# CHESTNUT: Improve serendipity in movie recommendation by an Information Theory-based collaborative filtering approach

Xiangjun Peng, Hongzhi Zhang, Xiaosong Zhou, Shuolei Wang, Xu Sun and Qingfeng Wang

University of Nottingham
UK | CHINA | MALAYSIA

University of Nottingham Ningbo China, 199 Taikang East Road, Ningbo, 315100, Zhejiang, China.

First published 2019



University of Nottingham
UK | CHINA | MALAYSIA

# CHESTNUT: Improve Serendipity in Movie Recommendation by an Information Theory-based Collaborative Filtering Approach (Invited Paper)

Xiangjun Peng, Hongzhi Zhang, Xiaosong Zhou, Shuolei Wang, Xu Sun and Qingfeng Wang

User-Centric Computing Group, University of Nottingham Ningbo China
xu.sun@nottingham.edu.cn

**Abstract.** The term serendipity has been understood narrowly in the Recommender System. Applying a user-centered approach, user-friendly serendipitous recommender systems are expected to be developed based on a good understanding of serendipity. In this paper, we introduce ***CHESTNUT***, a memory-based movie collaborative filtering system to improve serendipity performance. Relying on a proposed Information Theory-based algorithm and previous study, we demonstrate a method of successfully injecting insight, unexpectedness and usefulness, which are key metrics for a more comprehensive understanding of serendipity, into a practical serendipitous recommender system. With lightweight experiments, we have revealed a few runtime issues and further optimized the same. We have evaluated ***CHESTNUT*** in both practicability and effectiveness, and the results show that it is fast, scalable and improves serendipity performance significantly, compared with mainstream memory-based collaborative filtering. The source codes of ***CHESTNUT*** are online at https://github.com/unnc-idl-ucc/CHESTNUT/.

**Keywords:** Serendipity, Recommeder Systems, Information Theory

## 1 Introduction

In an era of an increasing need for personalized recommendations, serendipity has become an important metric for achieving such a goal. Serendipitous recommender systems have been investigated and developed, to generate such results for their customers. Such systems can now be found in certain applications, such as in music recommendation [21].

However, as a user-centric concept, serendipity has been understood narrowly within the Recommender System field, and it has been defined in previous research as receiving an unexpected and fortuitous item recommendation [20]. The understanding of serendipity, as a user-centered concept, has been a gap for a while. Until recently, an awareness of this gap has led a conceptual bridge, which introduced serendipity from Information Research into Recommender Systems,

by proposing an Information Theory-based algorithm [36]. To further investigate this algorithm, it needs to be implemented as an end-to-end recommender system, but it is difficult to do so.

The challenges of transferring this conceptual bridge into a real-world implementation are two-fold. Firstly, it is demanding to inject the understanding appropriately, since the implementation may forfeit the algorithm design, to develop such a run-time system. Secondly, even though the implementation can recommend serendipitous information, it is demanding to ensure an overall enhanced user experience. For example, the overall system performance may compromise a user's experience, if the system response time is slow, since serendipity is a very sensitive feeling.

Thus, it is important, that serendipitous systems are designed with an accurate understanding of the concept, while delivering a high level of performance. Hence, we present ***CHESTNUT***, a state-of-the-art memory-based movie recommender system to improve serendipity performance. Whereas prior research has produced many serendipitous frameworks, it has focused on applying algorithmic techniques, rather than transferring a basic understanding of serendipity into the system development (Section 3).

We have addressed the issues of developing serendipitous systems by following a user-centered understanding of serendipity (Section 3) and focusing on runtime failures while making predictions (Section 5). Furthermore, we have optimized ***CHESTNUT*** by revisiting and updating **significance weighing** statistically to ensure a high level of system performance.

More specifically, we have made three main contributions here:

**(1) *CHESTNUT* Movie Recommender System.** ***CHESTNUT*** applies an Information Theory-based algorithm, which aims to combine three key metrics based on a user-centered understanding of serendipity: insight, unexpectedness and value [36]. With regard to these metrics, ***CHESTNUT*** has three key functional units, respectively: 1) **cInsight** performs the making connections to expand a target user's profile, to collect all target-user-related items (Section 3.1); 2) **cUnexpectedness** filtered out all expected items from all target-user-related items, with the help of a primitive prediction model (Section 3.2); and 3) **cUsefulness** evaluates the potential value of those candidate items through prediction, and generates a list of recommendations by sorting them from high to low (Section 3.3). In addition, while developing ***CHESTNUT*** we revealed key implementation details (Section 4). The source codes of ***CHESTNUT*** are online at https://github.com/unnc-idl-ucc/CHESTNUT/.

