Time and content aware implicit social influence estimation to enhance trust-based recommender systems.

Armel Jacques Nzekon Nzeko'o, Adamou Hamza, Thomas Messi Nguele, Bleriot Pagnaul Betndam Tchamba

Time and content aware implicit social influence estimation to enhance trust-based recommender systems

Armel NZEKON[*1,3], **Adamou HAMZA**[*1,3], **Thomas MESSI**[*1,2,3], **Bleriot BETNDAM**[1]

[1]University of Yaounde I, FS, Computer Science Department, Cameroon
[2]University of Ebolowa, HITLC, Computer Engineering Department, Cameroon
[3]Sorbonne Université, IRD, UMI 209 UMMISCO, F-93143, Bondy, France

*E-mail : {armel.nzekon, adamou.hamza, thomas.messi, pagnaul.betndam}@facsciences-uy1.cm

Abstract

Nowadays, e-commerce sites such as Amazon, streaming platforms such as Netflix and YouTube, and social networks such as Facebook and Instagram play an important role in our daily lives. However, with the ever-increasing addition of items (products on Amazon, videos on Netflix and YouTube, posts on Facebook and Instagram) on these platforms, it is becoming difficult for users to manually select the items that interest them, hence the implementation of recommender systems whose main objective is to provide a list of items from among millions that best match users' preferences. To improve the performance of these recommender systems, some works in the literature incorporate explicit or implicit trust between platform users through trust-based recommender systems. Indeed, a large number of studies are based on explicit trust relationships entered into the platform by users. However, this information is not available on most digital platforms, as on one platform out of ten, users are willing and even able to provide such information. What's more, the studies that incorporate implicit trust do not take account of time, let alone the categories of items to be recommended, in the process of estimating implicit trust. And yet, the fact that a user u_1 replicates the behaviour of another user u_2 over time for all or certain categories of items is proof of the influence of u_2 on u_1. In this thesis, we are working on estimating implicit social influences extracted from the history of users' actions on items, which are a much more frequent source of information on digital platforms. In addition, we propose to take into account the time and categories of items in the process of estimating implicit social influences, which was not done in previous work in the state of the art. Once the new implicit temporal and categorical social influences have been estimated, we integrate them into graph, K-nearest neighbour (KNN) and matrix factorisation recommender systems, following the principles of trust-based recommendation systems. Experiments carried out on the Epinions and Ciao datasets show that taking account of the temporal aspect and item categories in the estimation of implicit social influences contributes significantly to improving the performance of these recommender systems.

Keywords

Time aware implicit social influence; Item category aware implicit social influence; Implicit trust; Trust-based recommender systems

I INTRODUCTION

For the past decade, online platforms have been offering a wide range of items (products, films, social network posts) to their users. For example, on an e-commerce platform like Amazon, products are sold to users; on streaming platforms like Netflix and Youtube, films and series are made available to users; on social networks like Facebook and Instagram, users consult publications.

However, with the ever-increasing number of items made available to users, it is becoming increasingly difficult for a given user to manually browse through all the products on the platform in order to select those that interest him or her. This manual search for products is time-consuming for the user, and can lead to a loss of interest in the platform, resulting in a loss of revenue for the platform's managers. It was with a view to alleviating these problems that recommenders systems were set up. These suggest products that are most likely to be of interest to the user [13].

Recommender systems are based on several approaches to filtering information, notably collaborative filtering, which assumes that users who have had the same preferences in the past will have the same preferences in the future [4]. We also have content-based recommendation systems, which assume that a user who has liked products in the past will like other products in the same category in the future [3]. Finally, we have hybrid recommender systems, which are in fact a combination of the two types of approach mentioned above [2].

The recommenders systems approaches cited above all have the limitation of not taking into account additional information relating to friendship and trust relationships between users. And yet, humans tend to replicate the actions of those they trust. It is to integrate such consideration that trust-based recommender systems have been proposed in the hope of improving the quality of recommendations.

Trust-based recommender systems incorporate explicit or implicit trust between platform users [19, 20]. Indeed, a large number of works are based on explicit trust relationships provided by users in the platform [10, 16], which is not reassuring because this information is not available in most digital platforms. This unavailability may be attributed to a variety of factors, including limited direct opinions, time constraints, privacy concerns, and benefits deemed insufficient for disclosure of this information. Furthermore, work that integrates implicit confidence does not take into account time and even less the categories of items to recommend in the process of estimating implicit confidence.

However, the fact that a user u_1 replicates the behavior of another user u_2 over time for all or for certain categories of items, is proof of the influence of u_2 on u_1. In other words, not considering time in the estimation of implicit trust assumes that the influence of one user on another does not change over time, but in reality this is not the case. Furthermore, not taking item categories into account in the estimation of implicit trust assumes that there is no variation in the influence of one user on another depending on the categories of items, but in reality, the influence of user u_2 on user u_1 can be very large for one category of items, and very small for another category. Faced with these shortcomings, we wonder how it would be possible to take time and categories of items into account in the process of estimating implicit confidence.

In this paper, we work on the estimation of implicit social influences extracted from the history of user actions on items, which are a much more frequent source of information in digital platforms. Furthermore, we aim to take into account time and item categories in the process

of estimating implicit social influences, which was not done in previous state-of-the-art work. Once the new implicit social influences have been estimated, we need to integrate them into trust-based recommender systems in order to evaluate their impact. This impact includes improved recommendation relevance, increased user satisfaction, and reduced recommendation errors.

The rest of the paper is structured as follows: in section II, a state of the art is presented, focusing on trust-based recommender systems. Section III outlines the strategies deployed to integrate the time factor and item categories into the process of estimating implicit social influences. The experiments and results obtained are detailed in section IV, and finally, this work is concluded with perspectives evoked in section V.

II TRUST-BASED RECOMMENDER SYSTEMS

In this section, we present existing work on recommender systems that incorporate information relating to trust and friendly relations between users, more specifically known as "trust-based recommender systems". To carry out this task, we start by presenting in section 2.1 the generalities on recommender systems, through the different approaches to information filtering namely collaborative filtering, content-based filtering and hybrid filtering. In section 2.2, we present recommender systems based on explicit trust, through the computation of trust and the integration of this computation in classical recommender systems. Subsequently, we present recommender systems based on implicit trust in section 2.3, focusing exclusively on how to estimate this implicit trust. Finally, we close the section by outlining some of the limitations of existing work.

2.1 General information on recommender systems

Recommender systems are software tools and techniques that provide suggestions for items that may be useful to a user. Suggestions relate to various decision-making processes, such as items to buy, music to listen to or online news to read [14]. As stated here [6], "the recommendation system helps to cope with information overload and provide personalized recommendations, content and services". We can also define a recommender system as a set of information filtering techniques that predicts the rank or preference a user assigns to an item among a set of similar items (films, music, books, news, images, web pages, etc.) that are likely to be of interest to him or her.

Recommender systems are based on several approaches to information filtering, including collaborative filtering, which assumes that users who have had the same preferences in the past will have the same preferences in the future, content-based filtering, which assumes that a user who has liked items in the past will like other items in the same category in the future, and hybrid filtering, which combines the two types of approach mentioned above.

The predominant collaborative filtering approach remains central to the literature on recommender systems. However, its limitations, notably the cold-start problem when a new user or product is introduced to the platform, as well as the challenges associated with sparse data in the user-item matrix (the matrix of ratings that users assign to items), can hinder the optimization of recommendation performance. Alongside user-item data matrices, information about users' social relationships is often available, and sometimes data clearly indicates which users trust the system, as observed on platforms such as Epinions and Ciao. These observations have prompted research into recommendation systems based on explicit trust. This work aims to solve some

of the problems inherent in collaborative filtering while exploiting information linked to social relations and trust.

2.2 Recommender systems based on explicit trust

Explicit trust occurs when trust information between users is available and provided by the users themselves. For example, a user u_1 may openly declare that he trusts another user u_2. To give an overview of this work, this section is divided into two main parts: the first is dedicated to models of trust and the second describes the integration of trust in recommender systems.

2.2.1 Trust models

In this subsection, we provide a brief overview of trust and the techniques that can be can be used to measure trust between users of social media in general and recommendation systems in particular. This information on trust will serve as a basis for improving conventional recommender systems. We begin this section by defining trust and its properties. The section continues with a description of global and local trust. It concludes with a description of how trust values are calculated, based solely on explicit trust informations provided by users.

Definition and properties of trust. **Trust**, in the social context, generally refers to the fact that one person has faith in the words and actions of another. It extends to relationships within social groups such as family, friends, a community or an organization [1] . In the field of recommender systems, trust is defined in terms of a user's ability to provide relevant recommendations to another user, as specified by Guo et al [18]. It should be noted that, in this study, only trust between users within the same system is considered, excluding other forms of trust such as a user's trust in a community.

