A Proposal of Discovering User Interest by Support Vector Machine and Decision Tree on Document Classification

Loc Nguyen
The University of Science, Ho Chi Minh city, Vietnam
E-mail: ng_phloc@yahoo.com

Abstract

User interest is one of personal traits attracting researchers' attention in user modeling and user profiling. User interest competes with user knowledge to become the most important characteristic in user model. Adaptive systems need to know user interests so that provide adaptation to user. For example, adaptive learning systems tailor learning materials (lesson, example, exercise, test…) to user interests. We propose a new approach for discovering user interest based on document classification. The basic idea is to consider user interests as classes of documents. The process of classifying documents is also the process of discovering user interests. There are two new points of view:

- *The series of user access in his/her history are modeled as documents. So user is referred indirectly to as "document".*
- *User interests are classes such documents are belong to.*

1. Introduction

Our approach includes four following steps:

1. Documents in training corpus are represented according to vector model. Each element of vector is product of term frequency and inverse document frequency. However the inverse document frequency can be removed from each element for convenience.
2. Classifying training corpus by applying decision tree or support vector machine. Classification rules (weight vectors W^*) are drawn from decision tree (support vector machine). They are used as classifiers.
3. Mining user's access history to find maximum frequent itemsets. Each itemset is considered a interesting document and its member items are considered as terms. Such interesting documents are modeled as vectors.
4. Applying classifiers (see step 2) into these interesting documents in order to choose which classes are most suitable to these interesting documents. Such classes are user interests.

This approach bases on document classification but it also relates to information retrieval in the manner of representing documents. Hence section 2 discusses about vector model for representing documents. Support vector machine and decision tree on document classification are mentioned in section 3, 4. Main technique to discover user interest is described in section 5. Section 6 is the conclusion.

2. Vector model for representing documents

Suppose our corpus **D** is the composition of documents $D_i \in \mathbf{D} = \{D_1, D_2,\ldots, D_m\}$. Every document D_i contains a set of key words so-called *terms*. The number of times a term occurs in a document is called *term frequency*. Given the document D_i and term t_j, the term frequency tf_{ij} measuring the importance of term t_j within document D_i is defined as below: $tf_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$

Where n_{ij} is the number of occurrences of term t_j in document D_i, and the denominator is the sum of number of occurrences of all terms in document D_i.

Suppose we need to search documents which are most relevant to the query having term t_j. The simple way is to choose documents which have highest term frequency tf_{ij}. However in situation that t_j is not a good term to distinguish between relevant and non-relevant documents and other terms occurring rarely are better ones to distinguish between relevant and non-relevant documents. This will tend to incorrectly emphasize documents containing term t_j more, without giving enough weight to other meaningful terms. So the *inverse document frequency* is a measure of general importance of the term. It is used to decrease the weight of terms occurring frequently and increase the weight of terms occurring rarely. The inverse document frequency of term t_j is the ratio of the size of corpus to the number of documents that t_j occurs.

$$idf_j = \log \frac{|corpus|}{|\{D: t_j \in D\}|}$$

Where $|corpus|$ is the total number of documents in corpus and $|\{D: t_j \in D\}|$ is the number of documents containing term t_j. We use *log* function to normalize idf_j so that it is less than or equal *1*.

The weight of term t_j in document D_i is defined as product of tf_{ij} and idf_i

$w_{ij} = tf_{ij} * idf_i$

This weight measure the importance of a term in a document over the corpus. It increases proportionally to the number of times a term occurs in the document but is offset by the frequency of this term in the corpus. In general this weight balances the importance of two measures: term frequency and inverse document frequency.

Suppose there are n terms $\{t_i, t_2, \ldots, t_n\}$, each document D_i is modeled as the vector which is composed of weights of such terms.

$D_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{in})$

Hence the corpus becomes a matrix m x n, which have m rows and n cols with respect to m document and n terms. D_i is called document vector.

The essence of document classification is to use supervised learning algorithms in order to classify corpus into groups of documents; each group is labeled. In this paper we apply two methods namely support vector machine and decision tree for document classification.

3. Document classification based on Support Vector Machine

Support vector machine (SVM) [1] is a supervised learning algorithm for classification and regression. Given a set of n-dimensional vectors in vector space, SVM finds the separating hyper-plane that splits vector space into sub-set of vector; each separated sub-set (so-called data set) is assigned one class. There is the constraint for this separating hyper-plane: "it must maximize the margin between two sub-sets".

