Recommendation Information Diffusion in Social Networks Considering User Influence and Semantics

Dionisis Margaris
Department of Informatics and Telecommunications
University of Athens, Greece

Costas Vassilakis
Department of Informatics and Telecommunications
University of the Peloponnese, Greece

Panagiotis Georgiadis
Department of Informatics and Telecommunications
University of Athens, Greece

Corresponding author information:
Costas Vassilakis
E-mail: costas@uop.gr
Telephone: +302710372203
Fax: +302710372160

Recommendation Information Diffusion in Social Networks Considering User Influence and Semantics

ABSTRACT

One of the major problems in the domain of social networks is the handling and diffusion of the vast, dynamic and disparate information created by its users. In this context, the information contributed by users can be exploited to generate recommendations for other users. Relevant recommender systems take into account static data from users' profiles, such as location, age or gender, complemented with dynamic aspects stemming from the user behavior and/or social network state such as user preferences, items' general acceptance and influence from social friends. In this paper, we enhance recommendation algorithms used in social networks by taking into account qualitative aspects of the recommended items, such as price and reliability, the influencing factors between social network users, the social network user behavior regarding their purchases in different item categories and the semantic categorization of the products to be recommended. The inclusion of these aspects leads to more accurate recommendations and diffusion of better user-targeted information. This allows for better exploitation of the limited recommendation space, and therefore online advertisement efficiency is raised.

Keywords
Information Diffusion, Social Networks, Collaborative Filtering, Quality of Service, Semantic Information

1 INTRODUCTION

In view of the exponential growth of information generated by online social networks, such as Facebook (Facebook, 2015a) and Twitter (Twitter, 2015), used by millions of people every day, social network analysis is becoming important for many Web applications. Social networks are increasingly used to influence buying decisions of potential customers: according to a study performed by ODM group (Sprout Social, 2011), social networks influence 74% of consumers' buying decisions. Masroor (2015) reports that consumers resort to social media in order to retrieve information that can help them make buying decisions, because social media enable them to (a) keep up with trends, (b) take advantage of sweepstakes and promotions, (c) learn more about the products and services of a company and (d) provide feedback and join brand fan communities. It is worth noting that the first two of these reasons are inherently time-restricted (trends fade and promotions are valid for a limited amount of time), hence timely diffusion of relevant information to interested individuals is of high importance.

Social network data is widely available, however identifying the data relevant to each individual user that are highly useful to support information diffusion tasks at personal level -such as personalized recommendations- still remains a challenge. Nowadays, different types of recommenders exist, which may be based on collaborative filtering, social network data and metrics such as influence, the semantic similarity between items or the items' qualitative characteristics. However, existing algorithms exploit data from either a single or at most two of the aforementioned categories, thus missing opportunities to better tailor the recommendations to the receiving users' profiles.

Collaborative filtering (CF) synthesizes the informed opinions of humans (i.e. opinions that encompass the aspect of satisfaction), to make personalized and accurate predictions and recommendations. Providing CF-based recommendations is a widely-used approach to diffuse information stemming from user behavior and actions. The biggest advantage of CF is that explicit content description is not required (as in content-based systems): instead, traditional

CF relies only on opinions expressed by users on items either explicitly (e.g. a user enters a rating for the item) or implicitly (e.g. a user purchases an item, which indicates a positive assessment). In the context of CF, personalization is achieved by considering ratings of "similar users", under CF's fundamental assumption that if users X and Y have similar behaviors (e.g., buying, watching, listening) on some items, they will act on other items similarly (Hwang and Yoon, 1981). Traditional recommender systems assume that users are independent and ignore the social interactions among them. Due to this fact, they fail to incorporate important aspects that denote interaction, tie strength and influence among users, which can substantially enhance recommendation quality (Facebook, 2015b; He and Chu, 2010).

Social network data-based recommender systems consider static data from the user profile, such as location, age or gender, complemented with dynamic aspects stemming from the user behavior and/or social network state such as user preferences, items' general acceptance and influence from social friends (Facebook, 2015b; He and Chu, 2010). Furthermore, *tie strength* between users of social networks can be exploited to enhance the choice of recommenders, so as to consider the opinions and choices of users that have a high influence on the user for whom the recommendation is generated (Arazy et al., 2009; Oechslein and Hess, 2014; Quijano-Sanchez et al., 2011). A first approach to identifying highly influential individuals within the social network would be to consider those having high tie strengths, such as family members or friends with similar age. Nevertheless, the influence of such individuals may be limited only to certain item categories (e.g. one may trust her friends regarding vacation packages and sunglasses, but not when it comes to clothing or shoes). Moreover, selected individuals with low tie strength, such as actors and singers, may influence a user regarding some specific categories (e.g. clothing and shoes), while for some categories a user may not be influenced at all. For instance, a user may consider herself an expert in smartphones, hence she decides exclusively on her own, after examining the qualitative characteristics of the products such as battery life, or camera resolution.

Recently, it has been identified that recommender systems should take into account qualitative aspects of items, such as price and reliability, the individual user behavior regarding purchases in different item categories, and semantic categorization of products (Boulkrinat et al., 2013; Margaris et al., 2015a). For example, if a user typically buys shoes in the price range of $80-$150, it would not be appropriate to recommend a $500 pair of shoes because some user having a high influence on the particular item category has made that purchase. A more fitting approach would be to recommend a pair of shoes *of the same style* with the $500 pair but costing less (e.g. a replica), to fit the profile of the user receiving the recommendation.

In this paper, we propose a novel algorithm for diffusing information to social media users, in the form of recommendations. The proposed algorithm considers two input information flows: (a) new items or items with modified characteristics (e.g. special offers or warranty extension) and (b) clicks on recommended items or item purchases made by influencers. These data streams are processed and appropriate information is diffused towards social network users in the form of recommendations. Accepted recommendations will trigger further information propagation to the social network members, since they constitute items in the second stream. We exploit the notion of social influence (Anagnostopoulos, Kumar and Mahdian, 2008; Guille et al., 2013) and identify influential individuals (Ver Steeg and Galstyan, 2012) at a personal level, in order to cascade information through appropriate paths. As noted above, timely diffusion of information is important so as to allow users to keep up with trends and exploit promotional offers. For new products in particular, this is reaffirmed by an early study done by the Chief Marketing Officer (CMO) Council and Lithium, which revealed that 80% of respondents "tried new things based on friends' suggestions" (Olenski, 2012).

The proposed algorithm enhances the state-of-the-art recommendation algorithms used in social networks since it combines the following features:

a) it considers the influencing factors between social network users and the social network user behavior regarding their purchases. Both these aspects are examined in a *per item category basis*, to increase recommendation accuracy.

b) it has the ability to allow users to follow trends, by recommending items that are highly similar to the trending ones but within the buying habits of each user (e.g. within the usual purchase price range in the particular category)

c) it takes into account qualitative aspects of the recommended items, such as price and reliability.

d) it exploits the semantic categorization and semantic distance of the products to be recommended in order to select the information to be diffused.

e) it has been designed for incorporation into online systems, by considering streams of events (information flows regarding new items, items with modified characteristics and click streams) and being structured using *offline* and *online* phases, so as to achieve adequate performance.

Finally, experiments have been performed to tune algorithm parameters and validate its performance, both in terms of execution speed and recommendation quality.

The rest of the paper is structured as follows: section 2 overviews related work, while section 3 presents the algorithm's prerequisites. Section 4 presents the recommendation formulation for information diffusion algorithm, while section 5 evaluates the proposed algorithm. Finally, section 6 concludes the paper and outlines future work.

2 RELATED WORK

Bakshy et al. (2012a) examine the role of social networks in online information diffusion with a large-scale field experiment that randomizes exposure to signals about friends' information and the relative role of strong and weak ties in information propagation. Bakshy et al. (2012b) measure social influence via social cues on an economically relevant form of user behavior and average rates of response. Their results demonstrate the substantial consequences of including minimal social cues in advertising and quantify the positive relationship between a consumer's response and the strength of their connection with an affiliated peer. Both these works establish that an appropriate recommendation algorithm is a valuable tool for successful information diffusion in social networks. Another contribution of this work is the consideration of methods for enhancing the effectiveness of the diffused information, e.g. using social cues to maximize the probability that a recommendation is adopted. Oechslein and Hess (2014) also assert that a strong tie relationship has a positive influence on the value of a recommendation.

In the domain of recommender systems, numerous approaches for formulating recommendations have been proposed in the relevant literature. Collaborative filtering formulates personalized recommendations on the basis of ratings expressed by people having similar tastes with the user for whom the recommendation is generated for; taste similarity is computed by examining the resemblance of already entered ratings (Schafer et al., 2007). Research has proven that the CF-based recommendation approach is the most successful and widely used approach for implementing recommendation systems (Zhang et al., 2013). CF can be further distinguished in user-based and item-based approaches (Herlocker et al., 2004). In user-based CF, a set of nearest neighbours for the target user is first identified, and the prediction value of items that are unknown to the target user is then computed according to this set. On the other hand, item-based CF proceeds by finding a set of similar items that are rated by different users in some similar way. Subsequently, predictions are generated for each candidate item, for example, by taking a weighted average of the active user's item ratings on these neighbour items. It has been shown that item-based CF can achieve prediction accuracies that are comparable to or even better than user-based CF algorithms (Balabanovic and Shoham, 1997).

Recently, with the advent of social networking, social network recommendation has received considerable research attention. Konstas et al. (2009) investigate the role of social networks' relationships in developing a track recommendation system based on a common CF item recommendation method, by taking into account both the social annotation and friendships inherent in the social graph established among users, items and tags. Arazy et al. (2009) outline a conceptual recommender system design within which the structure and dynamics of a social network contribute to the dimensions of trust propagation, source's reputation and tie strength between users, which are then taken into account by the system's prediction component to generate recommendations. Quijano-Sanchez et al. (2011) enhance a content-based recommender system by including in the recommendation algorithm the trust between individuals, users' interaction and aspects of each user's personality; this enhancement results to a 12% improvement of recommendation accuracy.

Walter et al. (2009) combine findings from recommender systems, ubiquitous systems and market analysis, investigating which types of retail stores would benefit from a personalization strategy and to which extent. Furthermore, they elaborate on building a recommender system tailored to the needs of a retail environment, considering different recommendation approaches (item-to-item CF, user-to-user CF and trust-based recommender) to achieve personalization. Finally, they propose a layered architecture for recommender systems named "LARS", which can serve as a blueprint for real-world implementations.

Walter (2011) discusses the issue of combining user communities and electronic marketplaces, showing that when users consider trust relationships during coalition formation on social networks, they are able to team up with users that are trustworthy even if they are far away in the social network. This allows users to achieve better match of preferences or better prices without disadvantages such as uncertainty about who they are interacting with. This work proposes a model in which agents use their trust relationships in order to determine who to form coalitions with, and shows that agents can learn who is trustworthy and who is not, without having any initial knowledge about the trustworthiness of other agents. Jamali and Ester (2010), employ matrix factorization techniques and a mechanism of trust propagation to create formulate recommendations in social networks.

Cai et al. (2010) propose a model that captures the bilateral role of user interactions within a social network and formulate CF methods to enable people to people recommendation. In this model users can be similar to other users in two ways – either having similar "taste" for the users they contact, or having similar "attractiveness" for the users who contact them. A neighbor-based CF algorithm was also developed to predict, for given users, other users they may like to contact, based on user similarity in terms of both attractiveness and taste. He and Chu (2010) analyze data from a social network and establish that friends have a tendency to select the same items and give similar ratings. They also show that using social network data within the recommender system improves prediction accuracy and also remedies the data sparsity and cold-start issues inherent in CF.

The issue of trust enhancement has been recently studied in social network-based recommenders, since trust building has been shown to increase purchasing probability (Bakshy et al., 2012a). Shuiguang et al. (2014) propose a social network-based service recommendation method with trust enhancement. First, a matrix factorization method is utilized to assess the degree of trust between social network users and next an extended random walk algorithm is used to obtain recommendation results. Li et al. (2014) model the user's social network using social network analysis and mining methods. They introduce a set of new measures for user influence and social trust. Wanget et al. (2015) present Friendbook, a novel semantic-based friend recommendation system for social networks, which recommends friends to users based on their lifestyles instead of social graphs. They introduce a similarity metric for user lifestyles, and calculate users' impact in terms of lifestyles with a friend-matching graph.

