **84(5)**, 888–918, (1977).

[2] Mustafa Bilgic and Raymond J. Mooney, 'Explaining recommendations: Satisfaction vs. promotion', in *Proceedings of the Beyond Personalization Workshop in association with IUI*, pp. 13–18, (2005).

[3] Yooncheong Cho, Il Im, and Jerry Fjermestad Starr Roxanne Hiltz, 'The impact of product category on customer dissatisfaction in cyberspace', *Business Process Managment Journal*, **9 (5)**, 635–651, (2003).

[4] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan., and John Riedl, 'Is seeing believing?: how recommender system interfaces affect users' opinions', in *Proceedings of the SIGCHI conference on Human factors in computing systems*, volume 1 of *Recommender systems and social computing*, pp. 585–592, (2003).

[5] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl, 'Explaining collaborative filtering recommendations', in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 241–250, (2000).

[6] Thorsten Joachims, Laura Granka, and Bing Pan, 'Accurately interpreting clickthrough data as implicit feedback', in *ACM SIGIR conference on Research and development in information retrieval*, pp. 154–161, (2005).

[7] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti, 'Automatically assessing review helpfulness', in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 423–430, (2006).

[8] David N. Laband, 'An objective measure of search versus experience goods', *Economic Inquiry*, **29 (3)**, 497–509, (1991).

[9] Judith Masthoff, 'Group modeling: Selecting a sequence of television items to suit a group of viewers', *User Modeling and User Adapted Interaction*, **14**, 37–85, (2004).

[10] David Mcsherry, 'Explanation in recommender systems', *Artificial Intelligence Review*, **24(2)**, 179 – 197, (2005).

[11] Patrick E. Murphy and Ben M. Enis, 'Classifying products strategically', *Journal of Marketing*, **50**, 24–42, (1986).

[12] Hien Nguyen and Judith Masthoff, 'Using digital images to enhance the credibility of information', in *Persuasive Technology symposium in association with the Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB)*, pp. 1–8, (2008).

[13] John O'Donovan and Barry Smyth, 'Eliciting trust values from recommendation errors', in *International Journal of Artificial Intelligence Tools (IJAIT)*, (2006).

[14] Pearl Pu and Li Chen, 'Trust building with explanation interfaces', in *International conference on Intelligent user interfaces*, Recommendations I, pp. 93–100, (2006).

[15] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth, 'Dynamic critiquing', in *European Conference on Case-Based Reasoning (ECCBR)*, volume 3155 of *Lecture Notes in Computer Science*, pp. 763–777, (2004).

[16] Carl Shapiro, 'Optimal pricing of experience goods', *The Bell Journal of Economics*, **14 (2)**, 497–507, (1983).

[17] James B. Stiff, *Persuasive Communication*, chapter 5, 94–98, Guilford Press, 1994.

[18] Nava Tintarev, 'Explanations of recommendations', in *ACM Recommender Systems*, pp. 203–206, (2007).

[19] Nava Tintarev and Judith Masthoff, 'A survey of explanations in recommender systems', in *WPRSIUI associated with ICDE'07*, pp. 801–810, (2007).

A Example questionnaire - underestimation

Experiment on product information

Age: _ _ _ _ Gender: M/F (please circle the one that applies)

All data gathered in this study will be treated confidentially, anonymized, and will only be used for the purpose of the research.

Assume you are on a website looking for a particular product to buy (such as a camera, holiday, light bulb, movie). Based on the information given, you form an opinion of the product, and decide not to buy it and to spend the money on something else. Later you talk to a friend who used the product, and your opinion changes.

Consider the following scenarios, and indicate how your experience in each case effects your perception of that particular website. Note: each scenario is about a different website, even for similar products.

Product	**Your opinion of the product based on info on the website**(1 to 5 scale with 1 being really poor and 5 really good)	**Your opinion of the product after talking to your friend** (1 to 5 scale with 1 being really poor and 5 really good)	**How do you rate the information on this website given this experience?**						
			Very unhelpful						Very helpful
Camera	3	5	1	2	3	4	5	6	7
Holiday	1	3	1	2	3	4	5	6	7
Light bulb	2	4	1	2	3	4	5	6	7
Movie	1	3	1	2	3	4	5	6	7
Camera	2	4	1	2	3	4	5	6	7
Holiday	3	5	1	2	3	4	5	6	7
Light bulb	1	3	1	2	3	4	5	6	7
Movie	3	5	1	2	3	4	5	6	7
Camera	1	3	1	2	3	4	5	6	7
Holiday	2	4	1	2	3	4	5	6	7
Light bulb	3	5	1	2	3	4	5	6	7
Movie	2	4	1	2	3	4	5	6	7

Would you like to explain your answers? Please do this here:

Thank you for your participation! If you would like to know more about this study, or receive a summary of the results please contact me at n.tintare@abdn.ac.uk

Harnessing Facebook for the Evaluation of Recommender Systems based on Physical Copresence

Alexandre de Spindler and **Stefania Leone** and **Michael Grossniklaus** and **Moira C. Norrie** [1]

Abstract. Various mobile social applications have proposed the use of ad-hoc network connectivity as a means of detecting user encounters and shared social contexts. These applications range from simple opportunistic information sharing to techniques for collaborative filtering in mobile settings. However, it can be difficult and costly to test the underlying assumption that repeated physical copresence can be used as a measure of user similarity. We have therefore developed a framework that allows existing online social platforms such as Facebook to be coupled with simple, standard mobile applications in order to test such hypotheses. The central idea is to map the physical copresence of users to connections in virtual social networks and then exploit the rich support for developing pluggable applications to measure user similarity within these networks.

1 Introduction

Several mobile social applications have proposed the use of ad-hoc network connectivity as a basis for detecting user encounters. Physical copresence provides a means of not only sharing information opportunistically between mobile devices, but also determining shared social contexts which can be an indicator of some kind of similarity between users. Hence, the communication layer can be used to perform some of the filtering of information normally associated with recommender systems.

Some of these applications focus on copresence at specific events in order to exchange information closely related to these events, while others use repeated copresence in public spaces such as railway stations to build user communities and share general information such as jokes or images. We have taken the idea one step further and adapted collaborative filtering to mobile settings by using physical copresence as the basis for measuring user similarity [3].

One of the major challenges in such applications is to validate the underlying assumption that physical copresence can be used as a measure of user similarity. While it might seem obvious that copresence at an event such as a music concert can be taken as an indicator of similar tastes in music, questions arise as to whether this could be extended to other categories such as books and films. Also should we consider only events such as concerts as a basis for filtering recommendations, or could we extend it to other types of locations such as bars, restaurants, work places, public spaces and even transport systems?

Unfortunately, it is very complex to test such assumptions in real world settings. Since recommendations are based on chance encounters, a meaningful experiment would either require the set of locations or events to be restricted artificially or the recruitment of a vast number of participants. Our goal was to investigate ways in which existing social networking sites such as Facebook could support such experiments by using data collected on physical copresence to create connections in virtual networks. The rich support for developing pluggable applications offered by sites such as Facebook could then be used to measure user similarity within these networks. Similarities can be measured in terms of ratings of specific items such as films and travel destinations to less specific personality properties such as attitudes and priorities in life.

To support such experimentation, we have developed a framework that bridges physical and virtual social networks by allowing users to easily connect to other users encountered in the real world through Facebook. The framework consists of two main components. The first allows the physical copresence of participants to be tracked in a transparent and unobtrusive manner. The second component is a general copresence Facebook application that we developed to enable users to view and connect to other users who they encountered in the real world. Researchers investigating the use of copresence in recommender systems can then develop their own Facebook applications to test specific hypotheses about different forms of user similarity.

In this paper, we present our framework and discuss how it could be used as an experimental platform for recommender systems that use physical copresence as a basis for filtering information. We begin in Section 2 with a discussion of related work and background information. In Section 3, we introduce a sample scenario that exemplifies how users interact with our infrastructure. Section 4 discusses the design of the framework and gives detailed information about the implementation of the various components involved. Section 5 then describes how the framework could be used to test specific ways in which copresence can be used as a measure for user similarity. Concluding remarks are given in Section 6.

2 Related Work

Several projects have investigated ways of detecting physical copresence in mobile environments and the analysis of copresence data as well as the usefulness of physical copresence for community detection or information sharing. Different goals of analysing the data collected have been pursued, ranging from identifying contact patterns as a basis for designing new data forwarding algorithms [17] to the identification of users [18], community detection [11] and serendipitous matchmaking systems for the benefit of the users [5]. We outline a few of these projects below to indicate the range of applications that rely on copresence data.

Freyne et al. [8] propose a system for message delivery in a mobile environment. Collocation is one possible trigger for delivery and

[1] ETH Zurich, Switzerland, email: despindler@inf.ethz.ch

therefore the system features proximity detection. However, the copresence data is not used to identify relationships among users and therefore the data is removed from the system as soon as it serves its triggering purpose.

Serendipity [6] is a project conducted at MIT where Bluetooth-enabled mobile phones were handed out to 100 students who were asked to carry them around for 9 months. Apart from logging usage patterns of phone applications, the phone was set to scan its Bluetooth environment every 5 minutes and record any other Bluetooth-enabled devices discovered. A number of similar studies followed, all pursuing the goal of collecting collocation data in order to recognise contact patterns [1, 14].

The notion of a *familiar stranger* denoting unknown people one frequently meets has been picked up by projects trying to recognise the familiarity of people based on copresence data [16]. Furthermore, such data has been used to provide users with an awareness of the familiarity of people currently in their vicinity, i.e. to recognise a user's social situation [15].

Lawrence et al. [12] developed AIDE, a system which recognises so called copresence communities, groups that are co-located on a regular basis. As an example application, AIDE uses these communities to disseminate information which is supposed to be relevant to all of its members. There have been no attempts to measure the degree of common interests within a copresence community.

The TRACE [2] project adopted a wizard-of-oz style approach to try and study the relationship between copresence and user similarity. Visitors attending a particular event such as a karaoke evening were handed out business cards indicating a web site address where they would later be able to view photos and other information about the event. Users of the web site could also open discussions with other visitors to the web site and, in this way, the developers of TRACE were able to connect users who had been copresent and monitor their interactions. Through this simple experiment, they were able to show that there was some evidence supporting the assumption that copresence can be an indicator of user similarity. However, the study was relatively small scale and one issue was the fact that users who interacted via the event-related web site would often shift their communication to other channels such as regular email at which point the researchers lost track of their communication.

In our own previous research, we have adapted collaborative filtering (CF) to mobile settings by assuming that the degree of copresence between users is a measure of similarity [3]. Our algorithm uses connectivity in mobile ad-hoc networks and opportunistic information sharing to exchange ratings and recommendations among users. The approach avoids the need for a central server and heavy computation of user similarity measures. We have also shown that the algorithm is computationally equivalent to traditional user-based CF algorithms. However, the quality of the recommendations is dependent on the underlying assumption that user copresence is a good measure of user similarity.

Our initial efforts to test this assumption were based on questionnaires designed to assess the similarity of people interviewed at different locations of the Edinburgh festivals [4]. Each year in the month of August, several festivals and large events take place in Edinburgh, including the Fringe, a book festival, a film festival and a military tattoo. The festivals therefore cover several categories of events such as music, drama, dance, comedy, film screenings, book readings and interviews with artists and authors. We wanted to assess whether copresence at particular venues would not only be a good indicator of similarity of tastes with respect to the category of events held at that venue, but also whether it could be extended to other categories.

Although the results obtained support the hypothesis that people who attend the same events tend to share similar tastes and interests, such questionnaires tend to be basic and lack generality. Since visitors were interviewed as they entered or left a venue, the number of questions had to be kept small. Further, due to the nature of a questionnaire-based interview, the kind of possible similarities is determined beforehand during the design of the questionnaire and cannot be further explored interactively after evaluating initial responses. Finally, such studies were unable to track visitors attending multiple events, and it was therefore not possible to investigate if and how the degree of similarity depended on the number of encounters.

Apart from studies based on questionnaires, such a hypotheses could also be tested by deploying an application to the general public or developing a simulation of the algorithm. However both of these approaches also have severe drawbacks. Developing and deploying such an application may require a high investment and will also be high risk if there is no prior evidence backing the underlying assumptions that copresence is a good indicator of user similarity. It is also difficult within academia to carry out large-scale experiments due to the costs involved. Effective simulation models require the integration of a human behaviour model and not only a model of the system operation. Although such models exist, it is difficult to find appropriate ones where behaviour can be characterised in terms of taste and interest and which are computationally not too intensive.

Given the rise in interest in social networking sites such as Facebook, we decided to investigate ways in which existing online social platforms could be used to test such hypotheses without having to implement and deploy full-scale mobile applications to a large number of users. The main idea is to relate physical copresence of users to connections in virtual social networks. Currently, social platforms such as Facebook offer a rich development environment for pluggable applications. As a result, applications that allow specific forms of user similarities within a network to be measured may be easily developed and deployed. Similarities can be assessed in terms of preferences related to specific categories of items such as films, music or travel destinations or less specific personality properties such as attitudes and priorities in life.

The mapping between real-world and virtual social networks is not a completely novel idea. As part of the Cityware project[2], a Facebook application has been developed to notify users about other Facebook users met in the real world. However, the system is not built on copresence detected soley by the mobile devices carried by users but rather relies on the presence of desktop computers featuring Bluetooth connectivity. These nodes perform the scanning of the physical environment and send the copresence data to a central server. As a result, the system would require the installation of many such nodes in order to work on a larger scale. Our aim therefore was to develop a physical-virtual copresence framework in which the mobile devices perform the scanning themselves in order to detect copresence.

3 Scenario

Before describing our framework in detail, we first use a simple scenario to explain the general operation of the system and the user's view of the application. To participate in our framework, users must install our application to gather collocation events on their mobile device and register with our Facebook copresence application.

Before going to work in the morning, a user might start the application on their mobile device. If, for example, it is a Bluetooth-

[2] http://www.cityware.org.uk

enabled mobile phone, it then would periodically scan the user's environment for other Bluetooth-enabled devices. When the application discovers other Bluetooth-enabled devices in the vicinity, it stores the identifiers of these devices locally, together with the current time and, if available, a representation of the current location.

The user may keep the application running throughout the day so that it records copresence events on the way to work, at work, during lunch and in the evening when they meet friends at a bar, restaurant or movie theatre. All the copresence events gathered throughout the day are stored persistently and can, therefore, be uploaded to a desktop or laptop computer when the user arrives back home in the evening. From this machine, the data is sent to a central server that manages copresence data from all users. This server also maintains a registry that maps device identifiers to Facebook user identifiers.

When the user later logs into the Facebook web site, they will be presented with an overview of all Facebook users that their mobile device was able to discover during the day by means of the collected copresence data and the registry mapping. Using our Facebook copresence application, they can browse their profiles and connect to the other users as well as keeping track of where and when they encountered somebody. Copresence events can also be displayed on a geographical map in a similar way to points of interests in geographical information systems. A web-based interface displays all locations where a particular user was encountered as well as all users met within a particular region in space and time.

Once users have established links to other users through the copresence application, other Facebook applications can exploit these copresence networks to measure user similarity. For example, a researcher who wants to test the hypothesis that users who go to the same bars at the same time like the same films, could develop their own likeness application for films that uses the copresence data to check whether there is a correlation between encounters in specific locations that are known bars with film preferences.

4 Physical-Virtual Copresence Framework

Having presented a scenario from the user's perspective, we now describe our framework. The framework consists of the three main components shown in Figure 1.

A mobile component is used for gathering collocation data. It consists of an application implemented with Java ME[3], a widespread development platform and runtime environment for mobile applications. The application runs on mobile devices and periodically scans its physical environment for other mobile devices. When a device is discovered, its unique identifier is stored along with the time of the discovery and the location of the scanning device, if that information is available. Note that devices do not need to have the scanning application installed in order to be discoverable. Each detection of a mobile device constitutes a so-called copresence event. The set of copresence events is later exported to a file and uploaded to a central database that processes and manages copresence events from all users of the framework.

Various technologies can be used for automated physical copresense detection, such as Bluetooth [6, 1], wireless hotspot subscriptions [9, 13] or radio frequency [10]. For our copresence framework, we also adopted the approach of using the scanning range of a Bluetooth transceiver to detect physical copresence. Nowadays, most mobile phones have the required hardware already built in and are easily configured to periodically scan their environment as users carry them around. Also, the issue of assigning unique identifiers to mobile users is easily resolved by relying on the MAC address of the Bluetooth receiver. Due to the personal nature of mobile devices, in most cases there exists a one-to-one relationship between the users and their device. Finally, the scanning of a device's Bluetooth environment is simple to implement and, therefore, we do not provide more details here.

To detect the location, we are currently using GPS, but are also experimenting with other location technologies such as GSM cell tracking and wireless hotspots recognition. An issue that is frequently raised when using GSM cell identifiers for determining a location is the fact that the geographic position of GSM cell antennae is not readily available. There seems to be a growing interest for position data of GSM cells as well as wireless hotspots such as used by the Google maps application for the iPhone. Therefore, the number of available resources supporting this kind of location technique can be expected to increase. Our application is designed to cope with the fact that the location tracking module may change in the future. Currently, the simplest way to track positions from a developer's point of view is to use a built-in GPS receiver as it already exists in some mobile phones. The GPS receiver may also be an external device connected to the mobile phone using Bluetooth.

A central server is the heart of the system and maintains the data gathered by the users as well as a registry mapping device identifiers to virtual social network user identifiers. Users must register their mobile device with their social platform identifier. This is a requirement for other users to be able to recognise a physically encountered person as the corresponding social platform user. This server application is implemented using Java and db4o[4] to persistently store the mapping of social network identifiers to physical device identifiers as well as all copresence events. Note that the registry mapping device identifiers to virtual social network user identifiers is also available to the mobile device which can therefore inform its owner about the physical copresence of social platform contacts.

The third component is a Facebook application which notifies Facebook users about other users they have met in the real world whenever they log into the web site. Based on these notifications, users can manage their physical encounters within the virtual social network as we will describe next. This facebook application is implemented using state-of-the-art web technologies such as Java Servlet technology, XSLT and the Facebook REST [7] API[5] for accessing social platform data.

On the one hand, this application can be used to browse a list of physically encountered users as shown in Figure 2. The user may connect to any of those users, i.e. request to build a virtual relationship, or browse their profiles if they are already connected. These connections form the basis for comparing their taste and interests, either using profile data or custom Facebook applications as will be described in the next section. On the other hand, collocation events can be displayed on a geographical map allowing the users to keep track of where and when particular users were encountered. The map can be set to display all users met as depicted in Figure 3, all locations where a particular user was met or all users met within a given region in space or time.

5 Use of the Framework

Our framework allows collocation data to be collected by users and this data to be stored on a central server. We now go on to describe

[3] http://java.sun.com/javame

[4] http://www.db4o.com

[5] http://developers.facebook.com/documentation.php

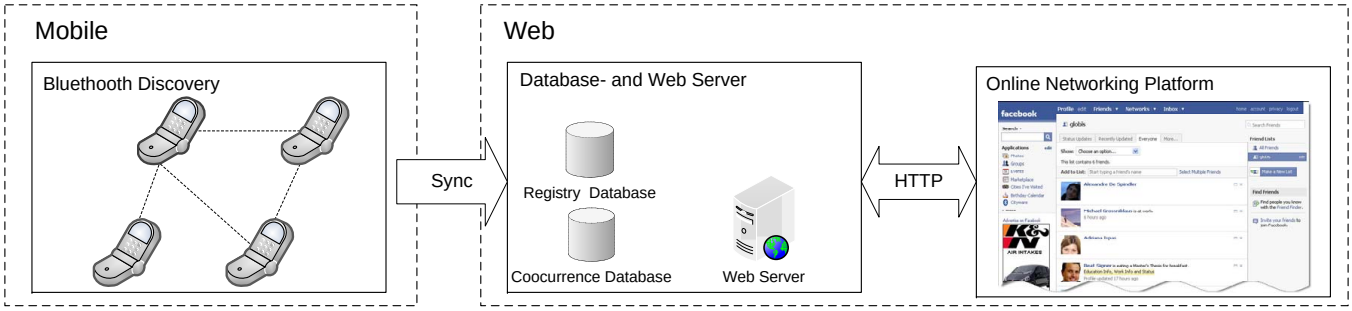


Figure 1. Framework Architecture

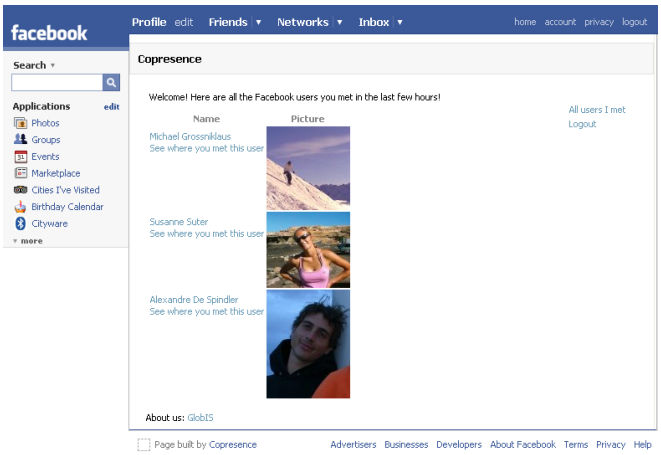


Figure 2. Overview of users met

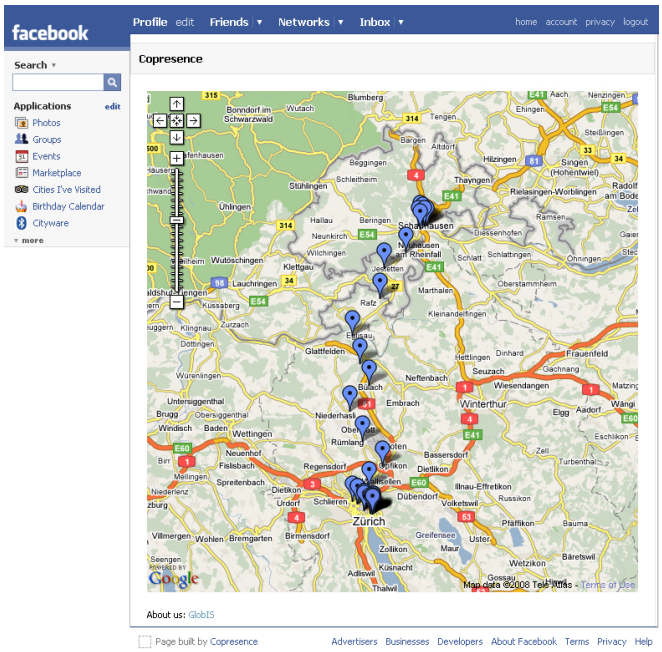


Figure 3. Map showing locations where users were met

how this data can be used to investigate the relationship between physical copresence and user similarity. For any user, we can build a graph representing their social network of real-world encounters. In this graph, users are represented by nodes which are connected by an edge if they have been in each other's vicinity. The edges are weighted with a value indicating the number of times that the users have been copresent.

In order to validate a hypothesis about user similarity measured by means of copresence, a relationship between the values of the edge weights and the actual similarity of the connected users must be shown to be significant. Having the Facebook identifiers of these users gives us the possibility of using that platform to validate such hypotheses. Since all collocation data gathered by our infrastructure is stored persistently, approaches to assess user similarities can be designed and applied at any time.

Additionally, our system allows sequences of user encounters to be analysed. For example, a user might be first met on a bus to a train station. Then the user could also be present on the train to another city. In that other city another user might be met repeatedly on the way from the train station to the office. If such repeated encounters on a user path are of relevance to a particular mobile social software, our system can be used to test its proper functioning.

The first step of any analysis within our framework consists in deciding on user characteristics determining the similarity. Depending on the application domain of a particular recommender system, such characteristics range from standard demographic data such as age, gender and education to domain-specific user taste and interests. An important part of this step is consideration about how users provide their data. The assessment of user characteristics is nothing new to Facebook and applications measuring user similarities are very popular. *Flixster* is a Facebook application with which users rate a predefined set of movies in order to compare their taste with that of their friends. *Likeness* is an application allowing users to be compared based on a wide variety of characteristics. Users are presented ten statements about a topic of their choice, such as top things to do on a day off or ten reasons to quit a job. They are asked to rank these statements according to their personal preferences and the resulting

sequence is used to compare them with those friends that have ranked the same statements.

As can be seen from these two examples, it is important that users can give their answers with as little effort as possible. For example, it is much easier for a user to rate ten movie items individually as opposed to putting the items in a descending order of preference. However, if users are asked to rate items using pre-defined rating values, the given values must cover the whole range of potential answers. If a movie is to be rated using the rating values "I liked" and "I disliked", a user cannot choose a suitable value in the case that the movie has not been watched. Furthermore, a negative opinion may be the reason why it has not been watched and it is unclear whether users would choose "I disliked" in this case.

Even though the Facebook platform already offers a wide variety of applications that measure user similarity, the data generated by these applications is not publicly available and hence cannot be used by other applications and frameworks such as ours. Therefore, a second step consists of developing a Facebook application for measuring user similarity based on the characteristics chosen in the first step. The idea is that users for whom collocation data is available install this application. The application can then gather data, either by accessing the users profile or by letting the user interact with it. The data gathered is then sent to the central server where it is stored along with the copresence data and available for analysis.

Our experience has shown that the complexity of developing a Facebook application is well within the skills of an experienced application developer. The developer's work consists of three tasks. First, a web application providing the means to assess the user properties determining the similarity must be developed. Facebook does not host any of the applications that are developed by external developers. Thus, the second task consists of deploying the web application on a hosted web server. Finally, the application has to be registered with the Facebook platform. The registration mainly consists of specifying the location of the web server and application. As a result, the application will appear as part of the Facebook platform to the user.

As an example, asume we want to test whether there is a correlation between copresence and movie preferences. Therefore, we could design a questionnaire as shown in Figure 4. The questionnaire contains a list of movies about which users state their opinion. In the simplest case, such a questionnaire may be implemented as a regular HTML page using text and form elements. As usual with HTML forms, a URL must be provided which points to a server-side component handling the submission of the response data. In this example, handling the response simply consists of making the data persistent so that it can be analysed at evaluation time. This behaviour may be implemented either in Java or with any web scripting language.

In order to set up such an experiment, a web server is required where the questionnaire as well as the response handling component is made accessible. We have been working with Apache Tomcat[6], an application server implementing Java Servlet and JavaServer Pages technologies. Using Java Servlet technology, the processing of response data as outlined in this simple example can be programmed using few simple lines of code. However, the resulting Servlet may be extended to also access the user's Facebook profile and store the additional data along with the user response. As mentioned before, Facebook offers a REST-based API that allows the Facebook platform to be accessed via HTTP GET and POST requests. This API also provides the means to notify the users through a Facebook built-in news feed, to write messages and invite them to use a particular application.

Once the characteristics used to measure similarity have been defined and a corresponding Facebook application has been deployed, actual user data must be gathered. Therefore, the recruitment of users is an important aspect of conducting an experiment. Apart from the incentives such as mapping real-world encounters to social network connections, displaying the location of these encounters and notifying users about the physical proximity of social network contacts, it is also important to ensure a high frequency of physical encounters. One possibility of addressing this issue is to leverage the concept of regional networks already present in Facebook. These networks could be used in order to recruit users from a restricted geographical region, effectively increasing the chance of these users being copresent at some time.

Finally, once the data required to prove or disprove the hypothesis has been gathered, it must be examined for possible correlations between collocation and user similarity. This can be done by applying standard statistical tests such as chi square (χ^2) or mean-value analysis. As our methodology to prove the existence or lack of correlation between datasets is based on straightforward statistical analysis, it is of little interest and lies outside the scope of this paper.

6 Conclusions

A frequent problem associated with mobile social software is the fact that underlying assumptions may be difficult to test. For example, a collaborative filtering approach based on opportunistic information sharing in mobile ad-hoc networks assumes that the number of times users are co-located correlates with their similarity. Previous projects have used a range of methods such as questionnaires, wizard-of-oz style experiments and the deployment of applications to try and validate such hypotheses but each can be severely restricted in terms of the amount of data collected due to practical considerations.