**(2) Optimizations of *CHESTNUT*.** Through system development, we observed that implementations following conventional methods could cause runtime failure in ***CHESTNUT***. We have formulated this problem (Section 5.1), and optimized ***CHESTNUT*** in two ways: First, we adjust the conventional designs while generating predictions for memory-based collaborative filtering techniques (Section 5.2); Second, we revisited the conventional optimization method, **significance weighting**, to further improve the performance and effectiveness of ***CHESTNUT***, with updates based on statistical analysis (Section 5.3).

**(3) Qualitative Evaluation of *CHESTNUT*.** We conducted an experimental study to assess the performance of ***CHESTNUT***, both a bare metal version and in optimized versions (Section 6). We have also benchmarked ***CHESTNUT*** with two mainstream memory-based collaborative filtering techniques, namely: item-based collaborative filtering and K-Neareset-Neighour user-based collaborative filtering from Apache Mahout. The results shows that ***CHESTNUT*** is **fast, scalable and extremely serendipitous**.

## 2 Background

***CHESTNUT*** is built on a series of works, which aimed to understand serendipity, to quantify serendipity in many use cases and to introduce serendipity understanding into the Recommender System (i.e. would be illustrated in detail further). We have also draw inspiration from the implementation and optimization of memory-based collaborative filtering techniques to enhance the system performance [9, 8, 27, 7].

Within the Recommender System field, serendipity has been understood as receiving an unexpected and fortuitous item recommendation [20]. Many efforts have been made in the development and investigation of serendipitous recommender systems [1, 2, 5, 6, 3, 10–12, 15, 14, 23, 24, 28, 29, 31–33]. Until recently, the main focus of the development of serendipitous recommender systems has centered on the algorithmic techniques that are being deployed, however, there are no existing systems which aim to bring an optional serendipitous user experience by applying a user-centered approach to the development of serendipitous recommender systems.

Unlike accuracy or other metrics, serendipity, as a user-centric concept, is inappropriate for taking this narrow view within this field. Understanding the serendipity has already raised considerable interest and it has been investigated for long in multiple disciplines [18, 19, 25, 30]. For instance, to better understand this concept, a number of theoretical models have been established to study serendipity [16, 17, 26]. More recently, previous research has highlighted how making connections is an important point for serendipitous engineering [13]. Based on previous research outcome from Information Research, an Information Theory-based algorithm has been proposed to better understand serendipity in the Recommender System [36]. Furthermore, a systematic context-based study among Chinese Scholars has been conducted and proves the effectiveness of the proposed algorithm [35].

This proposed conceptual bridge, which is based on a more comprehensive understanding of serendipity by merging **insight**, **unexpectedness** and **usefulness**, has been partly developed and studied in a movie scenario with early tryouts [34]. To bring together the above aspects, the system is expected to work sequentially in three steps, as follows: it first expands the user's profile by making connections; it then filters out unexpected items, according to the expanded profile and the original one; finally, it predicts ratings to calculate the value of all selected items to the target user, and then make appropriate recommendations.

However, it is still unclear how the proposed algorithm could be developed as an end-to-end recommender system in a real-world scenario, which is very practical, effective and suitable to deploy. Based on previous investigations, we have implemented ***CHESTNUT*** in a movie recommendation scenario. Below, we have presented a comprehensive overview of three major components to ensure and balance the three given metrics: **insight**, **unexpectedness** and **usefulness** (Section 3). In addition, we have presented the implementation details (Section 4) and optimization choices made during the development of ***CHESTNUT***, which have been employed to attempt to improve its reliability and practicality in the real world (Section 5).

## 3 *CHESTNUT* Overview

Before explaining the details of the implementation, we introduce the three major functional units of ***CHESTNUT***, which were developed consequentially with due consideration of the three metrics of serendipity mentioned above. There are three major functional units in ***CHESTNUT***: *cInsight*, *cUnexpectedness* and *cUsefulness*. These units function sequentially and ensure corresponding metrics, one by one.