None of the social networking-based recommender algorithms mentioned above provide the following features: (a) consider the influencing factors between social network users and the

social network user behavior regarding their purchases, (b) allow users to follow trends, by recommending items that are highly similar to the trending ones but within the buying habits of the relevant user (c) take into account qualitative aspects of the recommended items, such as price and reliability (d) exploit semantic categorization and semantic distance of products to be recommended for selecting the information to be diffused and (e) have a suitable task separation in offline and online procedures to enhance performance. The proposed algorithm aims to fill this gap, in order to provide more accurate and effective recommendations in an efficient way.

3 SOCIAL NETWORKING, SEMANTIC DATA MANAGEMENT AND QUALITY OF SERVICE FOUNDATIONS

In the following subsections we summarize the concepts and underpinnings from the areas of social networking, semantic data management, and quality of service (QoS), which are used in our work.

3.1 Influence in social networks

Within a social network, "social friends" greatly vary regarding the nature of the relationship holding among them: they may be friends or strangers, with little or nothing in between (Gilbert and Karahalios, 2009). Users have friends they consider very close and know each other in real life, and acquaintances they barely know such as singers, actors and athletes. According to Anagnostopoulos et al. (2008), three main causes of correlation in social networks exist: *influence* (also known as *induction*), *homophily* and *environment* (also referenced as *external influence*). Due to *influence*, an action of a user is triggered by one of his/her friend's recent actions (e.g. when a user buys a product because one of his/her friends has recently bought the same product). Homophily refers to the phenomenon that individuals often establish "social friendship" with others who are similar to them, and hence perform similar actions (e.g. sharing a common interest, such as mountaineering). Finally, *environment* refers to the phenomenon that external factors are correlated both with the event that two individuals become friends and also with their actions (e.g. two inhabitants of the same city posting pictures of the same landmarks in an online photo sharing system can become "social friends").

Bakshy et al. (2012b) suggest that a social network user responds significantly better to advertisements that originate from friends of the social network to who the user has high *tie strength*. The strength of the directed tie between users *i* and *j* is computed as:

$$W_{i,j} = \frac{C_{i,j}}{C_i} \tag{1}$$

where C_i is the total number of communications posted in a certain time period in the social network by user *i*, whereas $C_{i,j}$ is the total number of communications posted by user *i* on the social network during the same period and are directed at user *j* or on posts by user *j*. Bakshy et al. (2012b) use a 90-day period to compute tie strength.

The tie strength metric can be used to locate the influencers of a user. However, being a global metric, it cannot provide, by itself, information on which friends influence user *i* regarding a particular product category. To produce a more fine-grain metric and enhance recommendation accuracy, we exploit user interests that are collected either by search engines (e.g. Google interests, https://support.google.com/ads/answer/2842480) or the social network (e.g. Facebook interest targeting, https://www.facebook.com/help/188888021162119). Given that these interest lists are built transparently to the users when they simply browse the web/social network and/or post content, they are bound to include all product categories each user is actually interested in. Having the information about users' interests available, we may define the influence level of user *j* on user *i*, regarding item category C as:

$$IL_{i,j}(C) = \begin{cases} W_{i,j}, if\ C \in interests(i) \wedge C \in interests(j) \\ 0, otherwise \end{cases} \tag{2}$$

i.e. we use the value of the tie strength for categories in both users' interests and a value of 0 for all other categories. One possible enhancement for further improving the accuracy of

influence level estimation would be to count more strongly the communications whose content is related to category C. Another possible enhancement is to take into account the number of recommendations within category C that reached user *i* as a result of activities of user *j* (i.e. clicks on recommended items or item purchases made by user *j*) that were finally accepted (clicked) by user *i*. Therefore, the influence level can be defined as

$$IL_{i,j}(C) = \begin{cases} a * W_{i,j} + (1-a) * \dfrac{Rec_{i,j}^{acc}(C)}{Rec_{i,j}(C)}, if C \in interests(i) \wedge C \in interests(j) \\ 0, otherwise \end{cases} \quad (3)$$

where $Rec_{i,j}(C)$ is the total number of recommendations within category C, that reached user *i* as a result of activities of user *j*, $Rec_{i,j}^{acc}(C)$ corresponds to the number of these recommendations that were accepted by user *i* and *a* is a constant ($0 \leq \alpha \leq 1$) indicating the weight that is assigned to each influence factor (communications and acceptance of recommendations). Both these aspects will be considered as part of our future work.

3.2 Product semantic information and semantic similarity

In order to generate valid recommendations, the algorithm needs to be able to find which items are similar, and are thus candidate for recommendation when a new item is introduced or some item is viewed/purchased by a user. This is achieved through recording semantic information about products and using it to compute semantic similarity among products. In this work, we adopt a modified version of the similarity measure proposed by Hau et al. (2005) and adapted by Chedrawy and Abidi (2009). According to this approach, the semantic similarity between two items *I* and *J* is based on ratio of the common/shared RDF descriptions between *I* and *J* (*count_common_desc(I,J)*) to their total descriptions (*count_total_desc(I,J)*), i.e.:

$$SemSim(I,J) = \frac{count\_common\_desc(I,J)}{count\_total\_desc(I,J)} \quad (4)$$

However, when two products *I* and *J* are described in RDF, their descriptions might not be identical, yet be semantically close. For instance, two digital cameras C_1 and C_2 may have resolutions equal to 20.4 MPixels and 20.1 MPixels respectively. Clearly, these values are not identical, nevertheless it would not be appropriate to count them as totally different: in fact, they should be counted as highly similar. To tackle this issue, we modify the above formula to

$$SemSim(I,J) = \frac{\sum_{p \in I \wedge p \in J} sim_p(p_i, p_j)}{count\_total\_desc(I,J)} \quad (5)$$

where *p* is a property, p_i and p_j are the values of property *p* for items *I* and *J*, respectively, and sim_p is a function computing the similarity between values of property *p*. E.g., for the property *cameraResolution* of digital cameras, the relevant similarity function can be defined as

$$sim_{cameraResolution}(r1, r2) = 1 - \frac{|\mathrm{r1} - \mathrm{r2}|}{\mathrm{b} - \mathrm{a}} \quad (6)$$

where *b* and *a* are the minimum and maximum values in the set of available camera resolutions (a typical value normalization formula (Aslam and Montague, 2001; He and Wu, 2008)).

The same function can be used to compute the similarity between most numeric values, including prices, screen sizes etc. For values with a Boolean domain, the respective similarity value function is also straightforward:

$$sim_{boolean}(b1, b2) = \begin{cases} 1, if b1 = b2 \\ 0, otherwise \end{cases} \quad (7)$$

In other value domains, custom similarity metrics can be used. For instance a restaurant with Pakistani cuisine can be considered of high similarity to a restaurant with Indian cuisine, but of low similarity with a French restaurant. Admittedly, defining similarity functions for each distinct property within the ontology's RDF property list is a tedious task. However, the problem can be alleviated by employing automated similarity computation methods for specific domains, e.g. the metrics sim_g and sim_d proposed by Pirasteh et al. (2014) for movie genres and

movie directors; automated distance measures for colors (Androutsos et al., 1998); mechanistically computed similarity metrics for music genres and artists (Whitman and Lawrence, 2000; Schedl et al., 2008); and so forth. When in lack of a more elaborate comparison metric, the default similarity function shown in equation (8) can be used.

$$sim_{default}(v1, v2) = \begin{cases} 1, if v1 = v2 \\ 0, otherwise \end{cases} \quad (8)$$

Using such a function will provide performance identical to the method used in (Chedrawy and Abidi, 2009), while introduction of range-specific similarity functions will leverage the performance of the metric. An additional approach to alleviate the problem is to decrease the amount of needed similarity functions by considering only the products' salient features in similarity computation. For example, when considering digital cameras, resolution and available memory are salient features, but color is not. The properties corresponding to the salient features can be tagged appropriately within the semantic repository and the algorithm would then consider only those features when determining semantic similarities.

It is worth noting that the $SemSim$ similarity metric appropriately handles cases of products with overlapping functionalities. For instance, a smartphone will have a medium degree of similarity with a digital camera, since the smartphone includes a digital camera, and the camera-related properties of both items will be suitably processed according to the formulas presented above. The non-common attributes (such as attributes related to the smartphone's cell phone operation and application execution) will increase the value of the denominator in equation 5, leading to a lower similarity value, as would be expected.

3.3 QoS information concerning products

QoS may be defined in terms of attributes (ITU, 1988), while typical attributes considered are cost, response time, availability, reputation, security etc. (Cardoso, 2002). In this paper we will consider only the attributes cost (c) and reliability (rel), because these are two main attributes considered in purchases (users typically try to minimize cost and maximize reliability of purchased goods), adopting their definitions from (O'Sullivan et al., 2002). Inclusion of additional attributes (both in terms of model extension and processing) is straightforward, thus we have no loss of generality. Regarding the cost, actual prices are used; item reliability is encoded in a scale of 1-10, with larger values denoting higher reliability. An example of the items' qualitative characteristics values are shown in table 1.

Table 1. Sample QoS values within the repository

Item	cost	reliability
Samsung Galaxy S3	$200	8
Samsung Galaxy S4	$350	10
Samsung Galaxy S5	$470	9
. . .		

Reliability scores are typically pertinent to technology products including cars (Car reliability index, 2016), computers (e.g. Squaretrade, 2009), smartphones (e.g. Squaretrade, 2010), digital cameras (Digicamhelp, 2010) etc. When detailed reliability information on a specific model is not available (which is always the case for newly appearing products), model reliability can be approximated by considering the manufacturer's reliability record, under the assumption that the quality of the different models of a manufacturer is consistent (Wang, Huang and Chu, 2013). The reliability attribute may not pertinent for other goods or services categories, e.g. restaurants; in such cases, the same methodology can be used with other QoS attributes.

We note here that although more expensive items within a category are deemed to be more reliable than less expensive ones, this is not always the case. For instance, the Suzuki Swift is reported by (Reliability index, 2016) to be considerably more reliable than the Fiat Punto (the Suzuki Swift scores 71 marks while the Fiat Punto scores 121 marks; lower marking indicates

higher reliability), while the respective list prices are £8,999 (Suzuki Swift, basic model - http://www.suzuki.co.uk/cars/cars/new/swift/swift/price) and £10,990 (Fiat Punto, basic model http://www.fiat.co.uk/PublishingImages/price-list/Fiat_Price_List_November_2015.pdf). Analogous cases can be found in Digicam's report on digital cameras (Digicamhelp, 2010), where Panasonic's value cameras (under $300) have a lower failure rate (i.e. are more reliable) than Canon's premium cameras (over $300).

3.4 User's profile for enabling information diffusion through recommendations

As discussed in the introduction, certain users are influenced regarding their purchases by other users. The set of influencers may vary between item categories, whereas in some categories a user may not be influenced at all, or be influenced very little.

In order to accommodate these aspects in the recommender system, we follow the approach presented by Margaris et al. (2015b), adapting it appropriately. The item recommendation process is synthesized by executing in parallel (a) a CF-based recommendation algorithm and (b) a QoS-based recommendation algorithm. Then, the two individual results are combined into a single score, using a metasearch result combination algorithm (Aslam and Montague, 2001). In our case, the CF-based recommendation algorithm considers the opinions of the user's influencers for the particular item category. A distinct set of influencers is maintained in the user's profile for each item category, to increase the accuracy of the recommendations.

The QoS-based recommendation algorithm considers only the qualitative characteristics of the item. Additionally, for each user we store in the profile the average cost and reliability of purchases that the user makes for different item categories, so as to be able to determine how close each product is to the buying habits of the particular user.