We have shown how on-line social networking systems such as Facebook can be coupled with simple, standard mobile applications to provide a testbed for a variety of mobile social applications by mapping physical copresence onto connections in virtual social networks. Applications can then be developed using the support of the social networking systems to analyse, for example, the relationship between physical copresence of users and their similarity in terms of interest and taste. The simple application development environment offered by sites such as Facebook allows these applications to be easily designed and deployed.

Clearly, it is important to attract users to such applications, especially since it requires them to register their mobile phones. We believe that this can be done by offering additional functionality such as a geographical map showing where user encounters took place or the possibility of notifying users through their mobile device in the event of other Facebook users being currently in their vicinity.

Altogether, we believe that the association of virtual and real world social networks provides an important incentive for people to participate in registering with our framework. As a consequence, it should be possible to attract a sufficient number of users in order to conduct an experiment for validating assumptions underlying a wide variety of mobile social applications.

We have finished developing the framework presented in this paper and have integrated it in Facebook. We are currently experimenting with selected users collecting data, uploading it to the server and using the Facebook application. The next stage consists of releasing it to the public and developing further applications allowing user simi-

[6] http://tomcat.apache.org/

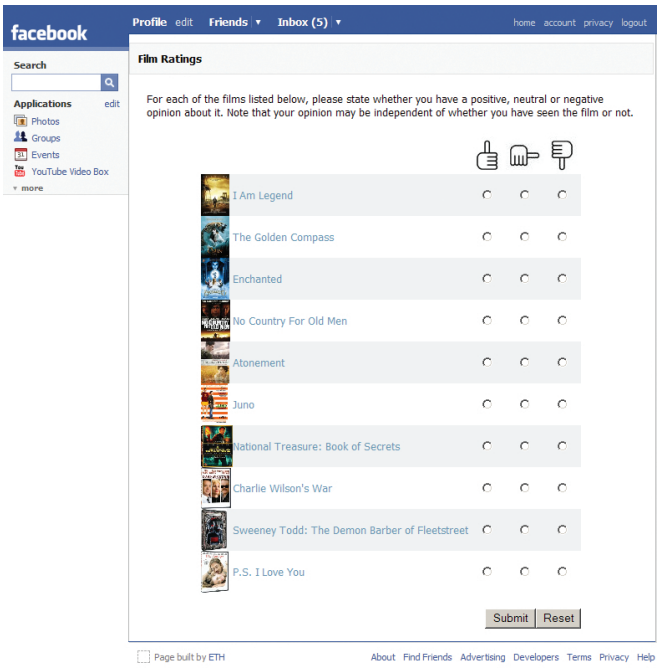


Figure 4. Example user similarity assessment

larities to be measured and put into relation with the collocation data collected.

REFERENCES

[1] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.

[2] S. Counts and J. Geraci. Incorporating physical co-presence at events into digital social networking. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1308–1311, New York, NY, USA, 2005. ACM.

[3] A. de Spindler, M. C. Norrie, and M. Grossniklaus. Collaborative filtering based on opportunistic information sharing in mobile ad-hoc networks. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 408–416, Berlin / Heidelberg, Germany, 2007. Springer.

[4] A. de Spindler, M. C. Norrie, and M. Grossniklaus. Recommendation based on Opportunistic Information Sharing between Tourists. *Information Technology & Tourism*, (to appear).

[5] N. Eagle and A. Pentland. Social Serendipity: Proximity Sensing and Cueing. Technical Report 580, MIT Media Laboratory, 2004.

[6] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.

[7] R. T. Fielding and R. N. Taylor. Principled design of the modern Web architecture. *ACM Trans. Interet Technol.*, 2(2):115–150, 2002.

[8] J. Freyne, E. Varga, D. Byrne, A. F. Smeaton, B. Smyth, and G. J. F. Jones. Realising Context-Sensitive Mobile Messaging. In *OTM Workshops (1)*, pages 407–416, 2007.

[9] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 187–201, New York, NY, USA, 2004. ACM.

[10] L. E. Holmquist, J. Falk, and J. Wigstrm. Supporting group collaboration with interpersonal awareness devices. *Personal and Ubiquitous Computing*, 3:13–21, 1999.

[11] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *MobiArch '07: Proceedings of first ACM/IEEE international workshop on Mobility in the evolving internet architecture*, pages 1–8, New York, NY, USA, 2007. ACM.

[12] J. Lawrence, T. R. Payne, and D. D. Roure. Co-Presence Communities: Using Pervasive Computing to Support Weak Social Networks. In *WETICE '06: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 149–156, Washington, DC, USA, 2006. IEEE Computer Society.

[13] M. McNett and G. M. Voelker. Access and mobility of wireless pda users, 2005.

[14] A. Natarajan, M. Motani, and V. Srinivasan. Understanding Urban Interactions from Bluetooth Phone Contact Traces. pages 115–124, 2007.

[15] T. Nicolai, E. Yoneki, N. Behrens, and H. Kenn. Exploring Social Context with the Wireless Rope. In *OTM Workshops (1)*, pages 874–883, 2006.

[16] E. Paulos and E. Goodman. The familiar stranger: anxiety, comfort, and play in public places. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 223–230, New York, NY, USA, 2004. ACM.

[17] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *Communications Magazine, IEEE*, 44(11):134–141, 2006.

[18] J. Perkio, V. Tuulos, M. Hermersdorf, H. Nyholm, J. Salminen, and H. Tirri. Utilizing Rich Bluetooth Environments for Identity Prediction and Exploring Social Networks as Techniques for Ubiquitous Computing. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 137–144, Washington, DC, USA, 2006. IEEE Computer Society.

Collaborative Filtering via Concept Decomposition on the Netflix Dataset

Nicholas Ampazis[1]

Abstract. Collaborative filtering recommender systems make automatic predictions about the interests of a user by collecting information from many users (collaborating). Most recommendation algorithms are based in finding sets of customers or items whose ratings overlap in order to create a model for inferring future ratings or items that might be of interest for a particular user. Traditional collaborative filtering techniques such as k-Nearest Neighbours and Singular Value Decomposition (SVD) usually provide good accuracy but are computationally very expensive. The Netflix Prize is a collaborative filtering problem whose dataset is much larger than the previously known benchmark sets and thus traditional methods are stressed to their limits when challenged with a dataset of that size. In this paper we present experimental results that show how the *concept decomposition* method performs on the movie rating prediction task over the Netflix dataset and we show that it is able to achieve a well balanced performance between computational complexity and prediction accuracy.

1 INTRODUCTION

Collaborative filtering (CF) is a subfield of machine learning that aims at creating algorithms to predict user preferences based on past user behavior in purchasing or rating of items [15],[18]. CF recommender systems are very important in e-commerce applications as they contribute much to enhancing user experience and, consequently, to generating sales and increasing revenue as they help people find more easily items that they would like to purchase [19].

In October, 2006 Netflix released a large movie rating dataset and challenged the data mining, machine learning and computer science communities to develop systems that could beat the accuracy of their in-house developed recommendation system (Cinematch) by 10% [3]. In order to render the challenge more interesting, the company will award a Grand Prize of $1M to the first team that will attain this goal, and in addition, Progress Prizes of $50K will be awarded on the anniversaries of the Prize to teams that make sufficient accuracy improvements. Apart from the financial incentive however, the Netflix Prize contest is enormously useful for recommender system research since the released Netflix dataset is by far the largest ratings dataset ever becoming available to the research community. Most work on recommender systems outside of companies like Amazon or Netflix up to now has had to make do with the relatively small 1M ratings MovieLens data [12] or the 3M ratings EachMovie dataset [11]. Netflix provided 100480507 ratings (on a scale from 1 to 5 integral stars) along with their dates from 480189 randomly-chosen, anonymous subscribers on 17770 movie titles. The data were collected between October, 1998 and December, 2005 and reflect the distribution of all ratings received by Netflix during this period. Netflix withheld over 3M most-recent ratings from those same subscribers over the same set of movies as a competition qualifying set and contestants are required to make predictions for all 3M withheld ratings in the qualifying set. As a performance measure the company has selected the Root Means Square Error (RMSE) criterion between the actual and predicted scores. In addition Netflix also identified a "probe" subset of the complete training set consisting of about 1.4M ratings as well as the probe Cinematch RMSE value to permit off-line comparison with systems before submission on the qualifying set.

In this paper, we present the main components of one of our approaches to the Netflix Prize based on the *concept decomposition* method [5] and we show that it combines moderate computational complexity with good prediction accuracy on the RMSE criterion. However, due to the limits of the paper and our obvious interests, we intentionally do not publish all details of our method since some small but important details remain hidden. To this end we only report results evaluated on the probe subset of the Netflix dataset.

2 COLLABORATIVE FILTERING RECOMMENDER SYSTEMS

The goal of a CF algorithm is to recommend products to a target user based on the opinions of other users [6],[8],[14]. In a typical CF scenario, there is a list of n users $U = \{u_1, u_2, ..., u_n\}$ and a list of m items $I = \{i_1, i_2, ..., i_m\}$. For each user u_i we have a list of items I_{u_i} for which the user has expressed an opinion about. These opinions can be either explicitly given by the user as a rating score (as is the case with Netflix) or can be implicitly derived from the user's purchase records. Under this setting we consider a distinguished user $u_a \in U$ called the *active user* for whom the task of a collaborative filtering algorithm is to suggest other items that the active user might like. This suggestion can take either of the following two forms:

- **Prediction:** Provide a numerical value, $P_{a,j}$ expressing the predicted likeliness of item $i_j \notin I_{u_a}$ for the active user u_a. The predicted value should be within the same scale (e.g., from 1 to 5) as the opinion values provided by u_a in the past.
- **Recommendation:** Provide a list of N items, $I_r \subset I$, that the active user will like the most. Obviously the recommended list should only contain items not contained in I_{u_a}, that is $I_r \cap I_{u_a} = \Phi$. This kind of suggestion is also known as Top-N recommendation.

Most collaborative filtering based recommender systems represent every user as an m-dimensional vector of items, where m is the number of distinct catalog items and every item as an n-dimensional

[1] Department of Financial and Management Engineering, University of the Aegean, Greece, email: n.ampazis@fme.aegean.gr

vector of distinct users. These representations can also be combined together to form an $n \times m$ *user-by-item* matrix. Note that usually the user-by-item matrix is extremely sparse since each user has rated/purchased only a small fraction of the item's catalogue and some items may have very few ratings. For example Netflix dataset is more than 99% sparse since most users have only rated a fraction of the movies (most users have rated at most 200 movies) [3].

Using this matrix, a variety of methods can be applied for the prediction or the recommendation task. These methods range from user-to-user [18] (or item-to-item [17]) k-Nearest Neighbours techniques, to Singular Value Decomposition (SVD) [16] (or more advanced factorizations of the user-by-item matrix [9],[13]) or combinations of all the above techniques. For example this year's Progress Prize winners (Bellkor team) have used a blending of 107 different techniques whose combination corresponds to an 8.43% improvement over the Cinematch system [2].

Even though most of the above techniques can provide accurate predictions, they require, however, a computationally expensive training component. For example in a user-to-user k-Nearest Neighbours setting one may have to pre-compute all the correlations between users in order to otherwise having to repeat the same calculations on-the-fly. In addition batch matrix factorization techniques applied to a matrix of the scale of the Netflix user-by-item matrix may render the computation intractable on non-parallel computing architectures. Therefore these techniques may not be directly applicable for dynamic settings and may only be deployed in static settings where the known preferences do not vary much with time. However, a number of practical scenarios such as real-time personalization require dynamic collaborative filtering that can handle new users, items, and ratings entering the system at a rapid rate. In such situations, it is imperative for the recommender system to dynamically adapt its predictions using the new information, which in turn requires a fast and efficient training algorithm such as the one that we describe in the following sections.

3 CLUSTER MODELS

3.1 Overview of clustering methods

In order to find users who are similar to the active user, *cluster models* divide the user base into many partitions and treat the task as a cluster assignment problem. The algorithm's goal is to assign the user to the partition containing the most similar customers and then uses the purchases and ratings of the customers in the cluster to generate recommendations [21]. The partitions are typically created using an offline clustering (e.g. k-means [10]) or other unsupervised learning algorithm (e.g [7]). Once the algorithm generates the clusters it computes the user's similarity to each cluster and then chooses the cluster with the strongest similarity to accordingly assign the user into. Some algorithms may also assing users into multiple clusters with varying degrees by determining the strength of each relationship, like ,for example, Fuzzy c-means clustering [4].

Cluster models have better online scalability and performance than other collaborative filtering methods because they have to make comparisons of the active customer's vector to only a predefined number of cluster vectors. However recommendation quality may be low since the sparsity of the dataset may prevent the cluster models to group together the most similar customers resulting in an incosistent aggregation of ratings within a cluster [20].

3.2 Our item-clustering approach

Rather than grouping similar users to clusters, our item-clustering method matches each of the items' ratings in order to combine similar items into a group. The partitioning of the movie base is accomplished with the *concept decomposition* method which allows us to assing each item to more than one cluster with a well defined degree of membership. A full desctiption of the method is presented in section 4.

For the Netflix case the basic idea is to come up with a number of "representative" movies, with each one representing a group of movies with similar characteristics (genres). In this case, each cluster is really a "made-up" movie representing a group of movies. The data for each cluster are just user ratings, one rating value for each distinct user. For each user in every cluster we calculate its average rating from the movies who both belong to this cluster and have been rated by this customer, that is we replace the clusters rating for this user with the average. In this way movies are clustered into genres (e.g. action, adventure, comedy, science fiction, etc) by the users who have rated them and we allow each movie to belong with varying degrees to more than one genre. Therefore the predicted rating that the active user would give to a movie is a weighted combination of the average rating that this user usually gives to movies belonging to each particular genre, where the weights are determined by the membership of the particular movie in each genre.

Formally, in order to provide an active user's prediction on an unseen movie, we aggregate the active user's ratings from all the clusters that the unseen movie belongs to, and then we calculate the weighted average of all the clusters' predictions. Of course, the predictions of each cluster is the average prediction of all movies belonging to that cluster.

The algorithm can be summarized as follows: To make a prediction for user u on movie m:

- Find the k clusters that are most similar to movie m. Calculate the membership of the movie to each cluster according to a suitable similarity metric.
- The predicted rating P_{vm} for user u on movie m is the weighted average of the k clusters' ratings.

$$P_{um} = \frac{\sum_k r_{uk} w_{mk}}{\sum_k w_{mk}} \quad (1)$$

where r_{uk} is the average rating of user u in movie cluster k and w_{mk} is the membership (weight) of movie m in cluster k.

4 THE CONCEPT DECOMPOSITION METHOD

In this section, we study how to partition the high-dimensional and sparse movie vectors of the Netflix Dataset into disjoint conceptual categories (genres) and to determine the membership of each movie to every cluster. Towards this end, we briefly describe the spherical k-means clustering algorithm and the structure of the clusters it produces.

4.1 Concept Vectors

Given *m* item vectors $\boldsymbol{x}_1$,$\boldsymbol{x}_2$,...,$\boldsymbol{x}_m$ in R^n we denote by π_1,π_2,...,π_k their partitioning into *k* disjoint clusters such that

$$\cup_{j=1}^{k} \pi_j = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\} \text{ and } \pi_j \cap \pi_l = \Phi \text{ if } j \neq l \quad (2)$$

For each fixed $1 \leq j \leq k$, the *mean vector* or the *centroid* of the item vectors contained in the cluster π_j is

$$\boldsymbol{m}_j = \frac{1}{n_j} \sum_{\boldsymbol{x} \in \pi_j} \boldsymbol{x} \tag{3}$$

where n_j is the number of items in π_j. Since the mean vector $\boldsymbol{m}_j$ may not necessarily have a unit norm we can define the corresponding *concept vector* as

$$\boldsymbol{c}_j = \frac{\boldsymbol{m}_j}{\| \boldsymbol{m}_j \|} \tag{4}$$

Due to the Cauchy-Schwarz inequality, the concept vector $\boldsymbol{c}_j$ has the important property that for any unit vector $\boldsymbol{z}$ in R^n

$$\sum_{\boldsymbol{x} \in \pi_j} \boldsymbol{x}^T \boldsymbol{z} \leq \sum_{\boldsymbol{x} \in \pi_j} \boldsymbol{x}^T \boldsymbol{c}_j \tag{5}$$

Thus, the concept vector may be thought of as the vector that is closest in cosine similarity (in an average sense) to all the items vectors in the cluster π_j.

4.2 The Spherical k-means algoritm

The concept vectors can be calculated using the *spherical k-means* algorithm proposed in [5]. The outline of the algorithm is listed in Algorithm 1.

Algorithm 1 Spherical k-means

Require: A set of *m* data vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$ in R^n and the number of clusters *k*

Ensure: Unit-length cluster centroid vectors $\{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k\}$ are initialized to some (e.g. random) values

Method:

1: Data assignment:
For each $\boldsymbol{x}_i$ assign $\boldsymbol{x}_i$ to $\pi_k \Leftarrow \arg\max_k [\boldsymbol{x}_i^T \boldsymbol{c}_k]$

2: New centroid estimation:
For each cluster k, recalculate $\boldsymbol{m}_k = \frac{1}{n_k} \sum_{\boldsymbol{x} \in \pi_k} \boldsymbol{x}$ and let $\boldsymbol{c}_k \Leftarrow \frac{\boldsymbol{m}_k}{\|\boldsymbol{m}_k\|}$

3: **if** (*no $\boldsymbol{x}_i$ can be further reassigned*) **then**

4: exit;

5: **end if**

4.3 Locality and Sparsity of Concept Vectors

Since the Netflix item vectors (movies) are almost 99% sparse, consequently, the concept vectors that are produced by their clustering are also sparse. This means that at each cluster there is a small number of customers that contribute to the concept vector's coordinates. This observation allows us to associate a *user cluster* W_j within the movie cluster p_j as in [5] in the following way:

A user $1 \leq w \leq n$ is contained in W_j, if the value (weight) of that user in c_j is larger than the weight of that user in any other concept vector c_l, $1 \leq l \leq k$, $l \neq j$.

These user clusters allow us to idenfity groups of users within a cluster of movie vectors since most of the weights of a concept movie vector is concentrated in or localized to the corresponding user cluster [5].

4.4 Concept decomposition and matrix factorization

It has been shown in [5] that the spherical k-means clustering algorithm is directly related to matrix factorization in the sense that we can consider the approximation of each item vector by a linear combination of the concept vectors. This matrix approximation scheme is known as *concept decomposition*.

We define the *concept matrix* as a $n \times k$ matrix such that for $1 \leq j \leq k$, the j-th column of the matrix is the concept vector $\boldsymbol{c}_j$, that is

$$\boldsymbol{C}_k = [\boldsymbol{c}_1 \boldsymbol{c}_2 ... \boldsymbol{c}_k] \tag{6}$$

Assuming that the k concept vectors are linearly independent then it follows that the concept matrix has rank k.

For any partitioning of the item vectors, we define the corresponding concept decomposition $\tilde{\boldsymbol{X}}_k$ of the user-by-item matrix $\boldsymbol{X}$ as the least-squares approximation of $\boldsymbol{X}$ onto the column space of the concept matrix $\boldsymbol{C}_k$. We can write the concept decomposition as a $n \times m$ matrix

$$\tilde{\boldsymbol{X}}_k = \boldsymbol{C}_k \boldsymbol{Z}^* \tag{7}$$

where $\boldsymbol{Z}^*$ is a $k \times m$ matrix that can be determined by solving the following least-squares problem:

$$\boldsymbol{Z}^* = \arg\min_k \| \boldsymbol{X} - \boldsymbol{C}_k \boldsymbol{Z}^* \| \tag{8}$$

For this problem it is well known that a closed form solution exists namely,

$$\boldsymbol{Z}^* = (\boldsymbol{C}_k^T \boldsymbol{C}_k)^{-1} \boldsymbol{C}_k^T \boldsymbol{X} \tag{9}$$

This leads to the following approximation to the original user-by-item matrix:

$$\tilde{\boldsymbol{X}}_k = \boldsymbol{C}_k (\boldsymbol{C}_k^T \boldsymbol{C}_k)^{-1} \boldsymbol{C}_k^T \boldsymbol{X} \tag{10}$$

The matrix $\boldsymbol{D} = \boldsymbol{C}_k^T \boldsymbol{X}$ is the transpose of matrix $\boldsymbol{X}^T \boldsymbol{C}_k$ which can be thought of as the k-dimensional representation of the entire movie collection. In other words, this matrix contains the cosine similarities (since vectors are normalized) between the movie vectors and the concept vectors. Hence the elements of the matrix $\boldsymbol{D}$ determine the degree (membership) by which each movie is associated with each one of the k clusters. A more careful comparison between equations (1) and (10) clearly shows that $P_{um} \simeq \tilde{X}_{k_{(um)}}$ when one substitutes w_{mk} with D_{mk} and r_{uk} with the average user ratings provided by the un-normalized version of the concept matrix $\boldsymbol{C}_k$.

5 EVALUATION

The approach was evaluated on the entire Netflix dataset and experiments were run on a 3.4GHz Dual Core Pentium CPU with 3G RAM running Ubuntu 7.10 desktop x86_64 (Gutsy Gibbon) operating system. The spherical k-means clustering of the dataset was generated from a customized version of the *gmeans* software package (available from "http://www.cs.utexas.edu/users/dml/Software/gmeans.html") compiled with the Intel C/C++ Compiler Professional Edition for Linux. The algorithm converged in 10 iterations with a total running time of approximately 13.5 minutes.

Tables 1 to 8 show the top 5 movies (determined by their membership) in 8 representative (out of the 100) clusters produced by the spherical k-means clustering algorithm on the Netflix movie vectors. It is interesting to note the clear partitioning of the dataset into

the various genres: Science fiction (Table 1 / Cluster 4), Documentaries (Table 2 / Cluster 6), Rock Music (Table 3 / Cluster 11), Rop Music (Table 4 / Cluster 20), Religious (Table 5 / Cluster 21), Seasonal (Table 6 / Cluster 25), Children's (Table 7 Cluster 34), and Crime/Mystery (Table 8 / Cluster 42). The rest of the clusters (which we cannot present here due to space limitations) also exhibit similar coherence and reveal more fine-grained partitions between genres as is the case between Clusters 11 and 20 where they are both about music DVDs but in cluster 11 we find titles from rock music whereas in Cluster 20 we find pop music performances.

It is important to stress the fact that the algorithm has produced these partitions using solely the Netflix provided user ratings without the utilization of any external meta-data information about the movies (e.g. from IMDB [1]).

The efficiency of the approach on the probe dataset was evaluated with different combinations of the following parameters:

- C : the total number of clusters.
- k: the number of most similar movie clusters used for prediction.

The best prediction result was obtained with the combination $C = 100$ and $k = 5$, yelding an RMSE of 0.89151, which represents a 5.89% improvement over Cinematch's performance (0.9474). The time required to obtain the probe predictions was 20 seconds on the same computer as above (15 seconds to load the necessary datafiles of equation (10) and 5 seconds to apply equation (1) over the 1.4M probe user/movie pairs).

The results were also evaluated on the qualifying set which we however now hold back for obvious reasons. However, the probe result itself indicates the efficiency of the proposed technique which has been shown to combine moderate computational complexity with good prediction accuracy.

Table 1. Movies in Cluster #4

Star Trek: The Next Generation: Season 5
Star Trek: The Next Generation: Season 6
Star Trek: The Next Generation: Season 7
Star Trek: The Next Generation: Season 4
Star Trek: The Next Generation: Season 3

Table 2. Movies in Cluster #6.

National Geographic: Inside American Power: The Pentagon
National Geographic: The FBI
National Geographic: Inside American Power: The White House
National Geographic: Inside American Power: Air Force One
National Geographic: Ambassador: Inside the Embassy

Table 3. Movies in Cluster #11.

Eric Clapton: One More Car, One More Rider
Eric Clapton & Friends in Concert: The Crossroads Benefit
Eric Clapton: 24 Nights
Eric Clapton Unplugged
Stevie Ray Vaughan and Double Trouble: Live at the El Mocambo 1983

Table 4. Movies in Cluster #20.

Madonna: The Immaculate Collection
Madonna: The Girlie Show: Live Down Under
Madonna: The Video Collection 1993-1999
Madonna: Ciao Italia: Live from Italy
Britney Spears: Britney in Hawaii: Live and More

Table 5. Movies in Cluster #21.

Jeremiah: The Bible
Solomon: The Bible
Esther: The Bible
Great People of the Bible: The Apostle Paul
Great People of the Bible: Abraham, Sarah, Isaac, Jacob & Joseph

Table 6. Movies in Cluster #25.

Dr. Seuss' How the Grinch Stole Christmas
Rudolph the Red-Nosed Reindeer
A Charlie Brown Christmas
It's the Great Pumpkin, Charlie Brown
It's a Wonderful Life

Table 7. Movies in Cluster #34.

Sesame Street: Elmo's World: Springtime Fun
Garfield and Friends: Vol. 4
Sesame Street: Sing Along
Underdog: Nemesis
Popeye 75th Anniversary Collector's Edition

Table 8. Movies in Cluster #42.

Midsomer Murders: Blue Herrings
Midsomer Murders: The Electric Vendetta
Midsomer Murders: Garden of Death
Midsomer Murders: Dark Autumn
Inspector Morse 18: Who Killed Harry Field?

6 CONCLUSION

We have shown that the concept decomposition method may be successfully applied to the Netflix dataset in order to provide meaningful clustering of the movies into clusters (genres). This clustering information can be further utilized for the prediction task of a collaborative filtering recommender system. The method alone was able to provide an improvement of 5.89% over Cinematch's performance on the probe dataset wihtout using any other meta-data information apart from the user/movie ratings provided by Netflix. In addition the method has excibited moderate computational complexity as it can produce a whole set of probe predictions in less than 15 minutes (including the off-line clustering stage). These results indicate that the method is very promising and that in can be expecially useful in cases where its results can be blended with other efficient methods that are currently under our research scope with regards to the Netflix dataset.

References

[1] 'The internet movie database'. http://www.imdb.com/.

[2] Robert M. Bell and Yehuda Koren, 'Lessons from the netflix prize challenge', *SIGKDD Explor. Newsl.*, **9**(2), 75–79, (December 2007).

[3] J. Bennett and S. Lanning, 'The netflix prize', in *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2007).

[4] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[5] I. S. Dhillon and D. S. Modha, 'Concept decompositions for large sparse text data using clustering', *Machine Learning*, **42**(1), 143–175, (Jan 2001).

[6] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, 'Using collaborative filtering to weave an information tapestry', *Commun. ACM*, **35**(12), 61–70, (December 1992).

[7] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, N.Y.

[8] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl, 'Grouplens: applying collaborative filtering to usenet news', *Commun. ACM*, **40**(3), 77–87, (1997).

[9] Daniel D. Lee and Sebastian S. Seung, 'Learning the parts of objects by non-negative matrix factorization', *Nature*, **401**, 788–91, (October 1999).

[10] J. Macqueen, 'Some methods for classification and analysis of multivariate observations', in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, eds., L. M. Le Cam and J. Neyman, volume 1 of *Berkeley: University of California Press*, pp. 281–297, (1967).

[11] P. McJones, 'Eachmovie collaborative filtering data set'. Available from http://research.compaq.com/SRC/eachmovie/, 1997.

[12] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl, 'Movielens unplugged: experiences with an occasionally connected recommender system', in *Intelligent User Interfaces*, pp. 263–266, (2003).

[13] Jasson D. M. Rennie and Nathan Srebro, 'Fast maximum margin matrix factorization for collaborative prediction', in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pp. 713–719, New York, NY, USA, (2005). ACM.

[14] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl, 'Grouplens: an open architecture for collaborative filtering of netnews', in *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186, New York, NY, USA, (1994). ACM.

[15] Paul Resnick and Hal R. Varian, 'Recommender systems - introduction to the special section', *Commun. ACM*, **40**(3), 56–58, (1997).

[16] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, 'Incremental singular value decomposition algorithms for highly scalable recommender systems', in *Proceedings of the 5th International Conference in Computers and Information Technology*, (2002).

[17] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl, 'Item-based collaborative filtering recommendation algorithms', in *World Wide Web*, pp. 285–295, (2001).

[18] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl, 'Analysis of recommendation algorithms for e-commerce', in *ACM Conference on Electronic Commerce*, pp. 158–167, (2000).