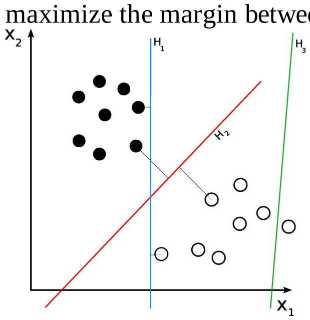

Figure 1: Separating hyper-planes

Suppose we have some n-dimensional vectors; each of them belongs to one of two classes. We can find many n-*1* dimensional hyper-planes that classify such vectors but there is only one hyper-plane that maximizes the margin between two classes. In other words, the nearest between a point in one side of this hyper-plane and other side of this hyper-plane is maximized. Such hyper-plane is called maximum-margin hyper-plane and it is considered as maximum-margin classifier.

Let $\{X_1, X_2, \ldots, X_n\}$ be the training set of vectors and let $y_i = \{1, -1\}$be the class label of vector X_i. It is necessary to determine the maximum-margin hyper-plane that separates vectors belonging to y_i=1 from vectors belonging to y_i = -1. This hyper-plane is written as the set of point satisfying:

$W^T \otimes X_i + b = 0$ (1)

Where $\otimes$ denotes the scalar product and W is a weight vector perpendicular to hyper-plane and b is the bias. W is also called perpendicular vector or normal vector. It is used to specify hyper-plane.

The value $\frac{b}{|W|}$ is the offset of the hyper-plane from the origin along the weight vector W.

To calculate the margin, two parallel hyper-planes are constructed, one on each side of the maximum-margin hyper-plane. Such two parallel hyper-planes are represented by two following equations:

$W^T \otimes X_i + b = 1$

$W^T \otimes X_i + b = -1$

To prevent vectors falling into the margin, all vectors belonging to two class $y_i=1$, $y_i=-1$ have two following constraints respectively:

$W^T \otimes X_i + b \geq 1$ (*for X_i of class y_i=1*)

$W^T \otimes X_i + b \leq -1$ (*for X_i of class y_i=-1*)

These constraints can be re-written as:

$Y_i (W^T \otimes X_i + b) \geq 1$ (2)

For any new vector X, the rule for classifying it is computed as below:

$f(X_i) = \text{sign}(W^T \otimes X_i + b) \in \{\leq -1, \geq 1\}$ (3)

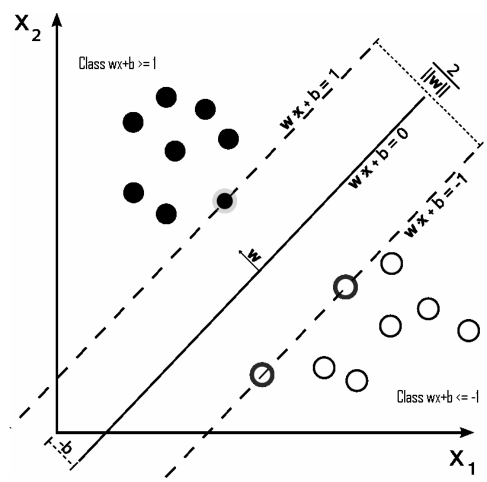

Figure 2: Maximum-margin hyper-plane

Because maximum-margin hyper-plane is defined by weight vector W, it is easy to recognize that the essence of constructing maximum-margin hyper-plane is to solve the following constrained optimization problem:

$$\underset{W,b}{Minimize}\ \frac{1}{2}|W|^2 \text{ subject to } y_i\ (W^T \otimes X_i + b) \geq 1,\ \forall i$$

This constraints can be re-written as below:

$$\underset{W,b}{Minimize}\frac{1}{2}|W|^2 \text{ subject to } 1 - y_i\ (W^T \otimes X_i + b) \leq 0,\ \forall i$$

(4)

The reason of minimize $\frac{1}{2}|W|^2$ is that distance between two parallel hyper-planes is $\frac{2}{|W|}$ and we need to maximize such distance in order to maximize the margin for maximum-margin hyper-plane. Then maximizing $\frac{2}{|W|}$ is to minimize $\frac{1}{2}|W|$. Because |W| is the norm of W being complex to compute, we substitute $\frac{1}{2}|W|$ with $\frac{1}{2}|W|^2$. This is the *constrained optimization problem* or *quadratic programming* (QP) *optimization* problem. Note that $|W|^2$ can be computed by scalar product of it and itself:

$$|W|^2 = W^T \otimes W$$

The way to solve QP problem in formula 2 is through its Lagrange dual. Suppose we want to minimize *f(x)* subject to *g(x)=0*, it exists a solution x_0 to set of equations:

$$\begin{cases} \frac{\partial}{\partial x}(f(x) + \alpha g(x))_{x=x0} = 0 \\ g(x) = 0 \end{cases}$$ where α is Lagrange multiplier.