In order to combine the individual results, we opt for a linear combination model (Aslam and Montague, 2001) and in particular the WCombSUM_i method (He and Wu, 2008). According to this method, the overall score for each item *i* personalized for user *u* is equal to:

$$WCombSUM_{i,u} = w_{CF,C(i),u} * score_{CF,i,u} + w_{QoS,C(i),u} * score_{QoS,i,u} \quad (9)$$

where $score_{CF,i,u}$ and $score_{QoS,I,u}$ are the recommendation scores for item *i* produced by the CF-based and the QoS-based recommendation algorithm respectively for user *u*, whereas $w_{CF,C(i),u}$ and $w_{QoS,C(i),u}$ are the weights assigned to the CF-based and the QoS-based recommendation algorithm, respectively. The algorithm weights are calculated for each user *u* individually (different weights may apply to different users) and with a granularity of *item category* (*C(i)* is used to denote the category of item *i*). Under this setting, a particular user U_l may have different values for $w_{CF,C(i),u}$ in regards to two distinct item categories, e.g. "smartphones" and "clothing".

In order to compute the values of the weights, we employ the following formulas:

$$w_{CF,C(i),u} = \frac{\left|ItemsClickedOrPurchased_{C(i),u} \cap ItemsSuggestedByInfluencers_{C(i),u}\right|}{\left|ItemsClickedOrPurchased_{C(i),u}\right|} \quad (10)$$

$$w_{QoS,C(i),u} = 1 - w_{CF,C(i),u} \quad (11)$$

Effectively, $w_{CF,C(i),u}$ is the ratio of the online advertisement clicks or purchases made by user *u* within item category *C(i)* and have been suggested by influencers, to the overall number of online advertisement clicks or purchases made by user *u* within item category *C(i)*. Obviously, the higher this ratio, the more receptive user *u* is to suggestions made by influencers, hence the weight assigned to the CF-based algorithm increases. This metric is analogous to the recall metric used in information retrieval (Manning et al., 2008).

Regarding the calculation of the set $ItemsSuggestedByInfluencers_{C(i),u}$, recall from section 1 that the proposed algorithm considers two information flows for selecting information to diffuse to other social network users: (a) new items or items with modified characteristics (e.g. special offers or warranty extension) and (b) clicks on recommended items or item purchases made by influencers. Based on this setting, an online advertisement impression is

deemed to have been suggested by an influencer if it has been triggered by information flow (b). A purchase is deemed to have been suggested by an influencer if (i) it has been performed after clicking on an advertisement that has been suggested by an influencer or (ii) the same item has been purchased by an influencer eight days before the user's purchase or less. The time frame of eight days has been chosen by considering the results presented in (Bakshy et al., 2012a), according to which the cumulative distribution of information lags between the subject and their first sharing friend reaches an almost steady-state within 8 days.

The formula computing the CF_{weight} ($w_{CF,C(i),u}$in the equations above) suffers from the cold start problem, i.e. the case that none (or very few) data are present for the specific category within the system. In particular, the quantity $w_{CF,C(i),u}$ is not computable according to the above formula when no items have been clicked or purchased in the specific category (e.g. a category just added in the user's interests), whereas if very few items have been clicked or purchased within the specific category, the value of $w_{CF,C(i),u}$ will not convey accurate information. In these cases, we set $w_{CF,C(i),u}$ to a default value of 0.4, based on the results presented in (Margaris et al., 2015b). According to these results, a CF_{weight} value of 40% ensures that the QoS levels desired by the user are maintained, while in parallel the CF dimension is adequately considered in the final recommendation.

4 THE RECOMMENDATION INFORMATION DIFFUSION ALGORITHM

As stated in section 1, the proposed algorithm considers two information flows for selecting information to diffuse to other social network users (a) new items or items with modified characteristics (e.g. special offers or warranty extension) and (b) clicks on recommended items or purchases made by influencers. In the rest of this section, the steps taken to initialize the algorithm, process the information flows and finally diffuse the information within the social network are described in detail.

Step 1 – Offline Initialization. A series of actions is initially performed in an offline fashion to bootstrap the algorithm. These actions are as follows:

- for each item category, the recommendation algorithm initially identifies the minimum and the maximum item cost in the category, using the formulas:

$$minCost(C) = \min_{item_i \in C}(cost(item_i)) \quad (12)$$

$$maxCost(C) = \max_{item_i \in C}(cost(item_i)) \quad (13)$$

  where *C* is a category. Similarly, the *minRel(C)* and *maxRel(C)* quantities are computed, corresponding to the minimum and maximum reliability of items in category *C*.

- The semantic similarity among pairs of items in the database is computed. Since item pairs whose semantic similarity score is low will never be needed (Karaiskos, 2013), the computation of such similarities is suppressed to save computing power. The semantic similarity computation formula given in equation (5) will certainly produce low scores when the categories in which items *I* and *J* belong to, have only few properties in common, as compared to the their overall number of properties (e.g. food and smartphones). Therefore, we do not compute or store similarities between items *I* and *J* belonging in categories C_1 and C_2 respectively where

$$StructuralSimilarity(C_1, C_2) = \frac{|properties(C_1) \cap properties(C_2)|}{|properties(C_1)| + |properties(C_2)|} < 0.4 \quad (14)$$

  Furthermore, for items that the semantic similarity *is* computed, but is found to be below the value of 0.4 (due to low similarity of property values within identical properties), the similarity score is not stored, to save storage space. When the similarity score of two items *I* and *J* is not found in the information repository, a value of zero is assumed,

indicating that the products are too dissimilar, and the recommendation algorithm should never propose *J* when triggered by item *I*, or vice versa.

The threshold of 0.4 has been adopted, by considering the results of the experiment presented in section 5.2 for determining the recommendation score threshold for information diffusion. These results show that the optimal threshold regarding the estimated user interest in an item, in order to diffuse an impression to the user is 0.4. Furthermore, the semantic similarity of the products contributes to the computation of the estimated user interest in an item (c.f. computation of $score_{CF,i,u}$ in step 2, below), hence item pairs with semantic similarity less than the recommendation score threshold are highly unlikely to contribute to the formulation of recommendations that will finally be diffused to the users. Note that the structural similarity of two categories C_1 and C_2 is an upper bound for the semantic similarity of any item pair i_1 and i_2 with $i_1 \in C_1$ and $i_2 \in C_2$. Indeed, the denominators of formulas $SemSim(i_1, i_2)$ and $StructuralSimilarity(C_1, C_2)$ are equal (the count of common properties), however the numerator of the $SemSim(i_1, i_2)$ formula will be equal to the numerator of the $StructuralSimilarity(C_1, C_2)$ formula only in the case that all common properties have identical values, and smaller in all other cases.

- For each item category (Google Inc., 2015a) and user, the recommendation algorithm computes the values of $w_{CF,C(i),u}$ and $w_{QoS,C(i),u}$, using the formulas presented in subsection 3.4. The mean cost and mean reliability of the items that the user has bought or viewed in the past are also computed.
- For each user and item category, the recommendation algorithm initially identifies its *top N* strongest influencers within the specific category as described in subsection 3.1. Then, the friends having the top N influence levels are maintained and stored in the user profile, tagged with the specific category. In this work, we use the value *N=8*, since we have experimentally determined that this value is adequate for producing accurate recommendations. The relevant experiment is described in subsection 5.1.

Step 2 – Online operation: Once the algorithm has been bootstrapped, it can be executed in an online fashion to produce recommendations. This part of the algorithm is executed in an event-based manner, with the steps described below being performed when events appear in information flow (a), which corresponds to new items or items with modified characteristics are introduced, or in information flow (b), which corresponds to clicks on recommended items or item purchases by influencers. The way that events are processed is described in the following paragraphs.

i) *Events appearing in information flow (a)*

An item *i* (either new or with modified characteristics) associated with such an event and belonging in category *C* is of potential interest to all users of the social network that have an interest in category *C*. To this end, initially the set of users whose profile includes category *C* are extracted. Subsequently, the algorithm estimates each user's interest in the particular item by considering (a) his/her purchase and impression view record on that particular category, which determines the item's QoS-score, and (b) the activities of her influencers regarding products in the specific category, which determine the item's CF-score. More specifically, the QoS score for each user is computed as follows:

$$score_{QoS,i,u} = price\_vicinity(u,i) * reliability\_vicinity(u,i) \quad (15)$$

where *price_vicinity* and *reliability_vicinity* are defined as follows:

$$price\_vicinity(u,i) = 1 - \frac{\min(|price(i) - MP(u,C)|, |offerPrice(i) - MP(u,C)|)}{maxCost(C) - minCost(C)} \quad (16)$$

$$reliability\_vicinity(u,i) = \begin{cases} 1 - \dfrac{|rel(i) - MR(u,c)|}{maxRel(C) - minRel(C)}, if\ rel(i) \leq MR(u,c) \\ 1, if\ rel(i) > MR(u,c) \end{cases} \quad (17)$$

In the equations above, *price(i)* is the original price of the item, *offerPrice(i)* is the offer price of the item (if the event corresponds to a new item and not an offer, *offerPrice(i)* is set equal to *price(i)*), *rel(i)* is the reliability of the item, and *MP(u,C)* and *MR(u,C)* are the mean price and the mean reliability respectively of purchases made and impressions viewed by user *u* within category *C*. Price vicinity indicates how close the item price is to the user's buying habits within the specific category. Price vicinity is examined both for the item's original price and the offer price and then the minimum distance is taken. The rationale behind this computation is to capture both the case that the user buys typically falling in her usual price range at a lower price (the item's original price is close to *MP(u,C)*) and the case where the usual amount of money *MP(u,C)* is spent, but this time to buy a product typically having a higher price (the item's offer price is close to *MP(u,C)*).

When computing reliability vicinity, we consider an item close to the user's preferences if its reliability is either equal to or higher than the mean reliability of the items that the user purchases in this category. The rationale behind this calculation is that products that are more reliable would typically be of interest to the user. For products having reliability less than *MR(u,C)*, a typical normalized distance metric is employed.

Recall from subsection 3.1 that we use the user interests that are collected by search engines or the social network to determine the categories that a user is interested in. Since these lists of interests are built automatically, based on the web/social network pages that the user views, the presence of some interest in these lists does not imply that the user has made purchases or viewed impressions of items within the particular category in the past. Frequent visits to informational pages classified in the particular category are adequate to get a category included in the interest lists. A user may also explicitly register her interest in a category she is starting to develop an interest in, but has not made any relevant purchases or viewed impressions yet (e.g. a parent having decided to enroll her daughter to a ballet class after summer vacation is over). In the absence of histories of purchases made and impressions viewed by a user, the quantities *MP(u,C)* and *MR(u,C)* are computed taking into account the relevant quantities of the user's influencers in the particular category. More specifically *MP(u,C)* is computed as

$$MP(u,C) = \frac{\sum_{IN \in influencers(u,C)} IL_{u,IN}(C) * MP(IN,C)}{\sum_{IN \in influencers(u,C)} IL_{u,IN}(C)} \quad (18)$$

where $IL_{u,IN}(C)$ is the influence level of user *IN* on user *u* regarding category *C* (c.f. subsection 3.1). An analogous computation is performed for *MR(u,C)*.

Regarding the CF-score, we first extract from the user's profile the N influencers for category C. Subsequently, for each of these influencers *IN*, we locate the item p_{IN} within category *C* that *IN* has purchased and has the greatest semantic similarity with item *i* (i.e. the item to which the event is associated). Then, the CF-score is computed as

$$score_{CF,i,u} = \frac{\sum_{IN \in influencers(u,C)} IL_{u,IN}(C) * SemSim(i, p_{IN})}{\sum_{IN \in influencers(u,C)} IL_{u,IN}(C)} \quad (19)$$

where $IL_{u,IN}(C)$ is the influence level of user *IN* on user *u* regarding category *C* (c.f. subsection 3.1) and $SemSim(i, p_{IN})$ is the semantic similarity between products *i* and p_{IN}.

Afterwards, the User's Interest Probability $UIP_{i,u}$ on the item is computed, using the weighted sum combination function discussed in subsection 3.4, i.e.

$$UIP_{i,u} = w_{CF,C(i),u} * score_{CF,i,u} + w_{QoS,C(i),u} * score_{QoS,i,u} \quad (20)$$

where $w_{CF,C(i),u}$ and $w_{QoS,C(i),u}$ are the weights assigned to the CF and QoS dimensions regarding recommendations made to user *u* for category *C(i)* (i.e. the category of the item

appearing in the event). Details on the computation of these weights are given in subsection 3.4.