[19] J. Ben Schafer, Joseph A. Konstan, and John Riedi, 'Recommender systems in e-commerce', in *ACM Conference on Electronic Commerce*, pp. 158–166, (1999).

[20] J. Ben Schafer, Joseph A. Konstan, and John Riedl, 'E-commerce recommendation applications', *Data Mining and Knowledge Discovery*, **5**(1/2), 115–153, (2001).

[21] V. Schickel-Zuber and B. Faltings, 'Using hierarchical clustering for learning the ontologies used in recommendation systems', in *Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2007).

Filler Items Strategies for Effective Shilling Attacks

Sanjog Ray[1] and **Ambuj Mahanti**[2]

ABSTRACT. Recommender systems have become a widely researched area because of its widespread application in ecommerce platforms. One area of research which has recently gained importance is the security of recommender systems. Malicious users may influence the recommender system by inserting biased data into the system. Such attacks may lead to erosion of user trust in the objectivity and accuracy of the system. Recent research on modeling of attacks have mainly focused in creating attack profiles which have high similarity with as many genuine users as possible. In this paper, we introduce attack strategies that are based on creating attack profiles that have high similarity with only those genuine users who have already rated the target item. Our approach considers the target item rating distribution to assign values to filler items. We show through experiments that our strategies on being applied to an average attack model results in substantial improvement over existing attack models.

1 INTRODUCTION

Recommender systems are technology based systems that provide personalized recommendations to users. Recommendations are generated from opinions and actions of other users with similar tastes. However, with increasing popularity of recommender systems in ecommerce sites they have become susceptible to shilling attacks. In shilling attacks, attackers try to influence the system by inserting biased data into the system. Recent researches have started focusing on attack models and attack detection strategies. [1, 2, 3 and 4]

An attack on a recommender system is mounted by injecting a set of biased attack profiles into the system. Each attack profile contains biased ratings data and a target item. Profiles are injected into the system by fictitious user identities created by the attacker. Every attack can be classified as a push attack or a nuke attack. In a push attack, the objective of the attacker is to increase the likelihood of the target item being recommended to a large section of the users in the system. While in a nuke attack, the objective is to prevent the target item from being recommended.

An attack is also classified as a *high-knowledge* attack or *low-knowledge* attack [1]. A *high-knowledge* attack requires more detail knowledge of the rating distributions of each item present in the system. While in a *low-knowledge* attack, to launch an attack, dependence on the recommender system for information on the items is minimal. The approach of construction the attack profile, based on knowledge about the items, products, and users of recommender systems is known as an attack model. The primary objective of an attacker is to build attack models that provide the most impact with minimal knowledge. The other concern of importance to an attacker is to create models which are hard to detect by attack detection algorithms.

The general form of a push attack profile is shown in figure 1.

[1] Indian Institute of Management, India, email: fp062004@iimcal.ac.in

[2] Indian Institute of Management, India, email: am@iimcal.ac.in

An attack profile consists of a set of m ratings for m items; where m is the total number of items present in the system. This attack profile of m ratings can be divided into four sets of items: a target item i_t, a set of selected items I_S, a set of filler items usually randomly chosen I_F, and a set of unrated items I_E. Attack models are defined by the rules by which the four set of items are identified and the way ratings are assigned to the items present in the sets. For some attacks set of selected items may be empty.

Selected Items (I_S)	Filler items (I_F)	Unrated items (I_E)	Target item (i_t)

Figure 1. A general form of attack profile

Most past researches have mainly focused on the set of selected items while creating new attack models. In this paper, we examine the importance of the target item rating distribution in improving the effectiveness of an attack. This paper proposes different strategies for selecting values for the filler items. It examines the effectiveness of each strategy for a push attack. We show that filler item strategy depends on the target item ratings distribution. Through experimental evaluation we show that having the right values for filler items can result in more effective attacks. Our paper provides an effective plan for mounting an average attack by intelligently selecting filler items values, values based on target item ratings distribution.

This paper is organized as follows. In section 2 we provide a brief summary of various attack models and their filler strategies. In section 3 user-based collaborative filtering algorithm and evaluation metric used are described. In section 4 we provide details of our work of selecting different strategies for assigning values to filler items. In section 5 we describe the experimental evaluation process and report the results obtained. Finally, we conclude the paper in section 6.

2 TYPES OF ATTACK

Various attack models have been proposed in previous researches on shilling of recommender systems. We discuss below, some of the popular attack models on which much research is focused on. A comprehensive study of different attack models can be found in [1].

Random attack: One of the initial attack models, attack profile has filler items chosen randomly and ratings are assigned to the filler items from a set of random values chosen from a distribution centered on the system mean. System mean is the mean for all user ratings across all items. This is *a low – knowledge* attack as minimal knowledge is required to obtain system mean value. It has been found that this model is not very effective [1].

Average attack: One of the most powerful attack models. In an average attack model, set of selected items is empty. Filler items are selected randomly, and each filler item is assigned its mean rating. Mean rating here corresponds to the average rating for the item across all users in the database who have rated it. Average attack is a *high-knowledge* attack as mean rating of each filler item is

required to mount an attack. However, in [5] it has been shown that this attack can be effective with limited knowledge i.e. a small set of filler items can perform an effective attack.

Bandwagon attack: In this model, the set of selected items contains few of those items that have high popularity among users. Thus, attack profiles created will have higher chances of being similar to a large number of users. Selected items and the target item are assigned maximum rating value. As in random attack, filler items are randomly selected and assigned mean rating of items across the whole system. Therefore, bandwagon attack can be seen as an extension of the random attack. Bandwagon attack is a *low–knowledge* attack as popular items data can be obtained from publicly available information sources.

Segmented attack: This attack is modeled to push the target item to those users who are most likely to be influenced by the recommendation. A segment is defined as a group of users having affinity for items of similar features. Group of users who have rated highly most of the popular horror movies is an example of a segment of users interested in horror movies. So, an attacker with intent to promote a horror movie will try to get his target item recommended to this segment of users as the likelihood of influencing them is higher.

In this model, the set of selected items contains few of those items that have high popularity among users of the targeted segment. Selected items and the target item are assigned maximum rating value. Filler items are identified randomly and given the lowest possible rating. It has been shown that segmented attack is the most effective model against in-segment users. It is a *low-knowledge* attack as selection of highly rated movie with similar features can be achieved from public information sources.

3 RECOMMENDATION ALGORITHM & EVALUATION METRIC

In this paper we have evaluated our filler item strategy for attacks against the user based collaborative filtering algorithm. In this section we describe the collaborative filtering algorithm, the evaluation metric used, and the notion of prediction shift.

3.1 User based collaborative filtering

In collaborative filtering, a user is recommended items that people with similar tastes and preferences liked in the past. This technique mainly relies on explicit ratings given by the user and is the most successful and widely used technique [6]. In user based collaborative filtering [7], firstly, neighborhood of *k* similar users is found for the target user. Then for generating prediction for an item not yet seen by the target user, weighted average of the ratings given by the *k* similar neighbors towards the predicted item is used.

To calculate similarity among users we use Pearson-r correlation coefficient. Let the set of items rated by both users *u* and *v* be denoted by *I*, then similarity coefficient ($Sim_{u,v}$) between them is calculated as

$$Sim_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r_v})^2}} \quad (1)$$

Here $r_{u,i}$ denotes the rating of user *u* for item *i*, and $\bar{r_u}$ is the average rating given by user *u* calculated over all items rated by *u*. Similarly, $r_{v,i}$ denotes the rating of user v for item *i*, and $\bar{r_v}$ is the average rating given by user *v* calculated over all items rated by *v*.

To compute the prediction for an item *i* for target user *u*, we use the following formula.

$$P_{u,i} = \bar{r_u} + \frac{\sum_{v \in V} Sim_{u,v}(r_{v,i} - \bar{r_v})}{\sum_{v \in V}|Sim_{u,v}|} \quad (2)$$

Where *V* represents the set of *k* similar users. While calculating prediction only those users in set *V* who have rated item *i* are considered.

3.2 Prediction shift

For the purpose of measuring the effectiveness of the attack we use the widely used metric prediction shift. Prediction shift of a target item is the difference of average predicted rating of the target item, after and before the attack, for all target users. Average Prediction shift of an attack is the average change in prediction for all target items. We use the same formula as in [8], which is defined as follows.

Let *U* and *I* be the sets of target users and target items. Let $\Delta_{u,i}$ denote the prediction shift for user *u* on item *i*. $\Delta_{u,i}$ can be measured as $\Delta_{u,i} = p'_{u,i} - p_{u,i}$, where $p'_{u,i}$ is the prediction value after the attack and $p_{u,i}$ before the attack. The average prediction shift for an item *i* over all users can be computed as

$$\Delta_{ui} = \frac{\sum_{u \in U} \Delta_{u,i}}{|U|} \quad (3)$$

The prediction shift for an attack model is the average prediction shift for all items tested. It can be computed as

$$\Delta = \frac{\sum_{i \in I} \Delta_i}{|I|} \quad (4)$$

4 FILLER ITEM STRATEGIES FOR AVERAGE ATTACK

Known attack models like average attack, bandwagon attack, and segmented attack are focused at creating attack profiles which have greater chance of having high similarity with as many users as possible. Because an attack profile with high similarity with a genuine user increases the chance of it being selected in top *k* similar neighbors of the user, thereby influencing the rating of the target item. Bandwagon attack and segmented attack have tried to achieve this objective by selecting popular items as part of their attack profiles. An attack profile consisting of the popular items will have similarity with higher number of users, which should finally result in an effective attack. However, it has been found that average attack is the more effective compared to bandwagon and segmented attack [1]. One possible reason for this unexpected result could be the presence of other factors which also affect the effectiveness of an attack.

Our proposed approach considers rating distribution of the target item as a critical factor that can affect the effectiveness of an attack. Unlike previous attack models which focus on creating attack profiles that are similar to as many users as possible from the set of all users; the objective of our approach is to create attack profiles that increase similarity with as many of those users who have rated the target item. Our approach proposes two strategies that are based on the rating distribution of the target item. Both strategies improve

similarity by assigning appropriate values to filler items. As we show below, our proposed approach performs substantially better than average attack model.

Our proposed approach is described below.

First, on the basis of the target item rating distribution, target item is categorized into T_L or T_H category.

T_L : Target item with majority of their ratings at lower end of the rating scale fall into this category. In our experiment we grouped items with 60 % or more of their ratings as 1 or 2 in this category.

T_H : Target item with majority of their ratings at higher end of the rating scale fall into this category. In our experiment we grouped items with 60 % or more of their ratings as 4 or 5 in this category.

Once we know the category of the target item, appropriate strategy for assigning values to filler items is used to construct attack profiles. We define below the two strategies.

4.1 Strategy L

This strategy is followed when the target item falls in T_L category. As majority of the users have rated the target item at the lower end of the rating scale, to improve effectiveness of the attack, we need to create profiles that are similar to the users who have rated the target item with a lower value. So, to improve similarity, a randomly selected filler item is assigned the average rating given to the filler item by those users who have rated the target item at the lower scale.

Let U_A be the set of all users who have rated the randomly selected filler item I_F. Let U_L be the set of all users who have rated the target item at a lower scale of rating and have also rated the randomly selected filler item. So, in this strategy, filler item I_F is assigned the average rating given to it by the set of users U_L. This approach differs from average attack in the way filler items are assigned values, as in an average attack item I_F was assigned the average rating given to it by the set of users U_A.

4.2 Strategy H

This strategy is followed when the target item falls in T_H category. As a large majority of the users have rated the target item at the higher end of the rating scale, to improve effectiveness of the attack, we need to create profiles that are similar to the users who have rated the target item with a higher value. So, to improve similarity, a randomly selected filler item is assigned the average rating given to the filler item by those users who have rated the target item at the higher scale.

Let U_A be the set of all users who have rated the randomly selected filler item I_F. Let U_H be the set of all users who have rated the target item at a higher scale of rating and have also rated the randomly selected filler item. So, in this strategy, filler item I_F is assigned the average rating given to it by the set of users U_H. This approach differs from average attack in the way filler items are assigned values, as in average attack item I_F was assigned the average rating given to it by the set of users U_A.

5 EXPERIMENTAL EVALUATION & DISCUSSION

We performed the experimental evaluation of our strategies on the publicly available MovieLens data set [9]. This is the most widely used dataset in recommender systems research. MovieLens consists of 100,000 ratings made by 943 users on 1682 movies. Each user in the data set has rated at least 20 movies and each movie has been rated at least once. A timestamp value is associated with each user, movie, and rating combination. The data set also contains information on the demographic detail (age, sex, occupation, and zip code) of each user and basic information (genre and release date) of each movie. The ratings are made in a scale of 1 to 5, where 5 indicate extreme likeness for an item and 1 dislike.

We evaluated effectiveness of both the strategies on average attack model. User based collaborative algorithm was used as the recommendation algorithm. For similarity calculation and prediction, equations 1 and 2 stated in section 3 were used. We used a neighborhood size of k = 20 for prediction calculation. Case amplification value of 10 was used while calculating correlation and only positive correlations values were considered for computing predictions.

To conduct our evaluation, we selected a sample 20 items. Out of the 20 items, 10 items belonged to T_L category while remaining 10 items to T_H category. All the 20 items were selected randomly from a larger set of items belonging to each category. We also randomly selected a sample of 50 target users. Target users selected were those who have never rated any of the 20 test items. Each of the target items was attacked individually and the prediction shift was calculated by averaging the prediction shift observed for each user. The final prediction shift for the attack is the average prediction over all items used in the test. Equation 4 was used to calculate the metric.

All experiments were conducted for "S*ize of attack*" values 1%, 3%, 6%, 12%, and 15%. "S*ize of attack*" represents number of attack profiles added as a percentage of pre-attack profiles. 1% "S*ize of attack*" implies 10 attack profiles were added to a system of 1000 genuine users. On the basis of the results reported in [1] that best results are reported when a filler size of 3% used in an average attack, we used a filler size of 3% for all our tests i.e. 3 % of 1682 items which is approximately 50 filler items. We used three strategies for filler items: Strategy L, Strategy H and average attack. . For average attack, filler item strategy used was the same as in an average attack i.e. the mean of the filler item was assigned to it. Category T_L, Category T_H, Strategy L, and strategy H were implemented the way explained earlier in section 4.

Figure 2 shows the prediction shift values of three attacks (Strategy L, Strategy H and average attack) for items belonging to T_L category. From the graph it's obvious that for items in T_L category, Strategy L is the most effective attack strategy at lower values of attack size. Similarly, figure 3 shows the prediction shift values for the three attack strategies for items belonging to T_H category. From the graph it can be concluded that for items belonging to T_H category, Strategy H performs much better than other attack strategies at lower values of attack size. Figure 4, shows the prediction shift of the three strategies for a set of 20 randomly selected target items. In figure 4 we observe that average attack performs better than Strategy L and Strategy H, this observation further strengthens the importance of tailoring of attack strategy on the basis of target item rating distribution.

Experimental results clearly show that our approach of selecting a strategy based on target item rating distribution outperforms the best available attack model i.e. average model. One drawback of our attack strategies is its high knowledge cost. However, automated software agents can help diminish the cost. One approach that can be used to decrease the cost is to use a subset of users while assigning values to filler items. For example, in Strategy H instead of assigning a filler item I_F the average rating given to it by the set of users U_H, we assign I_F the average rating

given to it by a subset of 5 randomly selected users from U_H. In future work we plan to experimentally verify the effectiveness of this cost reduction approach.

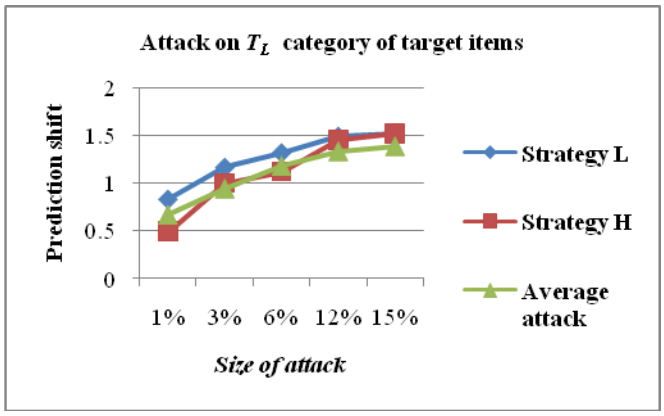


Figure 2: Attack on T_L category of items

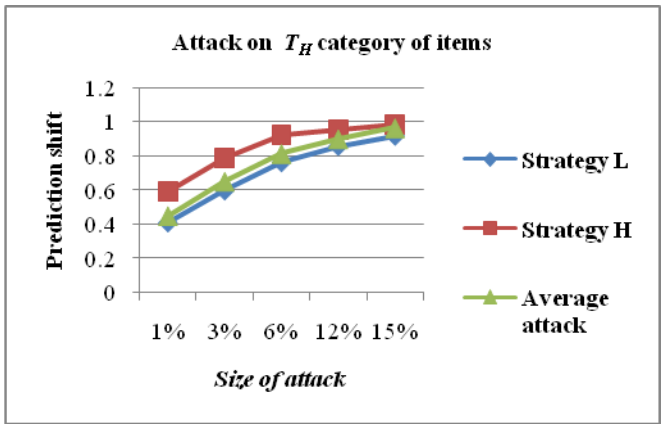


Figure 3: Attack on T_H category of items

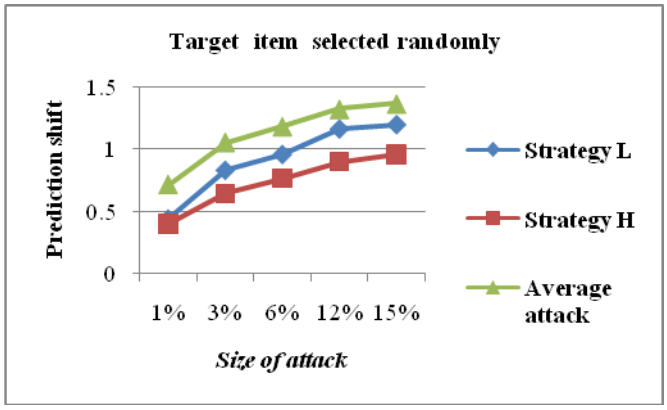


Figure 4: Attack on randomly selected target items

6 CONCLUSION

This paper examines different strategies that can increase the effectiveness of average attack model. Our approach provides a new perspective of using target item rating distribution to tailor different strategies for filler items. Through experiments we show that implementing the right filler strategy based on target item rating distribution, results in substantial improvement over the baseline average attack. While our attack strategies have focused only on assigning the right values to filler items for improving attacks effectiveness, we believe proper selection of filler items can also improve the effectiveness of an attack. In future, we plan to examine the filler items strategies for other attack models, and also create algorithms to improve robustness and stability of recommender systems against shilling attacks.

7 REFERENCES

[1] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, 'Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness', *ACM Transactions on Internet Technology*, **7,** 23:1-38, (2007).

[2] B. Mehta, T. Hofmann, and W. Nejdl, 'Robust Collaborative Filtering', *Proceedings of the 2007 ACM conference on Recommender systems*, 49-56,(2007).

[3] J.J.Sandvig, B. Mobasher, and R. Burke, **'**Robustness of collaborative recommendation based on association rule mining'**,** *Proceedings of the 2007 ACM conference on Recommender systems*, 105-112, (2007).

[4] S. Lam, and J. Riedl, 'Shilling recommender systems for fun and profit', *Proceedings of the 13th International WWW Conference*, (2004).

[5] R. Burke, B. Mobasher, and R. Bhaumik, 'Limited Knowledge Shilling Attacks in Collaborative Filtering Systems', *Proceedings of workshop on Intelligent Techniques for Web Personalization*,*(*2005).

[6] J.Konstan, B., Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl ,'GroupLens: Applying collaborative filtering to Usenet news'. *Communications of the ACM*, **40**, 3, 77–87, (1997).

[7] J.Herlocker, J. Konstan, A. Borchers, and J. Riedl, 'An algorithm framework for performing collaborative filtering', *Proceeding of SIGIR, ACM*, 77-87, (1999).

[8] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, 'Effective Attack Models for Shilling Item-Based Collaborative Filtering Systems', *Proceedings of the 2005 WebKDD Workshop*,(2005).

[9] MovieLens data set,*www.grouplens.org*

On the Scalability of Graph Kernels Applied to Collaborative Recommenders

Jérôme Kunegis, Andreas Lommatzsch, Şahin Albayrak [1] and **Christian Bauckhage** [2]
{kunegis, andreas, sahin}@dai-lab.de, christian.bauckhage@telekom.de

Abstract. We study the scalability of several recent graph kernel-based collaborative recommendation algorithms. We compare the performance of several graph kernel-based recommendation algorithms, focussing on runtime and recommendation accuracy with respect to the reduced rank of the subspace. We inspect the exponential and Laplacian exponential kernels, the resistance distance kernel, the regularized Laplacian kernel, and the stochastic diffusion kernel. Furthermore, we introduce new variants of kernels based on the graph Laplacian which, in contrast to existing kernels, also allow negative edge weights and thus negative ratings. We perform an evaluation on the Netflix Prize rating corpus on prediction and recommendation tasks, showing that dimensionality reduction not only makes prediction faster, but sometimes also more accurate.

1 Introduction

In information retrieval, the task of filtering and recommending items to users is often done in a content-based manner [1]. Collaborative filtering, by contrast, bases item rankings on ratings collected from users [13].

A collaborative filtering system therefore usually consists of a database of users, items (such as text documents or movies), and a collection of ratings that users have assigned to these items.

Collaborative rating databases are modeled as bipartite graphs, where users and items are represented by means of nodes, and the ratings by means of labeled edges. Recently, kernels have been used to tackle the task of recommendation. Graph kernels in particular are based on the rating database's underlying bipartite graph model. Traditionally, this type of kernel only copes with positively rated links. For a recommendation application this is a drawback since users may want to express their dislike and therefore may also assign negative ratings. In this paper, we therefore introduce kernels for collaborative rating prediction which also cope with negative ratings.

In order to scale to the size of current rating databases, collaborative recommendation algorithms must be able to process very large rating corpora. This necessitates some form of dimensionality reduction. Of course, dimensionality reduction will influence prediction accuracy and runtime. In this paper, we study the performance of collaborative recommendation algorithms in combination with dimensionality reduction.

Our contributions are therefore as follows: First, we study the prediction accuracy of graph kernels for recommendation on signed data, as opposed to unsigned data. Second, we introduce signed variants of all graph kernels that are based on the graph Laplacian. Third, we evaluate the influence of dimensionality reduction on the recommendation accuracy and runtime performance for both training and prediction.

Next, we first review related work; we then introduce terms and definitions used in collaborative filtering. The different graph kernels we consider in this paper are presented in the third section and the fourth section describes how to apply dimensionality reduction to each of these kernels. Afterwards, we discuss the application of graph kernels to recommendation algorithms. Finally we present and discuss our experimental evaluation.

2 Related Work

For a general description of collaborative rating prediction, we refer the reader to [2]. Graph kernels have been used for collaborative recommendations in [3]. The authors of [6] apply kernels to link analysis. In this study, the underlying graph is weighted by only positive values.

The matrix exponential has been used outside of computer science for sociometric analysis [8], and has been rediscovered for collaborative filtering recently [12].

Dimensionality reduction for collaborative filtering is discussed in [14]. This previous work however only apply dimensionality reduction to the adjacency matrix of the bipartite rating graph, without using graph kernels. This reference gives values for the optimal reduced rank between 5 and 15. We will refer to this method as *simple dimensionality reduction*.

As computation of dense kernels is too expensive in the case of large rating databases, sparse methods have to be employed, which in the general case scale much better than dense methods [4].

3 Definitions

Throughout this paper, we assume U to be the set of users where $|U| = m$, likewise we assume I to be the set of items where $|I| = n$. The rating database is represented by means of a sparse matrix $R \in \mathbb{R}^{m \times n}$ whose number of nonzero elements is denoted by r.

[1] DAI-Labor, Technische Universität Berlin, Germany
[2] Telekom Laboratories, Berlin, Germany

The sparse matrix R corresponds to a weighted bipartite rating graph $G = (V, E, W)$ where $V = U \cup I$ is the set of vertices and E the set of edges. For every rating $R_{ui} \neq 0$, E contains an edge (u, i), the corresponding edge weights are given by $W_{ui} = R_{ui}$. Since we want to consider positive and negative ratings, we do not restrict W to nonnegative values.

The adjacency matrix $A \in \mathbb{R}^{(m+n)\times(m+n)}$ of G is given by $A = \begin{bmatrix} & R \\ R^T & \end{bmatrix}$ and we also define a diagonal degree matrix D whose entries D_{ii} contain the sum of adjacent edge weights of the corresponding node i. The graph Laplacian is then given by $L = D - A$.

For our extension to the case of negative ratings, we also define the *absolute degree matrix* $\bar{D}$, which is a diagonal matrix, too, and contains the sum of *absolute edge weights* for each node. Analoguously, we define $\bar{L} = \bar{D} - A$.

4 Graph Kernels

In this section we present the kernels evaluated in this paper. For all except one of these, we introduce new variants in order to be able to deal with signed rating data. We also describe dimensionality reduction in itself, which is not a kernel but can be used in place of a kernel.

All kernels are based on the following observation: If $K \in \mathbb{R}^{V\times V}$ is a symmetric matrix, then the following function d is a dissimilarity matrix: $d(i, j) = K_{ii} + K_{jj} - K_{ij} - K_{ji}$. Its square root is an Euclidean metric in the space spanned by the eigenvectors of K, and its inverse is a similarity measure between any two nodes [9]. In the next paragraphs, we give expressions for the matrix K for the various kernels.

Rank reduced adjacency matrix kernel. The adjacency matrix A itself may be interpreted as a kernel, because if two nodes are similar (positively or negatively) they will be connected by an edge. However, in order to derive predictions for items that were not rated so far, this kernel is only useful after a rank reduction has been applied. We simply set

$$K_{\mathrm{DIM}} = A$$

Exponential diffusion kernel. [10] defines the exponential diffusion kernel using the matrix exponential:

$$K_{\mathrm{EXP}} = \exp(\alpha A) = \textstyle\sum_{i=0}^{\infty} \frac{1}{i!}\alpha^i A^i$$

Since A^n contains the number of paths of length n between any two nodes, this kernel represents an average of path counts between nodes, weighted by the inverse factorial of path length. Therefore, longer connections are less influential than shorter ones.

Resistance distance kernel. This kernel is also called the commute time kernel. It results from interpreting the graph G as a network of electrical resistances with resistance values given by the edge weights W. Given a pair of nodes, the total resistance induced by the network is a distance given by the following kernel [16]:

$$K_{\mathrm{RES}} = L^+ = (D - A)^+$$

where L^+ denotes the Moore-Penrose pseudoinverse of the Laplacian matrix.

In order to also account for negative edge weights, we define a signed resistance distance kernel [11]

$$K_{\mathrm{RES-S}} = \bar{L}^+ = (\bar{D} - A)^+$$

where we apply the absolute degree and Laplacian matrix as defined in the previous section.

Stochastic diffusion kernel. This kernel is based on a stochastic diffusion process and hence only applies to positive data [10].

$$K_{\mathrm{STO}} = (1 - \alpha)(I - \alpha D^{-1}A)^{-1}$$

The parameter α denotes the probability in the diffusion process of following a graph edge instead of returning to the starting node. The matrix $D^{-1}A$ is a stochastic diffusion matrix, and this kernel is therefore designed for positive data.

As with the resistance distance kernel, this kernel is a new variant of the stochastic diffusion kernel which also takes into account negative ratings

$$K_{\mathrm{STO-S}} = (1 - \alpha)(I - \alpha \bar{D}^{-1}A)^{-1}$$

Laplacian exponential diffusion kernel. The Laplacian exponential diffusion kernel applies the matrix exponential to the Laplacian [3, 15]:

$$K_{\mathrm{LEX}} = \exp(-\alpha L) = \exp(-\alpha(D + A))$$

We found this kernel to perform poorly in practice. However, a signed version leads to acceptable results.

$$K_{\mathrm{LEX-S}} = \exp(-\alpha \bar{L}) = \exp(-\alpha(\bar{D} + A))$$

In the evaluation, we will only use the signed Laplacian exponential diffusion kernel.