For multi constraints $g_i(x) = 0$ ($i=\overline{1,n}$), there is a Lagrange multiplier for each constraint

$$\begin{cases} \frac{\partial}{\partial x}(f(x) + \sum_{i=1}^{n}\alpha_i g_i(x)_{x=x0} = 0 \\ g_i(x) = 0 \end{cases}$$

In case of $g_i(x) \leq 0$, there is no change. If x_0 is a solution to the constrained optimized problem:

$$\underset{x}{Minimize}\ f(x) \text{ subject to } g_i(x) \leq 0\ \forall i$$

Then there must exist $\alpha_i \geq 0\ \forall i$ so that x_0 satisfies following equations:

$$\begin{cases} \frac{\partial}{\partial x}(f(x) + \sum_{i=1}^{n}\alpha_i g_i(x)_{x=x0} = 0 \\ g_i(x) \leq 0 \end{cases} \quad (5)$$

The function $f(x) + \sum_i \alpha_i g_i(x)$ is called Lagrangian denoted *L*. Let $f(X_i)$ be $\frac{1}{2}|W|^2$ and let $g_i(X)$ be $1 - y_i(W^T \otimes X_i + b)$, the constraint in (4) becomes:

$$\underset{\lambda}{Maximize}(\underset{W,b}{Minimize}L(W,b,\lambda)) \quad (6)$$

where

$$L(W,b,\lambda) = \frac{1}{2}|W|^2 + \sum_{i=1}^{n}\lambda_i(1 - y_i(W^T \otimes X_i + b)) \quad (7)$$

The Lagrangian *L* is minimized with respect to the primal variables *W* and *b* and maximized with respect to the dual variables λ. Setting the gradient of *L* w.r.t *W* and *b* to zero, we have

$$\begin{cases} \frac{\partial L(W,b,\lambda)}{\partial W} = 0 \\ \frac{\partial L(W,b,\lambda)}{\partial \lambda} = 0 \end{cases} \Leftrightarrow \begin{cases} W - \sum_{i=1}^{n}\lambda_i y_i X_i = 0 \\ \sum_{i=1}^{n}\lambda_i y_i = 0 \end{cases} \quad (8)$$

Substituting (8) into (7) we have:

$$L(\lambda) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{1}\lambda_i\lambda_j y_i y_j X_i^T X_j \quad (9)$$

Where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$

If (9) is re-written in matrix notation then the constrained optimization problem in (4), (6) leads to dual problem:

$$Maximize\quad L(\lambda) = \lambda^t I - \frac{1}{2}\lambda^T S\lambda \text{ subject to } \lambda \geq 0 \text{ and } \lambda^T y = 0 \quad (10)$$

Where S is a symmetric $n_x n$ matrix with elements $s_{ij}=y_i y_j X_i^T X_j$.

Suppose λ^* is a solution of (1) with condition (8). In order words, $L(\lambda^*)$ is maximized; so the weight vector representing maximum-margin hyper-plane is recovered by λ^* and X_i:

$$W^* = \sum_{i=1}^{n}\lambda_i y_i X_i \quad (11)$$

So the bias b is computed as below:

$$b^* = y_i - W^{*T} \otimes X_i \quad (12)$$

The rule for classification in (3) becomes:

$$f(X_i) = R = \text{sign}(W^{*T} \otimes X_i + b^*) \quad (13)$$

$$\mathbf{X} \longrightarrow \boxed{\mathbf{f}} \longrightarrow \mathbf{y} \in \{-1, 1\}$$

Figure 3: Classification function

It means that whenever we need to determine to which class a new vector X_i belongs, it is only to substitute X_i into $W^{*T} \otimes X_i + b^*$ and check the value of this expression. If the value is less than or equal *-1* (≤ -1) then X_i belongs to class $y_i = -1$. Otherwise, if the value is greater than or equal *1* (≥ 1) then X_i belongs to class y_i =

1. Hence the function ($W^{*T} \otimes X_i + b^*$) is called classification function or classification rule.

The Lagrange multipliers are non-zero when $W^T \otimes X_i + b$ is equal *1 or -1*, vectors X_i in this case are considered support vectors they are closest to the maximum-margin hyper-plane. These vectors lie on parallel hyper-planes. So this approach is called support vector machine.

