International Journal of Systems Science
Vol. 37, No. 13, 20 October 2006, 905–918

Text learning for user profiling in e-commerce

M. DEGEMMIS*, P. LOPS, S. FERILLI, N. DI MAURO,
T. M. A. BASILE and G. SEMERARO

Dipartimento di Informatica, Università di Bari, Via E. Orabona, 4 – 70125 Bari, Italia

(*Received 16 February 2006; in final form 7 April 2006*)

Exploring digital collections to find information relevant to a user's interests is a challenging task. Algorithms designed to solve this *relevant information problem* base their relevance computations on *user profiles* in which representations of the users' interests are maintained. This article presents a new method, based on the classic Rocchio algorithm for text categorization, able to discover user preferences from the analysis of textual descriptions of items in online catalog of e-commerce Web sites. Experiments have been carried out on several data sets, and results have been compared with those obtained using an inductive logic programming (ILP) approach and a probabilistic one.

Keywords: User modeling; Machine learning; Content-based filtering; Text categorization; Word Net

1. Introduction

E-commerce sites often recommend products they believe a customer is interested in buying. Often users are swamped with (product) information and have difficulty in separating relevant from irrelevant information. Many Web sites have started to embody recommender systems as a way of personalizing their content for users. Recommendation algorithms use input about a customer's interests to generate a personalized list of recommended items. A possible way to achieve personalization is to use static profiles that must be manually updated by users when their interests change. These limitations clearly call for alternative methods that infer preference information implicitly and support automated content recommendation. Machine learning techniques are being used to recognize regularities in the behavior of customers and to infer a model of their interests, referred to as a user profile.

This article presents a new method, based on the classic Rocchio algorithm for text categorization (Rocchio 1971), able to discover user preferences from the analysis of textual descriptions of items in the catalog of an e-commerce Web site. The novelty of the method can be summarized as follows:

(a) Positive and negative examples are weighted differently for each user, according to the rates given during the training phase. The classic Rocchio method uses specific control parameters that allow setting the relative importance of *all* positive and negative examples;

(b) The method is able to manage documents structured in different slots, each corresponding to a specific feature of an item, for example, title, authors, and abstract. This strategy permits us to give a different weight to words on the basis of the slot in which they appear, according to the idea that words appearing in the title may be more indicative of user preferences than words appearing in the body of the document.

In order to evaluate the effectiveness of the proposed approach, a comparison with different learning strategies has been carried out, namely an ILP approach and a naïve Bayes method. Our experiments evaluated the effects of the aforementioned methods in learning intelligible profiles of users' interests. The experiments were conducted on two data

*Corresponding author. Email: degemmis@di.uniba.it

International Journal of Systems Science
ISSN 0020–7721 print/ISSN 1464–5319 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207720600891794

sets: the first one in the context of a content-based profiling system for virtual bookshop on the World Wide Web, the other concerning recommendation services for movie sites. In these scenarios, a client side utility has been developed in order to download items (book or movie descriptions) for a user from the Web and to capture users' feedback regarding his liking/disliking on the downloaded items. Then, this knowledge can be exploited by the three different learning techniques so that when a trained system encounters a new item it can intelligently infer whether this new item will be liked by the user or not.

This article is organized as follows: section 2 describes the main principles for learning user profiles from textual description. Section 3 describes in more details the new Rocchio-based algorithm for inferring user profiles and gives an overview of the systems used for comparison. Section 4 presents the detailed description of the experiments. Finally, some conclusions are drawn in section 5.

2. Learning user profiles from textual descriptions

Recent research on intelligent information access and recommender systems has focused on the content-based information recommendation paradigm that exploits textual descriptions of the items to be recommended and relevance ratings given by users to infer a profile of user interests (Mladenic, 1999). Text categorization is commonly described as follows: given a set of classes $C = \{c_1, \ldots, c_n\}$ and a set of training documents labeled with the class the document belongs to, the problem is in building a classifier able to assign to a new document the proper class. We consider the problem of learning user profiles as a binary classification task: each document has to be classified as interesting or not with respect to the user preferences. The set of classes is restricted to c_+, the positive class (user likes), and c_-, the negative one (user dislikes). The application of text categorization methods to the problem of learning user profiles is not new: several experiments have shown that the naïve Bayesian classifier offers several advantages over other learning algorithms (Mooney and Roy 2000; Pazzani and Billsus 1997). Thus, we compared the proposed Rocchio-based algorithm with the naïve Bayesian classifier implemented in our Item Recommender system. Moreover, our research aims at comparing these techniques with a symbolic approach able to induce profiles that are more readable from a human understandability viewpoint.

2.1 *Documents representation*

The representation that dominates the text classification literature is known as *bag of words* (BOW). In this approach, each feature corresponds to a single word found in the training set. Usually a list of *stop words* that are assumed to have no information content is removed from the original text. In order to make the features statistically independent, typically a *stemming* algorithm is used to remove suffixes from words. In our application scenarios, items to be recommended are books or movies. Each item is represented by a set of *slots*, where each slot is a textual field corresponding to a specific feature of the item: title, author, and textual annotation, that is the abstract of the book, for books; title, cast, director, keywords, and summary for movies. The text in each slot is represented using the *BOW* model taking into account the occurrences of words in the original text. Thus, each instance is represented by a list of BOWs, one BOW for each slot. This strategy considers separately the occurrences of a word in the slots in which it appears. The idea behind this approach is that counting the number of occurrences separately in each slot could supply a more effective way to discover the informative power of a word in a document describing the item. Stemming and stop words removal have been applied to the documents used in the experiments.

2.2 *Related works*

Content-based systems have been used successfully in various domains including Web browsing, news filtering, and recommendation services.

Letizia is a content-based Web agent that suggests Web pages of interest to the user (Lieberman 1995). The system, a Web browser extension that tracks the user's browsing behavior, relies on implicit feedback and uses a set of heuristics to infer the user's preferences. For example, Letizia interprets bookmarking a page as strong evidence for the user's interests in the page. *NewsDude* (Billsus and Pazzani 1999) reads interesting news articles via a speech interface. The news source is Yahoo! News, with an initial training set of interesting news articles provided by the user. Length of listening time provides implicit user feedback on articles read out. A short-term user model is based on TFIDF (cosine similarity), and long-term model based on a naïve Bayes classifier. Both *Letizia* and *NewsDude* use implicit feedback, which does not require any *active* user involvement, in the sense that ratings are guessed by monitoring the user's activities. In our approach, we decided to adopt explicit feedback (the system requires the user to explicitly evaluate documents). This technique has the advantage of simplicity, both for system design and system implementation. Furthermore, in performing experiments, explicit feedback has the added advantage of minimizing one potential source of experimental error, inference of the user's true reaction. Our work on learning user profiles as Bayesian classifiers is mainly inspired by:

- *Syskill & Webert* (Pazzani and Billsus 1997) is a software agent that learns a user's interests saved as a user profile, and uses this profile to identify interesting Web pages. The learning process is conducted by first converting HTML source into positive and negative examples, and then using algorithms like Bayesian classifiers, a nearest neighbor algorithm and a decision tree learner.
- Mooney and Roy (2000) adopt a text categorization method in their *LIBRA* system, which makes content-based book recommendations exploiting the product descriptions obtained from the Web pages of the Amazon† online digital store, using a naïve Bayes text classifier. Book descriptions are structured into slots in order to maintain separately information about title, authors, abstract. The advantages of exploiting information about the structure of documents are pointed out also in the work by Ma *et al.* (2003). In this work, the authors propose a method which uses structuring conventions as a feature reduction method. Information extraction techniques are used to extract a synopsis of document structure, which contains only the most informative features. The main result is that the accuracy of the classifiers produced from the feature space is comparable to those reported in previous document classification efforts using much larger feature spaces. Based on these results, we decided to modify the categorization method by taking into account information about the structure of the documents.