Regularized Laplacian kernel. This kernel is a generalization of the random forest kernel [3].

$$K_{\mathrm{REL}} = (I + \alpha(\gamma D - A))^{-1}$$

The random forest kernel itself is based on random forest models [3]. It arises in the calculation of weighted counts of forests of G in which two nodes belong to the same tree.

$$K_{\mathrm{FOR}} = (I + L)^{-1}$$

We do not show the random forest kernel in the evaluation because it performs similarly to the regularized Laplacian kernel. We also define a signed variant of the regularized Laplacian kernel:

$$K_{\mathrm{REL-S}} = (I + \alpha(\gamma \bar{D} - A))^{-1}$$

We only use the signed regularized Laplacian kernel for evaluation, as it performs much better than the unsigned variant.

All these kernels are based on matrix inversion or exponentiation and cannot be computed directly.

5 Dimensionality Reduction

Given the huge but sparse adjacency matrix A, any computation of graph kernels will require dimensionality reduction.

If $A = Q\Lambda Q^T$ denotes the eigenvalue decomposition of the symmetric matrix A, a rank-k approximation of A is given by $\tilde{A} = Q_k\Lambda_k Q_k^T$, where $k \ll m + n$ is the desired rank and Q_k and Λ_k denote the corresponding truncations of Q and Λ.

This kind of dimensionality reduction is also known as latent semantic analysis and is frequently used for projecting high-dimensional data into lower dimensional spaces.

In order to apply this reduction to a kernel K, we observe that all kernels which we presented in the previous section can be expressed as $K = Qf(\Lambda)Q^T$, where Q and Λ are given by the eigenvalue decomposition of the a linear combination of the matrices A and D. The function $f(\Lambda)$ depends on the individual characteristics of the kernel. Note that $f(\Lambda)$ can be computed efficiently because it only only has to be applied to the diagonal matrix Λ. In the kernels we consider in this paper, three different types of $f(\Lambda)$ occur: Matrix inversion, the Moore-Penrose pseudoinverse, and the matrix exponential.

The rank reduced kernels are then computed as in the following example: If $\alpha A = Q\Lambda Q^T$ then $\tilde{K}_{\mathrm{EXP}} = Q_k \exp(\Lambda_k)Q_k^T$.

We also note that the truncation mode depends on the operation performed. For inversion and pseudo-inversion, we must retain the eigenvalues closest to zero, excluding zero eigenvalues for the pseudoinverse. For the matrix exponential, we retain the biggest eigenvalues and corresponding eigenvectors. For simple dimensionality reduction, we retain the eigenvalues with biggest absolute value.

6 Recommendation and Prediction

In this section, we describe the basic rating prediction algorithm, and the rating prediction algorithms based on graph kernels.

A common preprocessing step in collaborative filtering is to normalize the rating data. Normalization can be user-based or item-based [5]. For user-based normalization, each user's nonzero ratings are scaled to zero mean and unit variance, but zero entries of A remain unchanged.

In our implementation, we use a hybrid of user-based and item-based normalization. Given a rating r, the normalized rating $\hat{r}$ is computed using the user's and item's mean rating and rating standard deviation:

$$\hat{r} = (2r - \mu_u - \mu_i)/(\sigma_u + \sigma_i) \quad (1)$$

Once a rating has been predicted based on the normalized rating matrix, it has to be scaled back to the user's original range of ratings by inverting Equation (1).

Given a user u and item i and ignoring normalization, the baseline user-based rating prediction algorithm [13] proceeds as follows:

1. Retrieve all ratings of item i according to other users.
2. Compute the average over these ratings, weighted by the correlation between the other users and user u.

To predict ratings using a kernel, the correlation step in this algorithm is replaced by computing the similarity measure induced by the kernel. Since collaborative filtering considers user-user or item-item similarities, we consider distinct kernels for the sets of users and items respectively.

Recommendation is implemented by predicting ratings for all possible items, and choosing the items with the highest rating prediction, in function of the number of items searched.

7 Evaluation

We use the Netflix Prize corpus of ratings[3] for evaluation. Out of the whole corpus, we use a subset of 3,216 users, 1,307 items and 57,507 ratings. The corresponding rating matrix is filled to 1.37%. The test and training samples were drawn at random from the complete rating set.

We measure the accuracy of rating prediction using the root mean squared error (RMSE) which is the square root of the average over all squared differences between the actual and the predicted rating. This precedure is standard in the collaborative filtering literature [2].

the accuracy of recommendation is given by the normalized discounted cumulated gain (nDCG), as defined in [7].

[3] `http://www.netflixprize.com/`

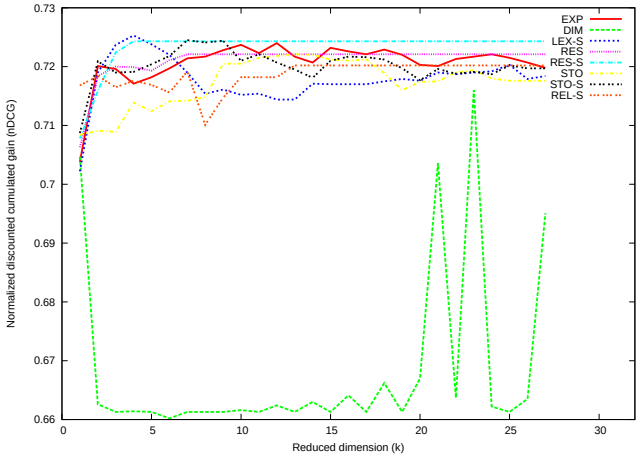


Figure 1. Comparison of recommendation accuracy in function of the reduced rank k for all kernels. This figure shows the normalized discounted cumulated gain (nDCG). Higher values denote better recommendation.

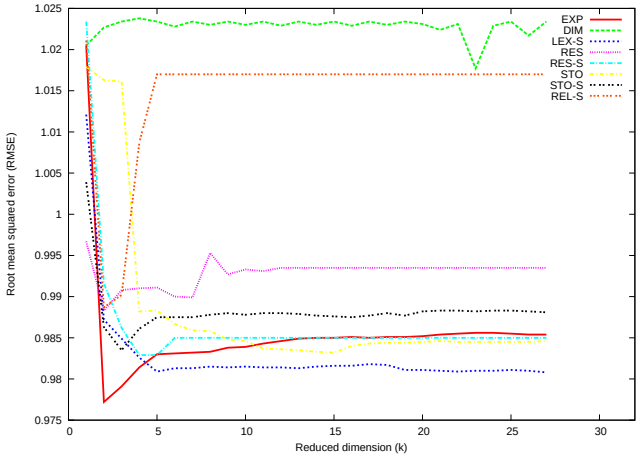


Figure 2. Comparison of rating prediction error in function of the reduced rank k for all kernels. This figure shows the root mean squared error (RMSE). Lower values denote more accurate rating prediction.

For testing the scalability of the different kernel methods, we varied the parameter k from 1 to 27 for each kernel. Figure 1 shows the nDCG in function of k for all kernels in the recommendation task. Higher nDCG values denote more accurate recommendations. Figure 2 shows the RMSE in function of k for all kernels in the prediction task. Lower RMSE values denote more precise predictions.

Asymptotic behavior. We observe two different patterns of asymptotic behavior. Some kernels attain their best performance asymptotically for big k, while others reach an optimum for a specific value of k. Table 1 summarizes these findings. We note that most kernels (EXP, REL-S, RES, RES-S, STO-S) show an inverted behavior on the recommendation task than on the prediction task. Although simple dimension-

ality reduction shows isolated peaks for specific values of k, there is no recognizable pattern for this kernel.

Table 1. Classification of graph kernels by their asymptotic behavior. The left column groups the kernels attaining their best performance at a specific, small value of k. The right column contains the kernels having asymptotic optimal behavior for big k. Kernels showing neither behavior are omitted.

	$k_{\text{best}} = k_0$	$k_{\text{best}} = +\infty$
Recommender	DIM	EXP, RES, RES-S STO, STO-S, REL-S
Prediction	EXP, RES, RES-S STO-S, REL-S	LEX, STO

Choice of k. Algorithms with asymptotically optimal performance reach their almost-optimum for $k = 5$. The other kernels peak between $k = 2$ and $k = 5$. Both observations suggest that a value $k > 5$ is not needed for this size of corpus.

Stability. At the task of rating prediciton, all kernels perform smoothly in function of k. At recommendation, only the resistance distance and regularized Laplacian kernels perform smoothly. The other kernels' performances vary much more with changing k. We must therefore recommend the resistance distance and regularized Laplacian kernels as their results are more predictable and consistent.

Good recommender but bad predictor. We observe that the regularized Laplacian kernel shows acceptable recommendation accuracy, but bad prediction accuracy except for a small peak at $k = 2, 3$. We interpret this performance as a correctly ranked prediction which however does not match the actual values.

Simple dimensionality reduction. Dimensionality reduction itself performs worse than all proper kernels as expected. Also, the accuracy of simple dimensionality reduction seems to oscillate between better performance for even k and worse performance for odd k. We explain this by the fact that the spectrum of A contains pairs of eigenvalues $\pm\lambda$ because the rating graph is bipartite. Apparently, using only one of these two eigenvalues and its respective eigenvector leads to lower accuracy.

Laplacian vs adjacency matrix. We observe that kernels based on the graph Laplacian perform better than kernels based on the adjacency matrix. The resistance distance kernel, which corresponds to the inverted Laplacian, is definitely better than simple dimensionality reduction, and Laplacian exponential kernel, while not more accurate than the exponential kernel, has more stable behavior for changing k.

Signed kernels better than unsigned. The signed variants all performed better than the unsigned counterparts. The near-exception are the stochastic diffusion kernels on the prediction task, where the unsigned variant is more accurate asymptotically. However, the overall peak is reached by the signed variant at $k = 3$.

Overall best kernel. For the choice of overall best kernel, we select the signed resistance distance and signed Laplacian exponential kernel. The exponential kernel comes close but has worse stability for varying k, making it difficult to recommend in practice.

8 Conclusion and Future Work

In this paper, we studied the prediction accuracy of collaborative recommender and rating prediction algorithms based on graph kernels. We considered eight different kernels, including three novel, signed variants.

We found that small reduced ranks are acceptable in most cases depending on the rating corpus and that kernels based on the graph Laplacian are usually better than kernels based on the adjacency matrix. Also, we showed that dimensionality reduction not only reduction the runtime but also makes collaborative recommenders more accurate. We also showed that most kernels can be used in the context of signed rating data, when new *signed* kernel variants are used.

REFERENCES

[1] Justin Basilico and Thomas Hofmann, 'Unifying collaborative and content-based filtering', in *Proc. Int. Conf. on Machine learning*, p. 9. ACM Press, (2004).

[2] John S. Breese, David Heckerman, and Carl Kadie, 'Empirical analysis of predictive algorithms for collaborative filtering', in *Proc. Conf. Uncertainty in Artificial Intelligence*, pp. 43–52, (1998).

[3] François Fouss, Luh Yen, Alain Pirotte, and Marco Saerens, 'An experimental investigation of graph kernels on a collaborative recommendation task', in *Proc. Int. Conf. on Data Mining*, pp. 863–868, (2006).

[4] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, October 1996.

[5] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl, 'An algorithmic framework for performing collaborative filtering', in *Proc. Int. Conf. on Research and Development in Information Retrieval*, pp. 230–237, (1999).

[6] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto, 'Application of kernels to link analysis', in *Proc. Int. Conf. on Knowledge Discovery in Data Mining*, pp. 586–592, (2005).

[7] Kalervo Järvelin and Jaana Kekäläinen, 'Cumulated gain-based evaluation of ir techniques', *ACM Trans. Inf. Syst.*, **20**(4), 422–446, (2002).

[8] Leo Katz, 'A new status index derived from sociometric analysis', *Psychometrika*, **18**(1), 39–43, (March 1953).

[9] D. J. Klein and M. Randić, 'Resistance distance', *Journal of Mathematical Chemistry*, **12**(1), 81–95, (1993).

[10] R. Kondor and J. Lafferty, 'Diffusion kernels on graphs and other discrete structures', in *Proc. Int. Conf. on Machine Learning*, pp. 315–322, (2002).

[11] Jérôme Kunegis and Stephan Schmidt, 'Collaborative filtering using electrical resistance network models with negative edges', in *Proc. Industrial Conf. on Data Mining*, pp. 269–282. Springer-Verlag, (2007).

[12] Joel C. Miller, Gregory Rae, Fred Schaefer, Lesley A. Ward, Thomas LoFaro, and Ayman Farahat, 'Modifications of kleinberg's hits algorithm using matrix exponentiation and web log records', in *Proc. Int. Conf. on Research and Development in Information Retrieval*, pp. 444–445, (2001).

[13] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl, 'GroupLens: An Open Architecture for Collaborative Filtering of Netnews', in *Proc. Conf. on Computer Supported Cooperative Work*, pp. 175–186, (1994).

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, 'Incremental svd-based algorithms for highly scalable recommender systems', in *Proc. Int. Conf. on Computer and Information Technology*, pp. 399–404, (2002).

[15] A. Smola and R. Kondor, 'Kernels and regularization on graphs', in *Proc. Conf. on Learning Theory and Kernel Machines*, pp. 144–158, (2003).

[16] F. Y. Wu, 'Theory of resistor networks: The two-point resistance', *Journal of Physics A*, **37**, 6653–6673, (2004).

Empirical Evaluation of Ranking Trees on the Problem of Recommending Learning Algorithms

Carla Rebelo[1] and **Carlos Soares**[2] and **Joaquim Pinto da Costa**[3]

Abstract. This paper addresses the recommendation of Machine Learning/Data Mining (DM) algorithms, which is a less common recommendation task than others, such as the recommendation of web pages or products in web sites. This problem is relevant because many different algorithms are available today for DM tasks such as classification and regression, and the computational cost of executing all off them on a given dataset is very high. Therefore, the user must decide which algorithm(s) should be tested. The Metalearning approach to this problem consists of using information about the past performance of a set of learning algorithms on a set of datasets to induce a mapping between characteristics of those datasets to the relative performance of those algorithms. In this paper we empirically evaluate the ranking trees algorithm on some instances of this metalearning problem. Our results do not confirm previous results that showed that ranking trees outperformed the k-Nearest Neighbors algorithm.

1 Introduction

Currently, the most popular field of application of Recommender Systems is e-business [1]. However, there are other applications with a similar nature. One such application is the recommendation of algorithms in Machine Learning and Data Mining. The overall goal of this application is the recommendation of an algorithm to be used on a given dataset, from the set that is available to the user. One approach to this problem is Metalearning, which consists of using a learning algorithm to model the relation between the characteristics of datasets and the relative performance of a set of algorithms [4]. As argued in Section 2, recommendations should be in the form of a ranking (of a subset) of the available algorithms. Therefore, a metalearning approach to this problem can be regarded as a recommender system.

However, typical algorithms used in recommender systems cannot be used directly in the algorithm recommendation problem, as discussed in Section 3. Specific methods for learning rankings which are suitable for problems such as this one have been developed, including the *k*-NN ranking algorithm [4]. Another ranking algorithm that has been used for metalearning is *Ranking Trees*, an adaptation of the Top-Down Induction of Decision Trees (TDIDT) for ranking tasks [14] (Section 4). A ranking tree is a decision tree in which the leaves do not predict one from a set of possible class values but, instead, predict a ranking of the set of class values (e.g., algorithms in the case of metalearning).

[1] LIAAD-INESC Porto LA, Universidade do Porto, Portugal email: crebelo@liaad.up.pt
[2] Faculdade de Economia do Porto, Universidade do Porto, Portugal email: csoares@fep.up.pt
[3] Faculdade de Ciências, Universidade do Porto, Portugal email: jpcosta@fc.up.pt

However, the algorithm was tested on a single metalearning problem. The goal of this work is to empirically evaluate ranking trees on several metalearning problems. We present our results in Section 5.

2 Recommendation of Learning Algorithms

Many different learning algorithms are available to data analysts nowadays. For instance, decision trees, neural networks, linear discriminants, support vector machines among others can be used in classification problems. The goal of data analysts is to use the one that will obtain the best performance on the problem at hand. Given that the performance of learning algorithms varies for different datasets, data analysts must select carefully which algorithm to use for each problem, in order to obtain satisfactory results. Let us consider the estimates of the classification accuracy of four algorithms on two datasets, given in Table 1. For instance, the best algorithm on d_1, a_1, obtains a classification accuracy of 90%. If the user executes a_3 rather than a_1, accuracy decreases by 8%, which is probably significant. If a_2 is executed, the accuracy obtained is further reduced to 61%.

Table 1. Accuracy of four learning algorithms on two classification problems.

	a_1	a_2	a_3	a_4
d_1	90%	61%	82%	55%
d_2	84%	86%	60%	79%

Selecting the algorithm by trying out all alternatives is generally not a viable option, as explained in [14]:

> In many cases, running an algorithm on a given task can be time consuming, especially when complex tasks are involved. It is therefore desirable to be able to predict the performance of a given algorithm on a given task from description and without actually running the algorithm.

The learning approach to the problem of algorithm recommendation consists of using a learning algorithm to model the relation between the characteristics of learning problems (e.g., application domain, number of examples, proportion of symbolic attributes) and the relative performance of a set of algorithms [4]. We refer to this approach as *metalearning* because we are learning about the performance of learning algorithms.

Many metalearning approaches to the problem of algorithm recommendation handle it as a supervised classification task. Thus, the recommendation provided to the user consists of a single algorithm. However, this is not the most adequate form of recommendation for

this problem because it does not provide any further guidance when the user is not satisfied with the results obtained with the recommended algorithm. The alternative of executing all the algorithms (which often have parameters that can be tuned) has a very high computational cost and is, thus, not a viable strategy. Nevertheless, it is often the case that the available computational resources are sufficient to run some of the available algorithms. If recommendation indicates the order in which the algorithms should be executed, then the user can execute as many as possible, thus increasing the probability that a satisfactory result is obtained. Therefore, the problem of metalearning is a good example of a recommendation problem that should be tackled as a ranking task.

3 Learning Rankings

The type of ranking problem that we are dealing with in this work is somewhat different from the problems that are typically addressed with recommender systems. In fact, it has more similarities with the problem of supervised classification. In classification we have a set of examples, characterized by attributes and each one is assigned to one of a set of classes. Given a new example, described by the values of the attributes, the objective in supervised classification is to predict the class it belongs to. In ranking the goal is to predict the order of the classes as applicable to each example. Thus, the input to a problem of learning rankings is a set of examples (or instances) as described by a set of attributes and with a known ranking of the classes (the *target ranking*). The goal is to obtain a model that, given a new example described by the same set of attributes, generates a ranking of all the classes (or items). One of the most important differences to the problems that are usually dealt with recommender systems is the number of items. Here, we deal with a few items (e.g., 10 or 20 algorithms), while recommender systems are applied to problems with a large number of items (e.g., hundreds or thousands of web pages).

In general, a ranking represents a *preference function* over a set of items [6]. Therefore, given a set of n items,

$$X = (X_1, X_2, ..., X_{n-1}, X_n) \tag{1}$$

we define a ranking as a vector,

$$R = (R(X_1), R(X_2), ..., R(X_{n-1}), R(X_n)) \tag{2}$$

where $R(X_i)$ (or R_i, for simplicity) is the rank of item X_i and the item with rank R_i is preferred to item with rank R_j if $R_i < R_j$ [11]. In a metalearning application, the examples are, for instance, classification or regression (base-level) datasets and the items are learning algorithms.

To induce a model, it is necessary to have a training dataset containing a description of a set of examples according to a set of m attributes, $(A_1, A_2, \ldots, A_m)$, and the corresponding target ranking, R (or a vector X with n scores, which can be converted to a ranking). In the algorithm recommendation problem, the target rankings of the meta-level dataset (which we will refer to as dataset) are obtained by estimating the performance of the algorithms, which is usually done by running them on the base-level datasets (i.e., the learning tasks which will be represented by the examples for our metalearning problem). The target rankings of the base-level datasets of Table 1 are given in Table 2.

The generalization ability of ranking methods, i.e., their ability to accurately predict the target rankings of new examples, can be estimated using the same strategies that are used for other learning problems: a sample of the training dataset is used to induce a ranking model; the model is used to predict the rankings of the remaining examples; the accuracy of the model is estimated by comparing the predicted rankings to the corresponding target rankings. Here we have used *Leave-one-out Cross Validation*, which consists of iteratively, for each example, computing the accuracy of the prediction made for the selected example using a model obtained on all the remaining examples [15].

Table 2. Target rankings representing the accuracy of four learning algorithms on the two classification problems of Table 1.

	a_1	a_2	a_3	a_4
d_1	1	3	2	4
d_2	2	1	4	3

We have used three measures of ranking accuracy: Spearman's Rank Correlation Coefficient, the Weighted Rank Correlation coefficient and the Log Ranking Accuracy, which are presented below.

3.1 Spearman Coefficient (SC)

Spearman's rank correlation coefficient, r_S, has been proposed in the early $20th$ century by Charles Spearman and is given by the expression:

$$r_S = 1 - \frac{6\sum_{i=1}^{n}(R(X_i) - R(Y_i))^2}{n^3 - n} \tag{3}$$

where X and Y are two sets of n values and $R(X_i)$ represents the rank of element i in the series X. Nothing is assumed about the distribution of values of the variables. The coefficient simply evaluates the monotonicity of two sets of values, i.e., if their variations are related. If they tend to increase or decrease together, the variables are positively correlated. However, if one tends to increase while the other decreases then they are negatively correlated. This is less restrictive than other coefficients, such as Pearson's because it does not assume that the relationship between the two variables is represented by a particular type of function [11].

The expression above is valid only if there are no ties (the same numerical value in two or more observations), although in the case of a small number of ties it can still be applied (using the average rank for the tied observations). If the number of ties is very large, then it is better to use the expression of Pearson's correlation coefficient of the two vectors of ranks, as this is an alternative way of finding the Spearman's coefficient in all situations.

$$r_S(X,Y) = \frac{\sum_{i=1}^{n}(R(X_i) - R(\bar{X}_i))(R(Y_i) - R(\bar{Y}_i))}{\sqrt{\sum_{i=1}^{n}(R(X_i) - R(\bar{X}_i))^2 \sum_{i=1}^{n}(R(Y_i) - R(\bar{Y}_i))^2}} \tag{4}$$

It is, however, computationally less efficient that the one above.

3.2 Weighted Rank Correlation (r_W)

Given a ranking of learning algorithms, it can be expected that the higher an algorithm is ranked, the higher the probability that it will be executed by the user. Similar scenarios are expected in other ranking applications. Generally, this is true in ranking problems where the prediction is only used as a recommendation. Therefore, the evaluation of ranking algorithms should assign greater importance to higher ranks. However, Spearman's coefficient treats all ranks equally. An alternative coefficient is the Weighted Rank Correlation Coefficient [10, 5]. Let $d_i^2 = (R(X_i) - R(Y_i))^2$ and

$$W_i^2 \quad = \quad d_i^2((n - R(X_i) + 1) + (n - R(Y_i) + 1)) \tag{5}$$

The first term of this product d_i^2 is the quadratic error of rank, exactly as in r_S, and represents the distance between $R(X_i)$ and $R(Y_i)$. The second term weighs the error of rank by the importance of the two ranks involved $R(X_i)$ and $R(Y_i)$. Based on these expressions, the Weighted Rank Correlation coefficient is defined as:

$$r_W(X,Y) = 1 - \frac{6\sum_{i=1}^{n} W_i^2}{n^4 + n^3 - n^2 - n} \quad (6)$$

A thorough study of this coefficient is presented in [10].

3.3 Log Ranking Accuracy (LRA)

Another weighted measure of accuracy, that gives even more importance to higher ranks than r_W is the Log Ranking Accuracy [11]:

$$r_{log}(X,Y) = 1 - 2 * \frac{6\sum_{i=1}^{n} log_{1+R(X_i)}(1 + R(X_i) - R(Y_i))^2)}{\sum_{i=1}^{n} log_{1+i}(1 + (i - (n - i + 1))^2)} \quad (7)$$

4 Ranking Trees

One of the advantages of tree-based models is how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background on learning algorithms. It is also possible to obtain information about the importance of the various attributes for the prediction depending on how close to the root they are used. The Top-Down Induction of Decision Trees (TDIDT) algorithm is commonly used for induction of decision trees [9]. It is a recursive partitioning algorithm that iteratively splits data into smaller subsets which are increasingly more homogeneous in terms of the target variable (Figure 1).

It starts by determining the split that optimizes a given splitting criterion. A split is a test on one of the attributes that divides the dataset into two disjoint subsets. For instance, given a numerical attribute A_2, a split could be $A_2 \geq 5$. One of the problems with the most simple version of the TDIDT algorithm is that it only stops when the nodes are pure, i.e., when the value of the target attribute is the same for all examples in the node. This usually leads the algorithm to overfit, i.e., to generate models that fit not only to the patterns in the data but also to the noise. One approach to address this problem is to introduce a stopping criterion in the algorithm that tests whether the best split is significantly improving the quality of the model. If not, the algorithm stops and returns a leaf node. A leaf node contains the prediction that will be made for new examples that fall into that node. This prediction is generated by a rule that solves potential conflicts in the set of training examples that are in the node. In classification, the prediction rule is usually the most frequent class among the training examples. If the stopping criterion is not verified, then the algorithm is executed recursively for the subsets of the data obtained based on the best split.

An adaptation of the TDIDT algorithm for the problem of learning rankings has recently been proposed [14], called *Ranking Trees*. This algorithm is based on the Clustering Trees algorithm [2]. The adaptation of this algorithm for ranking involves a few issues, including the splitting criterion, the stopping criterion and the prediction rule. Before discussing these issues, we note that a training example in the problem of learning rankings is described using a set of m attributes $(A_1, A_2, \ldots, A_m)$ and is associated with a ranking $(R_1, R_2, \ldots, R_n)$ as defined in Section 3.

Input: D
Output: Tree model
BestSplit ← Test of the attributes that optimizes the *splitting criterion* ;
if *stopping criterion is TRUE* **then**
  Determine the leaf prediction based on the target values of the examples in D;
  Return a leaf node with the corresponding prediction obtained with the *prediction rule*;
else
  LeftSubtree ← TDIDT($D_{BestSplit}$) ;
  RightSubtree ← TDIDT($D_{BestSplit}$) ;
end
return *(BestSplit, LeftSubtree, RightSubtree)*;

Figure 1. TDIDT algorithm

4.1 Splitting Criterion

The splitting criterion is a measure that quantifies the quality of a given partition of the data. It is usually applied to all the possible splits of the data that can be made based on individual tests of the attributes.

In *Ranking Trees* the goal is to obtain leaf nodes that contain examples with target rankings as similar between themselves as possible. To assess the similarity between the rankings of a set of training examples, we compute the mean correlation between them, using Spearman's correlation coefficient (Section 3.1). The quality of the split is given by the weighted mean correlation of the values obtained for the subsets, where the weight is given by the number of examples in each subset.

Table 3. Illustration of the splitting criterion

Attribute	Condition: True		Condition: False	
	values	rank corr.	values	rank corr.
A_1	a	0.3	b, c	-0.2
	b	0.2	a, c	0.1
	c	0.5	a, b	0.2
A_2	< 5	-0.1	≥ 5	0.1

The splitting criterion of ranking trees is illustrated both for nominal and numerical attributes in Table 3. The nominal attribute A_1 has three values (a, b and c). Therefore, three splits are possible. For the numerical attribute A_2, a split can be made in between every consecutive value. In this case, the best split is $A_1 = c$, with a mean correlation of 0.5 for the training examples that verify the test and a mean correlation of 0.2 for the remaining, i.e., the training examples for which $A_1 = a$ or $A_1 = b$.

4.2 Stopping Criterion

The stopping criterion is used to determine if it is worthwhile to make a split or if there is a danger of overfitting. In the original implementation of *Ranking Trees* the criterion is not described. Here we define that a split should only be made if the similarity between examples in the subsets should increase significantly. Let S_{parent} be the similarity between the examples in the parent node, D, and S_{split} the weighted mean similarity in the subsets obtained with the best split.

The stopping criterion is defined as follows:

$$(1 + S_{parent}) \geq \gamma(1 + S_{split}) \quad (8)$$

Note that the significance of the increase in similarity is controlled by the parameter γ

4.3 Prediction Rule

The prediction rule is a method to generate a prediction from the (possibly conflicting) target values of the training examples in a leaf node. In *Ranking Trees*, the method that is used to aggregate the rankings that are in the leaves is based on the mean ranks of the items in the training examples that fall into the corresponding leaf. Table 4 illustrates the prediction rule used in this work.