Finally, if the value of $UIP_{i,u}$ meets or exceeds the UIP threshold (subsection 5.3 discusses the computation of the UIP threshold's value), the information is finally diffused to user u.

The algorithm described above is illustrated using pseudocode in Figure 1.

```
diffuse_on_item_trigger(item) {
/*
    Input:  An item that has been newly added or that had its characteristics modified (e.g. an
            offer).
    Output: None
    Effects: Information has been diffused to the members of the social network for which the
            interest probability exceeds the appropriate threshold.
*/
category = item.category;
candidate_users = get_users_with_interest(category);

foreach user in candidate_users {
    influencers = get_influencers(user, category);

    /* Compute QoS score */
    if (history(user, category) ≠ ∅) {
        mean_price = history(user, category).get_mean_price();
        mean_reliability = history(user, category).get_mean_reliability();
    }
    else {
        /* use equation 19 to compute mean price and reliability as a weighted mean of the
           corresponding influencers' values */
        mean_price = mean_price_from_influencers(influencers, category);
        mean_reliability = mean_reliability_from_influencers(influencers, category);
    }
    min_price = category.min_price();
    max_price = category.max_price();
    min_reliability = category.min_reliability();
    max_reliability = category.max_reliability();
    pv = price_vicinity(item, mean_price, min_price, max_price);
    rv = reliability_vicinity(item, mean_reliability, min_reliability, max_reliability);
    qos_score = pv * rv;

    /* Compute CF score */
    foreach inf in influencers {
        influencer_items = history(inf, category);
        candidate_item_array[inf] = get_item_with_max_sem_sim(candidate_items, item);
    }
    /* use equation 20 to compute the cf_score as a weighted mean */
    cf_score = compute_cf_score(influencers, candidate_item_array);

    /* compute user interest probability and diffuse if appropriate*/
    uip = qos_score * user.qos_weight[category] + cf_score * user.cf_weight[category];
    if (uip >= uip_threshold)
        diffuse(item, user);
}
```

Figure 1. Pseudocode for the algorithm that diffuses information for new items or items with modified characteristics

In this figure we may note that the following quantities are precomputed during the offline phase (step 1 - offline initialization and step 3 – repository update): (a) the set of users with an interest in the specific category, (b) the category-specific influencers per user and their corresponding influence levels, (c) the minimum and maximum price and reliability per

category, (d) the mean price and reliability of users' purchases per category, (e) the semantic similarities between items, and (f) the weights of the QoS and CF dimensions per user and category. Using precomputed values for these quantities drastically reduces the time needed to execute the algorithm. Regarding the complexity of the algorithm, it can be analyzed as follows:

- In order to compute the QoS score for a single user, all items are precomputed, except for the case that the user's history on the category is empty. In the latter case, the mean price and reliability are computed as weighted averages of the corresponding values of the user's influencers, therefore the worst-case complexity is

$$complexity_{qosScore} = O(|influencers(u, c)|) \quad (21)$$

  Note that the number of influencers per category is bounded, with the value of 8 influencers being a prominent upper bound (c.f. subsection 5.1).

- In order to compute the CF score for a single user, the influencers' histories within the item's category are examined, extracting from each history the item with the highest similarity with the item being processed. Since item similarities are precomputed, the complexity of this procedure for a naïve implementation that simply iterates over the lists of influencers and their histories is:

$$complexity_{CFScore} = O(|influencers(u, c)| * |history(inf, c)|) \quad (22)$$

  However, the computation of the CF-score involves actually finding within each influencer's history the product with the maximum similarity with the item being processed. This is a typical case of a join-aggregate nested query (JA-type query) (Kim, 1982), where the tuples of the outer table (*influencers*) are matched against the maximum of an outer-tuple dependent aggregate (the maximum similarity item of the respective influencer). This type of query is very common in applications and the database community has designed highly efficient algorithms for processing these queries. Ganski and Wong (1987) report that existing solutions based on transformations followed by merge-joins can introduce savings accounting to the 80% of the naïve implementation (nested iteration). For our purposes, we simply code the query appropriately and delegate the responsibility of the optimization to the underlying database system.

- Since the computation of the QoS-score and the CF-score is performed for all users that are interested in the category of the item that is being processed, the overall complexity of the algorithm is

$$complexity_{itemTrigger} = O(|usersInCategory| * |influencers(u, c)| * |history(inf, c)|) \quad (23)$$

  under the naïve implementation; as noted above, this complexity is considerably reduced by employing the JA-type query evaluation optimizations. The term in equation (21) does not appear in equation (23), since the term in equation (22) dominates the complexity of the algorithm.

ii) Events appearing in information flow (b)

An event appearing in information flow (b) corresponds to item purchases or impression clicks for an item *i* made by a triggering user *tu* of the social network. The particular event may generate information diffusion to those users *u* of the social network for which user *tu* is among their influencers regarding the category *C* that item *i* belongs to. Consequently, initially the set of potential recipients of the information *S(i)* is computed as

$$S(i) = \{U|\mathrm{tu} \in \mathrm{influencers}(\mathrm{U}, \mathrm{C})\} \quad (24)$$

Subsequently, for each user $u \in S(i)$ we compute the information that should be diffused to the particular user. The rationale used here to select the information to be diffused is as follows:

- If the QoS parameters of item *i* are "close" to the QoS attributes that user *u* typically purchases or views within category *C*, then the User's Interest Probability $UIP_{i,u}$ on the item is computed. If the value of $UIP_{i,u}$ meets or exceeds the UIP threshold, then the same impression is forwarded to user *u*.
- If the QoS parameters of item *i* are too distant from the QoS attributes that user *u* typically purchases or views within category C, then the algorithm searches for an item *i*' in category *C* that (a) is highly similar to item *i* and (b) its QoS attributes are close to the QoS attributes that user *u* typically purchases or views within category *C*. For this item *i'*, the metric $UIP_{i',u}$ is computed, and the impression in *i'* is forwarded to user *u* if the value of $UIP_{i,u}$ meets or exceeds the UIP threshold.

In more detail, the computation of the information to be diffused proceeds as follows: initially, the QoS score of item *i* for user *u* is computed, using the formula

$$score_{QoS,i,u} = price\_vicinity(u,i) * reliability\_vicinity(u,i) \quad (25)$$

similarly to the case of events appearing in information flow (a). If $score_{QoS,i,u}$ is greater than 0.68 (a discussion on this value is given in the conclusions section), then the QoS parameters of item *i* are considered to be close to the QoS attributes that user *u* typically purchases or views within category C. In this case, the User's Interest Probability $UIP_{i,u}$ on the item is computed as in the case of events appearing in information flow (a), and if the computed value $UIP_{i,u}$ meets or exceeds the UIP threshold, an impression on item *i* is forwarded to user *u*.

In the case that $score_{QoS,i,u}$ is less than 0.68, the QoS attributes of item *i* are considered to be distant from the QoS attributes that user *u* typically purchases or views within category *C*. In this case, the items of category *C* are searched to find an item *i'* having a high similarity with the triggering item *i* and being close to the QoS attributes that user *u* typically purchases or views within category *C*. The item selected is the one that maximizes the following formula:

$$rating_{i'} = SemSim(i,i') * score_{QoS,i',u} \quad (26)$$

For this item *i'*, the metric $UIP_{i',u}$ is computed, and the impression in *i'* is forwarded to user *u* if the value of $UIP_{i,u}$ meets or exceeds the UIP threshold.

The pseudocode for the algorithm that diffuses information on item purchases and impression views is depicted in Figure 2. Regarding the complexity of this algorithm, for each user that is influenced by the user that performed the action (purchase or click) we have the following cases:

- *Best case – the item's QoS score exceeds the threshold*. In this case, only the QoS score of the item needs to be computed. The complexity of this operation is given by equation 23, above.
- *Worst case – the item's QoS score does not exceed the threshold*. In this case, all items in the category need to be searched, and for each one the QoS score needs to be computed. The worst-case complexity of this operation is

  $$O(|items\_in\_category(c)| * |influencers(u,c)|) \quad (27)$$

  Afterwards, for the item attaining the highest "appropriateness" score, its UIP value is computed, with a complexity of

  $$O(|influencers(u,c)| * |history(inf,c)|) \quad (28)$$

  In equation (29), the factor $|influencers(u,c)| * |history(inf,c)|$ corresponds to the evaluation of a JA-type query (Kim, 1982), hence it is subject to efficient optimizations, as noted above.

  Furthermore, it is possible to limit the items in the category that need to be examined, in order to find the one that scores the highest $rating_{i'}$ value (cf. equation 26). First, the item *i'* with the highest value of $SemSim(i,i')$ is examined (recall that the item similarity

```
diffuse_on_user_trigger(user, item) {
/*
      Input:  An item that has been purchased or viewed by a specific user.
      Output: None
      Effects: Information has been diffused to the members of the social network for which the
               interest probability exceeds the appropriate threshold.
*/
category = item.category;
candidate_users = get_influenced_by(triggering_user, category);

foreach user in candidate_users {
    influencers = get_influencers(triggering_user, category);

    /* Compute QoS score, using the same method as in Figure 1 */
    qos_score = compute_qos_score(user, item);

    if (qos_score>= QoS_THRESHOLD) { /* QoS parameters are "close" to user habits */
        diffuse(user, item);
    }
    else {
        foreach cand_item in category.getItems() { /* Check candidate items in the category */
            /* compute QoS score, as in Figure 1*/
            cand_qos_score[cand_item] = compute_qos_score(user, cand_item);
            appropriateness[cand_item] = cand_qos_score[cand_item] * SemSim(item, cand_item);
        }
        test_item = item_with_max_appropriateness_value;
        /* compute UIP value, as in Figure 1*/
        uip_value = compute_uip_value(user, item);
        if (uip_value >= UIP_THRESHOLD) {
            diffuse(user, item);
        }
    }
}
```

Figure 2. Pseudocode for the algorithm that diffuses information on item purchases and impression views

values have been computed in the offline stage, hence no computation cost is incurred). It is clear that any item *i''* within category *C* having $SemSim(i, i'') < rating_{i'}$ cannot attain a rating higher than $rating_{i'}$ (since $score_{QoS,i',u} \leq 1$), and therefore the search list can be pruned by omitting all items satisfying the above condition. Pruning is repeated whenever an item with higher rating than the former maximum is discovered.

Overall, the worst-case and naïve implementation complexity of the algorithm that diffuses information on item purchases and impression views is

$$O(|influenced\_by(tu)| * (|items\_in\_category(c)| * |influencers(u,c)| + |influencers(u,c)| * |history(inf,c)|)) \quad (29)$$

Step 3 – Repository update. The contents of the social network and product information are dynamic, hence a number of information elements of our model need to be updated to maintain their consistency with the respective sources. The following updates need to be performed:

i) Each time a new item is introduced or has its characteristics modified, a check needs to be made whether the minimum and maximum values of the QoS parameters within that category need to be updated. Additionally, the semantic similarity between the newly introduced product and other products in the database needs to be computed.

ii) After an impression is clicked or a purchase is made by some user, the profile of the user is updated regarding the mean QoS attributes (price and reliability) of items within the clicked/bought item's category.

iii) The weights assigned to the CF dimension ($w_{CF,C(i),u}$) and the QoS-based dimension ($w_{QoS,C(i),u}$) need to be recomputed when the underlying data (items clicked or purchased by a user and impressions suggested by influencers) change.

iv) The top influencers of each user *u* within each category *C* need to be recomputed when the underlying data (number of communications and/or categories of interest) change.

Regarding the above list, updates (i) and (ii) are computationally more efficient accessing only few database items, while updates (iii) and (iv) perform extensive computations on the whole database. Thus, updates (i) and (ii) are performed synchronously with their triggering events, while updates (iii) and (iv) are performed on a periodical basis, e.g. using an 8 day period, to provide a balance between information timeliness and computation overhead minimization.