Trust can be measured in two main ways: binary or continuous. Binary trust simplifies the expression by limiting it to two possible states: one user trusts another, and the other does not. Platforms such as Amazon and eBay illustrate this approach using binary trust values (0 and 1). Continuous trust, on the other hand, offers greater finesse by assigning real numbers to represent the degrees of trust relationships.

With regard to the properties of trust, we can cite transitivity, asymmetry, context dependence and personalization [7, 12, 18]. These properties are extracted from the definition of trust between people and form the basis for the creation of trust measurement algorithms. Transitivity enables explicit trust to be propagated along the paths of the trust network to reach other users. Trust is asymmetrical, as it is a subjective and personal relationship between users. It is also context-dependent, as trusting someone on one subject does not guarantee trust on other subjects. For example, a user who is trustworthy in technology is not necessarily trustworthy in gastronomy. Finally, trust is personalized, as the degree of trust one person has in another can vary from person to person. This property is used to define and formulate local trust.

Global and local trust measures. Trust is defined as a relationship between two individuals, where the weights associated with trust reflect the degree of credulity one individual accords to another. Trust metrics facilitate the calculation of trust weights between network users. This view situates trust as a local attribute. Beyond this local view, it is possible to conceive of trust as a global measure. In this way, each user is assigned a global value that reflects his or her reliability on the scale of the entire network.

Global trust metrics are simpler and less time-consuming to calculate than local metrics, since local metrics are calculated for each pair of users. However, local trust models can represent

a user's interests more accurately than the global approach. Indeed, local trust models bring personalization to the calculation of trust.

Calculation of trust values. On some digital platforms, users explicitly declare their trust in other users. This can be seen on platforms such as Epinions and Ciao, where users formally express their trust in others. These declarations of trust play a crucial role in improving user recommendations. When a user u_1 declares that he trusts u_2, a value representing u_1's degree of trust in u_2 is calculated. Consequently, the number of values calculated depends closely on the number of explicit trust relationships available.

Jian-Ping Mei et al. [19] have proposed two approaches to calculating trust using data from Epinions. In the first approach, the number of items rated by u_2 is used to estimate u_1's trust in u_2. In the second approach, the number of users declaring that they trust u_2 is used to estimate this trust. However, this work has its limitations, notably a method of estimating global trust that does not take into account the personalization of trust. In addition, it neglects the temporal aspect, assuming that trust between two users remains constant over time, whereas in reality it can vary. Futhermore, it does not take into account the variation of this influence across item categories, whereas one user can be influenced by another strongly for a certain category of item and weakly for another.

We can also mention the research by Nzekon et al. [20], where the method consists of assigning the value 1 in the case of an explicit declaration of confidence and 0 in the opposite case. Here too, we note that the calculated confidence does not take time into account and variation of trust according to item categories.

Once explicit trust information has been exploited to calculate the level of trust between users, it becomes essential to integrate it into conventional recommender systems. This makes it possible to create trust-based recommender systems.

2.2.2 Integrating trust into recommender systems

Users are more willing to accept recommendations from trusted friends than from strangers, as studies show [5]. Building trust into recommender systems improves the relevance of suggestions and the user experience. These trust-based systems use trust metrics and collaborative filtering techniques for more personalized recommendations, overcoming challenges such as cold startup and sparse data. They fall into two categories: memory-based and machine learning model-based.

Collaborative filtering based on memory and trust. An alternative approach to traditional collaborative filtering emphasizes friendly relationships over stranger connections. This method incorporates trust data to reinforce recommendations from trusted users while limiting the impact of others. For example, Mei et al.[19] used trust data from Epinions to improve the K-nearest neighbor (KNN) model. They computed an inter-user trust matrix, then predicted user scores on items based on trusted neighbors. The formula for predicting user u's score on item i is given by the equation 1:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in P_u(i)} Trust(u,v).(r_{vi} - \mu_v)}{\sum_{v \in P_u(i)} |Trust(u,v)|} \tag{1}$$

where $Trust(u, v)$ represents the trust that u places in v, and $P_u(i)$ corresponds to the set of users whom u trusts and who have rated item i.

Model-based and trust-based collaborative filtering. In this category, model-based collaborative filtering techniques, in particular matrix factorization, are widely employed. These approaches are based on the idea that users' preferences are similar to or influenced by those of trusted users. For example, Mei et al.[19] incorporate trust into matrix factorization. The underlying idea is to adjust a user's latent factor (numerical values that describe a user) to bring them closer to those of users he trusts and those who trust him.

Following the analysis of the section 2.2 on explicit trust-based recommendation systems, it is observed that a significant number of works rely on explicit trust relationships declared by users on the platform. However, this data is not commonly available on most digital platforms. The lack of information about explicit trust relationships has given rise to a new line of research, that of recommender systems based on implicit trust.

2.3 Recommender systems based on implicit trust

To talk about of recommender systems based on implicit trust, two conditions must be met. Firstly, the trust between users must be estimated, and then this trust must be integrated into a conventional recommender system to produce a trust-based system. The second condition has already been presented in the case of explicit trust. Thus, in this section, we will focus exclusively on the estimation of implicit trust.

In implicit trust models, two scenarios are distinguished: the case where explicit trust information between users is available, but in addition to this, trust propagation algorithms are used to infer implicit trust information. In the second case, explicit trust information is not available, and the objective is to use other information to construct a network with trust values between users.

2.3.1 Explicit Information and Trust Propagation

In a user network, it is possible to ask each user to evaluate others and assign a value indicating their reliability. While theoretically feasible, in reality, evaluations are limited due to laziness, lack of time, or the fact that a user may have direct opinions about only a small number of users. To address this, techniques are developed to predict the reliability of unevaluated users based on existing trust information. These methods aim to predict confidence scores between all pairs of network nodes, even those for which no explicit opinion has been provided. These approaches typically rely on the assumption of transitivity within the network. They use algorithms that leverage paths in the network to propagate and deduce confidence values.

In some works, manipulated trust values are binary, while in others, they are continuous. In a binary trust network, edges are categorized as "approved" or "not approved." To infer the binary trust values t_{sd} from the source s to the target node d a voting algorithm is proposed [17]. In a continuous trust network, the most intuitive method to deduce trust weights is to calculate the average of trust values. The source queries its neighbors to obtain the trust weights assigned to the target node, repeating this process for each neighbor. When a node receives weights from multiple neighbors, it takes the average. This approach often forms the foundation of other techniques for inferring continuous trust values. For example, trust inferred from paths [10] where when two users are connected through other users in a path, a propagation technique is used to calculate their mutual trust. There is also a method based on matrix factorization [16].

In these works, it is also observed that the trust estimation process through propagation does not consider the temporal dynamics of trust, let alone the variation of trust based on item categories.

2.3.2 Implicit Trust without Explicit Information

Here, the aim is to use other information to build a network with trust values between users. In [8], Ziegler shows that there is a relationship between similar user preferences and trust between them. This means that people who share the same interests and tastes tend to trust each other more. We can therefore conclude that it is reasonable to use measures of user preference similarity to infer implicit trust values. Most of these techniques are based on the similarity of user profiles and the history of explicit ratings.

- **Similarity of user profiles :** Similarity between two users is defined by whether they are linked in a social network, have friends in common, or like the same items or categories of items.
- **History of explicit ratings :** Similarity is high between two users, if they rate the same items in the same way. Users who assign similar ratings are more likely to trust each other.

Using the criterion of explicit rating history, Nzekon et al. [20] estimated the implicit trust between users of the Epinons and Ciao platforms, which they then integrated into the graphs. To estimate implicit trust, they used the Jaccard similarity presented by the equation 2

$$Jaccard(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{2}$$

with I_u representing the set of items purchased by user u and I_v the set of items purchased by v.

This way of estimating trust between users does not require explicit trust information. Instead, it relies on the history of explicit user ratings for items, which is reassuring given that this information is more widely available. However, this approach to confidence estimation has a number of limitations. Firstly, it seems to treat trust as a symmetrical relationship, whereas in reality, trust is an asymmetrical relationship. For example, if a user u_1 grants a trust of 0.8 to u_2, this does not necessarily mean that u_2 grants the same trust to u_1. Furthermore, this similarity does not take into account the temporal aspect of trust, let alone its variation according to item categories. The fact that one user u_1 reproduces the behavior of another user u_2 over time, whether for all or some item categories, is an indication of the influence of u_2 on u_1.

In this section on work in the field of trust-based recommendation systems, Ziegler's work has highlighted the possibility of estimating implicit trust by exploiting user rating histories and similarities between their profiles. Nzekon et al.'s work has also explored estimating implicit trust using Jaccard similarity. However, a common gap in these studies is the lack of consideration for time in trust estimation, as well as the failure to account for the variation of this influence across item categories. Replicating user u1's behavior by another user u2 over time for all or certain item categories indicates u1's influence on u2. To address these shortcomings, the following section proposes integrating this temporal aspect and consideration of categories into trust estimation. Due to the asymmetrical nature of trust reflected in our approach, we adopt the term "Social Influence."