Table 4. Illustration of the prediction rule, where e_i represents training example i

	R_1	R_2	R_3	R_4
e_1	1	3	2	4
e_2	2	1	4	3
Predicted	1	2	3	4

5 Experimental Results

We empirically tested the *Ranking Trees* algorithm on some ranking problems obtained from metalearning applications. For comparison purposes, the k-Nearest Neighbors (KNN) algorithm, which was previously applied on the same problems was also implemented [4]. Based on the results reported in that work, we used a single neighbor ($k = 1$). Additionally, we compared the results with a simple baseline, the *default ranking*, which is the mean ranking over all training examples [4]. The code for all the examples in this paper has been written in R (www.r-project.org). We also investigate the effect of varying the value of the γ parameter of the stopping criterion.

The performance of the methods was estimated using leave-one-out because of the small size of the datasets. Both algorithms were evaluated using the three ranking accuracy measures described earlier. However, given that the results obtained with Spearman's correlation coefficient and the Weighted Rank coefficient are similar, we only present the former.

5.1 Datasets

The data provided to the algorithm for learning rankings consists of two matrices. The first one contains a set of attributes that describe the dataset, referred to as *metafeatures*. In the case of metalearning, it often contains variables that represent general and statistical properties of the datasets, such as number of examples and mean correlation between numerical attributes. The second matrix contains the target rankings, based on the performance of the algorithms on the datasets.

We used the following meta-learning problems in our experiments:

Classification these data represent the performance of ten algorithms on a set of 57 classification tasks (base-level datasets). Two sets of metafeatures are used to describe base-level datasets. The first one is a small set with which KNN has been reported to achieve good results [4]. The second is a larger set, containing metafeatures that were used in the StatLog project [7, 3].

Regression these data represent the performance of nine algorithms on a set of 42 regression tasks (datasets). Also in this case, two different sets of metafeatures were used. The first one is, again, a small set with which KNN has been reported to achieve good results [11]. The second is a larger set that has been been based on measures proposed in the MetaL project [8].

SVM these data represent the performance of different variants of the Support Vector Machines algorithm on the same 42 regression datasets as in the previous set [11]. Five different sets of SVM variants were considered: in three of them, the sets include the same set of 11 settings of the σ parameter, and variations are obtained by using different values of the ϵ parameter of SVM; the other two contain five and 21 settings of the σ parameter, respectively. Additionally, three sets of metafeatures were used in this metalearning problem. The first is the MetaL set of metafeatures for regressions described earlier. The second is a set of SVM-specific metafeatures [12] and the last is a combination of the two. In total, we had $5 * 3 = 15$ different (but related) SVM metalearning problems.

More information about these meta-datasets can be found in [11].

5.2 Varying the Value of γ

We started by investigating the effect of the stopping criterion parameter. We tested a few values of γ above and below 1. Note that values greater than 1 mean that a split can be accepted even if the child nodes are less homogeneous than the parent node. On these datasets, using values lower than 0.995 generates trees with a single or very few nodes. This happens for values that are very close to 0.995, such as 0.95. When values larger than 0.995 are used, then the algorithm quickly overfits, generating trees with almost as many leaf nodes as examples. Reasonable results were obtained with values that are quite close to 0.995, as shown in Figure 2. This indicates that $\gamma = 0.995$ is a good choice for the stopping criterion parameter.

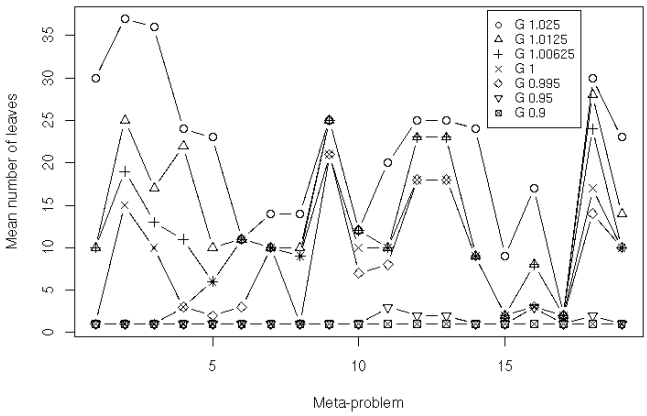


Figure 2. Variation of the number of leaves with the value of γ. The values presented are means over all examples (i.e., base-level datasets) in the metalearning dataset.

5.3 Comparing Ranking Trees with Other Methods

Comparing ranking trees with the KNN algorithm in terms of Spearman's coefficient (SC), we observe that the latter generally obtains

better results (Figure 3), except in the regression problem and in the SVM problems using the combination of the two types of metafeatures. The results of ranking trees may be explained with the small size of the dataset, which makes the induction of a model with good generalization ability very hard. This is also supported by previous results, in which the use of KNN on these problems with larger values of k leads to worse results [11]. On the other hand, these results are somewhat contradictory with a previous comparison between ranking trees and KNN, in which better accuracy is reported for the former algorithm [14]. This difference may be explained by the different experimental setup that is used. However, we note that a larger number of datasets was considered here.

Comparing ranking trees with the default ranking indicates that the former method is not able to predict the ranking of algorithms on new datasets very accurately. However, the observation for KNN is not very different except on a few cases. We note that the metalearning is a very difficult problem, especially taking into account that the datasets are small.

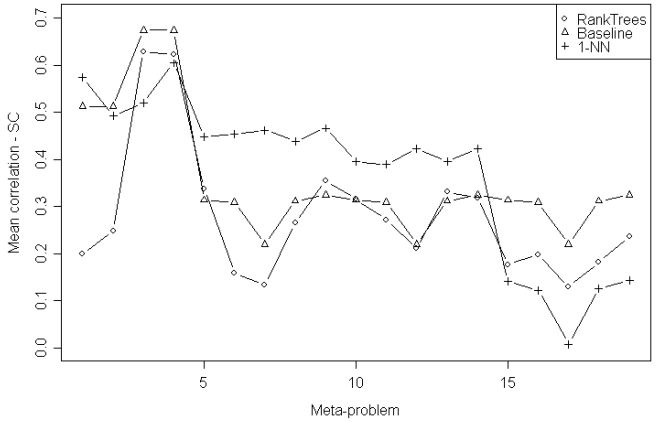


Figure 3. Comparison of the ranking accuracy measured with the Spearman's coefficient, obtained by Ranking Trees ($\gamma = 0.995$), the KNN algorithm and the default ranking baseline. The first four points represent the classification and regression datasets, respectively (two different sets of metafeatures each). Each of the following three sets of five points, represents the combination of the three sets of metafeatures with the five SVM problems.

Similar observations can be made from the results in terms of the LRA measure (Figure 4). However, according to this measure, the curves of the three methods seem to be closer, which indicates that the difference between the methods is smaller when higher importance is given to the top ranks.

6 Conclusions

In this paper, we address a somewhat uncommon recommendation problem, which is the recommendation of learning algorithms. We follow a metalearning approach, in which a learning algorithm is used to generate models that relate the characteristics of a (base-level) dataset to the performance of the algorithms. The goal of this work is to empirically evaluate ranking trees on some metalearning problems.

We started by choosing a value for the stopping criterion of ranking trees. Our choice of $\gamma = 0.995$ was based on the number of nodes generated. Smaller values of γ will generally lead to trees with a single node. On the other hand, for $\gamma > 1$ the number of leaves is close

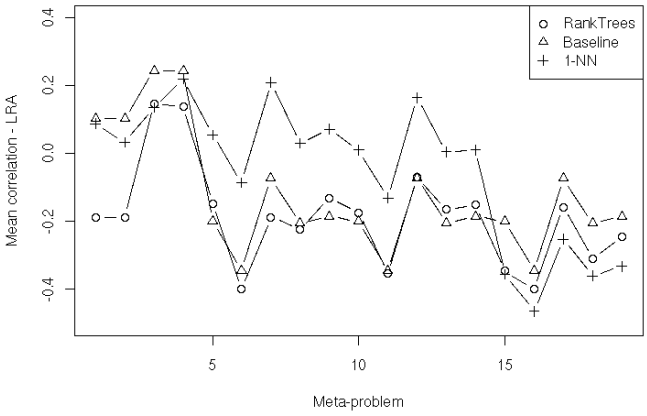


Figure 4. Comparison of the ranking accuracy measured with the Log Ranking Accuracy measure, obtained by Ranking Trees ($\gamma = 0.995$), the KNN algorithm and the default ranking baseline. The order of the metalearning problems is explained in Figure 3.

to the number of examples of the datasets, indicating that there is overfitting.

We compared ranking trees with the k-Nearest Neighbors (KNN) algorithm, which was previously applied on the same problems, and a baseline method. In our experiments, ranking trees did not generally outperform KNN. These results are in contradiction with earlier results [14], which may be explained with the different experimental setup used in those experiments. Additionally, we note that the difficulty of obtaining a ranking tree with a good generalization ability may be explained by the small number of examples in the datasets. It is, thus, necessary to test the methods on larger datasets.

Evaluating the results solely in terms of ranking accuracy is insufficient because, from the point of view of the user of the algorithm recommendation model, the goal is to obtain a model with a performance (value) as good as possible on his/her dataset with as less computational effort (cost) as possible. Therefore, it is necessary to evaluate the results in terms of the cost/benefit compromise that can be obtained by executing a varying number of algorithms [4].

Concerning the algorithm presented here, we plan to evaluate alternative splitting criteria, prediction rules and stopping criteria. In terms of the splitting criterion, we will test measures of similarity that give more importance to the top ranks, such as the ones used to assess ranking accuracy in this work.

Finally, we plan to test the method on different ranking problems. In particular, we plan to carry out experiments on *meta-recommendation*, i.e., using ranking trees and other ranking algorithms on the problem of recommending recommender systems. A similar problem is addressed by the AWESOME system, which dynamically chooses recommender models for a website [13].

ACKNOWLEDGEMENTS

This work was partially supported by project *Rank!*, funded by Fundação para a Ciência e Tecnologia (PTDC/EIA/81178/2006).

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Trans. on Knowl. and Data Eng.*, **17**(6), 734–749, (2005).

[2] Hendrik Blockeel, Luc De Raedt, and Jan Ramon, 'Top-down induction of clustering trees', in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 55–63, San Francisco, CA, USA, (1998). Morgan Kaufmann Publishers Inc.

[3] P. Brazdil, J. Gama, and B. Henery, 'Characterizing the applicability of classification algorithms using meta-level learning', in *Proceedings of the European Conference on Machine Learning (ECML94)*, eds., F. Bergadano and L. de Raedt, pp. 83–102. Springer-Verlag, (1994).

[4] Pavel B. Brazdil, Carlos Soares, and Joaquim Pinto Da Costa, 'Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results', *Mach. Learn.*, **50**(3), 251–277, (2003).

[5] J. Pinto da Costa and L. Roque, 'Limit distribution for the weighted rank correlation coefficient, r_w', *REVSTAT - Statistical Journal*, **4**(3), (2006).

[6] Johannes Fürnkranz and Eyke Hüllermeier, 'Preference learning', *Künstliche Intelligenz*, **19**(1), 60–61, (2005).

[7] R.J. Henery, 'Methods for comparison', in *Machine Learning, Neural and Statistical Classification*, eds., D. Michie, D.J. Spiegelhalter, and C.C. Taylor, chapter 7, 107–124, Ellis Horwood, (1994).

[8] C. Köpf, C. Tayllor, and J. Keller, 'Meta-analysis: From data characterization for meta-learning to meta-regression', in *Proceedings of the PKDD2000 Workshop on Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*, eds., P. Brazdil and A. Jorge, pp. 15–26, (2000).

[9] Thomas M. Mitchell, *Machine Learning*, McGraw-Hill Higher Education, 1997.

[10] Joaquim Pinto da Costa and Carlos Soares, 'A weighted rank measure of correlation', *Australian & New Zealand Journal of Statistics*, **47**(4), 515–529, (2005).

[11] Carlos Soares, *Learning Rankings of Learning Algorithms*, Ph.D. dissertation, Department of Computer Science, Faculty of Sciences, University of Porto, 2004. Supervisors: Pavel Brazdil and Joaquim Pinto da Costa.

[12] Carlos Soares and Pavel Brazdil, 'Selecting parameters of SVM using meta-learning and kernel matrix-based meta-features.', in *SAC*, pp. 564–568, (2006).

[13] Andreas Thor and Erhard Rahm, 'AWESOME: a data warehouse-based system for adaptive website recommendations', in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pp. 384–395. VLDB Endowment, (2004).

[14] Ljupco Todorovski, Hendrik Blockeel, and Saso Dzeroski, 'Ranking with predictive clustering trees', in *ECML '02: Proceedings of the 13th European Conference on Machine Learning*, pp. 444–455, London, UK, (2002). Springer-Verlag.

[15] Ian H. Witten and Eibe Frank, *Data mining: practical machine learning tools and techniques with Java implementations*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

Recommender Systems for Lifelong Learning inclusive scenarios

Olga C. Santos[1] **and Jesus G. Boticario**[1]

Abstract. Lifelong learning scenarios have particular differences in their need for personalised recommendations that make not possible reusing existing general approaches of recommender systems. Moreover, in the former scenarios, inclusive approaches are even more critical than in the later. This situation poses several challenges to be addressed by the new trend of recommender systems for lifelong learning. The paper describes those challenges and presents a hybrid proposal that combines different recommendation techniques.

1 INTRODUCTION

The Lifelong Learning (LLL) paradigm supports the idea that learning should occur throughout a person's lifetime [1]. This paradigm promotes a user-centred approach that removes social, physical and cognitive barriers, where dynamic support may foster attitudes and skills to improve the effectiveness of the learning process. In mediating this process, technology is playing an important role. In this sense, a dynamic support that recommends learners what to do to achieve their learning goals is desirable.

Traditionally, Intelligent Tutoring Systems (ITS) intend to provide direct customized instruction to students by finding the mismatches between the knowledge of the expert and the actions that reflect the assimilation of that knowledge by the student [2]. Their main limitations are: 1) ITS are specific of the domain for which they have been designed (since they have to be provided with the expert knowledge) and 2) it is unrealistic to think that it is possible to code in a system all the possible responses to cover the specific needs of each student at any situation of the course.

Our approach (defined at aLFanet project as the combination of design and run time adaptations throughout a pervasive usage of standards to allow for knowledge interoperability [3]) draws partly on the ITS ideas but leaves the knowledge information outside the system. Firstly, our approach benefits from recent educational specifications such as IMS Learning Design (IMS-LD) [4] that allows defining the instructional design of any type of course with any pedagogical theory. In this way, at design time it is possible to describe the course from its structural information (learning objectives, activities, services and resources and the relationships among them and for different user profiles). Secondly, the users' behaviour in the course is monitored at runtime to identify troublesome (i.e., lack of knowledge) and promising situations (i.e., high interest), and perform remediation or support actions by applying recommendation strategies. The design in terms of IMS-LD provides a detailed model of the course structure and context (mainly in terms of the learning objectives being worked) that is useful for selecting the appropriate recommendations. Furthermore, our approach is pedagogy neutral and has been applied in different learning scenarios [3].

[1] aDeNu Research Group, Artificial Intelligence Department, Computer Science School, UNED, c/Juan del Rosal, 16. 28040 Madrid, Spain {ocsantos,jgb}@dia.uned.es

In this paper, we present a hybrid approach to recommendative support in LLL inclusive scenarios which is based on a multi-agent architecture.

2 RS AND LLL INCLUSIVE SCENARIOS

Recommender systems (RS) support users in finding their way through the possibilities offered in web-based settings by pre-selecting information a user might be interested in. However, there are several distinct differences for recommendations to consumer products (e.g. music, news or movies) in contrast to recommendations to be provided to learners, which are translated into specific demands for these systems. In particular, recommender systems in LLL need to improve the learning effectiveness and do not depend just on the user's tastes. For instance, the preferred activity by a learner might not be pedagogically adequate [5].

Moreover, unlike for customer products, learners are not so motivated in rating each content they read or every activity they do. Implicit ways of getting this information have to be used. In turn, it is more likely that explicit information can be filled in advance, since it can be presented to the user as mandatory tasks to be done as part of the course requirements. For instance, learners can be asked 1) to fill in some information about their learning styles, their technological level and their accessibility preferences as part of the process for signing up in the learning management system (LMS) of the institution, and 2) to carry out some tests (e.g. on their previous knowledge on the course contents, their learning goals and interests) as part of the course activities.

Professors are also used to design activities and contents for different types of learners' needs, and can annotate them with metadata, provided that this extra workload is not disproportionate. This extra workload allows also for future reusability, which in the whole reduces the professor's workload.

Research works argue that memory-based recommendation techniques are the most adequate for RS in technology-enhanced learning settings [6], where LLL scenarios are deployed. In particular, collaborative filtering techniques [7-9] such as user-based (users that rated the same item similarly probably have the same taste), item-based (items rated similarly by users are probably similar) and stereotypes or demographics collaborative filtering (users with similar attributes are matched), and content-based techniques [10-12] such as case-base reasoning (if a user likes a certain item, she will probably like similar items in terms of the attributes they own), and attribute-based techniques (matching of item attributes to the user profile) are applicable. From those two types of techniques, several studies demonstrate the superiority of hybrid techniques (cascading, weighting, mixing, switching) [13-16]. In hybrid approaches collaborative- (or social-) based filtering (specially user-based and item-based techniques) are combined with content- (or information-) based filtering. If the former can be applied, learners benefit from the experience of others.

To cope with inclusiveness, dynamic support is also required to complement the universal design approach. Learners experience a disability when there is a mismatch between the learners' needs (or

preferences) and the education or learning experience delivered [17]. Thus, the system should take into account the preferences of the learners and the device capabilities when interacting with the system, and map them to specific content features. In particular, those related to the display, control and selection of learning content, so that alternative contents can be provided. For instance, small scale evaluations in ALPE project (eTen-2005-029328) showed that although adaptation of eLearning contents had been done at design time to suit the needs of users' with disabilities and adult learners, the provision of some dynamic support would have improved the learning performance and the user satisfaction [17].

3 RECOMMENDING STRATEGIES FOR LLL

In this section we present the key issues of our proposal (summarized in Fig. 1). In their definition, we have taken into account 1) the structural information of the course available in the IMS-LD (which allows to know at run time the course context in which the learner is at any time), 2) the user model information, 3) the particularities for LLL identified in the previous section, and 4) the characteristics of the different recommendation techniques.

1. Instead of top-down approaches from traditional knowledge-based RS where pedagogical knowledge is considered during the design of the system, a bottom-up approach can be used to provide pedagogical flexibility. This flexibility can be obtained by using domain independent techniques (which require no content analysis). The recommendation strategy decides internally the final recommendations, taking into account psycho-educational rules and learning strategies [6]. For instance, recommendations proposed by a collaborative-based technique can be checked to see if they fit with the learning style of the user. The course context and the user's features are required for this filtering process.
2. The professor should be able to specify generic recommendations to be applied in the course in terms of conditions on the user attributes and the course context.
3. Recommendations can be provided when no behaviour data exists. The 'cold start' problem (i.e. initial data set required) typical of collaborative approaches is avoided since it is expected that learners have provided basic information about their profile and that course contents are annotated by the course authors when designing the course. In this way, stereotypes and attribute-based techniques can be applied when new users enrol and new items are provided, respectively.

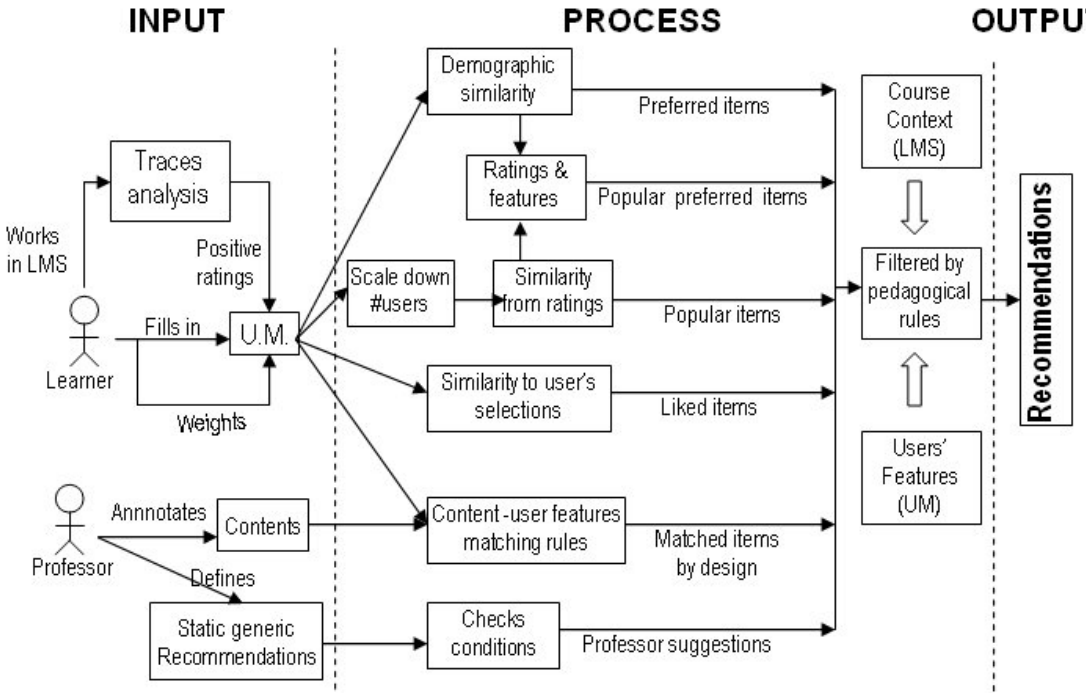


Fig. 1. Proposal for combining recommending techniques in LLL inclusive scenarios

4. Since demographic similarity and case-based reasoning are applied, not only popular items are recommended, but also those dependent on the user's preferences and her personal features. In this way, the sparsity problem (i.e. less recommendations for learners with unusual taste) is overcome.
5. Sparsity can also be reduced by combining past behaviour and users' features when computing the similarity measure [18].
6. Prioritized user-profile approaches can be used to implement more personalized recommendations by assigning different priority importance to each of the features of the user-profile [19]. This can be useful because the importance of each feature of a user-profile is not the same for all users.
7. Case-based reasoning can be applied when information about other users is not available. Moreover, case-based reasoning is limited to a pool of items that are similar to items the user knows. That is usually a limitation in RS for consumer products, but can turn into useful in LLL. For instance, when a learner wants to reach a higher competence level for a learning objective, it may require repeating some tasks till the competence is achieved [6].
8. Being recommended similar items to the ones preferred in the past is therefore useful, and applies to both situations, 'lack of knowledge' and 'high interest' [3].

9. Instead of requesting explicit ratings, positive ratings (those that have improved the learners' knowledge level) can be inferred from the user's interactions. For instance, if the learner's knowledge level in certain objective has increased after accessing an item that is associated to that objective.
10. When large amount of data are available to produce social recommendations, the learners can benefit from the experience of other successful learners. This allows learners to discover preferable items by serendipity.
11. If the number of potential similar users is large, data clustering to find densely populated groups of users with close similarities [5] or detecting active users [19] can be applied to scale down the candidate sets and also to guide collaborative filtering into a more focused space.
12. Inclusiveness can be supported provided that the user fills in her accessibility preferences and the professor annotates the accessibility features of items. By matching user preferences and items features, functional diversity issues are addressed.
13. Moreover, rule based reasoning can be applied with current specifications and standards. In particular, the IMS Learner Information Package combined with the IMS Metadata, and its extension for accessibility, the IMS-AccessForAll specification [4] (and the corresponding ISO/IEC JTC1 standard on Individualised Adaptability and Accessibility Learning, Education and Training -24751 under definition).
14. The quality of the recommendations improves over time. On the one hand, the user model attributes (filled in initially by the user) can be updated taking into account the behaviour of the user, producing a more accurate model along time which is used for demographic similarity and attributes matching. On the other hand, techniques based on user actions improve with the number of actions recorded (i.e more time, more actions).
15. Preliminary results suggest that recommendations can be grouped into different types, and some types are more effective than others for increasing the learning performance depending on the course situation [20].

4 THE MULTI-AGENT APPROACH FOR A HYBRID RECOMMENDER SYSTEM IN LLL

Since there are different recommending techniques, and each technique is more suitable in some situation than in others, we have proposed a multi-agent architecture where each agent can implement one of the recommendation techniques [3, 17]. The process works as follows (Fig. 2). The learner is using the LMS and before presenting some information to the user, the LMS asks a recommender service for available recommendations for the user. This RS service calls the multi-agent system where a coordinator agent activates the recommender agents. These recommender agents try to produce some recommendations adapted to the current user (needs and preferences stored in her model) and her context (passed in the request). From the recommendations produced by the recommender agents, the coordinator agent selects the appropriate ones and prioritizes them. The reason for this is that it may not be appropriate to show the whole list of available recommendations to the user, especially if the screen size of the device used is small or the user prefers a large font size to see the contents. Therefore, the potential recommendations for a learner in the current situation are prioritized to allow selecting the top ones with greater relevance. The number of selected recommendations depends on the device capabilities and the accessibility preferences.

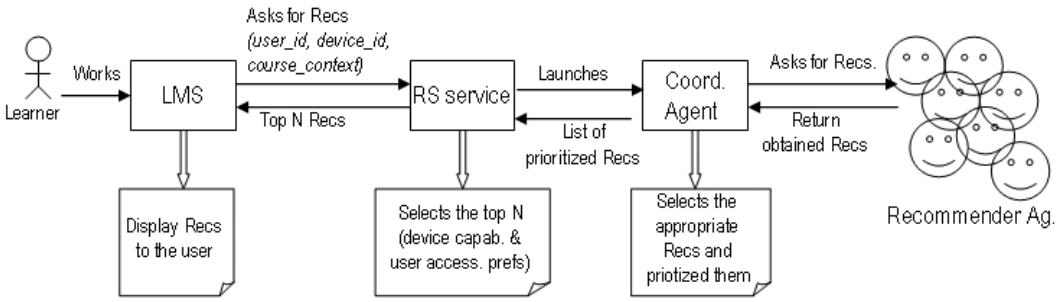


Fig. 2. Components involved in the generation of the recommendations for the LMS

The prioritising algorithm has to be tuned with the experience. However, we have started with the following criteria. First, it should select among contradictory recommendations. Second, it should give more priority to duplicated recommendations (i.e. equal recommendations generated by different agents). Next, it should consider the generic static recommendations. Finally, it should pay attention that recommendations are homogenously distributed by their type. Each of the above recommender agents can implement a different recommendation technique, as follows:

- Agent1: Select those static generic recommendations that match the conditions → Suitable recommendations from those defined by the professor for the course (Items suggested by the professor).
- Agent2: User-based collaborative filtering (user similarity from ratings) → Recommends positive rated items from similar learners (by ratings) (Popular items by alike users)
- Agent3: Item-based collaborative filtering (items rated similarly) → Recommends high-correlated items (sharing positive ratings from similar users) to user positive rated items (Items as popular as liked by user).
- Agent4: Demographic collaborative filtering (user similarity from attributes) → Recommends positive rated items from similar learners (by attributes) (User preferred items).
- Agent5: Combines user ratings and attributes to compute the similarity → Recommends positive rated items from similar learners (ratings and attributes) (Popular preferred items).
- Agent6 → Case-based reasoning (similar items to previous ones) → Recommends high-correlated items (in terms of attributes) to user liked items (Items similar to liked by user).
- Agent7: Attribute-based rules (IMS/ISO matching rules) → Matching items to user model (Items defined in the course design).

5 ON GOING WORKS

Current developments have focused on providing the infrastructure for the RS to allow offering recommendations in the LMS user interface. An initial prototype has been integrated in OpenACS/dotLRN LMS [21]. Recommendations consist on links to actions to perform in the LMS, such as offering help, suggesting to post a message in the forum, recommending the reading of a particular resource of the course.