As regards the work mainly related to the Rocchio algorithm for user profiling, *ifWeb* (Asnicar and Tasso 1997) is a system that supports users in document searching. User profiles are stored in the form of a weighted semantic network, that represents terms and their context by linking nodes (words) with arcs representing co-occurrences in some documents. *ifWeb* support explicit feedback and takes into account not only interest, but also explicit *dis*interest, and therefore presumably represents a user's idiosyncrasies more accurately. In this aspect, this approach is similar to our method based on the Rocchio relevance feedback that learns both a positive and a negative profile. SiteIF (Stefani and Strapparava 1998) is a personal agent for a multilingual news Web site that learns the user's interests from the requested pages that are analyzed to generate or to update a model of the user. Exploiting this model, the system tries to anticipate which documents in the Web site could be interesting for the user. SiteIF strongly resembles ifWeb (profiles and documents are stored as semantic networks also in this system), except that explicit user interaction is avoided.

3. A relevance feedback method for learning user profiles

The Rocchio algorithm is one of the most popular learning methods from Information Retrieval and document classification. In this algorithm, documents are represented with the vector space representation and the major heuristic component is the TFIDF (term frequency/inverse document frequency) word weighting scheme (Salton and McGill 1983), that reflects empirical observations regarding text:

$$\text{TFIDF}(t_k, d_j) = \underbrace{\text{TF}(\mathrm{t_k}, \mathrm{d_j})}_{\text{TF}} \cdot \underbrace{\log \frac{N}{n_i}}_{\text{IDF}} \tag{1}$$

where N is the total number of documents in the training set and n_i is the number of documents in which the term t_k appears. $\text{TF}(t_k, d_j)$ is a function that computes the frequency of the token t_k in the document d_j. Learning combines vectors of positive and negative examples into a prototype vector $\vec{c}$ for each class in the set of classes C. The method computes a classifier $\vec{c_i} = \langle \omega_{1i}, \ldots, \omega_{|T|i} \rangle$ for category c_i (T is the *vocabulary*, that is the set of distinct terms in the training set) by means of the formula:

$$\omega_{ki} = \beta \cdot \sum_{\{d_j \in POS_i\}} \frac{\omega_{kj}}{|POS_i|} - \gamma \cdot \sum_{\{d_j \in NEG_i\}} \frac{\omega_{kj}}{|NEG_i|} \tag{2}$$

where ω_{kj} is the TFIDF weight of the term t_k in document d_j, POS_i and NEG_i are the set of positive and negative examples in the training set for the specific class c_i, while β and γ are control parameters that allow setting the relative importance of *all* positive and negative examples. To assign a class $\tilde{c}$ to a document d_j, the similarity between each prototype vector $\vec{c_i}$ and the document vector $\vec{d_j}$ is computed and $\tilde{c}$ will be the c_i with the highest value of similarity. We propose a modified version of this method, that manages documents

†http://www.Amazon.com

represented using different slots. As described in section 2.1, an item is represented by its slots. If m is the index of the slot ($m = 1, 2, 3$ for the book descriptions and $m = 1, 2, \ 3, 4, 5$ for the movie descriptions), an item is represented by a list of m bag of words:

$$d_j = \langle w_{1j}^m, \ldots, w_{|T_m|j}^m \rangle$$

where $|T_m|$ is the cardinality of the vocabulary for the slot s_m and w_{kj}^m is the weight of the term t_k in the document d_j, in the slot s_m. Each weight w_{kj}^m is computed as follows:

$$\text{TFIDF}(t_k, d_j, s_m) = \text{TF}(t_k, d_j, s_m) \cdot \log \frac{N}{n_{km}} \tag{3}$$

$\text{TF}(t_k, d_j, s_m)$ is the frequency of term t_k in the document d_j in the slot s_m; the inverse document frequency of the term t_k in the slot s_m is computed as the logarithm of the ratio between the total number of documents N and the number of documents containing the term t_k in the slot s_m. Given a user u and a set of rated items in a specific genre (e.g., *Comedy* for movies and *Business* for books), the aim is to learn a profile able to recognize items liked by the user in that genre. Learning consists in inducing one prototype vector for *each slot*: these (three or five) vectors will represent the user profile. The rationale of having distinct components of the profile is that words appearing in a 'heavy' slot such as the title could be more indicative of preferences than words appearing in other slots such as the annotation, having a low informative power. For these reasons, each prototype vector of the profile could contribute in a different way to the calculation of the similarity between the vectors representing an item and the vectors representing the user profile. Another key issue of our modified version of the Rocchio algorithm is that it separately exploits the training examples: it learns two different profiles $\overrightarrow{p_i} = \langle \omega_{1i}^m, \ldots, \omega_{|T_m|i}^m \rangle$, for a user u and a category c_i by taking into account the ratings given by the user on documents in that category. The rating $r_{u,j}$ on the document d_j is a discrete judgment ranging from 1 to MAX. It is used to compute the coordinates of the vectors in both the positive and the negative user profile:

$$\omega_{ki}^m = \sum_{\{d_j \in POS_i\}} \frac{\omega_{kj}^m \cdot r'_{u,j}}{|POS_i|} \tag{4}$$

$$\omega_{ki}^m = \sum_{\{d_j \in NEG_i\}} \frac{\omega_{kj}^m \cdot r'_{u,j}}{|NEG_i|} \tag{5}$$

where $r'_{u,j}$ is the normalized value of $r_{u,j}$ ranging between 0 and 1 (respectively corresponding to $r_{u,j} = 1$ and MAX), $POS_i = \{d_j \in T_r | r_{u,j} > MAX/2\}$, $NEG_i = \{d_j \in T_r | r_{u,j} \leq MAX/2\}$, and ω_{kj}^m is the weight of the term t_k in the document t_j in the slot s_m computed as in equation (3), where the *idf* factor is computed over POS_i or NEG_i depending on the fact that the term t_k is in the slot s_m of an item rated as positive or negative (if the term is present in both positive and negative items two different values for it will be computed). Computing two different idf values for a term led us to consider the rarity of a term in positive and negative items, in an attempt to discover the informative power of a term in recognizing interesting items. Equations (4) and (5) differ from the classic formula in the fact that the parameters β and γ are substituted by the ratings $r'_{u,j}$ that allow us to give a different weight to each document describing the item in the training set.