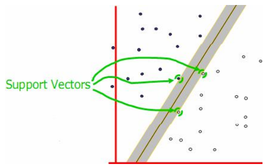

Figure 4: Support vectors

However the way to find λ^* in (10) is quadratic programming (QP) problem. There are many approaches to sole this problem; for instance, the sequential minimal optimization is described as below:

- A QP with two variables is trivial to solve
- At each iteration, a pair of (λ_i, λ_j) is picked up and QP is solved with these variables; repeat until convergence

3.2. Applying Support Vector Machine into document classification

Give corpus **D** = $\{D_1, D_2, D_3,\ldots, D_m\}$ in which every document D_i is modeled by a tf-idf weight vector. Suppose there are *n* terms $\{t_1, t_2,\ldots, t_n\}$, we have:

$D_i = (d_{i1}, d_{i2},\ldots, d_{in})$ where d_{ij} is product of term frequency and inverse document frequency $d_{ij} = tf_{ij}*idf_j$

If the index of document is ignored, document *D* is represented as below: $D = (d_1, d_2,\ldots, d_n)$

Given *k* classes $\{l_1, l_2,\ldots, l_k\}$, there is demand of classifying documents into such classes. The technique of classification based on SVM is *two-class* classification in which the classes is +1, -1 for $y_i=+1, -1$ respectively. So we need to determine unique hyper-plane referred to as *two-class* classifier. It is possible to extend *two-class* classification to *k-class* classification by constructing *k* *two-class* classifier. In means that we must specify *k* couple of optimal weight vector W_i^* and bias b_i^*. Each couple (W_i^*, b_i^*) being a *two-class* classifier is the representation of class l_i. The process of finding (W_i^*, b_i^*) in training corpus **D** is described in the section of support vector machine.

Class	Weight vector	Bias	Classification rule
l_1	W_1^*	b_1^*	$R_1= W_1^{*T} \otimes X + b_1^*$
l_2	W_2^*	b_2^*	$R_2= W_2^{*T} \otimes X + b_2^*$
…	…	…	…
l_k	W_k^*	b_k^*	$R_k= W_k^{*T} \otimes X + b_k^*$

Table 1: *k* couple (W_i^*, b_i^*) corresponds with *k* class $\{l_1, l_2,\ldots, l_k\}$

For example, classifying document $D= (d_1, d_2,\ldots, d_n)$ is described as below:

1. For each classification rule $R_i = W_i^{*T} \otimes X + b_i^*$, substituting each D into such rule. It means that vector *X* in such rule is replaced by document *D*.
   $Expression_i = W_i^{*T} \otimes D + b_i^*$
2. Suppose there is a sub-set of rules $\{R_{h+1}, R_{h+2},\ldots, R_{h+r}\}$ that the value of expression $Expression_i = W_i^{*T} \otimes D + b_i^*$ is greater than or equal *1*. We can conclude that document *D* is belongs to *r* classes $\{l_{h+1}, l_{h+2},\ldots, l_{h+r}\}$ where $\{l_{h+1}, l_{h+2},\ldots, l_{h+r}\} \subseteq \{l_1, l_2,\ldots, l_k\}$

Classification Expression	Value
$W_1^{*T} \otimes D + b_1^*$	$\geq 1 \; : D \in l_1$ $\leq -1 : D \notin l_1$
$W_2^{*T} \otimes D + b_2^*$	$\geq 1 \; : D \in l_2$ $\leq -1 : D \notin l_2$
…	…
$W_k^{*T} \otimes D + b_k^*$	$\geq 1 \; : D \in l_k$ $\leq -1 : D \notin l_k$

Table 2: Classifying document *D*

4. Document classification based on Decision Tree

Given a set of classes **C** = {*computer science*, *math*}, a set of terms **T** = {*computer*, *programming language*, *algorithm*, *derivative*} and the corpus **D** = {*doc1.txt*, *doc2.txt*, *doc3.txt*, *doc4.txt*, *doc5.txt*}. The training data is showed in following table in which cell (*i*, *j*) indicates the number of times that term *j* (column *j*) occurs in document *i* (row *i*).