So far, we have implemented the selection of the generic recommendations and carried out some initial experiments to find out if there are different factors that affect the learning performance and if they are dependent on the course situation. Preliminary results considering three situations (familiarization with the platform, familiarization with the operative approach of the course and the course itself) have shown that not all types of recommendations have the same relevance in all the situations [20]. In that experiment, different types of recommendations were given at each of the situations of the course. Comparing the performance with a group of students who were not given recommendations, the result was that the learners' performance has been increased. We are currently automatizing the process of results gathering to allow a bigger size for the sample and run experiments with a large population of learners, which will allow us better validate the proposal. In parallel, we are also implementing the combination of recommending techniques, as illustrated in Fig. 2.

6 CONCLUSIONS

Lifelong learning inclusive scenarios have particular differences in their need for personalized recommendations. This poses several challenges to be addressed by the new trend of recommender systems for LLL. In the paper we have analysed those challenges and proposed a way to combing memory-based recommender systems technologies to address these particularities. In particular, i) pedagogically guided instead of just by learners' taste, ii) not explicit ratings after each user action, iii) initial user profile available, iv) existing items annotation that match user profile, and v) matching of accessibility preferences and device capabilities.

The approach is based on pre-filled and pre-designed information at the beginning, when runtime information is not available and as interactions are produced, recommendations relay on more flexible and learnable techniques, which make learners benefit from the experience of other successful learners. Thus, if there is not enough information about items that were used in the past by similar learners (e.g. community driven information is not possible), sparsity and scalability problems can be avoided by switching to recommending similar items to the ones the learner preferred in the past (e.g. individual information). If users have filled in the attributes of their user model, then stereotypes (demographic collaborative filtering) can also be provided when the other collaborative-based techniques are not applicable and avoid the 'cold-start' problem. If explicit matching of item attributes to user profiles has been done, attribute based techniques can also avoid the 'cold-start' problem.

In implementing the approach, we have chosen a multi-agent architecture for its flexibility in combining different recommendation techniques. Preliminary experiments have been done applying generic recommendations (matching conditions defined by the professor) to an on-line course. A three layer evaluation process will be applied in future experiments.

REFERENCES

[1] Longworth, N. Lifelong learning in action – Transforming education in the 21st century. Kogan page (2003).

[2] Sleeman, D., Brown, J. S. Introduction: Intelligent Tutoring Systems. In D. Sleeman, J. S. Brown (Eds.), Intelligent Tutoring Systems (pp. 1-11). New York: Academic Press (1982).

[3] Boticario, J.G., Santos, O.C. An open IMS-based user modelling approach for developing adaptive learning management systems. Journal of Interactive Media in Education (JIME) September (2007)

[4] IMS Global Learning Consortium set of specifications. Available at: http://www.imsglobal.org/.

[5] Tang, T. and McCalla, G. Smart Recommendation for an Evolving E-Learning System. Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education (2003).

[6] Drachsler, H., Hummel, H. G. K., and Koper, R. Recommendations for learners are different: Applying memory-based recommender system techniques to lifelong learning. Proceedings of Workshop on Social Information Retrieval for Technology-Enhanced Learning, 2nd European Conference on Technology Enhanced Learning (2007).

[7] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. GroupLens: An open architecture for collaborative filtering of netnews. In Proc. of the ACM Conf. on Computer Supported Cooperative Work. ACM (1994) 175-186.

[8] Shardanand, U., Maes, P. Social information filtering: Algorithms for automating 'word of mount'. In Proc. SIGCHI Conf. on Human Factors in Computing Systems (1995). 210-217

[9] Breese, J.S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence. (1998) 43-52

[10] Lang, K. NewsWeeder: Learning to filter netnews. In Proc. of 12th Int'l Conf. on Machine Learning. (1995) 331-339.

[11] Pazzani, M.J., Muramatsu, J., and Billsus, D. Syskill & Webert: Identifying interesting web sites. In Proc. of the 13th Nat. Conf. on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conf. AAAI Press/MIT Press (1996) 54-61.

[12] Mooney, R.J., Bennett, P.N., and Roy, L. Book recommending using text categorization with extracted information. In Proc. of the AAAI Workshop on Recommender Systems , AAAI Press, p. 70-74 (1998).

[13] Setten, M. Supporting People In Finding Information: Hybrid Recommender Systems and Goal-Based Structuring. Telematica Instituut Fundamental Research Series, vol. 016, (2005)

[14] Burke, R. Hybrid recommender systems: survey and experiments. User Modeling and User-Adapted Interaction 12, 331-370 (2002).

[15] Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J. Combining collaborative filtering with personal agents for better recommendations. Proceedings AAAI99, 439-446 (1999).

[16] Melville, P., Mooney, R.J., Nagarajan, R. Content-boosted collaborative filtering for improved recommendations. 18th National Conf.on Artificial Intelligence, 187-192 (2002).

[17] Santos, O.C., Boticario, J.G. Recommendations for providing dynamic inclusive learning. Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies in press).

[18] Pazzani, M.J. A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review 13, 393-408 (1999).

[19] Rad, H.S, Lucas, C. On the Effectiveness of Prioritized User-Profile and Detecting Active Users in Collaborative Filtering Recommender Systems. WPRSIUI Workshop, ICDE 2007,(2007) 863 – 870.

[20] Santos, O.C., Boticario, J.G. Recommendation strategiews for promoting eLearning performance factors for all. Workshop Intelligent Techniques for Web Personalization & Recommender Systems in AAAI 2008 (in press).

[21] Santos, O.C., Raffenne, E, Granado, J. and Boticario, J.G. Dynamic support in OpenACS/dotLRN: Technological infrastructure for providing dynamic recommendations for all in open and standard-based LMS. Proceedings of the International Conference and Workshops on Community based environments (2008).

Help-Desk Agent Recommendation System Based on Three-Layered User Profile

YongBin Kang[1] and **Arkady Zaslavsky**[2] and **Shonali Krishnaswamy**[3]

Abstract. This paper proposes a novel approach for recommending a help-desk agent that may appropriately handle problems requested by clients. First, we identify a key problem of high tendency to depend on help-desk agent when dealing with a problem. To solve this problem, we present a three-layered user profile with a new concept of role information of users. Then, we emphasize how our new recommendation strategy is working based on the user profile, particularly using the individual/role information in the user profile. We finally demonstrate how our approach works with an example.

1 INTRODUCTION

The main role of help-desk agent (HDAgent) is to behave as a front-line interface to solve a service-call by utilizing accumulated knowledge and learned experience. A key problem of service management in help-desks lies in the high dependency on HDAgent when solving a service-call. The problem may cause two negative situations: *inconsistency* and *unreliability*. More specifically, retrieved solution may be inconsistent according to which a particular HDAgent handled the given service-call. In addition, according to different HDAgents, suggested solutions may be reliable or not. For example, if a novice HDAgent that may not have enough domain knowledge or experience solves a service call, it would not be guaranteed whether the solution from that HDAgent is appropriate or not. In both cases, these problems can be naturally linked to negative effects to the help-desk organization, such as the loss of the confidence and satisfaction of the clients [9].

A practical research area designed to address the issued problem can be found in application using Case-based Reasoning (CBR) approach [5, 15]. A key concern of CBR is how to design a *retrieval function* to generate possible solutions to a given service-call. However, one common weakness of CBR lies in that such function is usually derived by considering only limited two spaces, i.e., service-call and case space. Therefore, that issued problem (i.e., high dependency on HDAgent) still remains unsolved due to the ignorance of the inclusion of the HDAgent knowledge. To address the issue, this paper aims to present a new recommender system that recommends a HDAgent that may adequately handle a given service-call. In particular, we focus on designing a user profile that may represent enough information of users involved in help-desk domains and developing a new hybrid recommendation strategy based on that user profile.

[1] Caulfield School of IT, Monash University, Australia, email: yongbin.kang@infotech.monash.edu.au
[2] Caulfield School of IT, Monash University, Australia, email: arkady.zaslavsky@infotech.monash.edu.au
[3] Caulfield School of IT, Monash University, Australia, email: shonali.krishnaswamy@infotech.monash.edu.au

This paper is organized as follows. In the next section, we discuss the proposed user profile in detail. Section 3 presents the proposed recommendation process and Section 4 shows a demonstration. We review related work in Section 5, followed by the conclusion.

2 THREE-LAYERED USER PROFILE

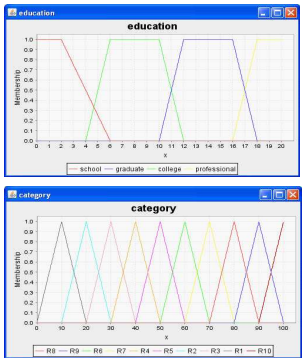
Our user profile is designed as an uniform profile that may represent enough knowledge of both the client and HDAgent. The basic intuition used here is that personalized information of the client and HDAgent can be uniquely decided on the following combination of three layers, which is motivated by the work [1]: *factual information*, *domain-specific problem features*, and *transactional information* of interactions between the clients and HDAgents. The main differences between the profile proposed in [1] and our profile are that the former model is mainly designed for capturing "purchasing behaviors of individuals" in e-commerce application, while the latter generalizes the former idea into the ITSM domain. In addition, the new concept of *role* is deployed into our user profile.

The first layer represents factual information that consists of four components, such as user identity, company description, role characteristics, and role category (see Table 1). The first two components represent domain-independent user information, which are initially generated by combining explicit user input and stereotypes. The *role characteristics* feature a set of important components of the *roles* (i.e., the task functions or positions of individual or a target group of users [16]) in the client company or help-desk organization. The main reason for defining the role characteristics is to identify the same or similar characteristics of the users, and to set a basis for utilizing both the individual and role information. Thus, it enables that user profiling might be individual or group-based [6]. These characteristics may be differently defined according to different industrial domains using various attributes carefully decided by the domain experts. In this work, some potential characteristics are identified that may address unique roles of the both client and HDAgent. This is consistent with our literature review in [9, 4, 13] as seen in Table 1. The goal of deriving *role category* is to provide better information to both the client and HDAgent when handling the service-call by classifying the role of the user based on the role characteristics. The role category is calculated using fuzzy logic which provides a human-like mechanism to imitate human decision that can be used to reason and aggregate strategy to reach optimal decisions [10]. Fuzzy logic has proved to be quite useful for developing many practical applications which need to enhance the capabilities of industrial automation due to its intelligent ability to formalize and manage inexact and vague information [13]. Having motivated the benefits of using fuzzy logic, the role characteristics are used for generating membership functions in

our fuzzy model, and our system assigns pre-defined role categories to users according to corresponding membership values using fuzzy logic. Each characteristic is defined by a membership function which helps to take the crisp input values, and transformed into certain degrees of membership (see also Fig. 1).

TYPES	ROLE CHARACTERISTICS	FUZZY SET
Demographic	Age	young, middle, old
	Education	school, college, graduate, professional
	Task Function	IT_General, IT_Operation, IT_Expert
Knowledge	Domain Knowledge	low, medium, high
	Training	low, medium, high
Experience	Current Experience	novice, medium, senior, expert
	Previous Experience	low, moderate, extensive
Proficiency	IT Speed	low, medium, fast
	Skill	novice, medium, senior, expert
	Attitude	alone, ordinary, positive, cooperative
Evaluation	Performance Satisfaction	poor, average, good

(a)



(b)

Age	Education	Task Function	Domain Knowledge	Training	Current Experience	Previous Experience	IT Speed	Skill	Attitude	Performance Satisfaction	Role Category
Young	School	IT General	Low	Low	Novice	Low	Low	Novice	Alone	Poor	R0
Middle	College	IT Operation	Medium	Medium	Medium	Moderate	Medium	Medium	Ordinary	Average	R1
Old	Graduate	IT Expert	High	High	Senior	Extensive	Fast	High	Positive	Poor	R2
Old	Professional	IT General	Medium	Low	Medium	Extensive	Fast	Low	Cooperative	Good	R3
Young	College	IT Operation	Low	High	Novice	Moderate	Medium	High	Positive	Average	R4

(c)

Figure 1. (a) shows the role types, the role characteristics, and how the role categorization is performed by defining fuzzy set for the role characteristics. Two sample membership functions for input (education) and output (category) using fuzzy logic are shown in (b), and some sample of fuzzy rules are seen in (c).

The second layer of the user profile denotes preferential information which represents the domain-specific feature of the problems (service-calls) encountered, closely related to a particular individual user or his/her role. Namely, this layer particularly reveals the reasons about how/why the user is deeply related to particular types of problems. As seen in Table 1, each problem is composed of problem identity, problem class (either problem taxonomy or troubleshooting option), a composition of attributes which may represent the main characteristics of the problem well enough, relevance weights of these attributes, and relevance weight of each problem indicating its relative importance among all the problems in this layer. The number of components in the second layer corresponds to the number of possible problems that have been encountered. Besides, as the client interacts with HDAgents, the second layer's components are increased, aggregated, and updated with the change of the transactional information in the third layer.

The third layer of the user profile maintains transactional information about how the problems have been handled by certain HDAgents. As seen in Table 1, a single transaction consists of HDAgent identity that solve the given problem, case identity which contains retrieved solutions by this HDAgent, a set of problem identities that are closely related to the solutions in the case identity, and a set of diagnosed problem features, and appropriateness component. The appropriateness component indicates an appropriateness value of the given case for the involved problem identities, which is evaluated by the feedback from the client or HDAgent. Namely, it represents how much the given case is adequate to the related problems. This component is used to update the value of 'performance satisfaction' component in the first layer, and thus also to contribute to updating

Table 1. The proposed three-layered user file structure.

LAYER	TYPE	COMPONENT	
FIRST LAYER	User Identity	Client / HDAgent ID	
		Name	
	Company Description	Affiliation	
		Location	
		Employee Numbers	
		Software in Use	
	Role Characteristics	Demographic	Age
			Education
			Task Function
		Knowledge	Domain Knowledge
			Training
		Experience	Current Experience
			Previous Experience
		Proficiency	IT Speed
			Skill
			(Problem-Solving) Attitude
		Evaluation	Performance Satisfaction
	Role Category	Classified Role	
SECOND LAYER	Problems (≥ 1)	Problem ID	
		Problem Class	
		Attribute ID	Attribute Type(Keyword | QA)
			Attribute Name
			Relevance in Problem (%)
		Relevance over Overall Problem (%)	
THIRD LAYER	Case Solved (≥ 1)	Case ID	
		HDAgent ID solved the given problem	
		Problem Identities (≥ 1)	
		Diagnosed-Attributes Sequence	
		Appropriateness(%)	

the 'role category' in that layer according to the increased number of the transactions. Moreover, whenever a new transaction occurs, this new record will be used to update the corresponding instance(s) of related problems in the second layer. Further, once the first layer is created by user explicit input or stereotypes, the whole body of the user profile is automatically built up in an unobtrusive way without extra burdening of both the client and HDAgent by observing the transaction information in the third layer.

3 RECOMMENDATION STRATEGY

This section describes our recommendation strategy that recommends a HDAgent who may appropriately handle a given problem, based on the proposed user profile. This strategy is composed of the following major three steps.

3.1 Weighting Computation Based on Client's Own Experience

Given a problem by a particular client, we calculate the weighting function of relevant HDAgents' role for the client based on their problem-solving experience. More precisely, to compute this weighting, we use the information about the frequencies of the retrieved cases and whose appropriateness values. Such information can be found in the third layer of the user profile. In other words, this weighting indicates how frequently a particular HDAgent have handled the problems given by the client and how appropriate the retrieved solutions suggested by the HDAgent are. The key idea behind this step is to mimic the paradigm of content-based filtering [3]. Namely, we compute the weighting on the basis of the assumption that the more a HDAgent's role is experienced in solving the problems given by the client, the better the HDAgent's role will solve a problem given by the client appropriately in the future.

Formally, let CR and SR be the roles of the entities, clients C and HDAgents S, respectively, and then the weighting function W_1 is conceptually expressed by the correlation between C and SR as: $W_1 : C \times SR \rightarrow Weightings$. The weighting function $W_1(c, s_r)$

of a particular HDAgent's role s_r for a given client c is computed as: $W_1(c, s_r)$ = $\sum$ (*appropriateness values of the cases solved by* s_r *in the given client profile*) / (*total number of such cases*).

3.2 Weighting Computation Based on Similar Client's Experience

In this step, we compute additional weighting function of relevant HDAgents' roles for a given client's role. This weighting function is based on the appropriateness values of the clients having the same role with the given client. The assumption applied in this step is that we would acquire increasingly accurate weighting by taking objective views of similar clients to the given client. The basic idea behind this step is to take advantage of the paradigm of collaborative-based filtering [3]. In other words, we compute the weighting taking into account appropriateness values assessed by those clients who have the same role with the given client.

Formally, the second weighting function W_2 can be represented as: $W_2 : CR \times SR \rightarrow Weightings$. More specifically, The weighting function $W_2(c_r, s_r)$ of the HDAgent's role s_r for a given client's role c_r is computed as $W_2(c_r, s_r)$ = $\sum$ ([*appropriateness values of the cases solved by* s_r *in the client profile having the same role with the given client's role* c_r]/[*total number of such cases*])/(*total number of clients having the same role with the given client*).

3.3 Final Weighting Computation using Linear Combination

As the final step, we combine two separate weightings computed in the previous steps. For this, we adopt linear combination approach due to its generality, simplicity, usefulness, and powerfulness [12]. Our combination formula is defined as, "$W(c, s) : W_1(c, s_r) * (1 - \tau) + W_2(c_r, s_r) * \tau$", where τ denotes a combination coefficient. Here, τ is derived from this formula, $\tau = \left(\frac{N(\hat{CR})}{N(CR)} + \frac{N(\hat{SR})}{N(SR)}\right)/2$, where the $\hat{CR}$ and $\hat{SR}$ are the set of clients and HDAgents that have the same roles with the client c and HDAgent s, respectively. Finally, the HDAgent having the highest weighting is recommended as the most appropriate HDAgent that would solve the problem given by the client.

4 DEMONSTRATION

To evaluate the validation of our recommendation approach, one of the best ways might be to obtain the assessment from real help-desk domains. Remaining that actual evaluation to our future work, we instead illustrate how our recommendation approach is processed with an example to help improve intuitive understanding of it.

Let us consider an example consisting of four clients and five HDAgents profiles, as shown in Fig. 2. Then, we can represent a set of formal definition of that domain as C = {CA, CB, CC, CD}, S = {IA, IB, IC, ID, IE}, CR={CR1, CR2, CR3}, SR = {SR1, SR2, SR3, SR4, SR5}. Now we assume that the client CA presents a new problem with the category ("printing"). To recommend an appropriate HDAgent who would handle the given problem, the following three steps are processed.

First, to compute the first weighting function $W_1(c, s_r)$, we need to know s_r seen in the client profile CA. Given example, note that there exist three HDAgents (i.e., $s \in$ {IA, IB, IC}) and whose three roles (i.e., $s_r \in$ {SR1, SR2, SR3}) that have handle the cases in the client profile CA. In this example, therefore, W_1 is calculated as follows: W_1(CA,SR1) = (0.7+0.25+0.15)/3=0.37, W_1(CA,SR2)=0.6, W_1(CA,SR3) = 0.65.

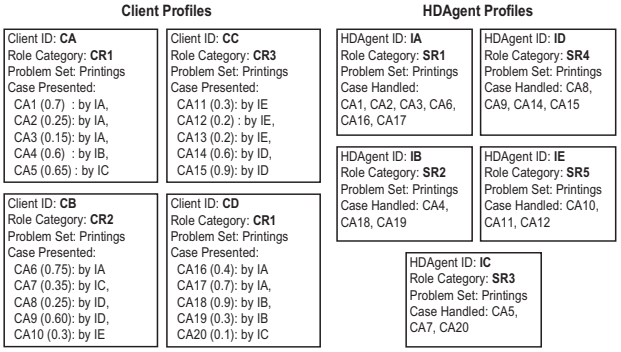


Figure 2. An example of the help-desk domain consisting of four clients and five HDAgents' profiles.

Next, to compute the second weighting function $W_2(c_r, s_r)$, we have to find a set of clients having the same role with the given client's role c_r. In this example, we can observe that there is the only one client CD that has the same role with the client CA's role CR1. Since the client CD has the cases handled by HDAgents IA, IB, and IC, W_2 is computed as: W_2(CR1,SR1) = ((0.4+0.7)/2)/1=0.55, W_2(CR1,SR2) = ((0.9+0.3)/2)/1=0.6, W_2(CR1,SR3) = (0.1/1)/1=0.1.

Lastly, the final weighting function is computed based on the linear combination of W_1 and W_2 as:

Computing the final weighting 'W':
W(CA,IA) = W_1(CA,SR1) x (1-0.35) + W_2(CR1,SR1) x 0.35 = (0.37x0.65) + (0.55x0.35) = 0.24+0.19 = 0.43
(τ = (0.5+0.2)/2 = 0.35)
W(CA,IB) = W_1(CA,SR2) x (1-0.125) + W_2(CR1,SR2) x 0.125 = (0.6x0.875) + (0.6x0.125) = 0.525+0.075 = 0.60
(τ = (0.25+0.2)/2 = 0.125)
W(CA,IC) = W_1(CA,SR3) x (1-0.125) + W_2(CR1,SR3) x 0.125 = (0.65x0.875) + (0.1x0.125) = 0.22+0.015 = 0.24
(τ = (0.25+0.2)/2 = 0.125)

According to that result, we recommend the HDAgent IB who has the role SR2 as the most appropriate HDAgent to handle the given problem in the example.

In order to improve an intuitive impression of the correctness of our approach, the question posed in verification used here is: "Do the recommended agent correctly ensure an appropriate agent ?" Based on this question, we explain how our approach is better than typical content-based and collaborative recommendation methods.

Let us consider a situation where we would apply content-based method with the above example, i.e., only using client CA's past experience. In this case, if the experience is regarded as a normalized summation of the appropriateness values of the cases, the HDAgent IC would be recommended due to the fact that IA=(0.7+0.25+0.15)/3=0.37, IB=0.6, and IC=0.65. But, such outcomes seem to be less confident since these are produced without considering the other client's opinions (i.e., the client CD), even if the client CD has the same role (i.e., similar problem-solving characteristics) with the client CA. Note that if we apply the same method for the client CD, the HDAgent IB will be selected. Thus, we can not guarantee that the IA is more appropriate than IB.

On the other hand, let us assume that we apply collaborative method. Since this method does not have the role information of

the clients when trying to recommend, all the experiences (cases) of the clients CB, CC, CD will be taken into account. Because such clients have concerned the same type of problem (i.e., 'printing'). In this case, probably the HDAgent IA could be recommended if we regard the experience as the same notion with the content-based method by means of (IA: (0.75+0.45+0.7)/3=0.62, IB: (0.9+0.3)/2=0.6, IC: (0.35+0.1)/2=0.225, ID: (0.25+0.60)/2=0.425, IE: (0.3+0.3+0.2+0.2)/4=0.225). However, this outcome also seems to be inadequate since the collaborative method includes the unnecessary or irrelevant experiences of the client CB and CC who have different roles (i.e., problem-solving characteristics) with the client CA. However, in our approach, we overcome those drawbacks by only considering the clients having the same role, i.e., the client CD. Therefore, we believe that our strategy would be intuitively correct compared with those two methods by considering both of the individual/similar group opinions based on the role information of the clients.

5 RELATED WORK

Help desk systems typically leverage CBR approaches. However, they are limited in many ways as our following discussion shows: Caseadvisor [14] is a representative interactive CBR system that provides solutions for customer's problems effectively in real-time. Its recommendation is usually done interactively by working with the customer through a requirement acquisition and definition process. However, this system tries to search optimal cases using only the given problem and stored cases, ignoring additional useful information such as the knowledge of the clients and help-desk agents.

To extend the limited decision space, knowledge management-centric help desk system is introduced [8]. This system attempts to use diverse knowledge source in the organization including database, files, experts, knowledge bases, and group chat to ensure high utilization and maintenance of knowledge store. Besides, utilizing semantic representation of the decision space [2] is proposed to improve the accuracy of the similarity matching between the problem and cases. Even though, these systems try to extend a decision space by utilizing more amount of knowledge involved in the case and semantical knowledge representation, these do not consider the personalized information about the client and help-desk agent. Moreover, the problem of high dependency on the help-desk agent is not still clearly addressed in these works.

There has been also approaches for reducing the search space for similarity matching and retrieving accurate case retrieval by focusing on the representation of the case. For example, the authors in [11] partition the case into the discrimination part and shared-featured part, and then apply a hybrid reasoning approach by integrating rule-base and case-base techniques. Meanwhile, the research in [7] proposes to use an object-oriented approach to model the domain, in order to overcome very limited knowledge about the structure and semantics of the domain based on attribute-value pairs, textual representation, and question-answer. However, these still rely on the agent's personal expertise to make suitable solutions to the problems.

6 FUTURE WORK AND CONCLUSION

This paper presented a new hybrid design approach for recommending an appropriate help-desk agent to solve the given service-calls based on the user profile. The user profile consists of a three-layered combination of factual, preferential and transaction information, which can richly conceptualize the knowledge of both the client and help-desk agent in an uniform way. The main feature of the user profile is that a new concept of roles of the users was deployed using fuzzy logic based on problem-solving characteristics. Then, we described how our hybrid recommendation strategy using individual/role information of the users is designed.

In the future, we plan to evaluate our approach on real help-desk domains. Moreover, since our approach remains cold-start and sparsity problem unsolved, we will incorporate knowledge of cases into our approach to address such issues.

REFERENCES

[1] Oshadi Alahakoon, Seng Loke, and Arkady Zaslavsky, 'Capturing buying behaviour using a layered user model', *International Conference on E-Commerce and Web Technology (EC-Web 2007)*, 109–118, (2007).

[2] Olli Alm, Eero Hyvnen, and Antti Vehvilinen, 'Opas : An ontology-based library help desk service', *4th European Semantic Web Conference 2007 (ESWC 2007)*, (2007).

[3] Robin Burke, 'Hybrid recommender systems: Survey and experiments', *User Modeling and User-Adapted Interaction*, **12**(4), 331–370, (2002).

[4] Robert Bushey, Jennifer Mitchell Mauney, and Thomas Deelman, 'The development of behavior-based user models for a computer system', *7th International Conference on User Modeling (UM99)*, 109–118, (1999).

[5] Christine W. Chan, Lin-Li Chenb, and Liqiang Genga, 'Knowledge engineering for an intelligent case-based system for help desk operations', *Expert Systems with Applications*, **18**(2), 125–132, (2000).

[6] Ibrahim Cingil, Asuman Dogac, and Ayca Azgin, 'A broader approach to personalization', *Communications of the ACM*, **43**(18), 136–141, (2000).

[7] Mehmet H. Göker and Thomas Roth-Berghofer, 'Development and utilization of a case-based help-desk support system in a corporate environment', in *ICCBR '99: Proceedings of the Third International Conference on Case-Based Reasoning and Development*, pp. 132–146. Springer-Verlag, (1999).

[8] Luz Minerva Gonzlez, Ronald E. Giachetti, and Guillermo Ramirez, 'Knowledge management-centric help desk: specification and performance evaluation', *Decision Support System*, **40**(2), 389–405, (2005).

[9] Robert Heckman and Audrey Guskey, 'Sources of customer satisfaction and dissatisfaction with information technology help desks', *Journal of Market-Focused Management*, **3**(1), 59–89, (1998).

[10] Mohammad Jamshidi, Nader Vadiee, and Timothy J. Ross, *Fuzzy logic and control: software and hardware applications*, Prentice-Hall, Inc, 1993.

[11] Yuh Foong David Law, Sew Bun Foong, and Shee Eng Jeremiah Kwan, 'An integrated case-based reasoning approach for intelligent help desk fault management', *Expert Systems with Applications*, **13**(4), 265–274, (1997).

[12] Qing Li and Byeong Man Kim, 'An approach for combining content-based and collaborative filters', *Proceedings of the sixth international workshop on Information retrieval with Asian languages*, **11**, 17–24, (2003).

[13] Satya Shah, Rajkumar Roy, and Ashutosh Tiwari., 'Development of fuzzy expert system for customer and service advisor categorisation within contact centre environment', *10th Online World Conference on Soft Computing In Industrial Applications*, (2004).

[14] Qiang Yang, Edward Kim, and Kirsti Racine, 'Caseadvisor: Supporting interactive problem solving and case base maintenance for help desk applications', *In Proceedings of the IJCAI 97 Workshop on Practical Applications of CBR*, (1997).

