REBECCA: A Trust-based Filtering to Improve Recommendations for B2C e-Commerce

Domenico Rosaci and Giuseppe M.L. Sarné

Abstract Recommender systems usually support B2C e-Commerce activities without to provide e-buyers with information about the reputation of both products and interlocutors. To provide B2C traders with suggestions taking into account gossips, in this paper we present REBECCA, a fully decentralized trust-based B2C recommender system that also guarantees scalability and privacy. Some experiments show the advantages introduced by REBECCA in generating more effective suggestions.

1 Introduction

Business-to-Consumer (B2C) activities play a relevant role in the e-Commerce (EC) [2] and recommender systems (RS) [21] can support both customers and merchants with suggestions about products and services. To perform such a task, a representation (*profile*) of customer's interests and preferences needs and it can be exploited to implement RSs by using a multi-agent system (MAS) [19] to suitably consider users' orientations [3, 10, 14]. However, MASs need to consider their "social" interactions for allowing an agent to maximize (minimize) its income (outcome) by selecting its interlocutors [7] also by using a trust-based approach [17, 18].

In this paper, we present **RE**putation-**B**ased **E**-**C**ommerce **C**ommunity of **A**gents (**REBECCA**), to support the generic merchant M in generating suggestions for the generic customer C, based on a collaboration between their associated agents. More in detail, (*i*) a merchant agent works as a hybrid RS [21] while (*ii*) a customer agent filters its suggestions based on the customers' satisfaction (i.e. "reputation") about products provided by other agents, weighted in terms of reliability and reputation in the MAS. Many RSs, as [1, 9], consider reliability (reputation) to generate suggestions to present only relevant information according to the personal point of view

Domenico Rosaci, Giuseppe M.L. Sarné
University Mediterranea of Reggio Calabria, 89123 Reggio Calabria (Italy) e-mail: {domenico.rosaci, sarne}@unirc.it

of other trusted users To this aim, several trust models consider user's orientations [5, 8, 16] in computing reliability and/or reputation combining them in a single trust measure [4, 6, 13, 15]. In [11] it is obtained by using a real parameter $\beta \in [0, 1]$, where 1 (0) means all the relevance to reliability (reputation), while REBECCA also considers both the satisfaction of C about a given product and those of the customers for presenting to C only suggestions for high-reputed products. The decentralized architecture makes REBECCA scalable with respect to the MAS size and preserves the C's privacy since (*i*) each merchant agent monitors the C's behaviour in visiting its site and (*ii*) each customer agent locally computes its trust measures.

Finally, the results of some experiments performed by using a JADE prototype of REBECCA clearly show its advantages, in terms of suggestions effectiveness.

2 The REBECCA Framework

In REBECCA each customer C and each merchant M is associated with a personal agent (resp., c and m) using a common *Catalogue* K storing each *product category* (*pc*) and its *product instances* (*pi*). Agents monitors Cs' site visits to each *pi* and *pc* to build their profiles and measure the associated interests. To determine collaborative filtering suggestions, each agent m computes the similarity among its visitors by using the interest values shown in each offered *pi*. Finally, c stores an internal trust model of the products that is based both on the direct past experiences and on suggestions coming from each other customers. More in detail, each user U (either customer or merchant) is associated with an agent a that manages a *User Profile* (UP) consisting of the tuple $\langle WD, BD, TD \rangle$, where:

- WD contains the system parameters α, β and γ (see below), and the current K.
- BD describes the U's past behaviour. Thus for C (M), BD stores, for each visited (offered) *pc*: (*i*) the interest Cr in that *pc* and (*ii*) L_{pc}, a list of elements l_{pi} associated with a $pi \in pc$, visited by C (offered by M). Each l_{pi} stores some data about the C's behaviour (the behaviour of all the customers visiting the site)in visiting *pi* and, in particular, Pr represents the interest in *pi*.
- TD is only in the C'profile and consists of 4 arrays of real values set in [0,1], namely: (*i*) SAT stores the C's global satisfaction for each *pi* based on the other customers' evaluations; (*ii*) $RECC$ contains the suggestions of other customer agents provided to c about the expertise of other customers agents on a *pc*; (*iii*) REP stores customer agents reputation rates computed using suggestions; (*iv*) REL contains the reliability values computed by c using its direct experiences;

Note that the meaning of Cr (Pr) for C is his interest rate in the linked *pc* (*pi*) but for M it is that of all the visitors of his site.