5 EXPERIMENTAL EVALUATION

In this section, we report on our experiments aiming to:

(a) determine the values of parameters that are used in the algorithm; in particular, this set of experiments is targeted to (i) identify the optimal value for parameter *N*, corresponding to the number of influencers that must be maintained per item category so as to offer useful and effective recommendations, (ii) estimate the number of categories that should be retained within the profile of each user (i.e. the product categories that the user is interested in), in order to assess the memory requirements and the scalability of the approach, and (iii) compute the recommendation score threshold *TH* over which a recommendation is diffused to the user it was generated for.

(b) evaluate the performance of the proposed approach, in terms of (a) execution efficiency (the time needed to make the recommendations) and (b) users' satisfaction regarding the offered recommendations.

For our experiments we used two machines. The first machine was equipped with one 6-core Intel Xeon E5-2620@2.0GHz CPU and 16 GBytes of RAM, which hosted the processes corresponding to the active users (browser emulators). The second machine's configuration was identical to the first, except for the memory which was 64 GBytes. The second machine hosted (i) the algorithm's executable, (ii) a database containing the users' profiles including the influence metrics per category and the lists of top N influencers and the data regarding the purchases made by each user and (iii) the product database, which includes product semantic information, computed product similarity metrics and product QoS data. The machines were connected through an 1 Gbps local area network.

To assess recommendation quality, we conducted a user survey in which 85 people participated. The survey took place in two phases: during the first phase a synthetic merchandise dataset was used, while in the second phase the merchandise dataset was real (details are given below). Each phase lasted for two weeks. The participants were students and staff from the University of Athens community, coming from 4 different academic departments (computer science, medicine, physics and theatre studies). 49 of the participants were women and 36 were men and their ages range between 19 and 46 years old, with a mean of 29. All participants were regular Facebook users, i.e. they use Facebook for two hours per day on average, post comments, upload pictures, engage in chatting activities, and click on items and advertisements of interest and had been members for at least two years. At the end of the experiment, each participant was given a gift card.

Participant recruitment was done in 6 groups, with each group containing from 10 to 18 persons and all members of the group were pairwise friends in Facebook. This setting permitted us to obtain the necessary permissions to access friends' profile information that was important for our algorithm. The relevant data were extracted using the Facebook Graph API (https://developers.facebook.com/docs/graph-api). Regarding the participants' profile and behavior within Facebook, the minimum number of Facebook friends among the participants was 122 and the maximum was 629, with a mean of 271. For each person, we computed the

relevant tie strengths with all of his Facebook friends in an offline fashion. Since page and impression views historical data was not available via the Graph API, we seeded the purchases/impression views history by asking each participant to choose 0-10 products from each category that they either had bought or would buy.

Regarding the merchandise data, two datasets have been used. The first dataset was a synthetic repository generated using the Berlin SPARQL Benchmark (Bizer and Schultz, 2009; Bizer and Schultz, 2010). The dataset included 2,000 items, falling in the categories of clothing (trousers, jeans, and shirts), shoes, mobile phones/smartphones, digital cameras and portable computers. The cost attribute values in this repository were set according to the items' current prices, while the reliability attribute values were uniformly drawn from the domain [0,10]. Four additional attributes were added in each category to be used as input for the semantic distance computation. For clothing and shoes these were brand, fashion style, color and category (e.g. shirt, trousers, loafers, boots etc.); for mobile phones/smartphones the attributes were brand (for phones) or operating system (for smartphones), amount of memory, processor power and camera resolution; for digital cameras the attributes were image resolution, video resolution, type (compact, advanced compact, DSLR) and optical zoom; and for portable computers the attributes were processor type, amount of memory, hard drive capacity and screen size.

The second dataset included 2,000 items falling in the same categories as above, which were retrieved from the Amazon product catalog using the Amazon Product Advertising API (Amazon, 2015). Amazon maintains no reliability attribute, hence the value of the *audience rating* attribute was used instead.

During the first experiment phase (where synthetic data were used), the first information flow (new items or items with modified characteristics) was fed with randomly created new items. The second information flow (clicks on recommended items or item purchases made by influencers) was fed with impression clicks from other subjects parallelly participating in the experiment. In the second experiment phase (where real-world data were used), the first information flow was fed with information from corresponding shops (i.e. garment, shoes and technology shops), both physical "brick & mortar" stores and e-shops. The second information flow was again fed with impression clicks from other users parallelly participating in the experiment.

In both experiment phases, only minimal differences were noted in the results, indicating that the algorithm has uniform performance on both synthetic and non-synthetic data. In the following subsections, we present and analyze the results, commenting on the cases that differences were noted in the two experiments.

5.1 Determining the number of influencers

The first experiment is aimed at determining the number of influencers N that must be maintained per user and per item category, in order to produce accurate recommendations. Recall that for each product category, we seek to take into account the opinion of the strongest influencers within the specific category when generating recommendations. Therefore, in this experiment we vary the number of strongest influencers considered, seeking the point at which adding more influencers does not alter the recommendations generated. The recommendations are expected to converge when the number of influencers considered increases: since strongest influencers are added first to the influencers set, considering larger influencer sets will contribute with influencers having gradually weaker levels of influence. Therefore, beyond some point newly incorporated influencers will not be capable of altering the recommendation outcome.

To find the value of N after which recommendations converge, we selected 1,000 *(user, category)* pairs to generate recommendations for. Subsequently, we generated the relevant recommendations for different values of N varying from 1 to 30, and finally we calculated the probability that a recommendation generated for a user considering her n strongest influencers (*recom@n*) is different than the corresponding recommendation generated considering her $n+1$ strongest influencers (*recom@n+1*). The results are shown in Figure 3 and demonstrate that the

number of strongest influencers considered can be fixed to 8; considering higher numbers of strongest influencers will only marginally modify the recommendations generated (less than 1% of the recommendations are modified). Considering these results, in the subsequent experiments we will consider only the 8 strongest influencers per category for each user.

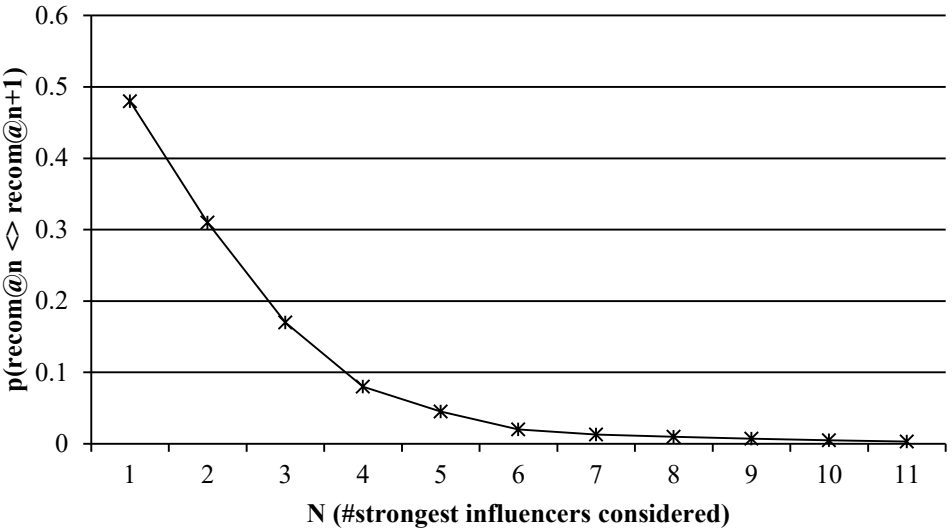

Figure 3. Different recommendations made, due to the fact of considering 1 more recommender

The results shown in Figure 3 have been found to be fairly independent of the number of items within a category, provided that the category has more than 60 products, which is expected to be the typical case. For categories with less than 60 products, less influencers (6-7) are adequate for generating accurate recommendations. However, the savings accomplished by such a fine tuning are minimal, hence the number of influencers is set to 8, regardless of the number of items present in the category.

5.2 Estimating the number of categories of interest per user

To estimate the number of categories of interest per user, the advertisement settings of the participants' Google profiles were analyzed. Google maintains a list of interests per signed in user (Google Inc., 2015b), and the lists related to the experiment participants were retrieved and analyzed. The number of interests of users ranged from 15 to 56, with an average number of 28. Therefore, in combination to the results drawn from subsection 5.1, each user profile should on average accommodate 28 item categories and 224 recommenders (8 per category). Since only user ids of recommenders are retained, the overall increase of the user's profile size due to the maintenance of per-category influencers is less than 2 KBytes (assuming 64-bit ids), therefore the storage overhead can be handled by the current technology.

5.3 Determining the recommendation score threshold for information diffusion

Recommendations that are generated by the proposed system are tagged with a score which we call *User's Interest Probability* (UIP – c.f. section 4), which reflects the estimated probability that the user will be interested in the recommendation. Through the user study described above, we estimated the probability that the user will actually click on the recommendation in relation to the UIP metric. The results of this experiment are illustrated in Figure 4 and show that the probability that a user clicks a recommended impression is a monotonically increasing function of the interest probability metric. Hence, in order to provide to the users the impressions that are most likely to be clicked, the impressions with the highest UIP values should be selected. To this end, we need to determine a threshold for the UIP value. The recommendations having a UIP value matching or exceeding the threshold will be diffused to their target users; contrary, recommendations having a UIP value less than the threshold will be dropped and not be diffused. In Figure 4 we can observe that the probability of impression clicking in the real

dataset is slightly higher, owing to brand and product recognizability (impressions of recognizable brands' products have a higher probability to be clicked).

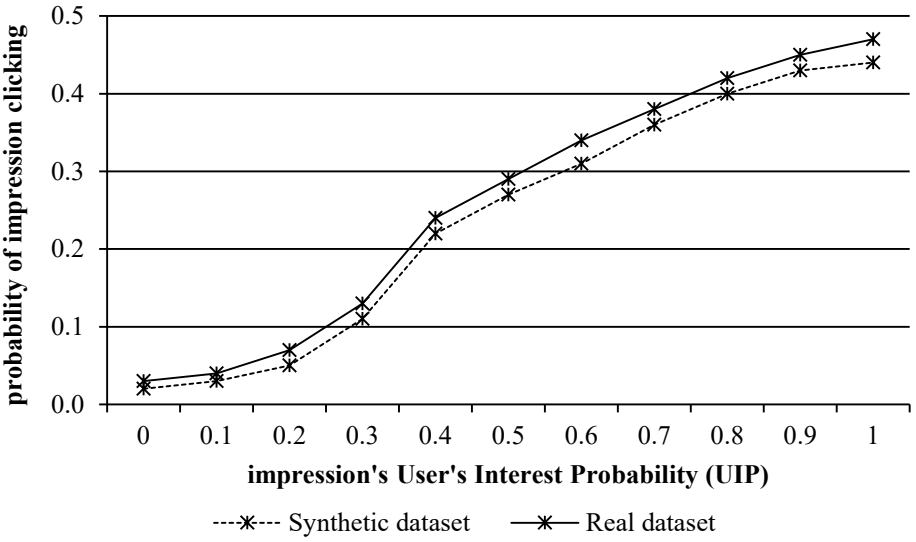

Figure 4. Impression clicking probability in relation to the user's interest probability (UIP)

The optimal value for the UIP threshold will be the one that best serves the two following contradicting goals in combination:

a) affords the maximum probability that a displayed impression is clicked on, i.e. maximizes the quantity

$$Prec = \frac{|\{impressions\_of\_real\_interest\} \cap \{displayed\_impressions\}|}{|\{displayed\_impressions\}|}$$

The set *impressions_of_real_interest* corresponds to the set of impressions that *would be clicked by the users* and may contain both impressions that have been displayed (passed the UIP threshold criterion) and impressions that have been dropped (not having met the UIP threshold criterion). The *Prec* metric is analogous to the *precision* metric used in information retrieval (Manning et al., 2008) and is best served by high UIP values.

b) maximizes the probability that a banner that *would be clicked* is actually shown (and not rejected due to not meeting the UIP threshold criterion), i.e. maximizing the quantity

$$Rec = \frac{|\{impressions\_of\_real\_interest\} \cap \{displayed\_impresions\}|}{|\{impressions\_of\_real\_interest\}|}$$

The *Rec* metric is analogous to the *recall* metric used in information retrieval (Manning et al., 2008) and is best served by low UIP values.