III TIME AND CONTENT AWARE IMPLICIT SOCIAL INFLUENCE ESTIMATION

In this section, we present the taking into account of time and item categories in the estimation of social influence from implicit trust relationships (user ratings). To carry out this task, we start by presenting in section 3.1 the computation of social influences with consideration of time and item categories, and then show in section 3.2 how to integrate these previously computed social influences into three models used in recommender systems namely K nearest neighbors, matrix factorization and graph.

3.1 Calculation of social influences, taking time and item categories into account

This sub-section presents the estimation of implicit social influence, taking time and item categories into account. To do so, we present in sub-section 3.1.1, the consideration of time in the estimation of implicit social influence and in sub-section 3.1.2 the consideration of both time and item categories in the estimation of implicit social influence.

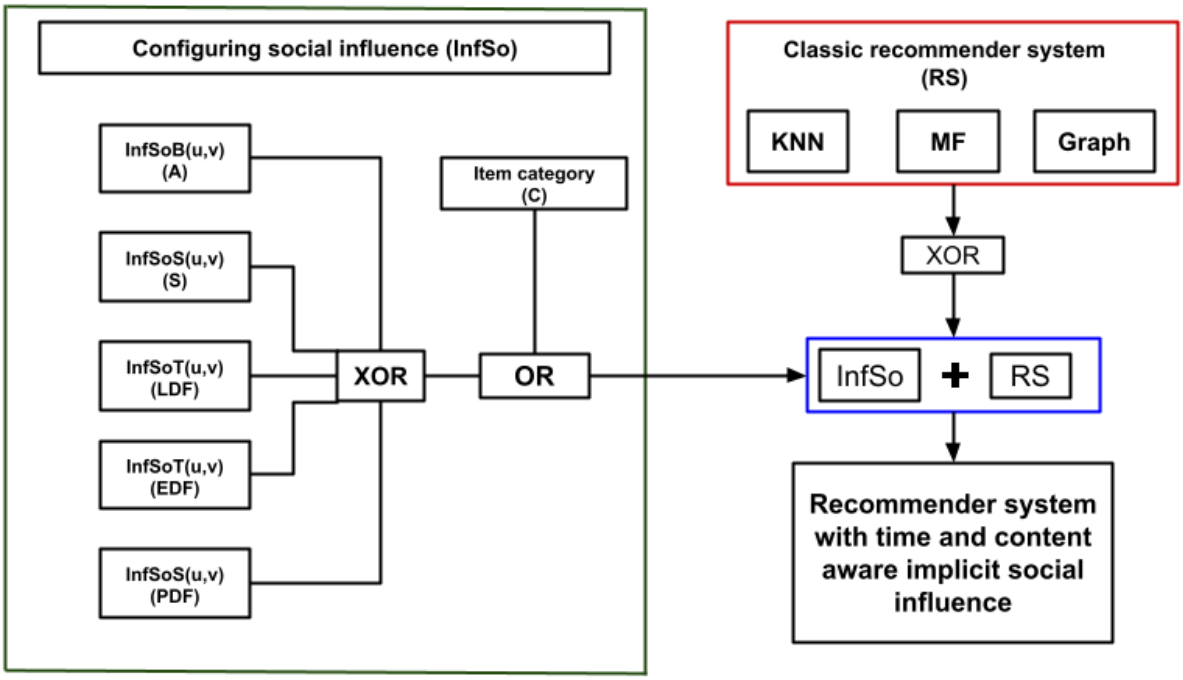

Figure 1: Architecture of the proposed recommender system.

Figure 1 shows the general architecture of our work. In the left rectangle, we choose the social influence matrix under consideration (**A** for asymmetrical social influence, **S** for social influence that takes into account purchase sequencing, and the next 3 for taking time into account with the EDF, LDF and PDF temporal decay functions). If we also want to integrate item categories, we consider calculating matrices for each category. We then integrate the social influence estimated above into a classic recommendation system to produce a **recommender system based on implicit trust that takes time and/or item categories into account**.

3.1.1 Temporal implicit social influence

The approach proposed in this paper is inspired by Jaccard's similarity measure as used by Nzeko'o et al. [20]. This is a symmetrical similarity measure, i.e. the similarity between the

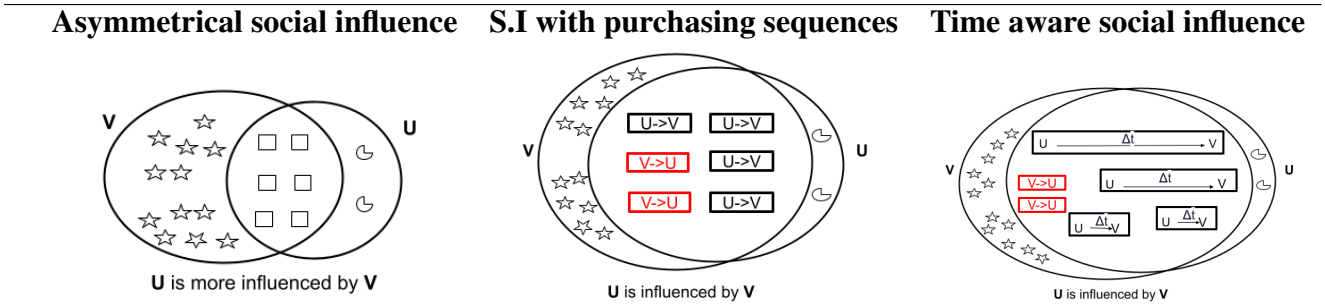

Figure 2: Time aware implicit social influence.

user u and v is the same as that between v and u. The symmetry of this similarity makes it unsuitable for estimating influence between users, as social influence is an asymmetrical measure.

Jaccard's similarity is as follows:

$$Jaccard(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{3}$$

We have therefore modified this previous formula to better estimate the influence between two users. The one we propose is as follows:

$$InfSoB(u, v) = \frac{|I_u \cap I_v|}{|I_u|} \tag{4}$$

I_u is the set of items that user u has rated positively. This measure aims to better capture the influence exerted by v on u, by expressing the proportion of items that u appreciates because of v's (the influencer's) prior appreciation of these items. Figure 2 (left side) best illustrates this idea. The circle V contains the items that user v liked on the platform, while the circle U contains the items liked by user u. The items at the intersection of the two circles are those appreciated by both users. Here, for example, we can see that both users enjoyed six items in common, but that u enjoyed almost all of his items in common with v, while v enjoyed only a tiny fraction of his items in common with u. We can therefore say that u is more influenced by v than v is by u.

It should be noted that a gap arises when v (the influencer) acquires the item after u. In such a scenario, it is irrelevant to conclude that u has been influenced. To overcome this limitation, we have adjusted the formula, incorporating the condition that u must have purchased the item after v. The modified formula 5, presented as the second version of our approach, takes this consideration into account for a more accurate assessment of influence.

$$InfSoS(u, v) = \frac{|u \rightarrow v|}{|I_u|} \tag{5}$$

The arrow pointing to the right expresses the fact that u follows v and therefore that u is influenced by v. The numerator refers to the number of items u purchased after v. Figure 2 (middle side) better illustrates this second version of our approach. Among the six items purchased and

enjoyed by u and v, it's notable that for four of them, u made the purchase after v, while for the other two items marked in red, the reverse is true. So, in applying this new version, we only consider the four items for which u made the purchase after v.

Let's consider the case where u buys the item two (02) days after v, and let's also consider the case where u buys the item two years after v. In reality, the influence that v exerts on u in the first case is much greater than in the second, because the shorter the time between u's purchase of the same item and v, the greater the influence exerted on u. However, the second version of our idea does not take this into account. For this reason, we propose a third version of our idea, which takes time into account when estimating the influence a user has. The formula 6 describes this.

$$InfSoT(u,v) = \frac{\sum_{|u \to v|} f(t_u - t_v)}{|I_u|} \tag{6}$$

t_u and t_v are respectively the times at which u and v rated the item, and f is a temporal decay function. The idea behind this type of function is to give a high weighting to influence relations for which the duration between the two purchases (that of u and v) of the same item is small, and to reduce the weighting in the opposite case. There are several temporal penalty functions. In this article, we used three (03) functions, two (02) of which were used in the graphs to penalize the weight of the oldest edges and give a high weight to the most recent edges [20] :

- **Exponential Decay Function *(EDF)*:** It is illustrated in figure 3 (left side) and has the expression $f(x) = e^{-x.ln(2)/to}$. to is the half-life, i.e. after a time to elapsed between the purchase of v and that of u, the weight of the influence of v on u in relation to a given item diminishes by half.
- **Logistic Decay Function *(LDF)*:** It is illustrated in figure 3 (middle side) and has the expression $f(x) = 1 - 1/(e^{-K(x-to)} + 1)$. K is the slope of the decay curve and to the half-life as well. The difference with the exponential function is that the logistic function cancels when $x = 2 * to$.
- **Power Decay Function *(PDF)*:** it is illustrated in figure 3 (right side) and has expression $f(x) = x^{(log_{to}(1/2))}$. where to is also half life. The difference with the first two functions is that the power function decays more slowly than the others.