The similarity between a profile $\overrightarrow{p_i}$ and an item $\overrightarrow{d_j}$ is obtained by computing m partial similarity values between each pair of corresponding vectors in $\overrightarrow{p_i}$ and $\overrightarrow{d_j}$:

$$\text{sim}(\overrightarrow{d_j}, \overrightarrow{p_j}) = \sum_{s=1}^{m} \text{sim}(\overrightarrow{d_j^s}, \overrightarrow{p_j^s}) \cdot \alpha_s \tag{6}$$

A weighted average of the m (m is the number of slots) values is computed, assigning a different weight α_s to reflect the importance of a slot in classifying an item.

In our experiments on movies, we used $\alpha_1 = 0.1$ (title), $\alpha_2 = 0.15$ (director), $\alpha_3 = 0.15$ (cast), $\alpha_4 = 0.25$ (summary), and $\alpha_5 = 0.35$ (keywords). In the experiments on books, we used $\alpha_1 = 0.5$ (title), $\alpha_2 = 0.4$ (authors), and $\alpha_3 = 0.1$ (annotation).

The values α_s were decided according to experiments not reported in the paper for the sake of brevity. We considered different values for each α_s and repeated the experiments reported in section 4 using the selected values. The values reported here are those that allowed us to obtain the best predictive accuracy. Weakest results are obtained when setting higher the α_s values of slots containing proper names. The main outcome of the experiments was that slots containing 'free text' (and not proper names alone, like slots *authors*, and *cast*, and *director*) should be 'heavier' in computing similarity. Since the user profile is composed by both the positive and the negative profiles, we compute two similarity values, one for each profile. The document d_j is considered as interesting only if the similarity value of the positive profile is higher than the similarity of the negative one. The method has been implemented by developing a system called RocchioProfiler, which was used to evaluate the proposed algorithm.

3.1 *INTHELEX*

INTHELEX (INcremental THEory Learner from EXamples) is a learning system for the induction of hierarchical theories from positive and negative examples. It is fully and inherently incremental: in addition to the possibility of taking as input a previously generated version of the theory, learning can also start from an empty theory and from the first available example. INTHELEX can learn simultaneously various concepts, possibly related to each other, expressed as (sets of) function free clauses to be interpreted according to the Object Identity assumption (Semeraro 1998). Examples describe the observations by means of only basic non-negated predicates of the representation language, and specifies all the classes for which the observed object is a positive example and all those for which it is a negative one. A positive example for a concept is not considered as a negative example for all the other concepts, unless it is explicitly stated. INTHELEX incorporates two inductive operators, one for generalizing definitions that reject positive examples and the other for specializing definitions that explain negative examples. Both these operators, when applied, change the set of examples the theory accounts for. In particular, when a positive example is not covered, completeness is restored in one of the following ways:

- Replacing a clause in the theory with one of its generalizations against the problematic example;
- Adding a new clause to the theory, obtained by properly turning constants into variables in the problematic example;
- Adding the problematic example as a positive exception.

When a negative example is covered, consistency is restored by performing one of the following actions:

- Adding positive literals that are able to characterize all the past positive examples of the concept (and exclude the problematic one) to one of the clauses that concur to the example coverage;
- Adding a negative literal that is able to discriminate the problematic example from all the past positive ones to the clause in the theory by which the problematic example is covered;
- Adding the problematic example as a negative exception.

We were led by a twofold motivation to exploit INTHELEX on the problem of learning user profiles. First, its representation language (First-Order Logic) is more intuitive and human readable than values exploited and provided by numeric/probabilistic approaches. Second, incrementality is fundamental in the given task, since new information on a user is available each time he issues a query, and it would be desirable to be able to refine the previously generated profile instead of learning a new one from scratch. Moreover, a user's interests might change in time, a problem that only incremental systems are able to tackle.

Each item (book/movie) description is represented in terms of its components that are three in the case of the books, i.e., title, author, and annotation, and five in case of the movies, i.e., title, cast, director, keywords, and summary. Such components were codified in the description of the books by using predicates `slot_title(b,t)`, `slot_author(b,au)`, and `slot_annotation(b,an)`, indicating that the objects `t`, `au`, and `an` are, respectively, the title, author, and annotation of the book `b`, and in the description of the movies by means of the predicates `slot_title(m,t)`, `slot_cast(m,c)`, `slot_director(m,d)`, `slot_keywords(m,k)` and `slot_summary(m,tr)`, with the same meaning above reported. Any word in the item description, both books and movies, is represented by a predicate corresponding to its stem, and linked to both the item itself, book or movie, and the single slots in which it appears. For instance, in case of book description, predicate `prolog(slott, slottitleprolog)` indicates that the object `slottitleprolog` has stem 'prolog' and is contained in slot `slott`; in such a case, also a literal `prolog(book)` is present to say that stem "prolog" is present in the book description. Formerly, INTHELEX was not able to handle numeric values; thus, a discretization was needed. In the new version, it can represent numeric information and manipulate numeric intervals, so the number of occurrences of each word in each slot was represented by means of a predicate `occ(Y,X)`, indicating that term Y occurs X times. Instead of learning a definition for each of the possible votes, just two possible classes of interest are learnt: 'likes', describing that the user likes an item (ratings from 6 to 10 for books preferences and from 4 to 6 for movies preferences), and its opposite 'not(likes)' (ratings from 1 to 5 for books and from 1 to 3 for movies). Such a discretization step is automatically carried out by an abstraction operator embedded in INTHELEX, whose cost is negligible since each numeric value is immediately mapped onto the corresponding discretized symbolic value. Figure 1 shows an example for the class `likes`. The clause means that the user `likes` the book with *id=50147799* that contains in the slot *title* a word whose stem is *practic* (one or two times) and a word whose stem is *prolog* (one or two times), in the slot *authors* the word *l_sterling* (one or two times). Figure 2 shows a rule learned by INTHELEX for the class `likes` in the movie recommending scenario.

```
likes(50147799) :-
 slot_title(50147799,slott), practic(slott,slottitlepractic),
 occ_1(slottitlepractic), occ_12(slottitlepractic),
 prolog(slott, slottitleprolog), occ_1(slottitleprolog),
 occ_12(slottitleprolog), slot_authors(50147799,slotau),
 l_sterling(slotau,slotauthorsl_sterling), occ_1(slotauthorsl_sterling),
 occ_12(slotauthorsl_sterling), slot_annotation(50147799, slotan),
 l_sterling(50147799), practic(50147799), prolog(50147799).
```

Figure 1. First-order representation of a book.

```
likes(A) :-
     disnei(A), famili(A), featur(A), kid(A), anim(A),
     slot_keywords(A, D),
     anim(D, E), occ_12(E), occ_1m(E),
     disnei(D, F), occ_12(F), occ_1m(F),
     famili(D, G), occ_1(G), occ_12(G), occ_1m(G),
     featur(D, H), occ_1(H), occ_12(H), occ_1m(H),
     not(alexander(A)).
```

Figure 2. Rule learned by INTHELEX.