Figure 5 illustrates the quantities *Prec* and *Rec* in relation to the UIP recommendation threshold. Note that the *Prec* measure plotted in Figure 5 is not the same as the *probability of banner clicking* plotted in Figure 4: Figure 4 depicts the individual impression's probability that it is clicked in relation to its User Interest Probability metric. Figure 5, on the other hand, illustrates the conditional probability that any impression is clicked, provided that it has been displayed, i.e. given that it surpasses the value of the UIP threshold. Again, small differences were observed among the synthetic and the real dataset, owing to brand recognizability.

The two metrics, *Prec* and *Rec* can be combined into a single effectiveness metric *EM* by computing their harmonic mean, i.e. $EM = 2 * \frac{Prec*Rec}{Prec+Rec}$, analogously to the combination of the precision and recall metrics in information retrieval to produce the *F-measure* (Manning et al.,

2008). Figure 6 illustrates the harmonic mean of the *Prec* and *Rec* metrics, in relation to the UIP recommendation threshold. In this figure we can notice that *EM* attains its maximum value for UIP threshold=0.4 for both datasets (0.491 for the synthetic dataset and 0.507 for the real dataset). Hence, in the subsequent experiments we will set the parameter UIP threshold to 0.4.

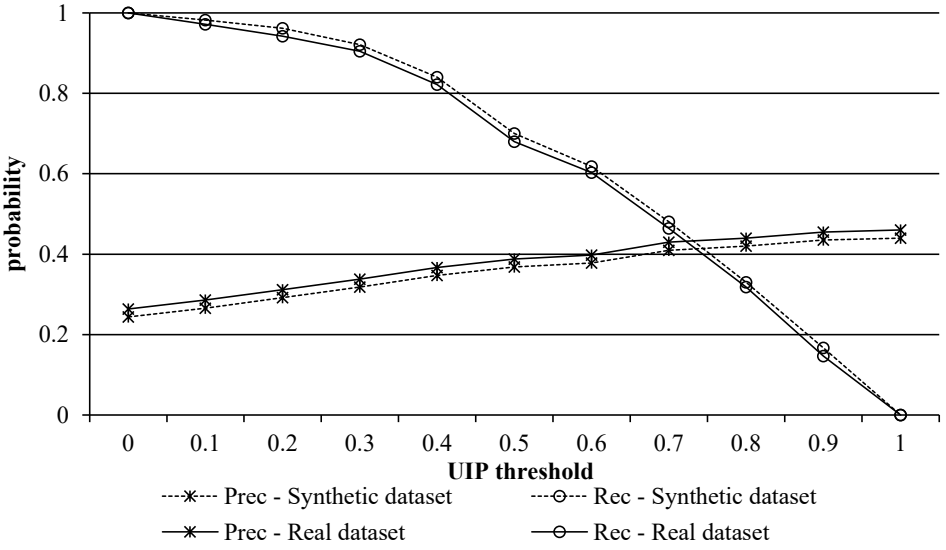

Figure 5. Displayed impressions click probability and probability of displaying impressions of interest, in relation to the UIP recommendation threshold

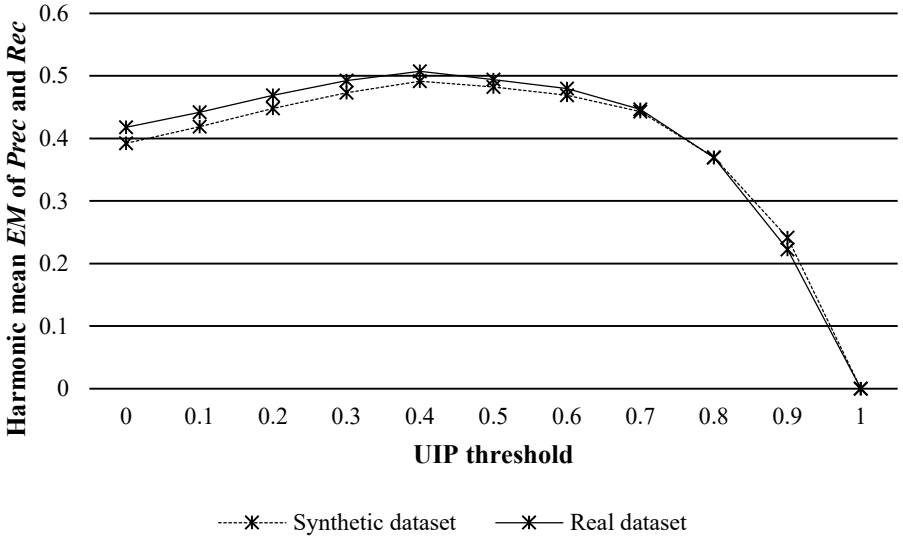

Figure 6. The harmonic mean *EM* of the *Prec* and *Rec* metrics, in relation to the UIP recommendation threshold

5.4 Recommendation formulation and diffusion time

The next experiment is aimed at quantifying the time needed to formulate the appropriate recommendations and diffusing the relevant information to the appropriate users. For this experiment we have considered three network sizes (1,000 users, 10,000 users and 50,000 users) and different numbers of concurrent events to be processed. Data regarding users and information flows were synthetically generated. In the data set, each user had from 100 to 1000 friends, with an average of 190 friends, following the mean value of friends in the Facebook social network (https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859). The average number of product categories in which users were interested in was 28, as estimated in the experiment presented in subsection 5.2. Each user

was set to have a history of 1-20 impression clicks and purchases within the last 8 days, with a mean of 8 items. All repositories (the items' semantic repository, the items' qualitative repository, the users' top recommenders and each user's purchases) were implemented as in-memory hash-based structures, which proved more efficient than using a separate (memory or disk-based) database, such as HSQLDB (HSQLDB, 2015) (memory-based) or MySQL (MySQL, 2015) (disk-based).

The results of this experiment are depicted in Figure 7. We can observe that for small-sized networks (1,000 users) the performance is very good (recommendation and diffusion time is in the 100 msec range). However the required time significantly increases with the size of the network; this is particularly true for the case of the large-sized network and a concurrency level of 3. In the latter case, a sharp increase appears, which is owing to the depletion of the second workstation's resources at this load level. Clearly, having available more execution units in the machine performing the recommendation, or offloading some processing to other machines, would result to smaller overheads. It is worth noting that the algorithm presented in section 4 is clearly parallelizable, by partitioning the set of users to be examined in sets and assigning each subset to a different processor/machine. Hence, in a real-world social network setting, where the hosting infrastructure is server farm-based (e.g. the Facebook infrastructure was estimated to over 60,000 servers in June 2010 (Data Center Knowledge, 2012)), the overall recommendation time is expected to significantly drop.

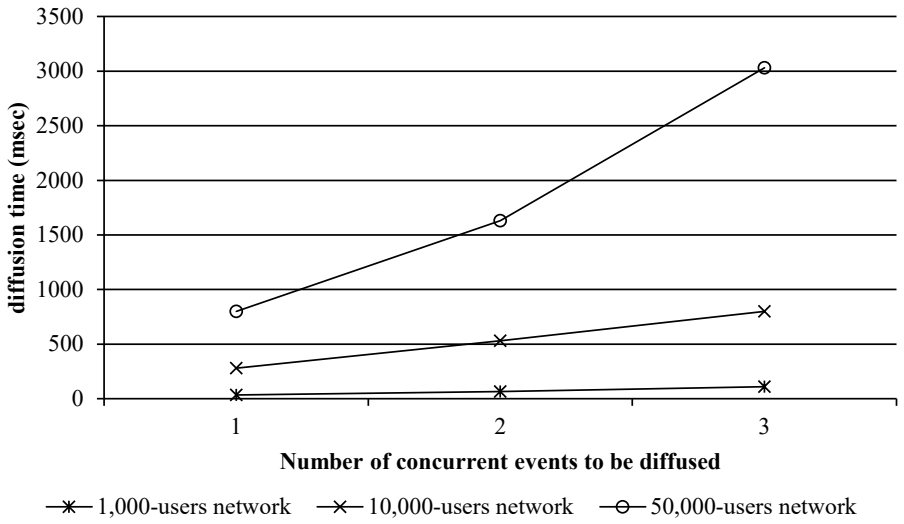

Figure 7. Recommendation formulation and diffusion time for varying degrees of concurrency and number of users in the social network

5.5 User satisfaction

Figure 8 depicts the participants' satisfaction from the recommended items, on a scale of 1 (totally unsatisfactory) to 10 (totally satisfactory), for different recommendation techniques. The participants rated the perceived recommendation usefulness for a number of impressions, ranging from 40 to 80. In this experiment, we considered the following recommendation formulation techniques:

a) the proposed algorithm with a UIP threshold equal to 0.4 (which is the optimal value, calculated in subsection 5.3),

b) the proposed algorithm with a UIP threshold equal to 0.3; this setting was tested to allow more impressions to reach end users,

c) a plain CF algorithm (the algorithm in section 4 taking into account only the cumulative influence and not considering the QoS dimension),

d) a plain QoS-based algorithm (the algorithm in section 4 without the CF dimension) and

e) the proposed algorithm, without considering per-category influencers for each user, and using a single set of influencers for all categories.

On average (last column on Figure 8) it is clear that the proposed algorithm using the optimal UIP threshold value outperforms the other algorithms, attaining an overall user satisfaction of 6.8. The runner up is the same algorithm with a UIP threshold value equal to 0.3 which scored an overall user satisfaction of 6.0 (or the 88% of the performance of the same algorithm with the optimal threshold). The plain CF algorithm was ranked 3rd, the proposed algorithm modified to use a single set of influencers was ranked 4th and the plain QoS-based one was ranked 5th, with their performance being at the 75%, 71% and 60%, respectively, of the proposed algorithm with an optimal threshold, respectively. The same results, complemented with algorithm ranking statistics and standard deviation metrics, are depicted in table 2.

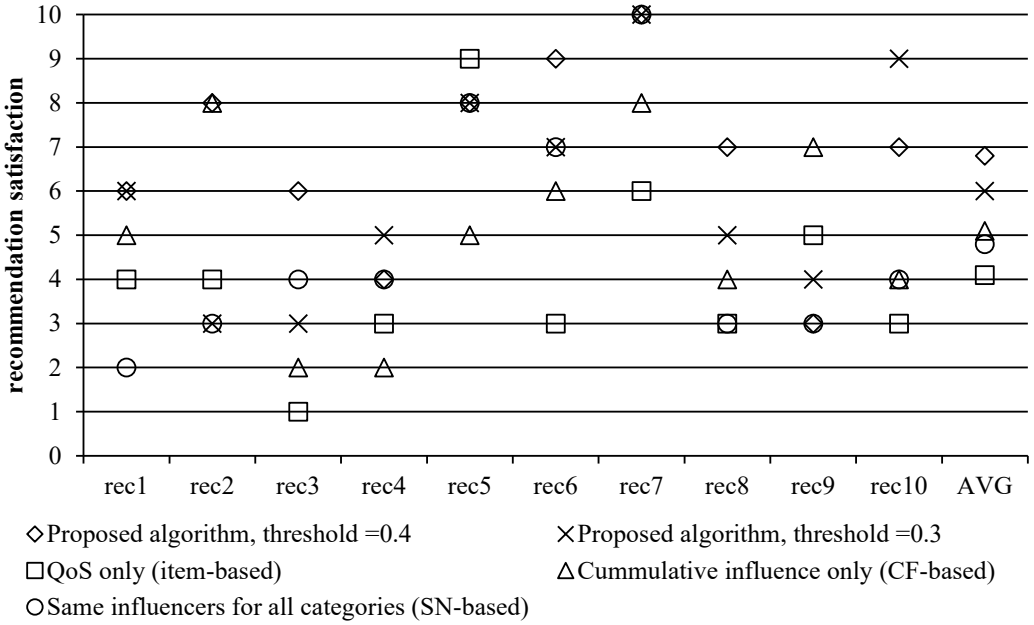

Figure 8. Users' satisfaction of recommendations made using different recommendation algorithms

Within Figure 8 we have also included user ratings for 10 individual recommendations; these have been chosen to demonstrate that algorithm performance is not uniform across all cases. In most occasions, the proposed algorithm with the optimal UIP threshold value produces the most favorably ranked recommendation, even at a tie with another algorithm. However, in other cases the recommendation of the proposed algorithm with the optimal UIP threshold value has been ranked in lower positions. It has to be noted however that in more than 80% of the cases, the recommendation of the proposed algorithm with an optimal UIP threshold was ranked first or second (first: 58%, second: 27%; third: 12%; fourth: 3%; fifth: none). Figure 9 details on the ranking frequency of the recommendations of the different algorithms. Further investigation of the cases where the recommendation of the proposed algorithm received a poor ranking (inferior to the rankings of other algorithms by more than one mark (24.3% of the total number of cases) or lower than 50% (16.7% of the total number of cases)) is part of our future work. For the latter case (i.e. recommendation rating lower than 50%), analysis of the results have shown that 43% of the cases where such ratings were given correspond to recommendations that had low price vicinity to the buying habits of the user. This reaffirms that price vicinity is an important factor and suggests that a separate threshold could be employed especially for price vicinity, complementary to the threshold value of 0.68 used for the $score_{QoS,i,u}$ metric. Further investigation of these issues is part of our future work.