Figure 2 (right side) illustrates the third version of our approach. Δt refers to the time that elapsed between the moment when v made the purchase of the item and the moment when u replicated the purchase of the item. It is this duration that is sent to one of the temporal decay functions. These operations are performed by considering only those transactions for which u purchased the item after v.

3.1.2 Temporal implicit social influence by item category

In the previous section, we introduced the temporal dimension into the estimation of social influence. In this section, we also take into account item categories. Influence between two users can vary according to the item category considered. Thus, it is essential to estimate the influence of a user v on a user u in a specific category c. To do this, we adapt the previous formulas 4, 5 and 6 by calculating a separate social influence matrix for each item category. For example, in the context of a streaming platform with films in different categories (action, drama,

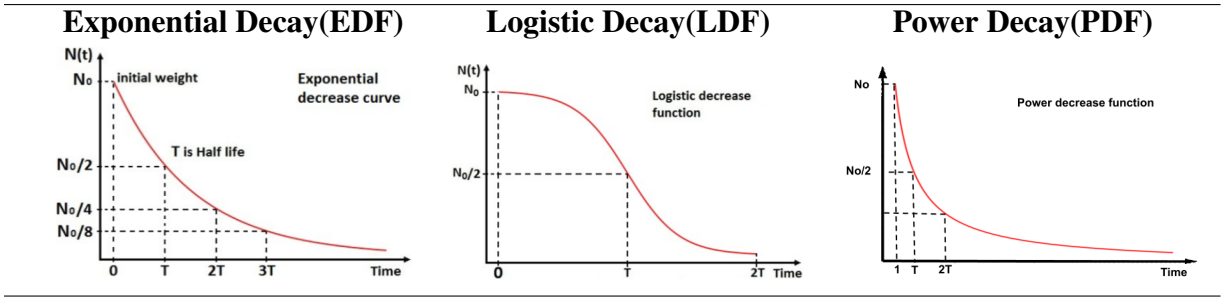

Figure 3: Time decrease functions.

romance, comedy), we calculate a matrix for each category. Using the equation 4 as an example, we adjust the calculations for each category. For the "drama" category, the numerator represents the number of items that u and v acquired jointly in this category, while the denominator is the total number of items acquired by u in this category.

The equations 7, 8, and 9 formalize the calculation of social influences by integrating item categories. These formulations apply respectively to scenarios where sequences of purchases between users are neglected, where they are taken into account, and where a temporal decay function is applied.

$$InfSoBC(u, v, c) = \frac{|I_{uc} \cap I_{vc}|}{|I_{uc}|} \tag{7}$$

$$InfSoSC(u, v, c) = \frac{|u \rightarrow v|_c}{|I_{uc}|} \tag{8}$$

$$InfSoTC(u, v, c) = \frac{\sum_{|u \rightarrow v|_c} f(t_u - t_v)}{|I_{uc}|} \tag{9}$$

For the above equations, c is the category concerned, I_{uc}, the set of items that u liked and that belong to category c , $|u \rightarrow v|_c$ the number of items that u bought after v and that belong to the c category, t_u the time at which u rated the item and finally t_v the time at which v rated the item.

3.2 Integrating implicit social influence into classic recommender systems

In this section, we show how to integrate the social influences estimated in section **??** into the three basic recommendation algorithms we have chosen. The choice of these algorithms was guided by taking some memory-based (User-KNN and Classic Bipartite Graph) and others model-based (Matrix Factorization).

3.2.1 K nearest neighbors - KNN

User-KNN is a neighborhood-based algorithm. It uses similarities between users to make item recommendations to a user based on a matrix of ratings. The last two steps in this algorithm are to identify the target user's k nearest neighbors and to predict that user's score for a given item. To integrate social influences, we have replaced the Pearson similarity matrix with the influence matrix we have proposed, both for the choice of neighbors and for the prediction of user ratings. For further exploration, we have considered two scenarios:

- The case where the Pearson matrix is used for neighborhood selection and the social influence matrix is used to calculate score prediction.
- The case where the social influence matrix is used for neighborhood selection and for calculating score prediction.

Let's recall here the formula for calculating the score prediction that the user u gives to an item i that he hasn't yet selected:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in P_u(i)} Sim(u,v).(r_{vi} - \mu_v)}{\sum_{v \in P_u(i)} |Sim(u,v)|} \tag{10}$$

Integrating social influences, the formula becomes :

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in P_u(i)} infSo(u,v).(r_{vi} - \mu_v)}{\sum_{v \in P_u(i)} |infSo(u,v)|} \tag{11}$$

with $infSo(u,v)$ being the social influence that v exerts on u. Note that $infSo(u,v)$ represents one of the influences expressed in equations 4, 5, and 6. In other words, $infSo(u,v) \in \{InfSoB(u,v), InfSoS(u,v), InfSoT(u,v)\}$

For the integration into the K-nearest neighbor model of social influences that take into account item categories, we also consider the two cases mentioned in the previous section. The important thing here is to capture the influence that one user exerts on another in relation to a given category. In practice, an item may belong to several categories. In this case, the rating prediction formula must take into account the influences in relation to all the item categories in question. It therefore becomes :

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{c \in Cat(i)} \sum_{v \in P_u(i)} infSo(u,v,c).(r_{vi} - \mu_v)}{\sum_{c \in Cat(i)} \sum_{v \in P_u(i)} |infSo(u,v,c)|} \tag{12}$$

Where $Cat(i)$ is the set of categories for item i. For the choice of neighborhood, when the item belongs to several categories, the matrix used is the average of the influence matrices of all the categories of item i.

3.2.2 Matrix factorization

Matrix factorization is a model used in recommender systems to predict the scores that users will give to items. Initially, we have a matrix of ratings $R \in R^{m*n}$ concerning m users and n items.

The idea behind matrix factorization is to learn a model of user latent factors $U \in R^{m*k}$ and item latent factors $V \in R^{n*k}$ so that the reconstruction $R^{'} = U * V^T$ between the two matrices best estimates the initial matrix R and therefore predicts the initially unknown scores of the starting matrix. k is the number of latent factors per user and per item, and is therefore a parameter of this model. u_i is the vector representing the k latent factors of user i and v_j the vector representing those of item j.

The matrix factorization model is based on the values of the U and V matrices. So it's important to devise a trick to find the right values (latent factors) for these two matrices. To do this, this model uses an optimization algorithm called **stochastic gradient descent**.[19].

The purpose of this model is to find the latent factors that minimize the function 13:

$$min\ J = \frac{1}{2}\sum_{(i,j)\in S}(r_{ij} - \hat{r}_{ij})^2 + \frac{\lambda}{2}\left(\sum_{i=1}^{m}\|u_i\|^2 + \sum_{j=1}^{n}\|v_j\|^2 + \sum_{i=1}^{m}o_i^2 + \sum_{j=1}^{n}p_j^2\right) \tag{13}$$

Algorithm III.1 Matrix factorization

Input: rating matrix R, set S of ratings $r_{ij} \neq 0$, k the number of latent factors, parameters α and λ, number of d'epochs N

Output: Matrix of latent factors U of users and V of items; Vector O of user bias and P of item bias

```
random initialization of U , V, O et P
  l ← 0
  while l < N
      ∀r_ij ∈ S
          update vector u_i by u_is = u_is + α.(v_js.e_ij − λ.u_is) ∀s ∈ {1, .., k}
          update vector v_j by v_js = v_js + α.(u_is.e_ij − λ.v_js) ∀s ∈ {1, .., k}
          update user biais o_i by o_i = o_i + α.(e_ij − λ.o_i)
          update item biais p_j by p_j = p_j + α.(e_ij − λ.p_j)
      l ← l + 1
  end
```

To incorporate social influences between users in the matrix factorization model, we drew on the work of Mei et al. [19]. They formulated social influence using explicit trust information and incorporated it into the matrix factorization and K-nearest neighbor models.