	computer	*programming language*	*algorithm*	*derivative*	***class***
doc1.txt	5	3	1	1	computer
doc2.txt	5	5	40	5	math
doc3.txt	20	5	20	55	math
doc4.txt	20	55	5	20	computer
doc5.txt	15	15	4	0.3	math
doc6.txt	35	10	45	10	computer

Table 3: Term frequencies of documents

	computer	*programming language*	*algorithm*	*derivative*	**class**
doc1.txt	0.5	0.3	0.1	0.1	computer
doc2.txt	0.05	0.05	0.4	0.5	math
doc3.txt	0.2	0.05	0.2	0.55	math
doc4.txt	0.2	0.55	0.05	0.2	computer
doc5.txt	0.15	0.15	0.4	0.3	math
doc6.txt	0.35	0.1	0.45	0.1	computer

Table 4: Normalized term frequencies

Because the expense of real number computation is so high, all term frequencies are changed from real number into nominal value:

1. $0 \le frequency < 0.2$: *low*
2. $0.2 \le frequency < 0.5$: *medium*
3. $0.5 \le frequency$: *high*

	computer	*programming language*	*algorithm*	*derivative*	**class**
doc1.txt	high	medium	low	low	computer
doc2.txt	low	low	medium	high	math
doc3.txt	medium	low	medium	high	math
doc4.txt	medium	high	low	medium	computer
doc5.txt	low	low	medium	medium	math
doc6.txt	medium	low	medium	low	computer

Table 5: Nominal term frequencies

The basic idea of generating decision tree [2] is to split the tree into two sub-trees at the most informative node. Such node is chosen by computing its entropy or information gain. Following figure shows the decision tree generated from our training data.

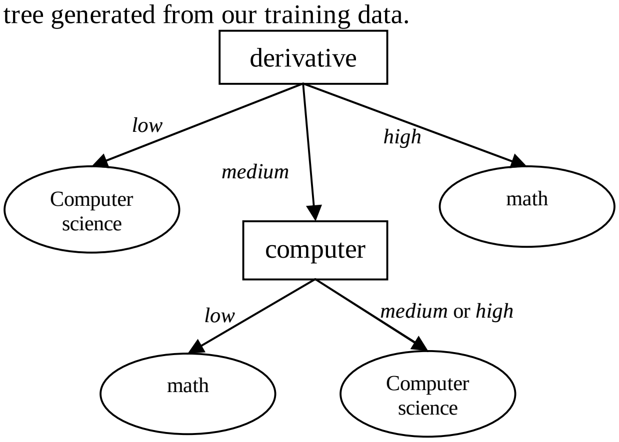

Figure 5: Decision tree

We can extract classification rules from this decision tree:

Rule 1	If frequency of term "derivative" is *low* then document belongs to class *computer science*
Rule 2	If frequency of term "derivative" is *medium* and frequency of term "computer" is *medium* or *high* then document belongs to class *computer science*
Rule 3	If frequency of term "derivative" is *medium* and frequency of term "computer" is *low* then document belongs to class *math*.
Rule 4	If frequency of term "derivative" is *high* then document belongs to class *math*

Table 6: Classification rules deriving from decision tree induction

Suppose the numbers of times that terms *computer*, *programming language*, *algorithm* and *derivative* occur in document *D* are 5, 1, 1, and 3 respectively. We need to determine which class document *D* is belongs to. *D* is normalized as term frequency vector.

D = (*0.5*, *0.1*, *0.1*, *0.3*)

Changing real number into nominal value, we have:

D = (*high, low, low, medium*)

According to rule 2 in above table, *D* is *computer science* document because in document vector *D*, frequency of term "derivative" is *medium* and frequency of term "computer" is *high*.