### 3.1 cInsight

The design of *cInsight* aims to stimulate the making connections process, which is a serendipitous design from Information Research, to expand the profile of target users.

Details of the functional process of *making connections* are as followed. With the users' profiles uploaded[1], according to a *referencing attribute*[2], *making connections* would direct target users from their own information towards the most similar users in this selected attribute[3]. This whole process is denoted as a *level*. The repetition of this process, by starting from the output in the previous level, would finally end with an *active user* or a set of *active users*, when the similarity between *active user* and *target user* reaches the threshold.

*cInsight* is not parameter-free: there are two parameters which need to be set in advance. First, the *referencing attribute* should be determined as the metric for *making connections*, and it should be related information, such as side information categories[4]. Second, is the threshold to determine if the repetition shall end. Since more *levels* are formed by *making connections*, there is a larger distance between *active users* and *target users*. This threshold aims to make sure *active users* are not too far from the *target user*. Here, the thresholds could be the mathematical abstractions of similarity[5]. *cInsight* performed the *making*

[1] Those users denoted as *target users*
[2] Attribute(s) to guide making connections
[3] Those users denoted as *active users*
[4] In movie recommendations, for instance, it could be directors, genres and so on
[5] For example, Pearson Correlation Similarity, and so on

*connections* process by starting with the *target user* profile. The repetitions of multiple *levels* would terminate and form a direction from *target users* to *active users*. *cInsight* would finally re-organize all *active users'* profiles[6] for further processing. Here, assuming *referencing attribute* is **director** of movies, an example would be introduced as a brief explanation of *making connections* process:

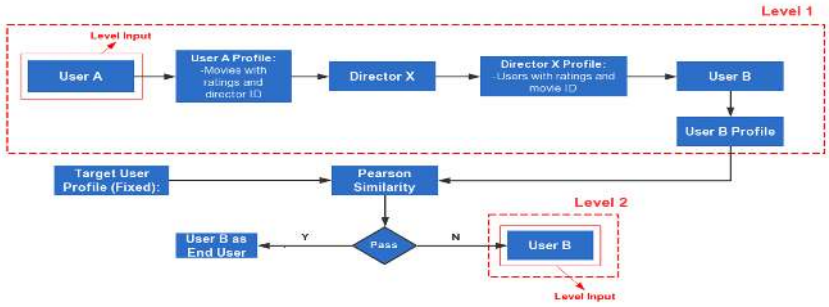


**Fig. 1.** An Example of the Connection-making Process

For a *Target user* who will be recommended with serendipitous information, *cInsight* works by analyzing his or her profile, and selects corresponding information from the profile as the starting point, which will depend on which attribute has been selected to reference.

As Figure 1 shows, the movie *Director D1*, who received the most movie ratings from *User A*, is selected as the attribute in this example. Then, according to *D1*, another *User B*, can be selected who is a super fan of *D1* and who contributes the largest number of movie ratings for *D1* throughout the whole movie database. If *User A* and *User B* satisfy the defined threshold on similarity, then *User B* is considered as the *active user* to recommend movies to *User A*. Otherwise, the algorithm continues to find another *User C*, by selecting another *Director D2*, on the basis of *User B*'s profile, until *User Z* is found to meet the threshold between *User Z* and the *Target user A*.

### 3.2 cUnexpectedness

After *cInsight*, all **relevant** items, generated by *making connections*, have been passed forward to *cUnexpectedness*. The design of *cUnexpectedness* aims to make sure all remaining items are indeed unexpected by the target user.

The functional process of *cUnexpectedness* proceeds in two steps, respectively. Firstly, it aims to identify what items a target user expects, based on a broader

[6] More specifically, their items

view of results from *cInsight*. Here, applying the primitive prediction model, *cUnexpectedness* expands the original target users' profiles into a **target-users-would-expect** profile. Secondly, based on the expected items generated by the first step, *cUnexpectedness* would remove all intersections between them and all items passed from *cInsight*[7]).