The aim of this integration of social influence is to update a user's latent factors in such a way as to bring them closer to the latent factors of the users he influences, while also bringing them closer to the latent factors of his influencers. The equation 14 is the new equation to be minimized with this integration.

$$min\ F = min\ J + \frac{\beta}{2}\sum_{i}\left(\frac{\sum_{l\in N_i} infSo(i,l)\|u_i - u_l\|^2}{\sum_{l\in N_i} infSo(i,l)} + \frac{\sum_{w|i\in N_w} infSo(w,i)\|u_i - u_w\|^2}{\sum_{w|i\in N_w} infSo(w,i)}\right) \tag{14}$$

With N_i being the set of influencers of user i and β the weight that controls the contribution of social influence. The expression that has been added is made up of two parts, the first concerns i's influencers and the second refers to the w users that i influences.

Equation 15 is the new equation for updating the latent factors of user i :

$$u_{is} = u_{is} + \alpha.\left(v_{js}.e_{ij} - \lambda.u_{is} - \beta.sum1\right) \tag{15}$$

with

$$sum1 = \left(\frac{\sum_{l\in N_i} infSo(i,l)(u_{is} - u_{ls})}{\sum_{l\in N_i} infSo(i,l)} + \frac{\sum_{w|i\in N_w} infSo(w,i)(u_{is} - u_{ws})}{\sum_{w|i\in N_w} infSo(w,i)}\right) \tag{16}$$

To integrate item categories into latent factor learning, we adjust a user's latent factors a number of times equivalent to the number of categories to which the item he has purchased belongs. For example, if an item j is associated with three categories in a transaction (i, j, r), the latent factors of user i are adjusted three times, each according to the respective category. Equation 17 is the new equation for updating the latent factors of user i :

$$u_{is} = u_{is} + \alpha.\left(v_{js}.e_{ij} - \lambda.u_{is} - \beta.sum2\right) \tag{17}$$

with

$$sum2 = \left(\frac{\sum_{l \in N_i} infSo(i,l,c)(u_{is} - u_{ls})}{\sum_{l \in N_i} infSo(i,l,c)} + \frac{\sum_{w|i \in N_w} infSo(w,i,c)(u_{is} - u_{ws})}{\sum_{w|i \in N_w} infSo(w,i,c)}\right) \tag{18}$$

and c being the item category.

3.2.3 Recommendation graphs

The user-item graph is defined as a bipartite, undirected graph $G = (N_u \cup N_i, A)$ with N_u being the set of nodes representing users and N_i that of nodes representing items. A is the set of edges in the graph. All these edges exist only between users and items. An edge of A exists between user u and item i if and only if u has rated item i.

For example, figure 4 shows a rating matrix and its associated user-item graph:

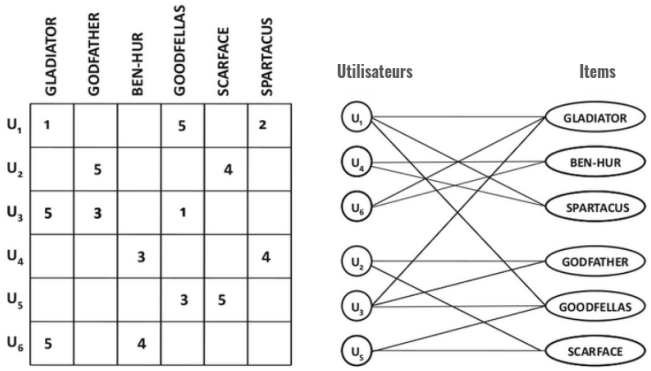

Figure 4: Rating matrix and associated classical bipartite graph.

Once the graph has been constructed, a random walk algorithm is applied. The recommendation hypothesis here is based on the transitivity of user tastes and the proximity of the target user to items that match his preferences. Thus, the objective is to recommend N items that the target user has not yet purchased and that are closest to him and his neighborhood.

PageRank, a random walk algorithm developed by Google's co-founders, was originally designed to rank web pages in order of importance. An adapted version, personalised PageRank, was then developed to recommend items to users [9]. This algorithm identifies the N items most likely to interest a user by performing a random walk starting from the node associated with that user. Node weights are stored in a vector PR, calculated iteratively according to the equation 19. M is the adjacency matrix of the constructed graph, $\alpha \in [0, 1]$ is the personalization attenuation factor and d the PageRank personalization vector. To recommend items to the

user, the corresponding scores in PR are sorted in descending order, and the N highest-scoring items not yet selected by the user are recommended.

$$PR = \alpha \ * M \ * PR + (1 - \alpha) \ * d \tag{19}$$

To integrate social influence into the classical bipartite graph, we adopted an approach inspired by the work of Nzeko'o et al. [20], who use Jaccard similarity in this context. We have adapted their method to include social influences in our study. The main idea is to adjust the personalization vector d. In the basic personalized PageRank, the node associated with the target user has a value of 1 in d, while the others are zero. However, to integrate social influences, this value (1) is split between the target user's node u and those of the users who influence it. For further details, see :

$d(u) = 1 - l$ with u being the target user

$$d(v) = \frac{(l) \ * \ infSo(u, v)}{\sum_{v \in Q_u} infSo(u, v)} \text{ if } v \in Q_u \text{ and } d(v) = 0 \text{ otherwise}$$

With Q_u being the set of nodes associated with users who influence u and l the value we share with nodes associated with users who influence u.

To incorporate social influences while taking into account item categories in the classic bipartite graph, we designed a new bipartite graph that takes into account the relationships between users and item categories. Each time a user u buys an item i, we create uc nodes for each category c to which i belongs, and establish links between i and these uc nodes, while maintaining the link between u and i. See figure 5 for an example in which item i belongs to categories $c1$ and $c2$ and j belongs only to category $c1$.

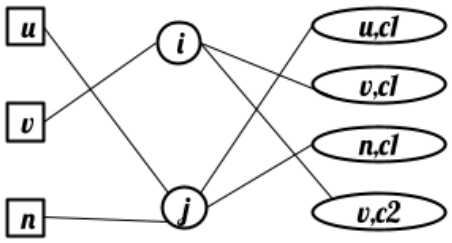

Figure 5: Bipartite graph with user-category relations taken into account

Using the graph we have proposed, we use the personalised PageRank algorithm by adjusting the personalization vector d so that the value (1) is distributed between the node associated with the target user u and the nodes linked to the user-category relationships for users who influence u in the context of the categories concerned. We therefore have :

$d(u) = 1 - l$ with u being the target user

$$d(v) = \frac{(l) \ * \ infSo(u,v,c)}{\sum_{c \in Cat} \sum_{v \in Q_{uc}} infSo(u,v,c)} \text{ if } v \in Q_{uc} \text{ with } c \in Cat \text{ and } d(v) = 0 \text{ otherwise .}$$

With Q_{uc} being the set of nodes associated with user-category relations for users who influence u with respect to category c. l is the value shared by nodes associated with user-category relationships for users who influence u in relation to categories c. Cat is the set of all categories in the dataset.

IV IMPLEMENTATION AND RESULTS

This section is dedicated to the presentation of the experiments carried out and the results obtained. It is structured into three (03) main subsections: the subsection 4.1 which sets out the datasets on which the experiments were carried out, the subsection 4.2 which details the experimental protocol, and finally the subsection 4.3 which presents the results, accompanied by comments.

4.1 Dataset

To carry out the experiments, we made use of publicly available dataset extracts from the Epinions and Ciao platforms [15] . These platforms are dedicated to publishing user reviews of a wide range of items, such as books, DVD, computers, and other items in various categories. Each of these datasets has been modeled as a set of tuples (u, i, c, r, t), meaning that user u has assigned a rating $r \in \{1, 2, 3, 4, 5\}$ to item i at time t, with c representing the category of item i. Both datasets also contain information on explicit trust between users. This data is in the form of tuples (u_1, u_2), meaning that user u_1 trusts user u_2.

Table 1 provides information on these two datasets. minU denotes the minimum number of appearances of a user in the dataset, while minI represents the minimum number of appearances of an item in the dataset. NbNotes quantifies the number of ratings present in the dataset, while NbUsers and NbItems refer, respectively, to the number of users and items present in the dataset.

	NbUsers	**NbItems**	**minU**	**minI**	**NbRatings**	**Period**	**NbCat**	**Conf**
Ciao	889	9053	1	1	12742	2007-2011	6	23385
Epinions	728	18141	20	2	58717	2006-2010	24	1381

Table 1: Description of the Epinions and Ciao datasets.

4.2 Experimental protocol

In this subsection, we present the segmentation of the dataset, the metrics for evaluating recommendation models and the description of some parameters.

4.2.1 Dataset segmentation

The task of dividing the dataset must be carried out in such a way as to guarantee the reliability of the resulting recommendation system. To assess this reliability, it is not enough to simply partition the data into training and test sets. It is necessary to implement a mechanism enabling repeated evaluation of the recommender system, thus providing a better insight into its reliability.