3.2 *Item recommender*

ITR (ITem Recommender) implements a Bayesian learning algorithm (Mitchell 1997) able to classify text belonging to a specific category as interesting or not interesting for a particular user. For example, the system could learn the target concept '*textual descriptions the user finds interesting in the category Computer and Internet*'.

According to the Bayesian approach to classify text documents, given a set of classes $C = \{c_1, \ldots, c_{|C|}\}$, the conditional probability of a class c_j given a document d is calculated as follows:

$$P(c_j|d) = \frac{P(c_j)}{P(d)} P(d|c_j)$$

In our problem, we have only two classes: c_+ represents the positive class (user-likes, corresponding to ratings from 6 to 10 for books and from 4 to 6 for movies), and c_- the negative one (user-dislikes, ratings from 1 to 5 for books and from 1 to 3 for movies). Since instances are represented as a vector of documents, (one for each BOW), and assumed that the probability of each word is independent of the word's context and position, the conditional probability of a category c_j given an instance d_i is computed using the formula:

$$P(c_j|d_i) = \frac{P(c_j)}{P(d_i)} \prod_{m1}^{|S|} \prod_{k=1}^{|b_{im}|} P(t_k|c_j, s_m)^{n_{kim}} \quad (7)$$

where $S = \{s_1, s_2, \ldots, s_{|S|}\}$ is the set of slots, b_{im} is the BOW in the slot s_m of the instance d_i, n_{kim} is the number of occurrences of the token t_k in b_{im}.

In (7), since for any given document the prior $P(d_i)$ is a constant, this factor can be ignored if the only interest concerns a ranking rather than a probability estimate. To calculate (7), we only need to estimate the terms $P(c_j)$ and $P(t_k|c_j, s_m)$, from the training set. Each instance is weighted according to the user rating r, normalized in order to obtain values ranging between 0 and 1:

$$w_+^i = \frac{r-1}{MAX}; \qquad w_-^i = 1 - w_+^i \quad (8)$$

The weights in (8) are used to estimate the two probability terms from the training set TR:

$$\hat{P}(c_j) = \frac{\sum_{i=1}^{|TR|} w_j^i}{|TR|} \quad (9)$$

$$\hat{P}(t_k|c_j, s_m) = \frac{\sum_{i=1}^{|TR|} w_j^i n_{kim}}{\sum_{i=1}^{|TR|} w_j^i |b_{im}|} \quad (10)$$

In (10), n_{kim} is the number of occurrences of the term t_k in the slot s_m of the ith instance, and the denominator denotes the total weighted length of the slot s_m in the class c_j. The length of b_{im} is computed by adding the occurrences of the words in the slot s_m of the ith instance. Therefore, $\hat{P}(t_k|c_j, s_m)$ is calculated as a ratio between the weighted occurrences of the term t_k in slot s_m of class c_j and the total weighted length of the slot. The final outcome of the learning process is a probabilistic model used to classify a new instance in the class c_+ or c_-. The model can be used to build a personal profile including those words that turn out to be most indicative of the user's preferences, according to the value of the conditional probabilities in (10).

4. Empirical evaluation

The goal of the experiments reported in this section has been the comparison of ITR, RocchioProfiler and INTHELEX in terms of several metrics. Two different data sets have been used in order to compare the systems in different domains: book recommending and movie recommending. The data sets are described

Table 1. Preprocessing operations performed on the EachMovie data set.

Slot	Tokenization	Stopwords	Stemming
Title	X		
Cast	X		
Director	X		
Summary	X	X	X
Keywords	X	X	X

in section 4.1 and section 4.2. Three main experiments have been conducted:

- Experiment 1: The classic Rocchio relevance feedback algorithm is compared to the novel approach implemented by RocchioProfiler in the task of book recommendation.
- Experiment 2: ITR, RocchioProfiler and INTHELEX are compared in the task of book recommendation.
- Experiment 3: ITR, RocchioProfiler and INTHELEX are compared in the task of movie recommendation.

4.1 *The EachMovie data set*

The EachMovie project was conducted by the Compaq Systems Research Center† over an 18-month period from 1996 to 1997. During this time, a large data set of user movie ratings was collected, consisting of 2,811,983 ratings for 1628 movies from 72,916 users. The movies are rated on a 6-point scale (from 1 to 6). The 1 to 6 star rating used externally on EachMovie is mapped linearly to the interval [0,1].

The original data set does not contain any information about the content of the movies. The content information for each movie was collected from the Internet Movie Database‡ using a simple crawler that, following the IMDb link provided in the original data set, collects information from the various links of the main URL. In particular the crawler gathers the *Title*, the *Director*, the *Genre*, that is the category of the movie, the list of *Keywords*, the *Summary* and the *Cast*. The retrieved content data is provided in a CSV (comma separated value) text file. After appropriate preprocessing (the operations performed on each document are listed in table 1), the content is organized and stored in a relational database.

The content of slots *title*, *director* and *cast* was only tokenized because we observed that the process of stopwords elimination produced some unexpected results: for example, after stopwords elimination, each slot *title* made exclusively of stopwords (like '*It*') became empty. Moreover, it makes no sense to process by stemming and stopword elimination on slots containing only proper names.

Table 2. 10 "Genre" data sets obtained from the original EachMovie data set.

Id genre	genre	Number of rated movies	Average length (in words) slot 'summary'	% POS	% NEG
1	Action	192	188.04	79	21
2	Animation	163	93.86	75	25
3	Art_Foreign	240	80.82	92	8
4	Classic	240	174.57	97	3
5	Comedy	239	107.07	63	37
6	Drama	249	133.02	84	16
7	Family	211	120.03	74	26
8	Horror	196	169.95	75	25
9	Romance	211	92.73	86	14
10	Thriller	187	141.43	85	15
		2128	130.15	81	19

Movies are subdivided into different genres: *Action*, *Animation*, *Art_Foreign*, *Classic*, *Comedy*, *Drama*, *Family*, *Horror*, *Romance*, and *Thriller*. For each genre or category, a set of five users was randomly selected among users that rated n items, $30 \leq n \leq 100$ in that movie category. In this way, for each category, a data set of at least 150 triples (user, movie, and rating) was obtained. Table 2 summarizes the data used for the experiments. Notice that the number of movies rated as positive and negative for each genre is balanced only in data set 5, (60–65% positive, 35–40% negative), while is slightly unbalanced in data sets 2, 7, 8 (65–75% positive, 25–35% negative), and is strongly unbalanced in data sets 1, 3, 4, 6, 9, 10 (over 75% positive). Rate frequencies of this data set are shown in table 3.