Here, we illustrate how the first step could be abstracted. The expected movie list ($EXP$) consists of two parts, namely those movies that could be expected by the users ($Eu$), and a primitive prediction model ($PM$) (e.g. those movies have been rated very high on average). And this are desribed in Equation (1).

$$EXP = Eu \cup PM \tag{1}$$

Through *cUnexpectedness*, items from *cInsight* have been confirmed as being unexpected by the *target user*, which satisfies the need of **unexpectedness**.

### 3.3 cUsefulness

Following the guarantees of *cInsight* and *cUnexpectedness*, the final unit is to identify which items are valuable to target users, so *cUsefulenss* has been developed to achieve this goal. To evaluate potential movies' value towards target user(s), generating prediction scores is the methodology applied in ***CHESTNUT***, conducted by *cUsefulness*. *cUsefulness* quantifies the value of each unexpected movie to target users by predicting how they would be rated by target users.

Since the development plan is collaborative-filtering based, the following equation, which is a conventional approach for prediction, is used to calculate the movie prediction score in *cUsefulenss*.

$$P_{a,i} = \bar{r_a} + \frac{\sum_{u \in U} (r_{u,i} - \bar{r_u}) \times W_{a,u}}{\sum_{u \in U} |W_{a,u}|} \tag{2}$$

In Equation (2), $\bar{r_a}$ and $\bar{r_u}$ are the average ratings for the user $a$ and user $u$ on all other rated items, and $W_{a,u}$ is the weight calculated by the similarity values between the user $a$ and user $u$. The summations are over all the users $u \in U$ who have rated the item $i$.

## 4 Implementation Details

After giving an overview of ***CHESTNUT***'s architecture and exploring the functionalities of the major components, in this section we will introduce some implementation details while developing ***CHESTNUT***, which enhanced the performance and practicality. ***CHESTNUT*** was developed in approximately 6,000 lines of codes in Java.

[7] Those items from *active users*, generated by the *target user*

### 4.1 Similarity Metrics

As for the similarity metrics, during the development of ***CHESTNUT***, *Pearson Correlation Coefficient* was selected as the similarity metric, which is described in Euqation (3).

$$W_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r_u})^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r_v})^2}} \tag{3}$$

In Equation (3), the $i \in I$ summations are over the items that both users $u$ and $v$ have rated, $r_{u,i}$ is the rating of *u-th* user on the *i-th* item and $\bar{r_u}$ is the average rating of the co-rated items of the *u-th* user.

### 4.2 cInsight

*cInsight* expanded its profile through the *connection-making* process, after collecting the user's profile, which relies on the *referencing attribute* from this target user. According to the number of movies rated by the user with respect to this very attribute and users' effective ratings, the most related ones[8] has been selected. With this selection, another user's profile could be generated which covers all the users that have rated movies, with this *referencing attribute*. Through sorting by the number of effective scores on this director from different users, the largest was chosen as the next user. This process would be repeated until the similarity between *target user* and selected user reached a threshold, which had been set in advance.

In ***CHESTNUT***, the *referencing attribute* has been set as director of movies, and the effective scores refer to those ratings above 4.0[9]. Moreover, this threshold has been set at 0.3[10]. These settings are based on cInsight-related studies previously [34].

### 4.3 cUnexpectedness

*cUnexpectedness* preserves the unexpected items by excluding those any expected items from all *active users'* items. Generating such expected items relies on the primitive prediction model.

In ***CHESTNUT***, through the primitive prediction model, *cUnexpectedness* expanded the *target user's* profile in two respects: first, it added all series movies, if any of those had appeared within the *target user's* profile. Second, it also added the Top Popular Movies.

As Figure 2 demonstrates, the work flow for generating the *target-user-expected* movies. While we implemented, we have specifically done in the following ways: for the first step, *cUnexpectedness* determines whether a movie belongs to a film series, by comparing their titles. To speed up this process, here