2.1 Computing customer interests and product reliability

When C visits the M's site then the C's (M's) agent c (m) monitors the C's behaviour to computes the interest rates Pr (Cr) for each visited $pi \in pc$ (pc) based on both the time spent in visiting the Web page containing pi (pc) and the time distance between two consecutive visits. As a consequence, $Pr = \alpha \cdot \delta \cdot Pr + (1 - \alpha) \cdot t$ (resp. $Cr = \alpha \cdot \delta \cdot Cr + (1 - \alpha) \cdot t$), where the past C's interest for the involved pi (pr) is weighted by α and δ (i.e., $\delta = (365 - \Delta)/365$ and decreases based on the number of days (Δ) between two consecutive visits).

To evaluate the trust value of each pi which C is interested in, the C's agent requires some recommendations to a subset AS of agents in its MAS, considered as experts in that pc which pi belongs to. For each agent $d \in AS$, each recommendation about d, related to the category pc and coming from the generic agent e is stored in $RECC(e, d, pc)$. Basing on these recommendations, c computes the reputation of d in pc, as $REP(d, pc) = \sum_{e \in AS} RECC(e, d, pc)/|AS|$.

Moreover, the agent c computes the reliability degree of a customer agent d for a pc, considering each opinion $op(d, pi)$ provided by d about each pi belonging to pc, and comparing such opinion with the *feedback* $feed(pi)$ that c provides about pi. Formally:

$$REL(d, pc) = \beta \cdot REL(d, pc) + (1 - \beta) \cdot \frac{1}{q_e \cdot |AS|} \cdot \sum_{e \in AS} \sum_{j=1}^{q_e} \frac{|op(d, pi_j) - feed(pi_j)|}{op(d, pi_j)} \quad (1)$$

where β is a real value, belonging to $[0, 1]$, that represents the importance that c gives to the past evaluations of the reliability with respect to the current evaluation.

Finally, the agent c can compute the product satisfaction of $pi \in pc$ coming from the other customer agents weighted by means of their reliability and reputation integrated in a unique trust measure using a coefficient γ to weight the importance of the reliability vs the reputation. Formally:

$$SAT(pi) = \frac{\sum_{d \in AS} (op(d, pi) \cdot (\gamma \cdot REL(d, pc) + (1 - \gamma) \cdot REP(d, pc)))}{|AS|} \quad (2)$$

3 The REBECCA Recommendation Algorithm

The recommendation algorithm of REBECCA (Figure 1) runs on merchants and customers sides and exploits a *content-based* (CB) and *collaborative filtering* (CF) approaches, using the trust measures described in the previous section.

More in detail, when a customer C visits a merchant site M, the function `Recommend`, executed by the merchant agent m, receives the M's profile and returns the lists CB and CF of pi that m will recommend to c. To this purpose, m

calls the function `extract_pc` receiving the *BD* section of the *M*'s profile for returning the list *L*1 (with all the $pc \in K$ offered by *M*) that is sent to *c*, which answer with the list *L*2 including the *v* (an integer parameter set by *C*) *pc* contained in *L*1 that better meet the *C*'s interests, ordered by *Cr* values. When *L*2 is received by *m*, `contentbased_pi` is called by *M* and returns a list *CB* containing, for each of the *v* $pc \in L2$, at most *y* (*y* is a parameter set by *M*) *pi* having the highest rates. To generate CF suggestions the array of lists *PC* is built by *m*; each element is a list associated with a customer *i*, that visited the site, and stores the *v* most interesting *pc*. To this aim, *m* calls the function `customersInterests`, receiving in input the *BD* data of the *m* profile and *v*. Each array element $PC[i]$ is a list built by *(i)* using the *Cr* values, *(ii)* ordering them in a decreasing order and *(iii)* selecting, for each user, the *v pc* with the highest *Cr* values. Then the function `collaborativefiltering_pi` is called by *m*; it receives the list *L*2 (provided by *c*), the array of lists *PC*, the *BD* section of the *m* profile and the two integers *z* and *x* (set by *M*). For selecting the *z* agents most similar to *c*, this function exploits *PC* to compute the similarity degree between *c* and each customer that interacted with *M* in the past. For each of the *z* most similar customers, at most *x pi* having the highest *Pr* rates are inserted in the list *CF*. The lists *CB* and *CF* are returned to *m* and sent to *c*.