5. Discovering user interest based on document classification

Suppose in some library or website, user *U* do his search for his interesting books, documents… There is demand of discovering his interests so that such library or website can provide adaptive documents to him whenever he visits in the next time. This is adaptation process in which system tailors documents to each individual. Given there is a set of key words or terms {*computer, programming language, algorithm, derivative*} that user *U* often looking for, the his searching history is showed in following table:

Date	*Keywords (terms) searched*
Aug 28 10:20:01	computer, programming language, algorithm, derivative
Aug 28 13:00:00	computer, programming language, derivative, algorithm
Aug 29 8:15:01	computer
Aug 30 8:15:06	computer

Table 7: User's searching history

This history is considered as training dataset for mining maximum frequent itemsets. The keywords are now considered items. An itemset is constituted of some items. The support of itemset *x* is defined as the fraction of total transaction which containing *x*. Given support threshold *min_sup*, the itemset *x* is called *frequent itemset* if its support satisfies the support threshold ($\ge$*min_sup*). Moreover x is *maximum frequent itemset* if *x* is frequent itemset and all super-itemsets of x are not frequent. Note that *y* is super-itemset of x if $x \subset y$. The itemset that has *k* items is called *k-itemset*. Tabel 9 shows the supports of *1*-itemsets

1-itemset	*support*
computer	4
programming language	2
algorithm	2
derivative	2

Table 8: *1-itemsets*

Applying algorithm Apriori, it is easy to find maximum frequent itemsets given *min_sup* = 2. The maximum frequent itemset that user searches are showed in below table:

N_o	*itemset*
1	computer, programming language, algorithm, derivative

Table 9: The maximum frequent itemset that user searches

We propose the new point of view: "**The maximum frequent itemsets are considered as documents and the classes of such documents are considered as user interests**". Such documents may be called interesting documents. Which classes such interesting documents belong to are user interests. It means that discovering user's interests involves in classifying interesting documents. Suppose we have a set of classes **C** = {*computer science, math*}, a set of terms **T** = {*computer, programming language, algorithm, derivative*} and the set of classification rules in table 6. Each maximum frequent itemset that user searches is modeled as a document vector whose elements are the support of its member items. Note that the supports of such items are showed in table 8.

N_o	*vector*
1	(computer=4, programming language=*2*, algorithm=*2*, derivative=*2*)

Table 10: Interesting document vectors

N_o	*vector*
1	(computer=*0.4*, programming language=*0.2*, algorithm=*0.2*, derivative=*0.2*)

Table 11: Interesting document vectors are normalized

N_o	*vector*
1	(computer=*medium*, programming language=*medium*, algorithm=*medium*, derivative=*medium*)

Table 12: Nominal interesting document vectors

It is possible to use SVM or decision tree to classify documents. Hence we use decision tree as sample classifier for convenience because we intend to re-use classification rules in section III. Otherwise we must determine the weight vector W^* if applying SVM approach. However SVM approach is more powerful than decision tree with regard to document classification in case of huge training data.

Applying classification rule *2*, the interesting document belongs to class *compute science* because the frequency of "derivative" and "computer" are *medium* and *medium*, respectively. So we can state that user *U* has only one interest: *computer science*.

6. Conclusion

Our approach includes following steps:

- Documents are represented as vectors
- Classifying documents by using decision tree or support vector machine
- Mining user's access history to find maximum frequent itemsets. Each itemset is considered a interesting document
- Applying classifiers into interesting documents in order to find their suitable classes. These classes are user interests.

Two new points of view are inferred from these steps:

- The series of user access in his/her history are modeled as documents. So user is referred indirectly to as document.
- User interests are classes that such documents are belong to

The technique of constructing vector model for representing document is not important to this approach. There are some algorithms of text segmentation for specifying all terms in documents. From this, it is easy to build up document vectors by computing term frequency and inverse document frequency. However the concerned techniques of document classification such as SVM, decision tree influence extremely on this approach. SVM is more effective than decision in case of huge training data set but it is not convenient for applying classifiers (weight vector W^*) into determining the classes of documents. Otherwise it is easy to use classification rules taking out from decision tree for this task.

Reference

[1]. N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines. Cambridge University Press, 2000.

[2]. T. Mitchell. Machine Learning. McGraw-Hill International, 1997

[3]. C.Cortes and V.Vapnik. Support vector networks. Machine Learning, 20:273–297, 1995

[4]. Mohammad Alrifai, Peter Dolog, Wolfgang Nejdl. Learner Profile Management for Collaborating Adaptive eLearning Applications. APS'2006: Joint International Workshop on Adaptivity, Personalization and the Semantic Web at the 17th ACM Hypertext'06 conference, Odense, Denmark, August 2006

[5]. Christos Papatheodorou. Machine Learning in User Modeling. Lecture Notes in Artificial Intelligence (LNAI), No 2049: Springer-Verlag, 2001, pp. 286-294