[8] information with regard to the *referencing attribute*

[9] In this rating scale, the full mark is 5.0

[10] Here, the similarity refers to *Pearson-Correlation Similarity*

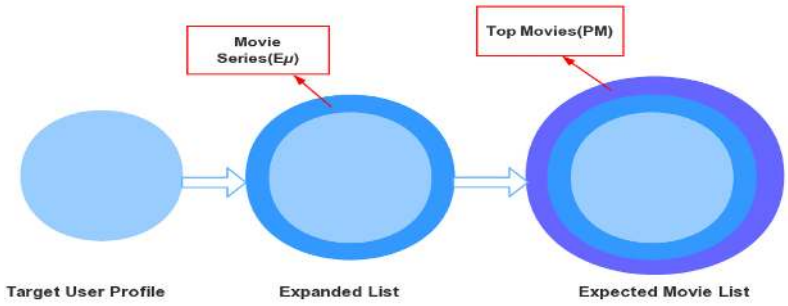


**Fig. 2.** Work Flow of Primitive Prediction Model

we applied a *dynamic programming approach*. In the second step, we selected *Top Two Hundred* because we observed that there is an obvious fracture in this very number, through sorting counts from high to low, based on the number of ratings have been given in the whole data set.

### 4.4 cUsefulenss

*cUsefulness* is responsible for examining the potential value of all movies, which have been filtered by *cUnexpectedness*. In the very first prototype development, *cUsefulness* functioned as the same as other memory-based collaborative filtering techniques, by exploring target users' neighbors, finding one with the most similarities and generating predictions according to the method mentioned in Section 3.3. However, through lightweight tests, we observed how this have caused run-time failures. We will discuss about it in Section 5.

### 4.5 User Interface

For user interactions, a website has been developed as a user interface for ***CHESTNUT***. After logging in, the user is able to view their rated movies, as shown in Figure 3. For each movie, the interface would offer an image of the movie poster, the title, the published year, the director and the rating from this user.

The follow-up pages, which enable users to view results and give feedback, are organized very similarly. However, when viewing the results, users are able to gather more information via their IMDB links (e.g. for more details or trailers), to present their own ratings, to answer the designed questionnaire and to leave comments.


**Fig. 3.** The User Interface

## 5 Optimization

In this section, we introduce some key insights for the related optimization of ***CHESTNUT***. Through lightweight tests, we found out that ***CHESTNUT*** could only produce one to two results for almost every user. To improve the system's overall performance and deployability, we optimized ***CHESTNUT*** by applying a new significance weighting and reforming the prediction mechanism. We first explored the problem, and then introduced them respectively.

### 5.1 Problem Formulation

After breakdown evaluations of each component in ***CHESTNUT***, we found that for every target user in the test set, only two to three items were predicted via *cUsefulness*, when the recommendation list was set to 1,000.

We believe this problem is two-fold. First, memory-based collaborative filtering relies on users' existing profiles to assist the prediction, and this method was directly conducted by searching co-rated items within the users' neighbors. However, with ***CHESTNUT***, neighbor users are very unlikely to have co-rated items. From our observations, almost every user could not be supported by their top two hundred neighbors in ***CHESTNUT***.

The second issue is more interesting. Owing to the characteristics of *Pearson-Correlation Coefficient*, the smaller the intersection between two users, the more the possibility that the value is higher. In other words, some similarities are not **trustworthy** and these led indirectly to ***CHESTNUT's*** runtime failures.

### 5.2 Mechanism Adjustment

Rather than searching a *target user's* neighbors from high similarity to low, *cUsefulness* applied a greedy approach to ensure the prediction process could proceed. Each time *cUsefulness* needs to make a prediction, it first selects all

users who have co-rated *need-to-predict* items. Then, within this group, *cUsefulness* cross-checks to find if there are any neighbors. If so, *cUsefulness* regroups and ranks from high to low, according to the similarity. With these settings, *cUsefulness* would proceed and make predictions for as many items as possible.

This mechanism adjustment demonstrated its benefits. First, it optimized the overall system performance. Since prediction is the most time-consuming element of ***CHESTNUT***, this adjustment ensured that the prediction would not reach a dead end, when finding predictable neighbors. Second, since it guaranteed the co-rated item in advance, it ensured that ***CHESTNUT*** would not have any runtime failures, caused by prediction interruptions.

However, this mechanism has intensified the formulated problem which mentioned previously. Since the computing sample size was smaller, owing to the features of serendipitous recommendation, the reliability of the similarity values would inevitably affect the overall recommendation quality.

### 5.3 Similarity Correction

We are not the first to recognize the necessity of similarity correction. Previous research has identified this kind of issue and has offered a solution known as **significance weighting** [8]. By setting a threshold, all similarity values, with fewer counts of co-rated items than this threshold, would divide a certain value to **correct** the value and maintain the exact similarity value.