In this paper, we have therefore opted to use cross-validation with a time window of increasing size. We chose this method because it ensures that all the data in the test set are more recent

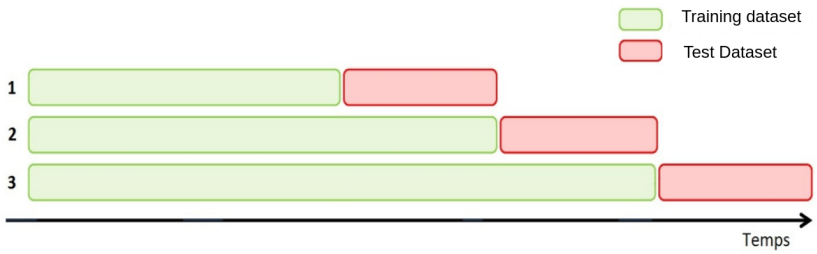

Figure 6: Cross-validation with increasing time window size.

than those in the training set. This method takes into account the temporal evolution of the data, where the size of the training set increases progressively with time. To implement this method, we define a test window size, which we set at four (04) months, denoted d_{jt}, and optionally an initial training set size, which we set at three (03) years, denoted d_{ja}. Thus, the first [training, test] sample covers the period $[d_{ja}, d_{jt1}]$, the second encompasses the interval $[d_{ja} + d_{jt1}, d_{jt2}]$, and so on, with the final sample covering data from the interval $[d_{ja} + d_{jt1} + ... + d_{jt(k-1)}, d_{jtk}]$. This method, used previously in [11], is illustrated in figure 6. In this work, we carried out our experiments using three separate samples.

4.2.2 Evaluation metrics

As evaluation metrics, we chose two metrics commonly used to assess the quality of score prediction, namely Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The first corresponds to the square root of the average square score prediction errors, while the second refers to the mean of the absolute error values. The prediction error here is the difference between the actual score and the predicted score. These two metrics are defined by $RMSE = \sqrt{\frac{\sum_{u,i \in r_{test}} (r_{ui} - \hat{r}_{ui})^2}{|r_{test}|}}$ and $MAE = \frac{\sum_{u,i \in r_{test}} |r_{ui} - \hat{r}_{ui}|}{|r_{test}|}$ with r_{test} the test data set..

For each of these metrics, the larger the value, the greater the model error and, consequently, the poorer the model's performance. The fact that the MAE or RMSE decreases indicates an improvement in the performance of the recommender system.

In addition to these metrics for evaluating the quality of prediction, we also considered 03 metrics for evaluating the quality of top-N recommendations. N corresponds to the number of items recommended to a user, but not yet purchased. These metrics are the *Mean Average Precision (MAP)*, the *Normalized Discounted Cumulative Gain (NDCG)* and the *Hit Ratio (HR)*.

Mean Average Precision (MAP) : Using this metric, we perform precision calculations taking into account the position of relevant items among the recommended N, which differs from the standard precision metric. The calculation of *MAP* is given by $MAP@N = \frac{\sum_{u \in U} AP_N(u)}{|U|}$ where $AP_N(u) = \frac{\sum_{k=1}^{N} \frac{hit_k(u)}{k} * h(k)}{hit_N(u)}$ and denoting the average accuracy of the top-N recommendations proposed to user u and $hit_k(u)$ the number of good recommendations for k items recommended to user u. U is the set of users in the dataset and $h(k) = 1$ if the item at position k is a good recommendation and 0 otherwise.

In addition to the *Mean Average Precision (MAP)*, another metric that also considers the position of items in the recommendation list is the *Discounted Cumulative Gain (DCG)*. It is defined by $DCG@N(u) = \sum_{i=1}^{N} \frac{Gain(i)}{\log_2(i+1)}$ with $DCG@N(u)$ representing the *DCG* considering the first N items recommended to user u. $Gain(i) = 1$ if the item at position i is a good recommendation and 0 otherwise. The DCG either increases with N, or remains the same. This means that

the larger N is, the higher the metric will be. One way of making comparisons fairer between different values of DCG for different values of N (top-N) is to normalize the DCG score by dividing it by the maximum possible DCG at each value of N. Hence the NDCG metric.

Normalized Discounted Cumulative Gain (NDCG) : *NDCG* is calculated by dividing the cumulative gain *(DCG)* of the list of recommended items by the *DCG* of the ideal recommendation list. This ideal recommendation list is the list where relevant items are ranked in the most optimal order, i.e. in the top positions. The *NDCG* varies from 0 to 1, with higher values indicating better performance. It is given by $NDCG@N(u) = \frac{DCG@N(u)}{IDCG@N(u)}$ With $IDCG@N(u)$ being the ideal *DCG* considering a recommendation of N items to user u. For all users of the dataset, the *NDCG* is given by $NDCG@N = \frac{\sum_{u \in U} NDCG@N(u)}{|U|}$

Hit ratio (HR) : The *Hit ratio* is the proportion of users to whom the recommender system has made at least one good recommendation. This metric is used to determine the proportion of satisfied users on the platform. It is given by $HitRatio@N = \frac{\sum_{u \in U} (hit_N(u) > 0)}{|U|}$.

4.2.3 Predefined parameter values

The parameters vary according to the classical recommendation models in which we have incorporated social influence. For the K-nearest neighbor model, the description of the parameters and their predefined values is summarized in table 2 . For the matrix factorization model, the parameters and predefined values are shown in table 3 . Finally, for the graph-based model, we have set out the description of the various parameters and their predefined values in table 4.

	Parameter description	**Predefined values**
K	Number of neighbors	2, 3, 5, 10, 20, 30
To	Half-life of EDF, LDF and PDF functions	30, 60, 120, 240, 360 days

Table 2: Predefined parameter values for KNN.

	Parameter description	**Predefined values**
K	Number of latent factors	2, 5, 10, 30
To	Half-life of EDF, LDF and PDF functions	30, 60, 120, 240, 360 days
β	Social influence rate	0.000001, 0.0001, 0.05, 0.1, 0.5

Table 3: Predefined parameter values for matrix factorization

	Parameter description	**Predefined values**
α	PageRank damping factor	0.25, 0.5, 0.75
To	Half-life of EDF, LDF and PDF functions	30, 60, 120, 240, 360 days
(l)	Influence factor attributed to u influencers	0.1, 0.3, 0.5, 0.7, 0.9

Table 4: Predefined parameter values for graphs

4.3 Results and comments

Before presenting the results, we first present in table 5 the configurations we subjected to experimentation. The results we present below refer to the best performance obtained, taking into account the multiple parameters we have adjusted. In other words, we will present the optimal performance when we have varied several parameter values for a given configuration.

	Code description
B	Basic model, i.e. no integration of additional information
J	Integrating Jaccard similarity into the model (equation 3)
A	Integration of social influence that does not take into account the sequencing of purchases between users (equation 4)
S	Integration of social influence, taking into account the sequencing of purchases between users (equation 5)
TE	Integration of time aware social influence with the EDF time decay function (equation 6)
TL	Integration of time aware social influence with the LDF time decay function (equation 6)
TP	Integration of time aware social influence with the PDF time decay function (equation 6)
C	Integration of social influence taking into account item categories (equation 7)
S-C	Integration of social influence, taking into account the sequencing of purchases between users and item categories (equation 8)
TL-C	Integration of time aware social influence with LDF function and item categories (equation 9)
TE-C	Integration of time aware social influence with EDF function and item categories (equation 9)
TP-C	Integration of time aware social influence with PDF function and item categories (equation 9)

Table 5: Description of different configuration codes.

4.3.1 Results for integrating temporal and categorical social influence in KNN, MF and Graphs

Table 6 presents the results of experiments on the Ciao and Epinions datasets. These results concern the metrics *MAE* and *RMSE* for the KNN and matrix factorization models (only for Ciao). In the "U-KNN (Pearson-Model)" column, users' neighbors are determined using the Pearson matrix. In the "U-KNN (Model-Model)" column, the same matrix (representing social influence) is used for both neighbor selection and score prediction. The last large column concerns the matrix factorization model. Lighter cells indicate better recommendation system performance, while darker cells (red) indicate less satisfactory performance. Note that performance improves when the time factor is taken into account in the estimation of social influence, and is even more promising when item categories are also taken into account.