In all cases, the distribution of users' evaluations for each recommendation algorithm followed a Gaussian distribution. The standard deviation for each algorithm is shown in the last row in table 2 (labeled as σ). From the standard deviation metrics, we can observe that the QoS-only

algorithm received the most uniform rankings (which were not favorable). Please note that the standard deviation shown in table 2 is calculated from all user responses, while the data rows of table 2 (rec_i rows) include user ratings for 10 individual recommendations, which have been specifically chosen to demonstrate that algorithm performance is not uniform across all cases, and are therefore not representative. The highest rating variance was recorded for the variant of the algorithm having a UIP threshold equal to 0.3.

Table 2. Users' satisfaction of recommendations and ranking comparisons

Scenario	Proposed algorithm, threshold =0.4	QoS only	Cumu-lative in-fluence only	Proposed algorithm, threshold =0.3	Same in-fluencers for all categories	Ranking of the proposed algorithm (threshold=0.4)	Score of proposed algorithm as % of the top-ranked algorithm
rec1	6	4	5	6	2	1	100%
rec2	8	4	8	3	3	1	100%
rec3	6	1	2	3	4	1	100%
rec4	4	3	2	5	4	2	80%
rec5	8	9	5	8	8	2	89%
rec6	9	3	6	7	7	1	100%
rec7	10	6	8	10	10	1	100%
rec8	7	3	4	5	3	1	100%
rec9	3	5	7	4	3	4	43%
rec10	7	3	4	9	4	2	78%
AVG	6.8	4.1	5.1	6	4.8	-	89%
σ	1.54	1.23	1.32	1.61	1.38	-	0.17

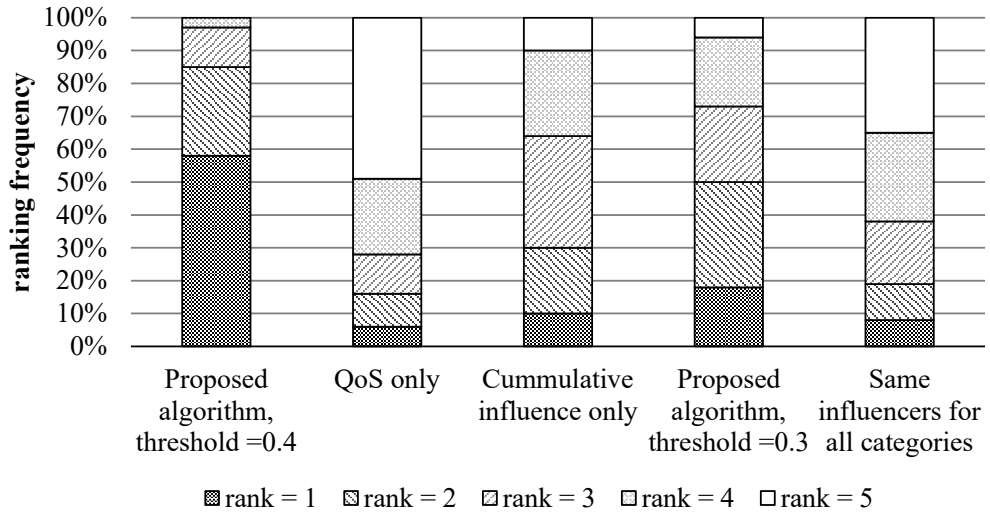

Figure 9. Ranking frequency of algorithms

The results of the user study were initially analyzed for statistical significance using the Kruskal-Wallis test (Kruskal & Wallis, 1952; Corder & Foreman, 2014). The parameters and results of this analysis are shown in table 3. The results illustrated in table 3 show that the null hypothesis is rejected, concluding that the differences in recommendation algorithm performance are statistically significant. Based on this result, we proceeded to applying the non-parametric Mann-Whitney U test (Corder & Foreman, 2014), examining whether the differences observed in the results between (a) the proposed algorithm with threshold = 0.4 and

(b) each one of the other algorithms are statistically significant. The results of these analyses are depicted in Table 4. Note that in the Mann-Whitney U test, the p-value reflects the probability that the two sample sets (algorithm evaluations in our case) come from populations with equal medians.

Table 3. Parameters and results of the Kruskal-Wallis test

Number of samples:	85
Number of populations:	5
Degrees of freedom:	4
Significance level α:	0.05
Null hypothesis H_0:	The samples come from populations with equal medians
Alternative hypothesis H_a:	The samples come from populations with medians that are not all equal
Rejection region R for Chi-Square test:	$x^2>9.488$
Result x^2	15.302
Result p-value	0.0041

Table 4. Results of the Mann-Whitney U-tests

Algorithms tested	p-value (two-sided)	Statistically significant at α=0.05?
Proposed algorithm, threshold =0.4 vs. QoS only	0.0007	Yes
Proposed algorithm, threshold =0.4 vs. Plain CF (Cumulative influence only)	0.02382	Yes
Proposed algorithm, threshold =0.4 vs. Proposed algorithm, threshold =0.3	0.26272	No
Proposed algorithm, threshold =0.4 vs. Same influencers for all categories	0.01552	Yes

These results establish with a confidence level of 95% that the proposed algorithm outperforms (a) the QoS only algorithm (b) the cumulative influence only algorithm and (c) the same influencers for all categories algorithm. On the other hand, statistical significance was not established in the comparison between the two variants of the proposed algorithm with different thresholds (0.4 and 0.3).

Considering the comparison with the QoS-only algorithm, the p-value is equal to 0.0007, establishing statistical significance at a level of 99%. The reason behind this performance difference is that the proposed algorithm considers the items that have been purchased by each user's influencers. This feature capitalizes on the trust relationships that exist between the user and her influencers (He and Chu, 2010; Shuiguang et al., 2014), formulating recommendations that have increased interest to the user (Bakshy et al., 2012a); this feature is also in-line with

the aspect of *homophily* (Anagnostopoulos et al., 2008), which includes performing similar actions to one's social neighborhood.

Regarding the comparison to the plain CF algorithm, the p-value is equal to 0.02382, establishing statistical significance at a level of 95%. The performance edge of the proposed algorithm is attributed to its ability to adapt its recommendations to the QoS habits of each individual user. Indeed, during the experiments many subjects were noticed to ignore numerous recommendations of the plain CF algorithm, even recommendations that were based on activities of highly-ranked influencers. When the subjects were asked why they did not select these recommendations, the most frequent responses were that "this is too expensive/too cheap" (38% of the answers) and "I would prefer something more reliable"/"I do not need this level of reliability" (23% of the answers). Note that the reply "I do not need this level of reliability" is again mostly associated with the price, since the particular items had prices that exceeded those recorded in the subjects' buying habits within the respective categories.

Considering the comparison with the variant using the same influencers for all categories, the p-value is equal to 0.01552, establishing statistical significance at a level of 95%. The superior performance of the proposed algorithm is due to the fact that it takes into account that a user may be influenced by one set of people regarding product category X and a distinct set of people regarding product category Y. Elaborate methods towards identifying influencers in different product categories are reported in recent research, e.g. (Liu et al., 2015). Incorporating such methods in the proposed algorithm is part of our future research.

Finally, considering the comparison with the variant that uses a different UIP threshold (0.3), the p-value is equal to 0.26272, not establishing statistical significance of the observed differences. This was expected to some extent, since the two variants differ only in the number of impressions that reach end-users (on average, 0.8 more impressions reach end users when the threshold is lowered to 0.3). Still, the variant using the threshold value 0.4 is ranked more favorably by users in the 44% of the cases, as opposed to the 23% that the variant using the threshold value 0.3 is preferred (in the remaining 33% of the cases we have a tie) and has a higher average score (6.8/10 as compared to 6.0/10 of the variant using the threshold value 0.3). Based on these observations, we can conclude that the threshold value of 0.4 appears to be a valid setting, even though statistical significance cannot be established. Further research can be conducted to this end to determine whether preference towards the lower threshold value is associated with certain properties of the user profile. In such a case, the threshold value could be set on a per-user basis, according to the assessment of the user's profile.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have presented an algorithm for fostering information diffusion in social networks through the generation of appropriate recommendations. The proposed algorithm contributes to the state-of-the-art by taking into account qualitative aspects of the recommended items, the influencing factors between social network users, the social network user behavior regarding their purchases in different item categories, and the semantic categorization of the products to be recommended. Furthermore, influencers in this algorithm are considered per category, to allow for formulation of more accurate recommendations and maximize the probability that the impression is clicked. The proposed algorithm has been experimentally validated regarding (i) its performance, and (ii) recommendation accuracy (users' satisfaction to the recommendations produced) and the results are encouraging.

One aspect of our future work will focus on studying the taxonomy levels a recommendation system must store, in order to provide more accurate recommendations. The user interests stored in Google's user's profiles may be in some cases too generic (corresponding to a top-level node in the Google products and services taxonomy) or too specific (corresponding, for instance, to a particular car make). We plan to investigate which level of abstraction is the most appropriate to maintain in the profile and additionally the conditions under which the level can be generalized or specialized.

Another direction of our future work is to conduct a user survey with a higher number of participants and more representative demographics. The current participant set was drawn from the University of Athens community, hence it is not a representative sample of the overall population and the results drawn may not be generalizable. A more comprehensive survey will address this issue and provide us with better insight on the satisfaction and needs of users with different profiles. In this survey, the value of 0.68 used as QoS similarity threshold in events arriving from information flow (b) will be further analyzed. The value of 0.68 used in the experiments conducted in this paper has been derived by asking participants of the experiment to rate whether 100 items were "close" or not to their QoS preferences and then computing the QoS threshold that maximizes the QoS-predictions F1-measure (Lipton et al, 2014) (the QoS-prediction considered here is that an item is considered "close" if its $score_{QoS,i,u}$ value is greater than or equal to the QoS threshold and "not close" if its $score_{QoS,i,u}$value is less than the QoS threshold). The extended survey will allow us to obtain a more comprehensive dataset regarding "closeness" perceptions and further investigate this issue, both regarding the value of the QoS threshold and in terms of whether the QoS threshold is uniform across all categories and/or user profiles. Furthermore, the QoS threshold mechanism leads to a behavior that the user is limited to viewing only information about products similar to those she has viewed or purchased in the past. This however limits the serendipity that may stem from recommendations, which is a desirable feature of recommender systems (Ge et al, 2010). To this end, mechanisms for allowing serendipity in recommendations will be investigated.

In this work, we have used a global UIP threshold value for all users; it is possible however that some users are more receptive to impressions than others, hence different UIP threshold values could be applied. The methods to calculate a personalized UIP value and the effect that this has on the effectiveness of information diffusion are also part of our future work.

We finally plan to elaborate on cases where the recommendation of the proposed algorithm received a poor ranking (inferior to those of other algorithms or lower than 50%), in order to identify the causes and further enhance the quality of the generated recommendations.