4.2 *The BOL data set*

The Bertelsmann Online (BOL) data set is a collection of textual book descriptions rated by real users according to their preferences. Eight book categories were selected at the Web site of a virtual bookshop. For each book category, a set of book descriptions was obtained by analyzing Web pages using an automated extractor and stored in a local database. Common preprocessing

†http://www.research.compaq.com/SRC/
‡IMDB, http://www.imdb.com

Table 3. Rate frequencies in EachMovie data set.

Category	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
1 Action	14	4	21	53	60	40
2 Animation	18	3	19	41	43	39
3 Art Foreign	2	2	14	59	95	68
4 Classic	2	0	5	49	103	81
5 Comedy	14	19	55	67	65	19
6 Drama	12	6	21	93	88	29
7 Family	21	11	22	62	51	44
8 Horror	10	12	26	60	46	42
9 Romance	1	2	27	72	92	17
10 Thriller	7	4	17	59	78	22
Mean	10.1	6.3	22.7	61.5	72.1	40.1

Table 4. Preprocessing operations performed on the BOL data set.

Slot	Tokenization	Stopwords	Stemming
Title	X		
Authors	X		
Annotation	X	X	X

operation have been performed on the book descriptions (table 4).

Each user involved in the experiments was requested to choose one or more categories of interest and to rate 40 or 80 books in each selected category, providing 1–10 discrete ratings. In this way, for each pair user category, a data set of 40 or 80 rated books was obtained (table 5). For each user we considered:

- *Rated books*: number of rated books with the indication of negative (rates in the range 1–5) and positive (rates in the range 6–10) ones;
- *Books with annotation*: number (and percentage) of books with a textual annotation (slot annotation not empty);
- *Average annotation length*: average length (in words) of the annotations;
- *Average rate* μ/σ: average rate provided by the user and standard deviation.

We can observe that the number of books rated as positive and negative for each user is balanced, except for the user 23 that has rated 39 books as positive and only 1 as negative. Then, almost the totality of books rated by the users contain annotations: the user 33 is the only one with a low percentage. As far as the average annotation length, we can notice that only users 26 and 33 have values lower than the others.

4.3 *Performance measures*

Classification effectiveness is measured in terms of the classic Information Retrieval notions of *precision*, *recall* and *accuracy*, adapted to the case of text categorization (Sebastiani, 2002).

Let *TP* (true positive) be the number of relevant test documents correctly classified, that is documents that both the system and the user deemed relevant. Then, recall and precision are computed as follows:

$$Re = \frac{TP}{\textit{number of documents the user deemed relevant}}$$

$$Pr = \frac{TP}{\textit{number of documents the system deemed relevant}}$$

Also used is F-measure, which is a combination of precision and recall:

$$F = \frac{2 \times Re \times Pr}{Pr + Re}$$

In other words, *precision* is the proportion of items classified as relevant that are really relevant, and *recall* is the proportion of relevant items that are classified as relevant. We adopted also *accuracy*, which represents the percentage of items correctly classified (as relevant or not relevant) computed as reported in (Sebastiani, 2002).

4.4 *Experimental setup*

In the design of all the experiments and the evaluation step, the concept of 'interesting item', that we call *relevant* item is central. In the movie recommendation task, users adopted a 6-point discrete scale for rating items: a movie in a specific category is considered as relevant by a user if the rating is greater or equal than 3. For books, we adopted a 10-point scale: if the

Table 5. BOL data set information.

User ID	Category	Rated books	Books with Annot.	Average Ann. Length	Average Rating μ/σ
37	SF, Horror and Fantasy	40 (22+, 18−)	40 (100%)	30.475	4.87/2.731
26	SF, Horror and Fantasy	80 (46+, 34−)	70 (87.5%)	19.512	5.49/3.453
30	Computer and Internet	80 (40+, 40−)	80 (100%)	56.425	5.31/2.462
35	Business	80 (30+, 50−)	78 (97.5%)	64.150	4.21/3.488
24c	Computer and Internet	80 (38+, 42−)	76 (95%)	49.100	5.71/3.174
36	Fiction and literature	40 (25+, 15−)	40 (100%)	40.225	5.87/1.805
24f	Fiction and literature	40 (27+, 13−)	38 (95%)	45.500	6.40/2.662
33	Sport and leisure	80 (35+, 45−)	49 (61.25%)	23.337	4.34/3.342
34	Fiction and literature	80 (42+, 38−)	70 (87.5%)	44.925	5.61/2.492
23	Fiction and literature	40 (39+, 1−)	36 (90%)	45.875	7.25/1.089
		640 (344+, 296−)	537 (83%)	41.965	

Table 6. Performance of the Rocchio algorithms on 10 different data sets. Notice that 9 users rated books in one category, while user 24 rated books in two categories.

	Precision		Recall		F1		Accuracy	
Id	$\beta=16$ $\gamma=4$	New Rocchio	$\beta=16$ $\gamma=4$	New Rocchio	$\beta=16$ $\gamma=4$	New Rocchio	$\beta=16$ $\gamma=4$	New
37	0.641	0.925	1	0.717	0.781	0.808	0.675	0.800
26	0.797	0.845	1	0.830	0.887	0.837	0.837	0.812
30	0.500	0.534	1	0.875	0.666	0.663	0.500	0.550
35	0.391	0.690	1	0.700	0.562	0.695	0.412	0.762
24c	0.471	0.675	0.966	0.583	0.633	0.626	0.475	0.687
36	0.616	0.767	0.916	0.700	0.737	0.732	0.600	0.675
24f	0.675	0.825	1	0.833	0.805	0.829	0.675	0.750
33	0.651	0.743	0.966	0.917	0.778	0.821	0.737	0.812
34	0.525	0.644	0.975	0.645	0.682	0.644	0.525	0.625
23	0.966	0.975	0.966	0.975	0.966	0.975	0.950	0.950
Mean	0.623	0.762	0.979	0.777	0.750	0.763	0.639	0.742

rating of the book is greater than 5, then the book is relevant, otherwise it is not relevant. The Rocchio-based profiling algorithm classifies an item as relevant if the similarity score of the class *likes* is greater than the one for the class *dislikes*, while ITR considers an item d_i as relevant if $P(c_+|d_i) \geq 0.5$, calculated as in equation (7). Symmetrically, INTHELEX considers as relevant those items covered by the inferred theory.

Experiment 1 This experiment has been conducted on the BOL data set. We executed one experiment for each user (table 5) in the data set: the ratings of each specific user and the content of the books rated have been used for learning the user profile and measuring its predictive accuracy, using the aforementioned measures.

Each experiment consisted of:

1. Selecting ratings of the user and the content of the books rated by that user;
2. Splitting the selected data into a training set *Tr* and a test set *Ts*;
3. Using *Tr* for learning the corresponding user profile;
4. Evaluating the predictive accuracy of the learned profile on the *Ts*, using the aforementioned measures.