The function `categoriesOfInterest` is executed by *c* when it receives from *m* the list *L*1 and calls `extract_pc` to obtain the list *PCI* with each *pc* of interest for *C*. Then, `intersection_pc` computes the intersection $L1 \cap PCI$ that is stored in $PCI*$. Finally, `select_pc` receives the list $PCI*$ and *v* for returning the first *v* product categories having the highest *Cr* values that are returned into the list *L*2.

Finally, `TBFiltering` is executed on the *c* side, accepting as input the lists *CB* and *CF* sent by *m* that computed them by exploiting `Recommend`. This function joins *CB* and *CF* in the list *P* and by means of `select_pi` provides to prune such lists from each *pi* considered as not interesting based on its $SAT(pi)$ value (see Section 2.1). Then `select_pi` returns each $pi \in P$ having a $SAT(pi) > \eta$ (a parameter set by *C*). This product instances are inserted in a list *FP* returned to *c*.

4 Experiments and Conclusions

This section presents some experiments devoted to show the REBECCA effectiveness in generating useful suggestions to support customers and merchants in their B2C activities. These experiments have been realized by using a JADE (jade.tlab.com) prototype involving 23 XML EC Web sites, a common *Catalogue* (described by a XML Schema) 852 product instances belonging to 10 product categories and a set of 48 real users monitored in their REBECCA B2C activities. To obtain an initial profile of the customers' orientation, the first 10 sites has been used without suggestion support and trusted customer agents. Profiles and trust information are used by the merchant agents to generate suggestions for the other 10 sites. Experiments involved REBECCA, the trust-based recommender system presented in [9] (identi-

```
void Recommend(merchantAgent m, ListOfProductInstancesCB, ListOfProductInstancesCF) {
    ListOfProductCategories L1=extract_pc(m.UP.BD);
    void send(ListOfProductCategories L1, c.Address);
    void receive(ListOfProductCategories L2, int v, int y);
    ListOfProductInstances CB=contentbased_pi(ListOfProductCategories L2, m.UP.BD, int y);
    ListsOfCustomersInterests PC[ ]=customersInterests(m.UP.BD, int v);
    ListOfProductInstances CF=collaborativefiltering_pi(ListOfProductCategories L2,
                                        ListsOfCustomersInterestsPC[ ], m.UP.BD, int z, int x);  }

ListOfProductCategories categoriesOfInterest(ListofProductCategories L1) {
    ListOfProductCategories PCI=extract_pc(c.UP.BD);
    ListOfProductCategories PCI*=intersection_pc(L1, PCI);
    ListOfProductCategories L2=select_pc(PCI*,v);
    return L2;  }

ListOfProductInstances TBFiltering(ListofProductInstances CB,CF, real η) {
    ListOfProductInstances P=union_pi(CB,CF);
    ListOfProductInstances FP=select_pi(P, η);
    return FP;  }
```

Fig. 1 The REBECCA recommendation algorithm.

fied as RS-TB), and the recommender systems EC-XAMAS [3] and TRES [12] that do not consider trust information.

To measure the effectiveness of the proposed suggestions, we have inserted in a list, called A, each *pi* suggested by the merchant agent and in a list, called B, the corresponding customer's choices, and we have compared the corresponding elements in the two lists. We adopted the standard performance metrics precision and recall [20] and set the customer and merchant parameters v, y, z and x to 2, 3, 2 and 3, respectively.

The results (see Table 1) show that REBECCA in terms of precision (resp. recall) performs 19 (resp. 15) percent better than EC-XAMAS, 9 (resp. 10) percent better than TRES and 20 (resp. 19) percent better than RS-TB. In conclusion, experiments show as REBECCA leads to generate more effective recommendations with respect the other systems mainly due to the use of trust information. Indeed, deactivating the use of the trust-based filtering, represented by the function `TBFiltering`. The result thus obtained, represented in Table 1, show that the performances of REBECCA significantly decrease.

As conclusion, in an EC community can be present unreliable or misbehaving interlocutors. To this aim, this paper has been presented REBECCA, a trust-based recommender system acting as CB and CF recommender to support B2C traders by considering customers' orientations, products reputation and traders' trustworthiness. The distributed architecture makes efficient the computation of trust measures and suggestions, gives high scalability and preserves customers' privacy. Some interesting experimental results have shown the advantages of REBECCA.