7 REFERENCES

Amazon (2015). Amazon Product Advertising API, https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html (Accessed January 17, 2016)

Anagnostopoulos, A., Kumar, R., Mahdian, M. (2008). Influence and correlation in social networks. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08), pp. 7-15.

Androutsos, D., Plataniotis, K. N., Venetsanopoulos, A.N. (1998). Distance measures for color image retrieval. Proceedings of the International Conference on Image Processing (vol.2), 1998, pp. 770 - 774.

Arazy, O., Kumar, N., Shapira, B. (2009). Improving Social Recommender Systems. IT professional, September 2009.

Aslam, J., Montague, M. (2001). Models for metasearch. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR 2001, W.B. Croft, D.J. Harper, D.H. Kraft, J. Zobel (eds), pp. 276-284.

Bakshy, E., Rosenn, I., Marlow, C., Adamic, L. (2012a). The role of social networks in information diffusion.Proceedings of the 21st international conference on World Wide Web, pp. 519-528.

Bakshy, E, Eckles, D., Yan, R., Rosenn I. (2012b). Social Influence in Social Advertising: Evidence from Field Experiments. Proceedings of the 13th ACM Conference on Electronic Commerce, pp. 146-161.

Balabanovic, M., Shoham. Y. (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, vol. 40, issue 3, 1997, pp 66-72.

Bizer, C., Schultz, A. (2009). The Berlin Sparql Benchmark. International. Journal of Semantic Web Information Systems, pp. 1–24.

Bizer, C., Schultz, A. (2010). Berlin SPARQL Benchmark (BSBM) - Dataset Specification. http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/Dataset/index.html (Accessed December 16, 2015)

Boulkrinat, S. Hadjali, A., Mokhtari, A. (2013). Enhancing recommender systems prediction through qualitative preference relations. 11th International Symposium on Programming and Systems (ISPS), pp. 74 – 80.

Bramantoro, A., Krishnaswamy, S., Indrawan, M. (2005). A semantic distance measure for matching web services. Proceeding of the 2005 international conference on Web Information Systems Engineering, 2005, pp 217-226.

Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Sok Kim, Y., Compton, P., and Mahidadia, A. (2011). Collaborative Filtering for People to People Recommendation in Social Networks. AI 2010: Advances in Artificial Intelligence, Volume 6464, pp 476-485.

Car reliability index (2016). http://www.reliabilityindex.com/ (Accessed January 15, 2016)

Cardoso, J. (2002). Quality of Service and Semantic Composition of Workflows. PhD thesis, Univ. of Georgia.

Chedrawy, Z., Abidi, S.S.R. (2009). A Web Recommender System for Recommending, Predicting and Personalizing Music Playlists. Proceedings of Web Information Systems Engineering (WISE 2009), pp 335-342.

Corder, G. W.; Foreman, D. I. (2014). Nonparametric Statistics: A Step-by-Step Approach. Wiley. ISBN 978-1118840313.

Data Center Knowledge (2012). The Facebook Data Center FAQ. http://www.datacenterknowledge.com/the-facebook-data-center-faq/ (Accessed January 8, 2016)

Digicamhelp (2010). Most reliable brand digital cameras. http://www.digicamhelp.com/buying-guide/checklist/most-reliable-brand-digital-cameras/ (Accessed January 15, 2016)

Facebook (2015a). Facebook home page, https://www.facebook.com

Facebook (2015b). Facebook ad targeting, https://www.facebook.com/business/products/ads/ad-targeting (Accessed January 12, 2016)

Fürnkranz, J, Hüllermeier E. (2010). Preference Learning. Springer; 2011 edition (October 13, 2010), ISBN: 3642141242.

Ganski, R.A., Wong, H. (1987). Optimization of Nested SQL Queries Revisited. Proceedings of the 1987 ACM SIGMOD international conference on Management of data, pp. 22-33.

Ge, M., Delgado-Battenfeld, C., Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. Proceedings of the fourth ACM conference on Recommender systems (RecSys '10), pp. 257-260.

Gilbert, E., Karahalios, K. (2009). Predicting tie strength with social media. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09), pp. 211-220.

Google Inc. (2015a). Google products and services taxonomy, https://developers.google.com/adwords/api/docs/appendix/productsservices (Accessed January 11, 2016)

Google Inc. (2015b). Control your Google ad, https://www.google.com/settings/u/0/ads/authenticated (note: user must sign in to Google to view own settings) (Accessed January 17, 2016)

Guille, A., Hacid, H., Favre, C.Zighed, D.A. (2013). Information Diffusion in Online Social Networks:A Survey. SIGMOD Record, vol. 42, no. 2, June 2013 Hau, J., Lee, W., Darlington,

J. (2005). A Semantic Similarity Measure for Semantic Web Services. Proceedings of WWW2005, May 10–14, 2005, Chiba, Japan.

He, J. and Chu, W.W. (2010). A Social Network-Based Recommender System (SNRS). Annals of Information Systems Volume 12, pp 47-74.

He, D., Wu, D. (2008). Toward a Robust Data Fusion for Document Retrieval. IEEE 4th International Conference on Natural Language Processing and Knowledge Engineering - NLP-KE.

Herlocker, JL., Konstan, JA., Terveen, LG., Riedl, JT. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems vol. 22, no 1, January 2004, pp. 5-53.

HSQLDB (2015). http://hsqldb.org/ (Accessed January 25, 2016)

Hwang, CL, Yoon, K. (1981). Multiple Criteria Decision Making. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag.

ITU (1988). Recommendation E.800 Quality of service and dependability vocabulary.

Kim, W. (1982). On Optimizing an SQL-like Nested Query. ACM Transactions on Database Systems, Vol. 7, No. 3, September 1982, Pages 443-469.

Jamali M., and Ester M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. Proceedings of the fourth ACM conference on Recommender systems, RecSys 2010, Barcelona, Spain.

Karaiskos, C. (2013). Enhanced Ontological Searching of Medical Scientific Information. Master's Thesis, University of Manchester.Konstas, I., Stathopoulos, V., and Jose, J.M. (2009). On social networks and collaborative recommendation". Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Boston, USA.

Li, W., Ye, Z., and Jin, Q. (2014). An Integrated Recommendation Approach Based on Influence and Trust in Social Networks. Future Information Technology, Lecture Notes in Electrical Engineering, Volume 309, pp 83-89.

Lipton, Z.C., Elkan, C., Naryanaswamy, B. (2014). Optimal Thresholding of Classifiers to Maximize F1 Measure. Proceedings of ECML PKDD 2014 (part II), pp 225-239.

Liu S., Jianga C., Linb Z., Dinga Y., Duana R., Xu Z. (2015). Identifying effective influencers based on trust for electronic word-of-mouth marketing: A domain-aware approach. Information Sciences Volume 306, June 2015, pp. 34–52

Manning, C.D., Raghavan, P., Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press; 1 edition (July 7, 2008), ISBN: 0521865719

Margaris, D., Georgiadis, P., Vassilakis, C. (2013). Adapting WS-BPEL scenario execution using collaborative filtering techniques. Proceedings of the IEEE 7th RCIS Conference, Paris, France, 2013.

Margaris, D., Georgiadis, P., Vassilakis, C. (2015a) A Collaborative Filtering Algorithm with Clustering for Personalized Web Service Selection in Business Processes. Proceedings of the IEEE 9th RCIS Conference, Athens, Greece.

Margaris, D., Vassilakis, C., Georgiadis P. (2015b). An integrated framework for adapting WS-BPEL scenario execution using QoS and collaborative filtering techniques. Science of Computer Programming, vol. 98, pp. 707–734.

Masroor, A. (2015). Is Social Media the Biggest Influencer of Buying Decisions? http://www.socialmediatoday.com/marketing/masroor/2015-05-28/social-media-biggest-influencer-buying-decisions (Accessed January 16, 2016).

MySQL (2015). http://www.mysql.com/ (Accessed December 14, 2015)

O'Sullivan, J., Edmond, D., Ter Hofstede, A., (2002). What is a Service? Towards Accurate Description of Non-Functional Properties .Distributed and Parallel Databases, vol. 12.

Oechslein, O., Hess. T. (2014). The Value of a Recommendation: The Role of Social Ties in Social Recommender Systems. 47th Hawaii International Conference on System Science, 2014, pp. 1864-1873.

Olenski, S. Are Brands Wielding More Influence In Social Media Than We Thought? http://www.forbes.com/sites/marketshare/2012/05/07/are-brands-wielding-more-influence-in-social-media-than-we-thought (Accessed January 16, 2016).

Paolucci, M., Kawamura, T., Payne, T., Sycara, T. (2002). Semantic Matching of Web Services Capabilities, Proceedings of the 2002 International Semantic Web Conference, 333-347.

Pirasteh, P., Jung, J.J. Hwang, D. (2014). Item-Based Collaborative Filtering with Attribute Correlation: A Case Study on Movie Recommendation. 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014, Proceedings, Part II, pp. 245-252.

Quijano-Sanchez, L., Recio-Garcia, J.A., Diaz-Agudo, B. (2011). Group recommendation methods for social network environments. 3rd Workshop on Recommender Systems and the Social Web within the 5th ACM International Conference on Recommender Systems (RecSys'11).

Schafer, JB., Frankowski, D., Herlocker, J., Sen, S. (2007). Collaborative Filtering Recommender Systems. In "The Adaptive Web", Lecture Notes in Computer Science Volume 4321, 2007, pp 291-324.

Schedl, M., Knees, P., Pohle, T., Widmer, G. (2008). Towards an automatically generated music information system via web content mining. Proceedings of the 30th European Conference on Information Retrieval (ECIR'08), Glasgow, Scotland, pp 585-590.

Shuiguang, D., Longtao H., and Xu, G. (2014). Social Network-Based Service Recommendation with Trust Enhancement. Expert Systems with Applications. 41(18): pp 8075-8084, 2014.

Sprout Social (2011). Social Networks Influence 74% of Consumers' Buying Decisions. http://sproutsocial.com/insights/social-networks-influence-buying-decisions/ (Accessed January 16, 2016)

Squaretrade (2009). Laptop & Netbook reliability. https://www.squaretrade.com/htm/pdf/SquareTrade_laptop_reliability_1109.pdf (Accessed January 15, 2016)

Twitter (2015).Twitter home page, https://twitter.com/ (Accessed December 15, 2015)

Ver, Steeg, G., Galstyan A. (2012). Information Transfer in Social Media. Proceedings of WWW 2012, April 16–20, 2012, Lyon, France, pp. 509-518.

Walter, F. (2011). Trust as the basis of coalition formation in electronic marketplaces. Advances in Complex Systems vol. 14, no 02, pp. 111-131

Walter, F., Battiston S., Yildirim, M., Schweitzer, F. (2012). Moving recommender systems from on-line commerce to retail stores. Information Systems and e-Business Management, Volume 10, Number 3, pp. 367-393.

Squaretrade (2010). SmartPhone reliability. http://www.squaretrade.com/cell-phone-comparison-study-nov-10 (Accessed January 15, 2016)

Wang, F-K., Huang, C-I, Chu, T-P. Reliability Analysis of Smartphones Based on the Field Return Data. Proceedings of the Institute of Industrial Engineers Asian Conference 2013, pp 1495-1502.

Wang, Z., Liao, J., Cao, Q., Qi, H. and Wang, Z. (2015). Friendbook: A Semantic-based Friend Recommendation System for Social Networks. IEEE Transactions on Mobile Computing, Volume 14, Issue 3, 2015, pp 538 – 551.

Whitman, B., Lawrence, S. (2002). Inferring descriptions and similarity for music from community metadata. Proceedings of the 2002 International Computer Music Conference, (Goteborg, Sweden), pp. 591-598.

Yu, J., Sheng, Q., Han, J., Wu, Y., Liu, C. (2012). A semantically enhanced service repository for user-centric service discovery and management. Data & Knowledge Engineering, vol. 72, Feb 2012, pp. 202-218.

Zhang, W., Chen, T., Wang, J., Yu, Y. (2013). Optimizing top-n collaborative filtering via dynamic negative item sampling. Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR ’13), pp. 785-788.