The methodology adopted for obtaining *Tr* and *Ts* was the *K*-fold cross-validation, that works by partitioning the data into *K* equal-sized segments and holding out one segment at a time for test purposes. We fixed K=10, and thus we ran 10 experiments on each user by averaging the evaluation measures computed in the test phase.

Table 6 and figure 3 show the results of the experiments using the modified version of the Rocchio algorithm and the classic Rocchio one, obtained by setting the values of the control parameters β and γ according

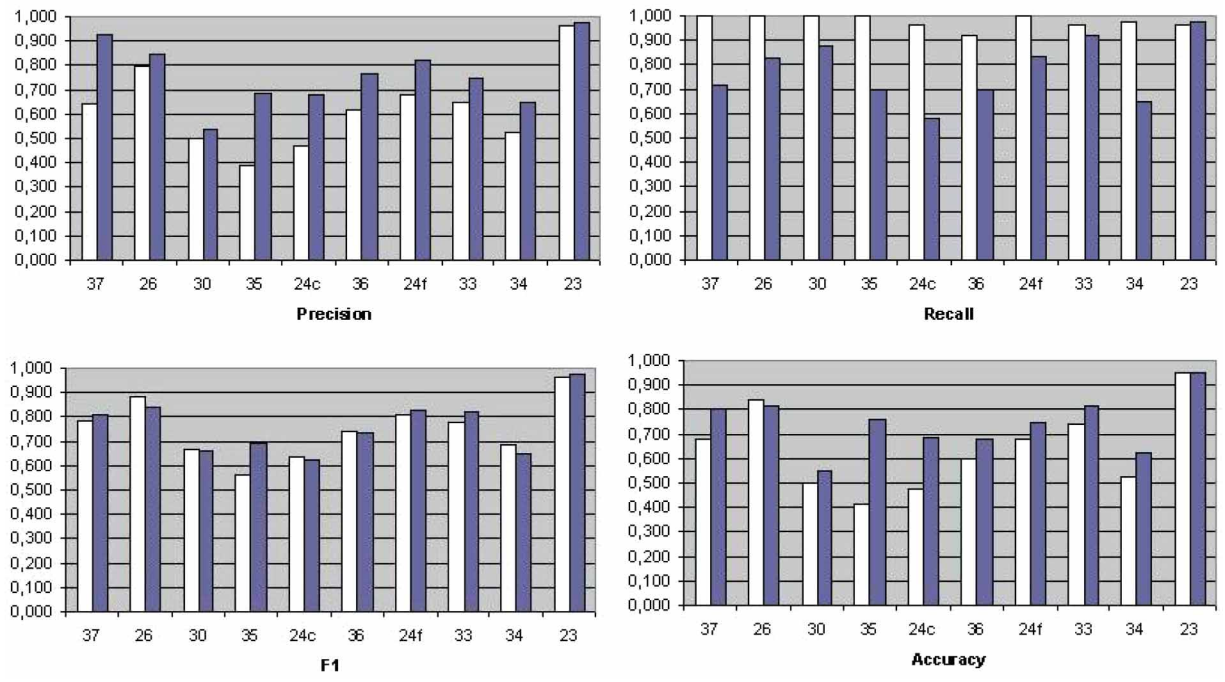

Figure 3. Performance of the Rocchio algorithms on 10 different data sets. The white bar represents classical Rocchio and the black bar New Rocchio.

to the literature (Sebastiani 2002). For pairwise comparison of the two methods, the nonparametric Wilcoxon signed rank test was used, since the number of independent trials (i.e., users) is relatively low and does not justify the application of a parametric test, such as the *t*-test (Orkin and Drogin 1990). Requiring a significance level $p < 0.05$, the test revealed that there is a statistically significant difference in performance both for precision and accuracy in favor of the modified Rocchio and for recall in favor of the classic Rocchio method ($\beta = 16$, $\gamma = 4$), but not as regards F1. These results led us to conclude that the new method is more effective than the traditional one in this domain, due to the fact that *trust* is a key word in giving recommendations: the systems should minimize false-positive errors. This means that it is better to provide users with a small number of high quality recommendations than to overload users with many recommendations that they should manually filter. In the next experiments, we compare the performance of the new Rocchio method with those of ITR and INTHELEX.

Experiment 2 This experiment has been conducted on the BOL data set.

Table 7 and figure 4 show the results of the experiment aimed at comparing the new Rocchio method with the ones implemented by INTHELEX and ITR in terms of average precision, recall, F1, and accuracy of the models learned in the 10 folds for each data set. The last row of the table reports the mean values, averaged on all data sets. The results of INTHELEX and ITR are described in more detail in the work by Esposito *et al.* (2003). The most important result is that the proposed method outperforms the other ones as regards accuracy. It is surprising to observe that the algorithm reaches high values of precision and recall for the users 26 and 37, even if the average annotation lengths of the documents rated by the users are among the shortest in the data set. This means that although the profiles contain few words for computing similarity on new documents, these words are indicative of the users' preferences. In general, the new Rocchio algorithm outperforms ITR in precision, but not INTHELEX (requiring a significance level $p < 0.05$ the systems are equivalent). Another remark worth noting is that theories learned by the symbolic system are very interesting from a human understandability viewpoint, in order to be able to explain and justify the recommendations provided by the system.

Experiment 3 The goal of this experiment is to compare RocchioProfiler, ITR and INTHELEX in the task of movie recommendation. Thus, this experiment has been conducted on the EachMovie data set.

For each 'genre' data set, we ran 5 experiments, one for each user in the data set: the triples (user,movie, rating) of each specific user and the content of the

Table 7. Performance of the systems on BOL data set.

	Precision			Recall			F1			Accuracy		
Id	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio
37	0.767	0.967	0.925	0.883	0.500	0.717	0.821	0.659	0.808	0.731	0.695	0.800
26	0.818	0.955	0.845	0.735	0.645	0.830	0.774	0.770	0.837	0.737	0.768	0.812
30	0.608	0.583	0.534	0.600	0.125	0.875	0.604	0.206	0.663	0.587	0.488	0.550
35	0.651	0.767	0.690	0.800	0.234	0.700	0.718	0.359	0.695	0.725	0.662	0.762
24c	0.586	0.597	0.675	0.867	0.383	0.583	0.699	0.467	0.626	0.699	0.599	0.687
36	0.783	0.900	0.767	0.783	0.300	0.700	0.783	0.450	0.732	0.700	0.513	0.675
24f	0.785	0.900	0.825	0.650	0.350	0.833	0.711	0.504	0.829	0.651	0.535	0.750
33	0.683	0.750	0.743	0.808	0.308	0.917	0.740	0.437	0.821	0.730	0.659	0.812
34	0.608	0.883	0.644	0.490	0.255	0.645	0.543	0.396	0.644	0.559	0.564	0.625
23	0.500	0.975	0.975	0.130	0.900	0.975	0.206	0.936	0.975	0.153	0.875	0.950
Mean	0.679	0.828	0.762	0.675	0.400	0.777	0.662	0.520	0.763	0.627	0.636	0.742