[15] Qiang Yang and Jing Wu, 'Enhancing the effectiveness of interactive case-based reasoning with clustering and decision forests', *Applied Intelligence*, **14**(1), 49–64, (2001).

[16] Arkady Zaslavsky, Claudio Bartolini, Abdel Boulmakoul, Oshadi Alahakoon, Seng Wai Loke, and Frada Burstein, 'Enhancing the it service desk function through unobtrusive user profiling, personalization and stereotyping', *Proceedings of the 14th Annual Workshop of HP Software University Association*, (2007).

ICARE: A Context-Sensitive Expert Recommendation System

Helô Petry, Patricia Tedesco, Vaninha Vieira and Ana Carolina Salgado

Abstract. In a competitive world, where time and resources are sparse, people can improve their productivity if they interact with others that possess hands-on experience in the task or problem being performed. Expert Recommendation Systems (ERS) aim at identifying experts that may help people to accomplish their tasks. However, existing ERS are expert-centered and usually do not consider users' and experts' contexts, which impacts the adequacy of their recommendations. This paper presents ICARE (Intelligent Context Awareness for Recommending Experts), an ERS that considers the context of people involved, providing recommendations that better fit users' needs. We discuss how context was considered and the heuristics used to perform experts ranking. The results of our experiment show that the context awareness in ICARE was deemed satisfactory and its recommendations were well evaluated by the participants.

1 INTRODUCTION

Increasing competition and decreasing time to market have compelled organizations to improve their working processes, pressing people to continually increase their productivity. However, individuals may have difficulties while performing their tasks, especially in knowledge intensive working environments. To make matters worse, such environments often lack adequate tools to help people overcoming their problems in an easy and fast way.

This state of affairs was pointed out to us in a questionnaire applied in five Computer Science research and development environments. We obtained fifty participants with different profiles and backgrounds. To the question "how did you solve the last problems that you faced while fulfilling your tasks?", 40% of the participants answered "I used Google", 20% said that they searched in documentation and 30% answered that they asked a friend. These results show that most people search for help in sources that cost time and effort (Internet and documentation). Also, asking a friend may not be successful because it is possible that the user has no friends with the needed knowledge.

In this kind of situation, people can save time and effort if they can interact with others that have the necessary knowledge obtained through experience, i.e. the experts. Researchers have observed that many affinities and opportunities for collaboration and knowledge sharing are discovered in informal interactions, such as chats in the coffee room [11]. However, it may be the case that potential collaborators may be not physically close to each other, and thus informal workplace meetings are not likely. Moreover, individuals may not possess *a priori* the knowledge that allows them to identify this possibility of interaction. One example was given to us by the director of a big cell phone company. He told us the case of an employee that faced a difficulty and did not find anyone to help. His superiors even contacted the organization in USA without success. After days of searching, they found out that an employee with the necessary knowledge worked in the same room that the one searching for help.

Thus, some form of computational support is necessary in order to make informal collaborations possible. A system that identifies potential collaborators and put them in touch can provide benefits to the work they are developing, since direct communication makes skill and tacit knowledge sharing easier and more effective. Expert Recommendation Systems (ERS) aim to identify experts which may help people in accomplishing their tasks.

However, existing ERS are expert-centered, since they focus mostly on expertise matching (e.g. [9], [12]). Further, they usually do not consider the context of people involved in the recommendation (the user requesting the recommendation and the recommended expert). This means that their suggestions are not always adequate to the situation of the user searching for help. Context is defined as the set of all relevant elements that can characterize people and their current situation while interacting with an application [14]. The use of context to provide services and information more adequate to users, considering their needs and preferences, is the goal of systems called context-sensitive systems.

In this light, we propose a context-sensitive ERS that aims at providing recommendations that better fit users' needs and situations (in this paper, the term 'user' refers to the person who is requesting the recommendation). This ERS is named ICARE (*Intelligent Context Awareness for Recommending Experts*). The use of context in ERS may enable better recommendations since it can be adapted to the user's current situation. For instance, for a busy user, ICARE might recommend the most available experts; or for a user with little experience, it might recommend experts in similar organizational position of such user. To verify the acceptance and correctness of ICARE, we performed an experiment whose results showed that the idea of embedding context-sensitivity into an ERS was well evaluated, that the defined heuristics for expert ranking worked properly, and that the recommendations were judged adequate to the users' needs.

The remainder of this paper is organized as follows: Section 2 introduces the concepts behind ERS; Section 3 presents ICARE; Section 4 describes the experiment performed to evaluate ICARE; Section 5 reviews some related work and compares them with ICARE; and, finally, Section 6 points out conclusions and directions for further work.

[1] Informatics Center, Federal University of Pernambuco, Brazil. Email: {**hp**, **pcart, vvs, acs**}**@cin.ufpe.br**

2 EXPERT RECOMMENDATION

Expert-seeking activities mostly happen in a dynamic work environment where people always deal with tasks that require new skills and knowledge. In this kind of environment, the ability to find experts in the required expertise is crucial to the successful completion of tasks [5]. Therefore, there is a need to use technology to scale from personal networks to the larger community of people within the organization [1]. Expert Recommendation Systems (ERS) are those that return references to individuals identified as experts in a requested domain and that can be used to connect human actors [12].

An ERS can help individuals by linking people who might never have an opportunity to meet face to face [1]. This kind of system can also provide benefits to the organization because, by representing employees experience and skill, they become a tool to represent tacit knowledge [1]. When existing knowledge gets applied to a problem, the organization saves time.

ERS can be considered as communication vehicles to the right person (i.e. the one with the adequate knowledge to our needs) [1]. Hence, ERS can be used to promote informal collaboration between people who need to share information. Yet, this interaction may help individuals to strengthen the tie between them. So, an ERS can help to augment personal networks by fostering connections with people inside the same organization [1].

Existing ERS use different: (i) types of information in order to indicate expertise (e.g. email communication, paper authorship); (ii) approaches to extract expertise data (e.g. linguistic techniques, mining data from the Web); and (iii) criteria to rank expert, such as expertise degree and social closeness.

When evaluating existing ERS (e.g. [7], [8], [10]), the lack of flexibility in expert definition becomes very clear. Their development is guided by a preconceived idea of expert (i.e. an expert is who reads documents about some topic, or expert is a paper author). They also need more flexibility when defining the expertise source of information. Besides, existing ERS usually do not consider user and expert contexts, or consider them isolated. At most, some systems use social networks. This way, their output is user-independent.

Researchers are beginning to realize this need to use context in expert recommendation, as the work of Balog *et al.* [2] indicates. The authors refined expertise retrieval models by incorporating topicality and organizational structure. So, the organizational unit a person belongs to was used as a context for that person. They tested efficiency of the models and found a positive impact of the context models for expert finding.

Therefore, we believe that a context-sensitive ERS can be more effective, providing more interesting opportunities for collaboration.

3 ICARE

In order to address the deficiencies identified in Section 2, we present ICARE (Intelligent Context Awareness for Recommending Experts) (Figure 1). ICARE has two main characteristics: it is context-sensitive and it uses different and many independent expertise sources. This system is designed to receive anytime the configuration to access a source. Regarding its context awareness, ICARE considers two main entities: the user that requests the recommendation and the expert that will be considered in the returned experts list. ICARE aims to improve its recommendations by using the user's and expert's context in its experts ranking function, thus prioritizing experts that better fit user's current needs.

ICARE uses the following contextual elements to characterize users and experts:

1. *Availability*: relates to how busy s/he is and in how many activities s/he is involved. It is more likely that someone with high availability will accept to engage in a collaborative session than someone with low availability;
2. *Approachability*: means how easy is to get him/her involved in a collaborative session. For instance, it is likely that a colleague (John), who works with his office's door opened, is more accessible than a colleague (Peter) who works with his door closed. However, if John is with his door opened and the room is full with people trying to speak with him, it demonstrates that John is accessible but is currently not available.
3. *Organizational level*: means his/her hierarchical position in a business or academic organization;
4. *Social distance*: is the number of people who know each other and that socially separate user from expert. Numerous studies have shown that one of the most effective channels for dissemination of information and expertise within an organization is its informal network of collaborators, colleagues, and friends [6], [4];
5. *Current task*: represents the task s/he is performing;
6. *Reputation*: is the expert overall quality as judged by users who contacted him. It is estimated through the feedback given by the users after using ICARE;
7. *Interests*: information related to the user that help on identifying the desired expertise;
8. *Expertise degree*: means how much someone is an expert in a given expertise.

Therefore, ICARE context sensitiveness helps to improve the odds of a successful collaboration between user and expert. For instance, if the user belongs to the second organizational level (of seven), it is possible that s/he prefers to interact with experts that belong to similar organizational levels (like 1, 2 or 3) because s/he might not feel comfortable in approaching someone of a high position in the organization.

The contextual elements used in ICARE are acquired through a generic context manager called CEManTIKA (Contextual Elements Management Through Incremental Knowledge Acquisition) [14]. This manager defines procedures and an infrastructure to manipulate contextual elements, managing the user's context that is shared with ICARE through its Context Interface (Figure 1). Another interface, the Recommendation Interface, is used to request recommendations to ICARE. It can be accessed, for instance, by a graphical user interface that helps the user in informing his/her desired expertise. The internal components of ICARE are explained in the next section.

3.1 ICARE Architecture

The system architecture is divided in four modules: *Experts Database Generator*, *Expertise Definer*, *Experts Retriever* and *Experts Presenter*. The details are presented in what follows.

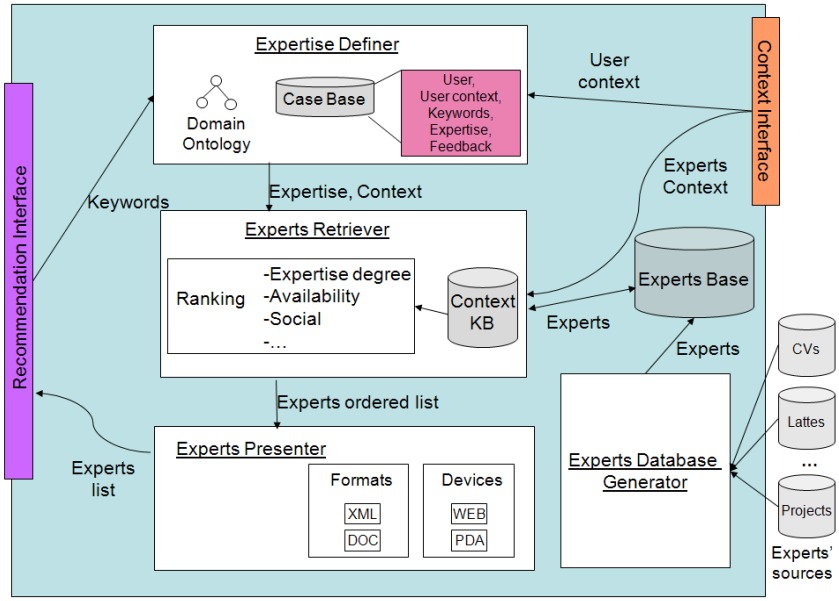


Figure 1. ICARE architecture

3.1.1 *Experts Database Generator*

This module is responsible for: (i) registering available experts' sources; (ii) receiving the configuration to access each expert source; and (iii) saving the information acquired in the experts' sources in the ICARE Experts Base. Notice that this database must be populated before a recommendation request. The set of stored experts is finite and defined, such as the set of employees in the organization.

When including a new expert source, the Experts Database Generator (EDG) receives a description of the provided information and how to access this source (e.g. SQL script to query the organizational projects database). Also, the EDG maps the information retrieved to a concept of the domain ontology.

Some kinds of expertise are inherently related to organizational activities practices. EDG uses a set of production rules to associate specific activities with the related expertise. These rules must be informed when including the new expert source.

3.1.2 *Expertise Definer*

The first step towards an expert recommendation is to discover the desired expertise. This is the Expertise Definer's (ED) goal. The ED input is a keyword and its output is one or more ontological concepts that match the keyword. It uses a domain ontology in which every concept is related to a set of keywords with a weight. The weights are used to identify the concepts most related to the keyword. Thus, ED searches the ontology to get the set of concepts related to the given keyword.

The ED also considers the expertise (concepts) identified in previous recommendation cases. To this end, ED selects the stored recommendation cases that had: (i) the same keyword as input; (ii) positive feedback; and (iii) similar context to the current user's. This last requirement uses a context similarity formula that calculates the degree of similarity between two contexts. This formula sums the absolute differences between the information contained in both contexts. It considers the following contextual elements: approachability, availability, organizational level, and current task.

Next, ED increases the weight of the identified expertise related to the user's interests. The retrieved expertise list is ordered according to each expertise weight. Finally, ED filters this list, maintaining the n first expertise, where n is a system configuration parameter.

3.1.3 *Experts Retriever*

After defining the searched expertise, the second step towards the expert recommendation is to find the corresponding experts. This is the Experts Retriever (ER) task. Its input is the expertise list provided by ED and its output is an ordered expert list.

For each expertise in the list, the ER accesses the Experts Base and retrieves the related experts. The results of these queries are unified and the duplicates are removed. Next, ER queries the Context Interface for the contexts of the identified experts. Then, the experts are ranked according to their context.

The ICARE experts ranking is based on the following contextual elements:

1. Expertise degree (ed);
2. Approachability (ap);
3. Availability (av);
4. Reputation (rep);
5. Social distance (socialDist); and
6. Difference between user and expert organizational levels.

Notice that the user's context is represented by the same contextual elements of the expert context (except for reputation and expertise degree).

Fitness is a measure that identifies the suitability of an expert to the recommendation. In order to calculate the fitness of a given expert to the recommendation, we created a formula simply separating the elements that are directly proportional from the inversely proportional to the expert fitness. For instance, the higher an expert's reputation is, the higher his fitness. Similarly, the lower the social distance between the expert and the user is, the higher the expert's fitness. Then, the expert fitness can be calculated by the following formula (which receives the **expert's** context):

$$Fitness = \frac{ed + ap + av + rep}{socialDist + \left|OL_u - OL_e\right|}$$

Where OL_u = user organizational level; and OL_e = expert organizational level.

By using the fitness measure, we obtain an expert ranking that is already context sensitive because more available experts will receive a higher fitness, for instance. But we can improve it by applying different weights to the contextual elements. This approach is interesting because a user may find one element more important than another. For instance, a user might prefer to contact the more accessible experts than the ones with good reputation. This way, the **user's** context is used to adjust the contextual elements' weights. In other words, the weight given to each element changes according to the user's context. Then, we introduce these weights in the formula that the system uses to calculate the fitness of each retrieved expert:

$$Fitness = \frac{\alpha_1 \times ed + \alpha_2 \times (ap + av) + \alpha_3 \times rep}{\alpha_4 \times socialDist + \alpha_5 \left|OL_u - OL_e\right|}$$

Where αi = contextual elements' weight. All the data applied to the formula are normalized in values between 0 and 1. In case the denominator results in 0, it is replaced by 1.

Therefore, expert fitness indicates his adequacy to the current recommendation. Thus, the retrieved experts are ranked according to their fitness.

These contextual elements' weights (αi) are adjusted by a Context Knowledge Base (CKB). It has an inference engine and a set of context rules. These rules define which elements should be favored given a user's context. They follow the pattern:

```
<if> contextual condition
<then> contextual elements weights setting
```

A sample rule that exists in the CKB is the following:

```
<if>
     Organizational level < 0.5
     Approachability < 0.3
<then>
     Expertise degree weight = 0.8
     Social distance weight = 0.2
```

This rule means that if the user occupies a low position in the organization and is not very accessible, the recommendation will favor experts with higher expertise degrees and will give less importance to the social distance.

Every time the system receives a recommendation request, the inference engine is informed with the user's context and is activated to calculate the contextual elements weights.

The context rules were defined after a preliminary experiment constituted by a questionnaire applied in several Computer Science research and development organizations. The experiment has identified the participants profile and their preferences when asking an expert recommendation. More specifically, people were asked whether or not they agreed with each contextual element and to rank the elements by importance. The contextual elements in question are: expertise degree, availability, organizational level, reputation, and social relationship. The analysis of the results helped us to create the rules that associate the user's context to a contextual elements weight setting. An interesting result was the most disapproved criterion: social relationship (by 64% of the participants). Meaning that people do not consider relevant the social relationship between them and the recommended expert.

3.1.4 Experts Presenter

The last step in the expert recommendation is to present the identified experts. The Experts Presenter aims to adjust experts' presentation according to the user's needs or preferences, concerning attributes such as the amount and format of the information. For instance, users might want to receive all the results found in HTML format.

3.2 Implementation

We have developed a prototype aiming to demonstrate ICARE functionalities and performance. Since we are interested in evaluating the use of context in expert recommendation, our implementation scope included the ICARE modules that use context: Expertise Definer, Experts Retriever, and Context Interface. In addition, we have developed a web user interface to illustrate the results of the recommendation.

We also have created the Experts Base using a third party system that automatically generates user profiles [13]. It identifies user interests by using textual information retrieval techniques applied to a standard curriculum platform proposed by a Brazilian research organization, named Lattes. We supplied the system with a set of curricula and stored the generated profiles in ICARE's database.

The Context Interface obtains the contextual information from the CEManTIKA context manager. However, it is still being designed. This way, the context information used in ICARE was, for now, simulated. Notice that ICARE and CEManTIKA use a context API which allows them to be developed separately.

4 EXPERIMENT WITH ICARE

We ran an experiment with ICARE addressing the objective to evaluate:

1. *Acceptance of ICARE strategies*. More specifically, we wanted to evaluate if the users agreed with the contextual elements, and whether or not they agreed with the weights' variation according to the user's context;
2. *If the steps of the recommendation process are executed as expected*. Such steps comprehend: (i) identifying expertise through the ED; and (ii) retrieving and ranking experts through the ER. Notice that the contextual elements' weights are also verified;
3. *If the recommended experts are adequate*, which means that the system returned suitable experts; and

4. *If the experts' generated profiles are consistent with their curricula*.

In order to allow the user to evaluate ICARE's results, its GUI shows the following internal and intermediate data of the recommendation: (1) the concepts identified; and (2) the contextual elements' weights inferred by CKB. Moreover, to allow the user to evaluate the experts' profile, we showed their short curricula. In addition, the contexts of the recommended experts were available.

The experiment counted with 21 participants; all of them were attendees of a course on Intelligent Agents belonging to the Computer Science Post-Graduation Program of our University. They were gathered in a lab where they could test ICARE at the same time. In the beginning of the experiment, we presented ICARE's goal and features and how to use it.

The participants were asked to inform their current context: approachability, availability, current task, and organizational level. The participants were also asked to request five recommendations. Afterwards, they answered a questionnaire that aimed at evaluating the aforementioned goals.

With regards to the first objective of the experiment, the results showed 86% of the participants agreed that the weights of the contextual elements should vary. Also, 85% of them evaluated the ranking elements as relevant. The other 15% of the participants specified the irrelevant elements, in their opinion, as: social relationship, organizational level and expertise degree.

Addressing the second objective, we obtained the following results. 47% of the participants said that the system identified adequate concepts related to the informed keyword. 34% of them said that the concepts were correctly identified, but incorrectly ordered. 65% of them approved the contextual elements' weights that the system generated.

Addressing the last two objectives, the results showed that 63% of the participants classified the recommended experts as adequate. In other words, they considered that the experts' curricula were consistent and that the system recommended the correct experts to the given request. Amongst the participants that disapproved the experts found, 19% (approximately half of them) also classified the expertise identified as inadequate (in the same recommendation). Therefore, the system could not find the correct experts if the expertise was not identified correctly first.

Summarizing, the experiment showed that ICARE was well accepted and its context-sensitiveness and contextual elements were well evaluated. Also, the contextual elements' weights, the identified concepts and recommended experts were, in general, adequate. Therefore, ICARE demonstrated being consistent and efficient.

5 COMPARATIVE ANALYSIS

In the following we analyze some existing ERS.

ReferralWeb [7] provides recommendations via chains of named individuals: their social network. This way, the user can find referral chains between him and experts on arbitrary topics. *ReferralWeb* goal is recognizing communities based in scientific paper co-authoring. The system uses the co-occurrence of names in close proximity in any documents publicly available on Web as evidence of a direct relationship. Such sources include list of coauthors in technical papers and citation papers. It uses a general search engine (AltaVista) to retrieve documents associated to a user. *ReferralWeb* has the advantage of identifying expert groups and the strength of the relationship between them. However, *ReferralWeb* has no specific expert search mechanism.

Expertise Recommender (ER) [10] uses social networks to tailor recommendations to the user. The recommendation is done in two steps. First, it finds a set of individuals who are likely to have the necessary expertise. Then they are matched to the requester by using a social network. Expertise indication is defined according to organizational criteria. The technique for profile construction also depends on the organization. Experts are ranked according to expertise degree and social closeness. ER's main advantage is its flexibility towards expertise indications and sources. But identifying social networks in the organization can be very costly.

HALe [9] aims at discovering implicit and explicit connections between people by mining semantic associations from their email communications. Thus, email communication is the expertise indication in HALe. Its approach to construct profile uses linguistic techniques. But HALe does not rank the experts and, therefore, the user has less information at hand to decide which expert to approach. Its transparency to the user is an advantage in *HALe*, since it does not disturb his/her activities. However, this system uses only one kind of information to indicate people expertise.

TABUMA (*Text Analysis Based User Matching Algorithm*) [12] generates users' profiles by extracting keywords from text documents using linguistic methods such as LSI (Latent Semantic Indexing). So, having (which implicitly means that the person read) text documents about a topic is the expertise indication in TABUMA. Its approach to construct profile uses linguistic techniques. Experts are ranked according to their expertise degree. An advantage in TABUMA is its independency from text document format. A disadvantage is that it needs to ask the user to choose a set of documents that s/he considers representative of her/his knowledge.

Liu *et al.* [8] present a RDF-based solution to expertise matching and integration. The main challenge addressed in their work is how people retrieval can be improved by extracting relevant information associated to an expert from different data sources and then semantically integrating them. It uses scientific and academic production as expertise indication. The profiles are constructed using wrappers that access each source. Experts are ranked according to expertise degree. A positive feature identified in this work is integration of information gathered from different sources. A disadvantage is that they do not consider any information related with the user.

When comparing the existing ERS with ICARE, we can observe the following:

- ICARE is flexible in the definition of expert: s/he can be defined by his/her authorship, or list of certifications, or time performing a given job. Therefore, ICARE is independent of the type of information used to characterize an expert. On the other hand, the existing ERS have a preconceived idea of expert;
- Except for the proposal of [8] and [10], the ERS usually are constructed according to a predefined expertise source. However, ICARE is flexible regarding the source of information used to acquire expertise data;
- In general, the existing ERS do not consider user and expert contexts. At most, some systems use social networks, such as *ReferralWeb* [7], ER [10] and *HALe* [9]. Moreover, as

our experiments indicated, this contextual information seemed to be the most unpopular between all contextual elements used in ICARE. Although the work of Balog *et al.* [2] is not a ERS, it uses contextual information that is distinct from social relationship (organizational position). On the other hand, ICARE uses a set of contextual elements and, this way, its output is user-dependent. This means that ICARE recommendations are tailored to the user and expert situation.

Therefore, ICARE provides more interesting opportunities for collaboration than the others ERS presented.

ICARE is restricted in the sense of using only predefined contextual elements (described in Section 3). Some researchers argue that context cannot be delineated and defined in advance, that the scope of contextual features is defined dynamically, and that context is particular to each occasion of activity or action (it is not stable) [3]. However, it becomes much more complex to develop context sensitive systems that regard this flexibility requirement. Therefore, we chose this approach driven from a technological perspective, focusing on the available computational resources, information and techniques. Moreover, we took care to detach the context related features in ICARE from the common functions. Therefore, new context elements might be included in the system with low effort.

6 CONCLUSIONS & FUTURE WORK

People are frequently involved in knowledge intensive activities that, naturally, raise doubts which usually must be solved quickly. In these situations, they need a tool to help them to find proper and efficient help. An ERS may help on identifying people with a given knowledge. Nevertheless, we have seen that existing ERS do not consider the context of people involved in the recommendation. Researchers are now realizing this necessity, as indicated in [2]. Our work presents a step further in this direction, since we proposed and developed a context-sensitive ERS which considers a set of contextual elements. Hence, we presented in this paper ICARE, a context-sensitive ERS that aims at facilitating the collaboration between people who can share knowledge.

ICARE uses context to customize its recommendations and, therefore, prioritize the most suitable experts for the user at the moment. The user's context is employed to adjust the weights applied to the contextual elements used to rank the experts who will be recommended. Hence, the recommendation results are different for each user, according to his/her context. The weights adjusting feature was develop after a preliminary experiment that identified people preferences when receiving recommendations.

The contributions of this work are:

- The inclusion of context sensitiveness in an ERS and its construction. This need was identified in the bibliography and confirmed in the experiment with ICARE;
- The proposed contextual elements used in recommendation (which were also approved in the experiment with ICARE);
- The experts ranking heuristics defined;
- The proposal to modify the relations between the contextual elements according to the user's context;
- The preliminary experiment that helped to understand what people expect from an ERS. It also evidenced their need for a tool which helps to locate knowledge.

ICARE was evaluated through an experiment. Its results show that our ERS demonstrated being consistent and efficient, since it generated adequate and successful recommendations.

As further work, we intend to apply ICARE in a real organizational environment. Also, we plan the integration with CEManTIKA and the addition of metadata representing the quality of the contextual information, such as accuracy and time.

ACKNOWLEDGMENTS

Authors want to thank CNPq and CAPES for their financial support.

REFERENCES

[1] Ackerman, M., Pipek, V., Wulf, V. "Sharing Expertise: Beyond Knowledge Management", The Mit Press (2003).

[2] Balog, K., Bogers, T., Azzopardi, L., Rijke, M., van den Bosch, A. "Broad expertise retrieval in sparse data environments", ACM SIGIR conference on Research and development in information retrieval (2007), pp. 551-558.

[3] Dourish, P. "What We Talk about When We Talk about Context", Personal and Ubiquitous Computing, v. 8, n. 1 (2004), pp. pp. 19-30.

[4] Galegher, J., Kraut, R., Egido, C. "Intellectual Teamwork: Social and Technological Bases for Cooperative Work", Hillsdale, NJ: Lawrence Erlbaum Associates. (1990).

[5] Ghazali, O., Shiratuddin, N. "Expert-seeking Activity Framework", Journal of Advancing Information and Management Studies, v. 1, n. 1 (2004), pp. 63-73.

[6] Granovetter, M. S. "The Strength of Weak Ties", The American Journal of Sociology, v. 78, n. 6 (1973), pp. pp. 1360-1380.

[7] Kautz, R., Selman, B., Shah, M. "ReferralWeb: Combining Social Networks and Collaborative Filtering ", Communications of the ACM, v. 40, n. 3 (1997), pp. 63-65.

[8] Liu, P., Curson, J., Dew, P. "Use of RDF for Expertise Matching within Academia", Knowledge and Information Systems, v. 8 (2005), pp. 103-130.

[9] McArthur, R., Bruza, P. "Discovery of Implicit and Explicit Connections between People Using Email Utterance", Conference on Computer Supported Cooperative Work (2003), pp. 21-40.

[10] McDonald, D. W., Ackerman, M. S. "Expertise Recommender: A Flexible. Recommendation System and Architecture", Conference on Computer Supported Cooperative Work (2000), pp. 231-240.

[11] Petry, H., Vieira, V., Tedesco, P., Salgado, A. C. "Um Sistema de Recomendação de Especialistas Sensível ao Contexto para Apoio à Colaboração Informal", Simpósio Brasileiro de Sistemas Colaborativos (2006).

[12] Reichling, T., Schubert, K., Wulf, V. "Matching Human Actors Based on their Texts: Design and Evaluation of an Instance of the ExpertFinding Framework", ACM SIGGROUP Conference on Supporting Group Work (2005), pp. 61-70.

[13] Ribeiro Jr, L. C. "Definição Automática de Perfis de Usuários de Sistemas de Recomendação", Escola de Informática. Universidade Católica de Pelotas (2005).

[14] Vieira, V., Tedesco, P., Salgado, A. C., Brézillon, P. "Investigating the Specifics of Contextual Elements Management: The CEManTIKA Approach", Modeling and Using Context (CONTEXT'07) (2007), pp. 493-506.