Acknowledge
This work has been partially supported by the TENACE PRIN Project (n. 20103 P34XC) funded by the Italian Ministry of Education, University and Research.

Table 1 Performances of REBECCA (with (A) and without (B) trust measures), RS-TB, *EC-XAMAS* and *TRES*.

	REBECCA (A)	*REBECCA* (B)	*RS − TB*	*EC-XAMAS*	*TRES*
Pre	0.892	0.783	0.819	0.698	0.802
Rec	0,878	0.795	0.807	0.721	0.778

References

1. P. Avesani, P. Massa, and R. Tiella. Moleskiing. it: a trust-aware recommender system for ski mountaineering. *Int. J. for Infonomics*, 20. 2005.
2. H. Chesbrough. Business Model Innovation: Opportunities and Barriers. *Long Range Planning*, 43(2-3):354 – 363, 2010.
3. P. De Meo, D. Rosaci, G.M.L. Sarnè, D. Ursino, and G. Terracina. EC-XAMAS: Supporting e-Commerce Activities by an XML-Based Adaptive Multi-Agent System. *Applied Artificial Intelligence*, 21(6):529–562, 2007.
4. M. Gómez, J. Carbó J., and C. Benac-Earle. An Anticipatory Trust Model for Open Distributed Systems. volume 4250 of *LNAI*, pages 307–324. Springer-Verlag, 2007.
5. G. Griffiths. Enhancing Peer-to-Peer Collaboration Using Trust. *Expert Systems with Applications*, 31(4):849–858, 2006.
6. T.D. Huynh, N.R. Jennings and N.R. Shadbolt An Integrated Trust and Reputation Model for Open Multi-Agent System. *Autonmous Agent and Multi Agent Sys.*, 13(2):119 – 154, 2006.
7. A. Jösang, R. Ismail, and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618 – 644, 2007.
8. M. Kim and J.H. Ahn Management of Trust in the e-Marketplace: the Role of the Buyer's Experience in Building Trust. *J. of Information Technology*, 22(2):119–132, 2007.
9. J. O'Donovan and V. Smyth Trust in recommender systems. In *Proc. 10th Int. Conf. on Intell. User Interfaces*, pages 167–174. ACM, 2005.
10. L. Palopoli, D. Rosaci, and G. Sarné. A Multi-tiered Recommender System Architecture for Supporting E-Commerce. *Studies in Computational Intelligence 446, Intelligent Distributed Computing VI*, pages 71–81, 2013.
11. D. Rosaci. Trust measures for competitive agents. *Knowledge-Based Systems*, 28(0):38 – 46, 2013.
12. D. Rosaci and G.M.L. Sarnè. TRES: A Decentralized Agent-Based Recommender System to Support B2C Activities. In *Agent and Multi-Agent Systems: Technologies and Applications*, volume 5559 of *LNCS*, pages 183–192. Springer, 2009.
13. D. Rosaci and G.M.L. Sarnè. Cloning Mechanisms to Improve Agent Performances. *Journal of Network and Computer Applications*, 36(1):402 – 408, 2013.
14. D. Rosaci and G.M.L. Sarnè. Recommending Multimedia Web Services in a Multi-Device Environment. *Information Systems*, 38(2):196–212, 2013.
15. D. Rosaci, G.M.L. Sarnè, and S. Garruzzo. Integrating Trust Measures in Multiagent Systems. *International Journal of Intelligent Systems*, 27(1):1–15, 2012.
16. J. Sabater and C. Sierra. REGRET: Reputation in Gregarious Societies. In *Proc. 5th Int. Conf. on Autonomous Agents*, pages 194–195. ACM, 2001.
17. J. Sabater and C. Sierra. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60, 2005.
18. D.H. Sarvapali, S.D. Ramchurn, and N.R. Jennings. Trust in Multi-Agent Systems. *The Knowledge Engineering Review*, 19:1–25, 2004.
19. C. Sierra and F. Dignum. Agent-Mediated Electronic Commerce: Scientific and Technological Roadmap. In *Agent Mediated Electr. Com.*, volume 1991 *LNCS*, pages 1–18. Springer, 2001.
20. C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.
21. K. Wei, J. Huang, and S. Fu. A Survey of E-Commerce Recommender Systems. In *Proc. of the 13th Int. Conf. on Service Systems and Service Management*, pages 1–5, 2007. IEEE.