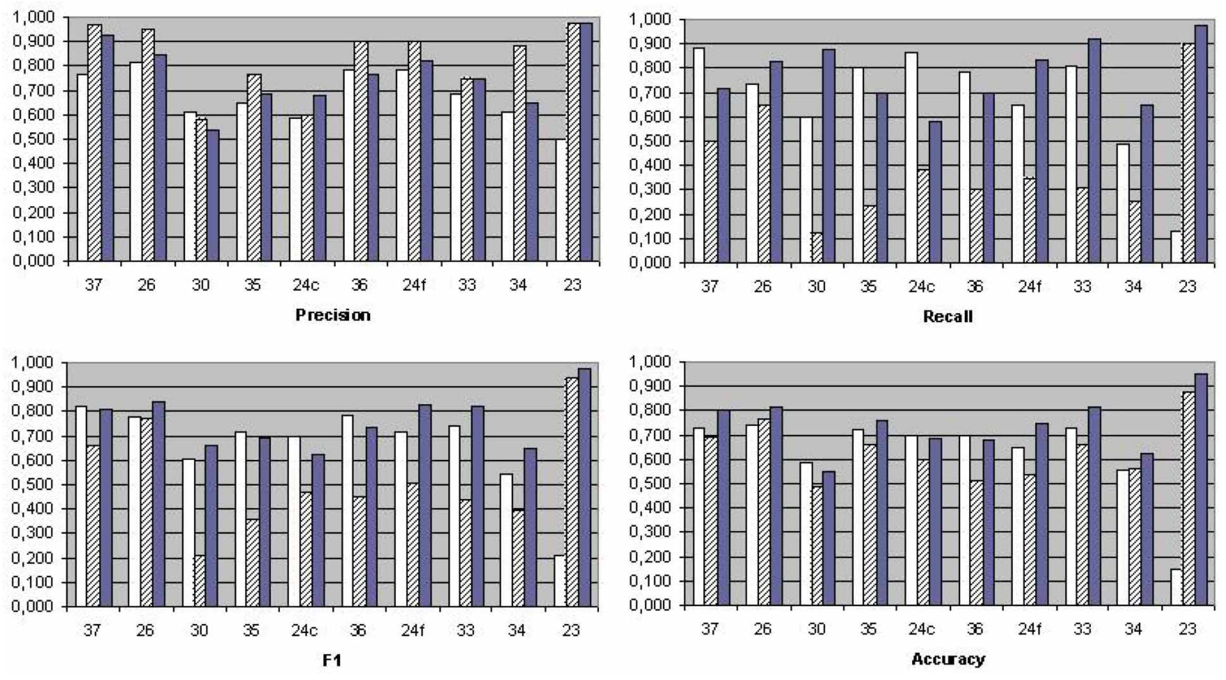

Figure 4. Performance of the systems on BOL data set. The white bar represents ITR, the strip one INTHELEX and the black bar New Rocchio.

rated movies have been used for learning the user profile and measuring its predictive accuracy, using the measures presented in section 4.3.

Each experiment consisted in:

1. Selecting the triples (user, movie, rating) of the user and the content of the movies rated by that user;
2. Splitting the selected data into a training set *Tr* and a test set *Ts*;
3. using *Tr* for learning the corresponding user profile;
4. evaluating the predictive accuracy of the learned profile on the *Ts*, using the aforementioned measures.

The methodology adopted for obtaining *Tr* and *Ts* was the 10-fold cross-validation. The results of each "genre" data set are reported in table 8 and figure 5.

The last row of the table reports the mean values, averaged on all data sets. We have carried out a pairwise comparison of the results, using the nonparametric Wilcoxon signed-rank test (Orkin and Drogin 1990). The most important result is that the new Rocchio

Table 8. Performance of the systems on the EachMovie data set.

	Precision			Recall			F1			Accuracy		
Id	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio	ITR	INTH.	New Rocchio
1	0.804	0.777	0.825	0.988	0.673	0.905	0.885	0.691	0.849	0.795	0.612	0.769
2	0.720	0.767	0.832	0.823	0.757	0.930	0.751	0.740	0.879	0.750	0.700	0.837
3	0.941	0.942	0.934	1.000	0.957	0.965	0.969	0.944	0.944	0.941	0.904	0.905
4	0.978	0.978	0.976	1.000	0.950	0.985	0.989	0.959	0.978	0.980	0.929	0.961
5	0.607	0.722	0.752	0.801	0.594	0.782	0.691	0.597	0.736	0.691	0.629	0.687
6	0.860	0.827	0.863	0.977	0.779	0.908	0.923	0.783	0.869	0.858	0.699	0.787
7	0.728	0.763	0.802	0.825	0.703	0.845	0.770	0.703	0.810	0.820	0.708	0.777
8	0.782	0.797	0.796	0.965	0.712	0.893	0.863	0.719	0.831	0.769	0.678	0.746
9	0.875	0.891	0.862	1.000	0.862	0.903	0.933	0.849	0.868	0.875	0.769	0.789
10	0.857	0.889	0.862	0.983	0.837	0.961	0.912	0.841	0.898	0.848	0.786	0.836
Mean	0.815	0.835	0.850	0.936	0.782	0.908	0.868	0.782	0.866	0.833	0.741	0.809