A Discussion on Multi-Criteria Recommendation

Nikos Manouselis[1]

Abstract. Recent studies have indicated that since multiple item characteristics may be taken into consideration when making a selection, it might not be sufficient to base recommendation on single-attribute ratings. They also highlighted the potential of applying Multi-Criteria Decision Making (MCDM) methods in recommender systems. This position paper aims to introduce multi-criteria recommender systems, review and assess current implementations of MCDM in recommender systems, as well as identify open issues for further discussion and investigation.

1 INTRODUCTION

The problem of recommendation has been identified as the way to help individuals in a community to find information or items that are most likely to be interesting to them or to be relevant to their needs [5]. It has been further refined to (i) predicting whether a particular user will like a particular item (prediction problem), or (ii) identifying a set of *N* items that will be of interest to a certain user (top-*N* recommendation problem) [3]. In a recommender system, the items of interest and the user preferences are represented in various forms, which may involve one or more variables. Particularly in systems where recommendations are based on the opinion of others, the incorporation of the multiple criteria that can affect the users' opinions into the recommendation problem, can potentially lead to more realistic recommendations [1].

To this direction, several recommender systems have already been engaging multiple criteria for the production of recommendations. Such systems, referred to as *multi-criteria recommenders*, early demonstrated the potential of applying multi-criteria decision making (MCDM) methods to facilitate recommendation in numerous application domains (e.g., movie recommendation, restaurant recommendation, product recommendation) [2]. A recent review and assessment of implementations of multi-criteria recommenders that have been proposed in the literature [5] has raised a number of issues for further discussion and investigation.

2 RECOMMENDATION AS MCDM PROBLEM

The recommendation problem can be formulated as follows (in the light of [1], [2], [5]): let *C* be the set of all users and *S* the set of all possible items that can be recommended. It may be assumed that there exists a utility function $U^c(s): C \times S \to \Re^+$ that can serve as an indicator (measure) of the appropriateness of recommending an item *s* to user *c*. It can also be assumed that this function $U^c(s)$ is not known for the whole *C* x *S* space but only on some subset of it. Therefore, in the context of recommendation, we want for each user $c \in C$ to be able to:

- either estimate (or approach) the utility function $U^c(s)$ for some item *s* of space *S* for which $U^c(s)$ is not yet known;
- or to choose a set of items $S' \in S$ that will maximize $U^c(s)$: $\forall c \in C, s = \arg\max_{s \in S'} U^c(s)$ (1)

In many recommender systems, the utility function $U^c(s)$ considers only one attribute of an item, e.g. its overall evaluation or *rating*. Recent studies (see [1],[2]) have indicated that this consideration might be insufficient, since the suitability of an item for a user can depend on more than one attributes (or criteria). The recommendation problem can be therefore modeled as a decision problem with multi criteria.

To model recommendation as a MCDM problem, we adopt the following steps of Roy's general modeling methodology for decision making problems [9]:

- *Object of the decision*. That is, defining the object upon which the decision has to be made and the rationale of the recommendation decision.
- *Family of criteria*. That is, the identification and modeling of a set of criteria that affect the recommendation decision, and which are exhaustive and non-redundant.
- *Global preference model*. That is, the definition of the function that aggregates the marginal preferences upon each criterion into the global preference of the decision maker about each item.
- *Decision support process*. That is, the study of the various categories and types of recommender systems that may be used to support the recommendation decision maker, in accordance to the results of the previous steps.

This modelling approach, together with a presentation of MCDM methods that can support automatic recommendation have been presented in previous work [5]. In addition, it has been used as a basis for designing, implementing, and experimentally studying a family of multi-attribute utility collaborative filtering algorithms [6].

3 EXISTING MULTI-CRITERIA SYSTEMS

In our previous work [5], a review and classification of multi-criteria recommender systems has also taken place. An aspect not included in this previous work has been the examination of their actual implementation and testing. Revisiting our sample of forty-two (42) multi-criteria recommender systems that have

[1] Greek Research & Technology Network (GRNET S.A.), 56 Mesogion Str., Athens, Greece. Email: **nikosm@ieee.org**.

been identified in the literature, we could note that the majority is proposed at only a design or early prototype level, which has never been tested by actual end-users. More specifically, for sixteen (16) systems only a proposed design is presented, whereas for nineteen (19) only an early prototype has been implemented. This means that these proposed systems have never been deployed and tested under actual usage conditions.

As far as their evaluation is concerned, more than half (i.e. 22 out of 42) of the systems in the sample have not been tested at all. For the remaining twenty (20), some evaluation has taken place – although not always in relation to the needs of some targeted users or intended application context. Our analysis revealed that:

- In only 4 out of these 20 tested systems evaluation focused on the system and/or its interface (e.g. satisfaction from the system or its usability). In the rest of the cases, only the recommendation algorithm has been tested (mainly its accuracy).
- In 3 out of 20 cases a pilot experiment that involved data collection from actual users took place (e.g. collecting ratings through a pilot experiment). In the rest 17 systems, a simulated execution of the recommendation algorithm took place. As far as these are concerned: (a) the data used for the simulation has been real (i.e. collected from the actual usage of the system) in only 5; (b) in 7 cases the data used for evaluation has been synthetic (simulated); and (c) in 5 cases a publicly available but single-attribute data set has been used, such as the MovieLens and EachMovie ones [4].

Overall, our analysis indicated that the evaluation of multi-criteria recommender systems cannot be considered systematic or complete. More than half of the systems have not been tested at all. In many cases, testing only comprised from a simulated execution of the algorithm. And in most occasions, the data sets used have been from different application contexts or under conditions that do not match the actual requirements of the intended usage environment (e.g. using single-attribute rating data sets instead of multi-attribute ones).

5 OPEN ISSUES

Based on our previous work and the results of our analysis presented above, we can identify a number of potentially interesting observations regarding multi-criteria recommender systems. First of all, it has been noted that the MCDM methods used for the construction of the preference models are mostly value-focused ones. In particular, the majority of systems uses Multi-Attribute Utility Theory (MAUT) and engages a linear additive value function for the representation of user preferences. This is a traditional decision making approach, widely applied and convenient to implement. On the other hand, assuming that the preference function is linear restricts the way user preferences are represented. Therefore, alternative forms for representing preferences in a MCDM manner should be explored.

Evaluation also poses a challenge for multi-criteria systems. One reason is that not many multi-criteria rating data sets from real-life applications are currently available. This means that in order to test a new multi-criteria recommender system, either experimental data have been collected through pilot user studies (as in the case of [6]) or synthetic (simulated) data sets have to be produced (as in the case of [7]). It would be desirable to have a way to represent, store and reuse multi-criteria ratings data sets, so that they can be shared among the researchers of this community. In addition, a simulation environment that can be used for the production of synthetic data sets to facilitate testing of multi-criteria recommendation algorithms would also be valuable. In this direction, we have implemented the *CollaFiS (Collaborative Filtering Simulation)* online environment for testing MAUT-based collaborative filtering algorithms [8].

Finally, an important limitation is the fact that the number of criteria increases calculation time and poses technical requirements. For instance, extending a product database in order to describe the multiple attributes of all products requires a larger storage space.

4 CONCLUSIONS

This position paper aims to serve as an introduction to the topic of multi-criteria recommender systems. It outlines how the recommendation problem is modelled as a MCDM one, assesses current implementations of MCDM methods in recommender systems, and identifies some open issues for further exploration.

ACKNOWLEDGEMENTS

Part of the work presented in this paper has been funded with support by the European Commission (project No ECP-2006-EDU-410012 Organic.Edunet).

REFERENCES

[1] G. Adomavicius and Y. Kwon, 'New Recommendation Techniques for Multi-Criteria Rating Systems', *IEEE Intelligent Systems*, **22**(3), 48-55, (2007).

[2] 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).

[3] M. Deshpande and G. Karypis, 'Item-based Top-N Recommendation Algorithms', *ACM Transactions on Information Systems*, **22**(1), 143-177, (2004).

[4] J.K Herlocker, J.A. Konstan, L.G. Terveen and J.T. Riedl, 'Evaluating Collaborative Filtering Recommender Systems', *ACM Transactions on Information Systems*, **22**(1), 5-53, (2004).

[5] N. Manouselis and C. Costopoulou, 'An Analysis and Classification of Multi-Criteria Recommender Systems', *World Wide Web Journal*, **10**(4), 415-441, (2007).

[6] N. Manouselis and C. Costopoulou, 'Experimental Analysis of Design Choices in Multi-Attribute Utility Collaborative Filtering', *International Journal of Pattern Recognition and Artificial Intelligence*, **21**(2), 311-331, (2007).

[7] N. Manouselis and C. Costopoulou, *Experimental Analysis of Multiattribute Utility Collaborative Filtering on a Synthetic Data Set*, 111-134, Personalization Techniques and Recommender Systems, Series in Machine Perception and Artificial Intelligence Vol. 70, World Scientific Publishing, (2008).

[8] N. Manouselis and C. Costopoulou, 'Designing a Web-based Testing Tool for Multi-Criteria Recommender Systems', *Engineering Letters*, 13(3), (2006) (http://infolab-dev.aua.gr/files/publications/en/1169683698.pdf).

[9] B. Roy, *Multicriteria Methodology for Decision Aiding*, Kluwer Academic Publishers, 1996.

Plug-in recommending for Eclipse users

Sebastian Draxler [1] and **Hendrik Sander** [2] and **Gunnar Stevens** [3]

Abstract. Products like MS Office, Mozilla Firefox or Eclipse have thousands of extensions, and each may be available in different versions. Because of this it is difficult for end-users to get an overview of the wide range of software extensions. It therefore seems that recommending extensions for component-based applications could become an important area of application for recommender systems. This is especially true for Eclipse, as it is a highly flexible, adaptable and extendible working environment. We therefore conducted a qualitative as well as a quantitative study on the process of configuring Eclipse in practice. We found that working in a team plays an important role in adopting new plug-ins. Mostly co-workers, who work in the same context currently act as 'recommender systems' to find suitable extensions. In this position paper we want to outline our findings in detail and discuss the domain specific issues and opportunities of team oriented recommender systems for plug-in extensions and their appropriation in the context of Eclipse.

1 INTRODUCTION

The Eclipse platform is a very good example for a current trend in software development. Software applications are nowadays often based on a kernel, which can be extended flexibly by additional components, so called plug-ins. For an Eclipse user this provides an opportunity to adapt an Eclipse installation to his work practice and needs. The Eclipse platform is usually delivered as Eclipse SDK, an Integrated Development Environment (IDE) for the Java programming language. This means the common Eclipse user is a software professional. Additional plug-ins can be installed as needed, after unzipping the IDE.

In the context of the CoEUD project we study to which extent recommending techniques can support end users in adapting their working environments to their needs.

In a first step we carried out a qualitative and a quantitative study to ascertain the existence and characteristics of the problem of finding and appropriating Eclipse extensions. We therefore focused on how software developers configure Eclipse as part of a daily work practice. In the qualitative study we cooperated with four different software companies from around 10 up to 250 employees. This Small and Medium Enterprises (SME) character is typical for the German software branch, where the average size of the enterprises is very small [1].

In each company we conducted at least 10 semi-structured interviews. Additionally, we visited two SME for a defined period of time (3-5 days) for a participatory observation [4]. The quantitative study was based on an online survey, which was announced in different online forums, mailing lists and two research institutes. 138 persons participated in the survey and 59 of 138 also sent us their Eclipse configuration data. We analyzed this configuration data in detail, because it showed us exactly which plug-ins users had installed[4]. As some persons had more than one Eclipse installation, we received 76 configurations for our analysis.

Even in the SME software companies we found a huge diversity of Eclipse installations and a highly dynamic evolution of installations.

2 PRACTICES OF ECLIPSE CONFIGURATION

There are many reasons that explain the dynamics and diversity of these installations. For example, we found that the companies we observed did not apply any guidelines or formal regulations for tool configuration, which lead to a certain freedom for the users in configuring their Eclipse installations.

Another fundamental reason for this phenomenon is the project-based organization of software development , which is highly dynamic and diverse itself. Special tools are needed in most projects , e. g. PHP development tools or an XSLT editor. These tools are often not included within default Eclipse IDE, but available as additional components. A typical project started with one or two developers, but the core team sometimes called upon other developers as experts for specific technologies or tasks (e.g. someone who knows a particular database or a UI design tool). As a consequence of this, the consulted experts had to synchronize their working environments with those of their colleagues in order to cooperate with each other.

In addition we found that copying and adapting the working environment is a normal way to instruct new team members:

"*If a new colleague starts working here, I would advise him to begin with the standard IDE, and as a starting point for further exploration, I would show him which extensions I have integrated in Eclipse could be interesting for him (based on his experience).*"

Furthermore, we also observed cross-organizational diffusion of Eclipse plug-ins based on the 'copy and adapt'-practice. This was a result of a semi-outsourcing practice, where developer of different companies work together on one project.

These observations demonstrate that teams are appropriate for recommending suitable plug-ins. We discovered in our online survey that 47% of the respondents mentioned their co-workers and team members as a source of information concerning useful plug-ins for a specific work context. 64% even stated they used the Internet for gathering information (multiple choices were allowed).

In the field of Eclipse, an interesting detail makes this style of working together possible. Most of the components are published under licenses which allows users to share these components with others. This results in an environment in which everyone is entitled

[1] University of Siegen, Germany, email: sebastian.draxler@uni-siegen.de
[2] Fraunhofer FIT, Germany, email: hendrik.sander@fit.fraunhofer.de
[3] Fraunhofer FIT, Germany, email: gunnar.stevens@fit.fraunhofer.de

[4] Eclipse configuration means the data, describing the footprint of the installed plug-ins.

to give away components or complete Eclipse installations ? for example to co-workers.

We found that team members utilized this freedom to recommend or share components, especially when their working context was similar to their co-workers' working contexts.

3 THE DATA OF ECLIPSE CONFIGURATIONS IN PRACTICE

The Eclipse platform applies an '*everything is a plug-in*' philosophy [2]. This means an Eclipse installation is decomposed into hundreds of plug-ins. We have therefore started to analyze sets or configurations of Eclipse plug-ins used in practice.

In a first step we were interested what the different plug-in sets look like in practice and whether this data is rich enough to set up a recommending solution based on it. Even in the small number of 76 configurations (cf. table 1) we were surprised to find 2,429 different plug-ins (4,944 if we take the different versions into account). The average number of 326 plug-ins per configuration gives a first impression of the plug-in network complexity of a typical Eclipse installation.

Table 1. Amount of installed plug-ins in n=76 Eclipse installations.

Overall num. of different plug-ins (disregarding different versions)	2,429
Overall num. of plug-ins (version sensitive)	4,944
Min. num. plug-ins found in an installation	89
Max. num. plug-ins found in an installation	1,008
Average plug-ins per installation	326
Standard deviation σ_p	190

The Eclipse ecosystem is characterized by a network of component vendors. We found that an Eclipse installation usually consists of plug-ins created by many different vendors. The dependencies of these plug-ins can be visualized as an acyclic graph (c.f. figure 1). This shows a complex system of dependencies between plug-ins, but more importantly it reveals complex dependencies between the different plug-in vendors. This makes it hard to choose certain plug-ins and at the same time make sure the installation stays executable.

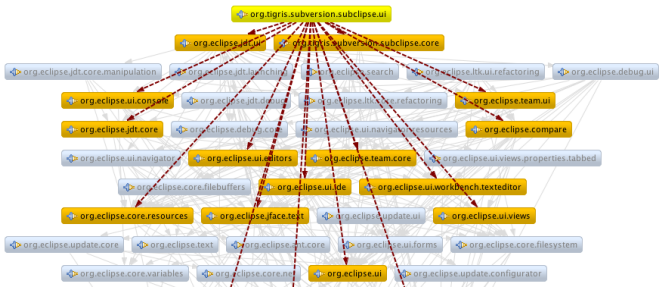


Figure 1. Snapshot of the dependency graph of the 3rd party extension called 'Subclipse', which relies on further plug-ins, created by IBM, Wind River and Embarcadero Technologies, Prosyst and others.

4 DISCUSSION

Thousands of plug-ins exist for Eclipse by now, which makes the management of Eclipse installations a complex task. This is especially important for getting an overview of the most useful plug-ins and their updates in a specific work context. One usual way to solve this problem are software product lines. With this approach a distributor bundles interesting components to a new application for a specific task [3]. This is especially interesting because software product line approaches can benefit from existing components. In the field of Eclipse the job of creating bundles is partly done by the Eclipse foundation by supplying end-users with default configurations (e.g. Java Development, Java EE Development, C++ Development) but users and teams usually adapt these configurations on their own. This means the work of integrating components into the software, which was previously done by distributors, is now shifted to the end-users. On the one hand this gives end-users a new freedom to adapt the software to their needs, but on the other hand we found that a new problem of finding appropriate and compatible components has risen.

The obvious way to support integration work of end users might be to take a look at the Eclipse installations of co-workers who work in the same context and might already have experience with suitable plug-ins.

At the workshop we would like to discuss if and how recommender systems could support users in dealing with this complexity in a similar way as the practice of over-the-shoulder learning [5] we observed in our empirical study of Eclipse. Although we are interested in how recommender systems can support the introduction of new plug-ins which have not yet been adopted by a team member.

The next step for CoEUD will be the implementation of a prototypical, team oriented recommender system for Eclipse plug-ins, based on our empirical studies. First we want to analyze and compare the configuration data (which also holds the data about the installed plug-ins) of different users more deeply. In particular we want to search for patterns in the data of team members as they typically work in the same context. This data might be extended by information which is collected while Eclipse is in use - the upcoming Eclipse Usage Data Collector[5] project is very interesting in this context. Information about the dynamically loaded plug-ins might be of particular importance, because it shows which plug-ins are really used and which are merely installed and remain unused. We are aware that features to enable end-users to tag or rate plug-ins could also be an interesting source of information on how to support end-users in adapting their working environment. In this context we are interested in how recommender systems can include this information or coexist with systems providing this information.

REFERENCES

[1] Michael Friedewald, H. Dieter Rombach, Petra Stahl, Manfred Broy, Susanne Hartkopf, Simone Kimpeler, Kirstin Kohler, Robert Wucher, and Peter Zoche, 'Softwareentwicklung in deutschland, eine bestandsaufnahme bestandsaufnahme (software development in germany. a survey)', *Informatik Spektrum*, **24**(2), 81–90, (2001).

[2] Erich Gamma and Kent Beck, *Contributing to Eclipse: Principles, Patterns, and Plugins*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2003.

[3] Linda M. Northrop, 'Sei's software product line tenets', *IEEE Softw.*, **19**(4), 32–40, (2002).

[4] Gunnar Stevens, Sebastian Draxler, and Tobias Schwartz. Appropriation: A work practice perspective to provisioning. http://www.eclipsecon.org/summiteurope2007/presentations/ESE2007-Appropriation_Perspective_to_Provisioning.pdf, 2007.

[5] Michael B. Twidale, 'Over the shoulder learning: Supporting brief informal learning', *Computer Supported Cooperative Work (CSCW)*, **14**(6), 505–547, (2005).

[5] http://www.eclipse.org/epp/usagedata/

Exploring the Support for Spoken Natural Language Explanations in Inference Web

Tejaswini Narayanan [1] and **Deborah L. McGuinness** [2]

Abstract

In this position-paper, we have outlined our initial explorations into provisioning for Spoken Natural Language Explanations in Inference Web (IW). We present our motivation for considering speech as an effective modality for presenting explanations, in terms of the interesting use-cases that we believe are enabled by the addition of this modality. We then describe the design and implementation of our prototype system. We highlight the salient features of our approach using an illustrative running example.

Author Keywords

Speech user-interfaces, explanation systems, semantic web

1 INTRODUCTION

Inference Web (IW) is a framework for explaining Semantic Web reasoning tasks [1]. IW leverages Semantic Web technologies and recommended standards such as the Web Ontology Language (OWL) [3], and provides a range of tools to support rich explanations. A reasoning task within IW is represented by a "Proof", and is encoded in a proof interlingua called Proof Markup Language (PML) [2]. A PML Proof describes the inference steps used to derive an answer for a proposed query. In addition, a PML Proof can also contain rich Provenance Metadata, which may be used in conjunction with the reasoning process captured in the proof-steps, to generate intuitive explanations and support follow-up questions.

1.1 Running example: Explaining KSL Wine Agent's wine recommendation for a meal

We use the KSL Wine Agent [4] as the running example to illustrate the features of our prototype. The Wine Agent leverages a Knowledge-Base (KB) of wines that is structured using an OWL ontology. Using a logical reasoning-engine on top of this KB, the Wine Agent can make recommendations for specific varieties of wine that pair well with specific varieties of food. The IW infrastructure can be used to provide explanations for the recommendations produced by the Wine Agent.

For example, the Wine Agent may make a recommendation of a white-wine drink for a Crab-dish based on the following assertions: (i) It is known that white-wine pairs well with seafood. (ii) It is known that crab is a type of seafood. Therefore, white-wine may be a suitable recommendation for a crab-dish. If this reasoning process is encoded in PML (together with Provenance Metadata about the sources of the assertions), then the IW tools can be used to effectively present an explanation to a curious user of the KSL Wine Agent, who may be interested in knowing *how* or *why* the system made a specific wine-recommendation.

1.2 Existing support for explanations in IW

The existing support for explanations in IW is provided through a set of tools. One specific tool that we will discuss here is called the *IW Browser (IW Toolkit, IW Explainer* being other such tools*)*. The IW Browser provides a mechanism for "browsing" PML proofs on a rendering surface. The presentation is made more intuitive by supplying suitable provenance data/metadata to the user (obtained from the underlying PML Proofs), in the *context* of the user's browsing operation.

Currently, the IW Browser offers three different 'Styles' for presenting proofs and explanations. Each Style enables a unique set of interactions between the user and the proof being browsed, thus facilitating easier understanding of the proof / explanation. The existing Styles are illustrated in Segments 1–3 in Figure 1.

2 EXPLORING SUPPORT FOR SPOKEN NATURAL LANGUAGE EXPLANATIONS: THE "SPOKEN" STYLE

We observe that the logic-sentences representing antecedents/conclusions of a PML Proof in IW are encoded in Knowledge Interchange Format (KIF). The IW infrastructure already provides support for textual Natural Language (English) explanations using a heuristic, case-driven KIF to English translator. For example, one may note that the translator component participates in the Narrative Style rendering of a proof in the IW Browser (Fig 1, Seg 3).

We propose to leverage this existing explanation support structure in IW, and extend the IW infrastructure to provide for Spoken Natural Language (English) explanations. We believe that this extension would enable some interesting explanation use-cases, viz. (i) Non-literate users (say, in developing countries) who may not be able read and comprehend a narrative explanation, can benefit from a spoken dialogue explanation (ii) Spoken explanations can also serve as an assistive technology for visually-challenged users (iii) The "Spoken" Style can effectively *complement* the Narrative Style for novice end-users.

[1] SV National Institute of Tech., India, email: tejaswini5@yahoo.com

[2] Rensselaer Polytechnic Institute, New York, email: dlm@cs.rpi.edu

This work was initiated while the authors were both at the Knowledge Systems Lab (KSL), Stanford University (Dr. McGuinness, as a senior research scientist, and Tejaswini Narayanan, as a Summer Visitor)

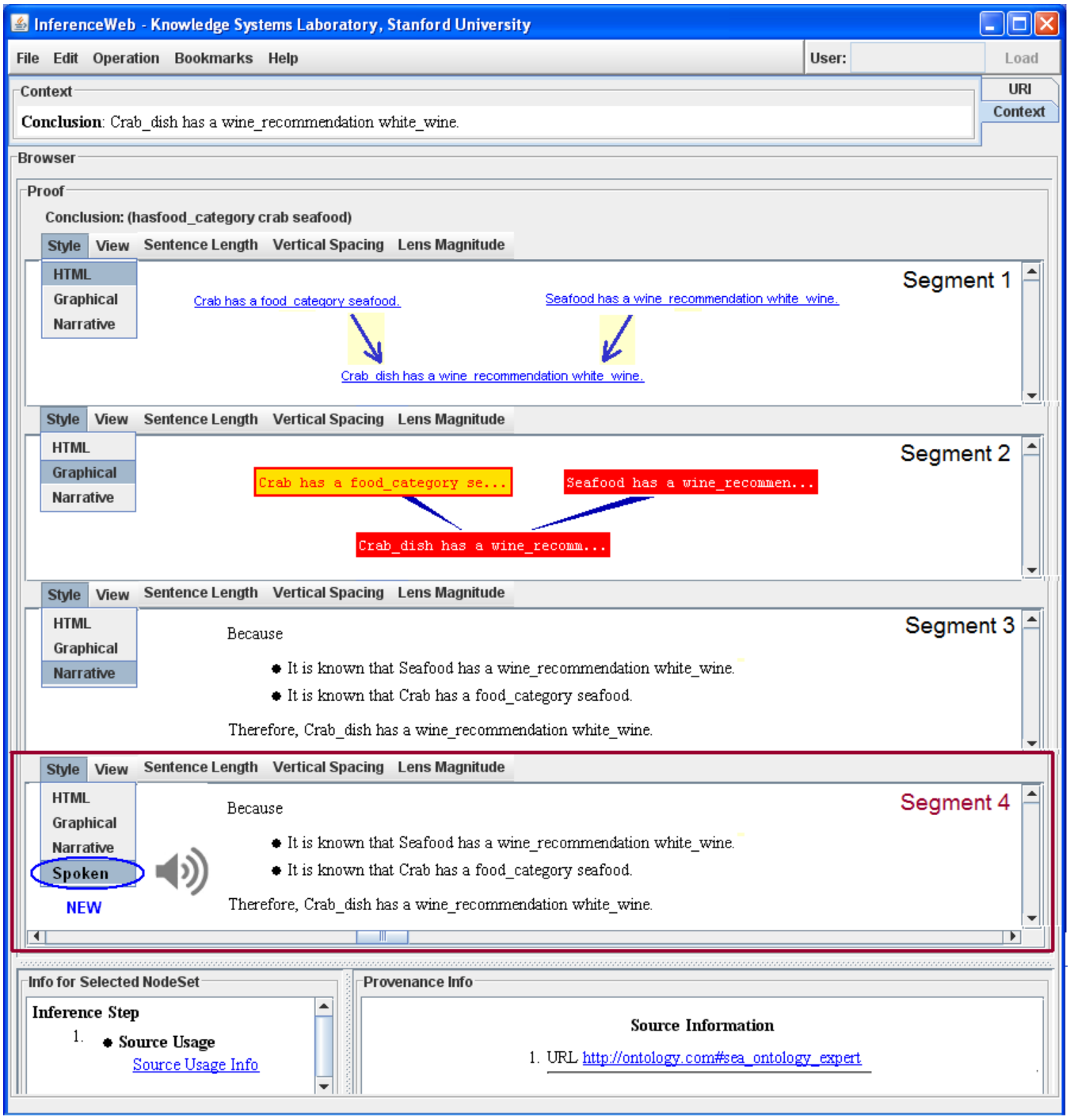


Figure 1: IW Browser: **Segment 1**: HTML Style (Static Presentation), **Segment 2**: Graphical Style (Dynamic Presentation),

Segment 3: Narrative Style (Sentential Presentation), **Segment 4**: New "Spoken" Style (Multi-Modal Presentation)

3 IMPLEMENTATION BRIEF

We leverage the FreeTTS 1.2 text-to-speech libraries [5] for integrating spoken natural language explanation support in the IW infrastructure. We then expose this functionality through the "Spoken" Proof Style, which is wired into the IW Browser as an additional Style presentation module. Segment 4 illustrates the enhanced IW Browser with the additional "Spoken" functionality.

SELECTED REFERENCES

[1] McGuinness, D.L et. al., Explaining Answers from the Semantic Web: The Inference Web Approach, *Intl. Semantic Web Conference, 2003*

[2] McGuinness, D.L et. al., PML 2: A Modular Explanation Interlingua, *Workshop on Explanation-aware Computing (ExaCt-2007), 2007.*

[3] McGuinness, D.L et. al., OWL Overview, *W3C Recommendation, 2004*

[4] KSL Wine Agent, http://onto.stanford.edu:8080/wino/

[5] FreeTTS 1.2, http://freetts.sourceforge.net