Table 6: KNN results and Matrix Factorisation with Ciao and Epinions

CIAO	U-KNN (Pearson-Model)		U-KNN (Model-Model)		Factorisation matricielle	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
B	0.73831	1.02394	0.73831	1.02394	0.71034	0.97938
J	0.71973	0.99342	0.72175	0.99478	0.71034	0.97938
A	0.71888	0.99333	0.72018	0.99499	0.71034	0.97938
S	0.70136	0.96306	0.70361	0.96574	0.71033	0.97935
TE	0.69944	0.96092	0.70319	0.96556	0.71033	0.97935
TL	0.68744	0.95384	0.6954	0.96614	0.71032	0.97934
TP	0.70093	0.96287	0.70431	0.96563	0.71033	0.97935
C	0.70691	0.97866	0.71344	0.98646	0.71034	0.97938
S-C	0.69686	0.95991	0.70142	0.96234	0.71034	0.97936
TL-C	0.6874	0.95377	0.69251	0.95805	0.71032	0.97933
TE-C	0.69531	0.958	0.70076	0.96179	0.71033	0.97934
TP-C	0.6967	0.95964	0.70177	0.96241	0.71034	0.97936

EPINIONS	U-KNN (Pearson-Model)		U-KNN (Model-Model)	
	MAE	RMSE	MAE	RMSE
B	0.90281	1.17958	0.90281	1.17958
J	0.84592	1.09995	0.84615	1.10008
A	0.84827	1.10318	0.84852	1.10324
S	0.84764	1.09948	0.84784	1.09953
TE	0.84797	1.10017	0.84817	1.10021
TL	0.83337	1.07178	0.8336	1.07241
TP	0.84767	1.09948	0.84787	1.09952
C	0.84769	1.09585	0.84762	1.09574
S-C	0.8479	1.09272	0.84783	1.09263
TL-C	0.8296	1.06226	0.82951	1.06252
TE-C	0.84816	1.09322	0.8481	1.09311
TP-C	0.84791	1.09269	0.84786	1.09261

Table 7 shows the results for the graph model on the Ciao and Epinions datasets, with the metrics *Hit Ratio (HR)*, *Mean Average Precision (MAP)* and *Normalized Discounted Cumulative Gain (NDCG)*. The color conventions are the same as in the previous tables. We also note that incorporating time and item categories significantly improves the performance of recommender systems.

Table 7: Results of the classic bipartite graph with Ciao and Epinions

CIAO	Hit Ratio			MAP			NDGC		
	HR@10	HR@30	HR@50	MAP@10	MAP@30	MAP@50	NDG@10	NDG@30	NDG@50
B	0.04422	0.08844	0.14626	0.00838	0.01118	0.01221	0.01287	0.0231	0.03329
J	0.04422	0.08844	0.12925	0.01044	0.01285	0.01378	0.01465	0.02551	0.03305
A	0.03401	0.08503	0.12925	0.00627	0.00845	0.00954	0.00981	0.02208	0.02963
S	0.04082	0.09184	0.13946	0.00931	0.01208	0.01282	0.01457	0.02531	0.03276
TE	0.04422	0.08844	0.13265	0.00839	0.01111	0.01212	0.01471	0.02349	0.0315
TL	0.04422	0.09184	0.14286	0.01105	0.01381	0.0149	0.01682	0.02656	0.03392
TP	0.04422	0.09524	0.13946	0.00991	0.01317	0.01365	0.01555	0.0262	0.03308
C	0.04762	0.11224	0.15306	0.01787	0.01905	0.02016	0.01979	0.03196	0.04068
S-C	0.04422	0.11224	0.15646	0.01543	0.01825	0.01829	0.01842	0.03189	0.04173
TL-C	0.04082	0.11905	0.17007	0.01553	0.01855	0.01976	0.01818	0.03216	0.04217
TE-C	0.04422	0.12245	0.15646	0.01523	0.01869	0.01944	0.01791	0.03402	0.04228
TP-C	0.04762	0.11224	0.15646	0.01585	0.01874	0.01839	0.01949	0.03235	0.04254

EPINIONS	Hit Ratio			MAP			NDGC		
	HR@10	HR@30	HR@50	MAP@10	MAP@30	MAP@50	NDG@10	NDG@30	NDG@50
B	0.0455	0.10582	0.1545	0.015	0.01682	0.01719	0.01517	0.02823	0.03874
J	0.04233	0.10476	0.15556	0.01288	0.01564	0.01618	0.01445	0.02827	0.03946
A	0.04127	0.10794	0.14921	0.01237	0.01549	0.01541	0.01408	0.02791	0.03758
S	0.04339	0.09206	0.13862	0.01459	0.01609	0.01689	0.01633	0.02728	0.03711
TE	0.04021	0.09312	0.1418	0.01335	0.01496	0.01483	0.01448	0.02664	0.03576
TL	0.03915	0.09206	0.14074	0.01305	0.01527	0.0146	0.01362	0.02544	0.03537
TP	0.04444	0.09524	0.13968	0.01484	0.0163	0.01676	0.01667	0.02795	0.03719
C	0.04868	0.12593	0.17566	0.01534	0.01856	0.01939	0.01729	0.03332	0.04497
S-C	0.04656	0.11852	0.17143	0.01339	0.01714	0.01823	0.01649	0.03204	0.04367
TL-C	0.04444	0.12169	0.17566	0.01264	0.01688	0.01803	0.01626	0.03203	0.04419
TE-C	0.04021	0.12169	0.1746	0.01298	0.01728	0.01834	0.01538	0.03232	0.04405
TP-C	0.04444	0.11852	0.17037	0.01316	0.01709	0.01819	0.01594	0.03181	0.04344

4.3.2 Comparison with the integration of explicit trust in KNN, MF, and Graphs

In the preceding tables, we have examined the effect of integrating time and item categories in the estimation of social influence, comparing them to the base cases of each model as well as to the integration of Jaccard similarity. We now want to compare ourselves with the integration of explicit trust information between users in classical recommender systems. To this end, we draw on the work of Mei et al. [19] (represented by **Activ** and **Trust**) and Nzekon et al. [20] (represented by **Bin**), whose methods were outlined in section **??**.

Table 8 shows the experimental results for the Ciao and Epinions datasets, including explicit trust information. We note that taking time and item categories into account gives better results than explicit trust for the Ciao dataset. For the Epinions dataset, taking time into account gives poorer results compared with explicit trust, but taking item categories into account gives results similar to those of explicit trust.

Table 8: KNN results and matrix factorization with Ciao and Epinions - Comparison with explicit trust

CIAO	U-KNN (Pearson-Model)		U-KNN (Model-Model)		Factorisation matricielle	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
B	0.73831	1.02394	0.73831	1.02394	0.71034	0.97938
J	0.71973	0.99342	0.72175	0.99478	0.71034	0.97938
Bin	0.719	0.99745	0.72547	1.0115	0.71032	0.97938
Activ	0.72047	0.99811	0.72705	1.01266	0.71034	0.97938
Trust	0.71952	0.99762	0.72627	1.01224	0.71033	0.97938
A	0.71888	0.99333	0.72018	0.99499	0.71034	0.97938
S	0.70136	0.96306	0.70361	0.96574	0.71033	0.97935
TE	0.69944	0.96092	0.70319	0.96556	0.71033	0.97935
TL	0.68744	0.95384	0.6954	0.96614	0.71032	0.97934
TP	0.70093	0.96287	0.70431	0.96563	0.71033	0.97935
C	0.70691	0.97866	0.71344	0.98646	0.71034	0.97938
S-C	0.69686	0.95991	0.70142	0.96234	0.71034	0.97936
TL-C	0.6874	0.95377	0.69251	0.95805	0.71032	0.97933
TE-C	0.69531	0.958	0.70076	0.96179	0.71033	0.97934
TP-C	0.6967	0.95964	0.70177	0.96241	0.71034	0.97936

EPINIONS	U-KNN (Pearson-Model)		U-KNN (Model-Model)	
	MAE	RMSE	MAE	RMSE
B	0.90281	1.17958	0.90281	1.17958
J	0.84592	1.09995	0.84615	1.10008
Bin	0.82376	1.04357	0.82397	1.04399
Activ	0.82378	1.04352	0.82409	1.04423
Trust	0.82377	1.04351	0.82414	1.04427
A	0.84827	1.10318	0.84852	1.10324
S	0.84764	1.09948	0.84784	1.09953
TE	0.84797	1.10017	0.84817	1.10021
TL	0.83337	1.07178	0.8336	1.07241
TP	0.84767	1.09948	0.84787	1.09952
C	0.84769	1.09585	0.84762	1.09574
S-C	0.8479	1.09272	0.84783	1.09263
TL-C	0.8296	1.06226	0.82951	1.06252
TE-C	0.84816	1.09322	0.8481	1.09311
TP-C	0.84791	1.09269	0.84786	1.09261

Table 9 also shows the results for the Ciao and Epinions datasets for the graph model. We note that in the graph model, taking into account time and item categories always gives better performance compared to explicit trust.