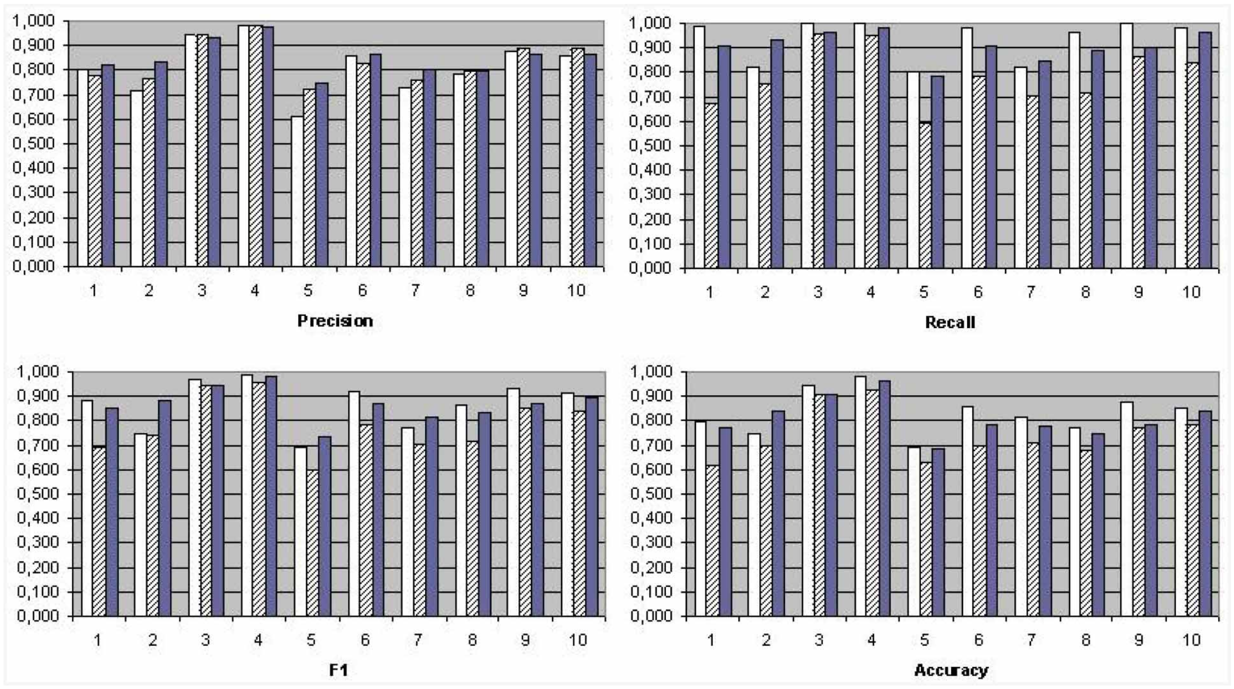

Figure 5. Performance of the systems on EachMovie data set. The white bar represents ITR, the strip one INTHELEX and the black bar New Rocchio.

algorithm outperforms both ITR and INTHELEX in precision (significance level $p < 0.05$), while experiment 2 showed that INTHELEX had a better precision.

In particular, the new Rocchio algorithm outperforms ITR on data set 5 (+14.5%), where the number of movies rated as positive and negative is balanced (table 2). This could be due to the different representation of the profiles adopted by the systems: ITR exploits positive and negative examples to learn a unique profile, while the Rocchio algorithm uses in a separate way positive and negative examples for learning two different profiles, one for the class “interesting movies” (positive class) and another one for the class “not interesting movies” (negative class). The conclusion is that the Rocchio method takes into account in a better way the negative examples. This conclusion is supported by the fact that for the other data sets, in which the percentage of negative examples is lower

(< 30%), the difference in precision, even if it persists again in favor of Rocchio, never reaches the value of 14.5%.

On the other hand, as regards Recall, the probabilistic and the statistical approaches outperform the symbolic approach, like in experiment 2: the difference between RocchioProfiler and ITR is statistically significant in favor of ITR, while in experiment 2 RocchioProfiler had a better performance. This could be due to the increased size of the data set, that allows ITR to produce more accurate estimates of the words found in the training set. As regards F1, the difference between ITR and RocchioProfiler, both outperforming INTHELEX, is not statistically significant. Finally, the overall predictive accuracy of profiles learned by ITR is higher than the ones of the other systems. These results led us to conclude that the probabilistic and the statistical approaches are more effective than the symbolic approach in the task of movie recommendation. This could be a general indication that numerical techniques perform better that symbolic techniques when the size of the documents become larger: the average length of the documents in the EachMovie data set is higher with respect to that of documents in the BOL data set. We measured the average length of the slot 'summary' for movies and the average length of the slot 'annotation' for books because in these slots there is more text than in the other ones. See tables 2 and 5 for more details. Finally, we observed also that all systems performed better when users have a tendency to express clear preferences, that means when they provide ratings on the boundaries of the voting scale. In this case, the examples used to train the systems represent a clear indication of user preferences. This is confirmed by the fact that all systems achieve their highest level of accuracy on data sets 3 and 4, in which there is strong positive evidence of user interests (both data set are strongly unbalanced in favor of positive examples and the most frequent ratings are 5 and 6, see tables 2 and 3).

From what said above, it seems that the approaches compared in this paper have complementary *pros and cons*. This naturally leads us to think that some cooperation could take place in order to reach higher effectiveness of the recommendations. For instance, since the probabilistic method has a better accuracy, it could be used for selecting which items should be presented to the user. Then, some kind of filtering could be applied on them, in order to present to the user first those items that are considered positive by the symbolic theories, which are characterized by a slightly low precision but that are more readable from a human understandability viewpoint, and thus can be used to explain and justify the recommendations provided by the system.

5. Conclusions

The paper proposed a new Rocchio-based method able to discover user preferences from textual descriptions of items in online catalogues of e-commerce Web sites. In order to evaluate the effectiveness of the approach, we performed an experimental session on several data sets. Results have been compared with respect to the performance of an ILP approach and a probabilistic one. The comparison highlighted the usefulness and drawbacks of each method, suggesting possible ways of integrating the approaches to offer better support to users.

Acknowledgments

This research was partially funded by the European Commission under the 6th Framework Programme IST Integrated Project VIKEF – Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems; more information at `http://www.vikef.net`).

References

F. Asnicar and C, Tasso, "ifweb: a prototype of user model-based intelligent agent for documentation filtering and navigation in the word wide web", in *Proceedings of 1st Int. Workshop on adaptive systems and user modeling on the World Wide Web*, 1997, pp. 3–12.

D. Billsus and Michael J. Pazzani, "A hybrid user model for news story classification", in *Proceedings of the Seventh International Conference on User Modeling. Banff Canada*, 1999, pp. 99–108.

F. Esposito, G. Semeraro, S. Ferilli, M. Degemmis, N. Di Mauro, T.M.A. Basile, and P. Lops, "Evaluation and validation of two approaches to user profiling", in *Proceeding of the ECML/PKDD-2003 First European Web Mining Forum*, 2003, pp. 51–63.

H. Lieberman, "Letizia: an agent that assists web browsing", in Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 924–929.

L. Ma, J. Shepherd and A. Nguyen, "Document classification via structure synopses", in *Proceedings of the Fourteenth Australasian database conference on Database technologies 2003*, 2003.

T. Mitchell, *Machine Learning*. New York, McGraw-Hill, 1997.

D. Mladenic, "Text-learning and related intelligent agents: a survey", *IEEE Intelligent Systems*, **14**, pp. 44–54, 1999.

R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization", in *Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195–204, San Antonio, US, ACM Press, New York, US, 2000.

M. Orkin and R. Drogin, *Vital Statistics*, New York, McGraw-Hill, 1990.

M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites", *Machine Learning*, **27**, pp. 313–331, 1997.

J. Rocchio, "Relevance feedback information retrieval," in Gerald Salton, editor, *The SMART retrieval system – experiments in automated document processing*, pp. 313–323. Englewood Cliffs, NJ, Prentice-Hall, 1971.

G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*. New York, McGraw-Hill, 1983.

F. Sebastiani,. "Machine learning in automated text categorization", *ACM Computing Surveys*, **34**, 2002.

G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli, "A logic framework for the incremental inductive synthesis of datalog theories," in N. E. Fuchs, editor, Logic Program Synthesis and Transformation, number 1463 in Lecture Notes in Computer Science, pp. 300–321. Springer-Verlag, 1998.

A. Stefani and C. Strapparava, "Personalizing access to web sites: the siteif project", in *Proc. of 2nd Workshop on Adaptive Hypertext and Hypermedia*, 1998.