Table 9: Results of the classical bipartite graph with Ciao and Epinions - Comparison with explicit trust

CIAO	Hit Ratio			MAP			NDGC		
	HR@10	HR@30	HR@50	MAP@10	MAP@30	MAP@50	NDG@10	NDG@30	NDG@50
B	0.04422	0.08844	0.14626	0.00838	0.01118	0.01221	0.01287	0.0231	0.03329
J	0.04422	0.08844	0.12925	0.01044	0.01285	0.01378	0.01465	0.02551	0.03305
Bin	0.01701	0.07483	0.10884	0.00482	0.00818	0.00832	0.00654	0.01826	0.02528
Activ	0.01701	0.07483	0.10884	0.00411	0.00696	0.0077	0.00647	0.01801	0.02501
Trust	0.01701	0.07483	0.10884	0.00516	0.0082	0.00833	0.00659	0.01865	0.02566
A	0.03401	0.08503	0.12925	0.00627	0.00845	0.00954	0.00981	0.02208	0.02963
S	0.04082	0.09184	0.13946	0.00931	0.01208	0.01282	0.01457	0.02531	0.03276
TE	0.04422	0.08844	0.13265	0.00839	0.01111	0.01212	0.01471	0.02349	0.0315
TL	0.04422	0.09184	0.14286	0.01105	0.01381	0.0149	0.01682	0.02656	0.03392
TP	0.04422	0.09524	0.13946	0.00991	0.01317	0.01365	0.01555	0.0262	0.03308
C	0.04762	0.11224	0.15306	0.01787	0.01905	0.02016	0.01979	0.03196	0.04068
S-C	0.04422	0.11224	0.15646	0.01543	0.01825	0.01829	0.01842	0.03189	0.04173
TL-C	0.04082	0.11905	0.17007	0.01553	0.01855	0.01976	0.01818	0.03216	0.04217
TE-C	0.04422	0.12245	0.15646	0.01523	0.01869	0.01944	0.01791	0.03402	0.04228
TP-C	0.04762	0.11224	0.15646	0.01585	0.01874	0.01839	0.01949	0.03235	0.04254

EPINIONS	Hit Ratio			MAP			NDGC		
	HR@10	HR@30	HR@50	MAP@10	MAP@30	MAP@50	NDG@10	NDG@30	NDG@50
B	0.0455	0.10582	0.1545	0.015	0.01682	0.01719	0.01517	0.02823	0.03874
J	0.04233	0.10476	0.15556	0.01288	0.01564	0.01618	0.01445	0.02827	0.03946
Bin	0.04021	0.09312	0.14074	0.01328	0.01476	0.01505	0.01297	0.02421	0.03398
Activ	0.04021	0.09312	0.13968	0.01339	0.01486	0.01512	0.01303	0.02439	0.03382
Trust	0.04021	0.09312	0.13862	0.01329	0.01477	0.015	0.01297	0.02424	0.03356
A	0.04127	0.10794	0.14921	0.01237	0.01549	0.01541	0.01408	0.02791	0.03758
S	0.04339	0.09206	0.13862	0.01459	0.01609	0.01689	0.01633	0.02728	0.03711
TE	0.04021	0.09312	0.1418	0.01335	0.01496	0.01483	0.01448	0.02664	0.03576
TL	0.03915	0.09206	0.14074	0.01305	0.01527	0.0146	0.01362	0.02544	0.03537
TP	0.04444	0.09524	0.13968	0.01484	0.0163	0.01676	0.01667	0.02795	0.03719
C	0.04868	0.12593	0.17566	0.01534	0.01856	0.01939	0.01729	0.03332	0.04497
S-C	0.04656	0.11852	0.17143	0.01339	0.01714	0.01823	0.01649	0.03204	0.04367
TL-C	0.04444	0.12169	0.17566	0.01264	0.01688	0.01803	0.01626	0.03203	0.04419
TE-C	0.04021	0.12169	0.1746	0.01298	0.01728	0.01834	0.01538	0.03232	0.04405
TP-C	0.04444	0.11852	0.17037	0.01316	0.01709	0.01819	0.01594	0.03181	0.04344

V CONCLUSION

In this paper, the aim was to estimate the implicit social influences extracted from the history of users' actions on items. Indeed, this information is much more frequent in digital platforms than explicit trust information provided by users themselves. What's more, the estimation of this social influence had to consider time and had to vary according to item categories, since the fact that one user $u1$ replicates the behavior of another user $u2$ over time for all or certain item categories, is proof of $u2$'s influence on $u1$.

After estimating these different social influences, we integrated them into K-nearest neighbor, Matrix Factorization and Graph recommendation systems, following the principles of the trust-based recommendation systems we studied. Experiments carried out on two datasets, Epinions

and Ciao, show that taking into account the temporal aspect and item categories in the estimation of implicit social influences contributes significantly to improving the performance of these recommender systems.

Specifically, we obtained an improvement of 38.45% for the Hit Ratio, 67.62% for the MAP and 47.27% for the NDGC compared to the basic graph models with the Ciao dataset. Similarly, we obtained an improvement of 19% for the Hit Ratio, 10.34% for the MAP and 18% for the NDCG compared to the basic graph models with the Epinions dataset. For the K-nearest neighbor models and matrix factorization, we go from a *RMSE* of 1.02394 (for the base case) to 0.95384 for KNN and from 0.97938 to 0.97933 for matrix factorization with the Ciao dataset. For Epinions, we have an improvement from 1.17958 to 1.06226 for KNN.

As a perspective of this work, we plan to use implicit social influence propagation algorithms exploiting the transitivity of social influence to obtain a much denser implicit social influence matrix. In addition, we plan to integrate social influences into other models such as support vector machines (SVMs) and neural networks. Another perspective of this work would be to conduct experiments on other datasets such as Movielens, Pompare and RetailRocket.

REFERENCES

[1] J. D. Lewis and A. Weigert. "Trust as a social reality". In: *Social forces* 63.4 (1985), pages 967–985.

[2] M. Balabanović and Y. Shoham. "Fab: content-based, collaborative recommendation". In: *Communications of the ACM* 40.3 (1997), pages 66–72.

[3] R. J. Mooney and L. Roy. "Content-based book recommending using learning for text categorization". In: *Proceedings of the fifth ACM conference on Digital libraries*. 2000, pages 195–204.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pages 285–295.

[5] R. R. Sinha, K. Swearingen, et al. "Comparing recommendations made by online systems and friends." In: *DELOS* 106 (2001).

[6] G. Adomavicius and A. Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". In: *IEEE transactions on knowledge and data engineering* 17.6 (2005), pages 734–749.

[7] J. A. Golbeck. *Computing and applying trust in web-based social networks*. University of Maryland, College Park, 2005.

[8] C.-N. Ziegler and J. Golbeck. "Investigating correlations of trust and interest similarity-do birds of a feather really flock together". In: *Decision Support Systems* 42.3 (2005), pages 1111–1136.

[9] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. "Video suggestion and discovery for youtube: taking random walks through the view graph". In: *Proceedings of the 17th international conference on World Wide Web*. 2008, pages 895–904.

[10] F. E. Walter, S. Battiston, and F. Schweitzer. "A model of a trust-based recommendation system on a social network". In: *Autonomous Agents and Multi-Agent Systems* 16 (2008), pages 57–74.

[11] N. Lathia, S. Hailes, and L. Capra. “Temporal collaborative filtering with adaptive neighbourhoods”. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 2009, pages 796–797.

[12] C. Castelfranchi and R. Falcone. *Trust theory: A socio-cognitive and computational model*. John Wiley & Sons, 2010.

[13] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[14] F. Ricci, L. Rokach, and B. Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, 2011, pages 1–35.

[15] J. Tang, H. Gao, and H. Liu. “mTrust: Discerning multi-faceted trust in a connected world”. In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pages 93–102.

[16] J. Tang, H. Gao, X. Hu, and H. Liu. “Exploiting homophily effect for trust prediction”. In: *Proceedings of the sixth ACM international conference on Web search and data mining*. 2013, pages 53–62.

[17] M. A. Abbasi, J. Tang, and H. Liu. “Trust-aware recommender systems”. In: *Machine Learning book on computational trust, Chapman & Hall/CRC Press* (2014).

[18] G. Guo, J. Zhang, D. Thalmann, A. Basu, and N. Yorke-Smith. “From ratings to trust: an empirical study of implicit trust in recommender systems”. In: *Proceedings of the 29th annual acm symposium on applied computing*. 2014, pages 248–253.

[19] J.-P. Mei, H. Yu, Z. Shen, and C. Miao. “A social influence based trust model for recommender systems”. In: *Intelligent Data Analysis* 21.2 (2017), pages 263–277.

[20] A. J. N. Nzeko’o, M. Tchuente, and M. Latapy. “A general graph-based framework for top-N recommendation using content, temporal and trust information”. In: *Journal of Interdisciplinary Methodologies and Issues in Sciences* 5 (2019).

A ACKNOWLEDGEMENTS

The success of this research project was made possible by the generosity and support of many people. We would like to express our gratitude to all the members of our working team, specifically the High Performance Data Science (HIPERDAS) team, who actively participated in stimulating exchanges and shared valuable ideas. In addition, the collaborative environment established within our university community played a fundamental role in the progress of this research. In conclusion, we would like to sincerely thank all those who, directly or indirectly, contributed to the success of this work.