In previous trials, **50** has been selected as the number for **significance weighting** to optimize the prediction process. However, in existing literature there is no explanation for how such a number has been obtained, and it appears to be a threshold obtained from previous experience. Since this threshold could be quite sensitive for the data set, we decided to explore and analyze its usage from a statistical perspective. As previously explained, the characteristics of Pearson-Correlation Coefficient could be too extreme when co-rated items are very limited (e.g. only one or two). Therefore, we have assumed the distribution shall be a normal distribution and we take advantage of the **Confidence Ratio** to illustrate this very problem.

All Pearson-Correlation values are computed and collected. All the values are then clustered and plotted on a new graph, with the average co-rated movie counters as y-axis and these values as x-axis. As shown in Figure 4, it is evident that this nonlinear curve can be fitted into a *GaussAMP model*, which illustrates that the global Pearson-Correlation Coefficients approximate a normal distribution.

Inspired by the **Confidence Ratio** in a Normal Distribution, we defined the quantity of edge areas as the *unlikelihood*. This *unlikelihood* aimed to quantify the unreliability of similarity values from global views. Based on the results presented in Figure 5, the Reliability, or the **Confidence Ratio**, could be abstracted as calculus mathematically. We then further selected four confidence ratios, in comparison with the initial value of **50**. According to the different ratios of the complete areas, determine the height reversely and apply into $M$ and calculate the corresponding $n$, Table 1 could be obtained:

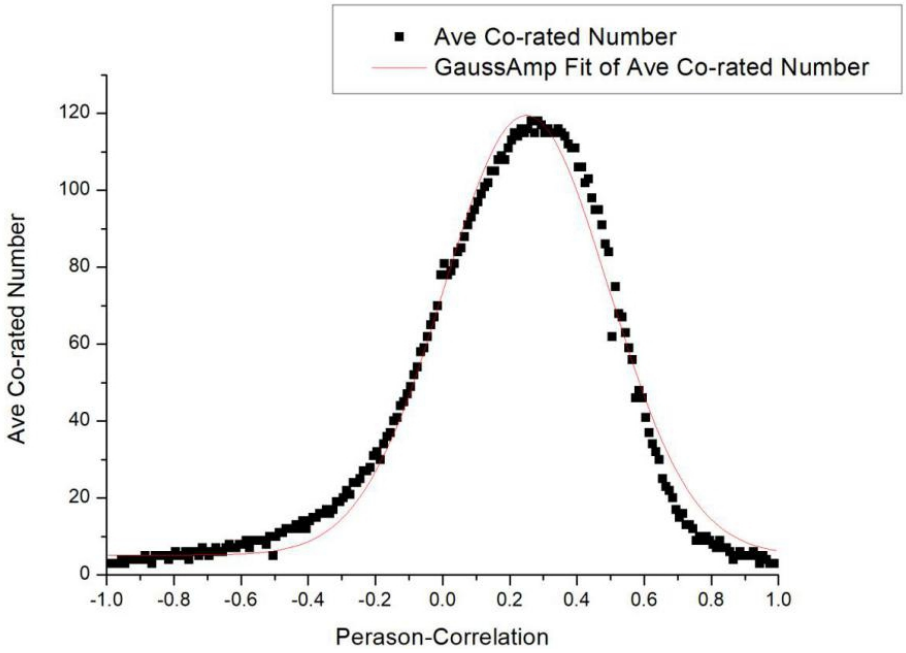


**Fig. 4.** Plotted Pearson-Correlation vs. Ave Co-rated Number

**Table 1.** The Average Number of Co-rated Items under Different Ratios

| Unlikehood | Confidence Ratio | Average Number of Co-rated Items |
|---|---|---|
| 1% | 99% | 5.2 |
| 5% | 95% | 8.5 |
| 10% | 90% | 19.25 |
| 20% | 80% | 42.5 |
| 22.5% | 77.5% | 50 |

We substituted the obtained results with the **significance weighting** respectively, and applied this similarity correction to improve the reliability of these values, in all related components of ***CHESTNUT***.

## 6 Experimental Study

In this section, we introduce details of ***CHESTNUT's*** experimental study. The *HeteRec 2011* data set was selected as the source data for this experimental evaluation. It contains *855,598* ratings on *10,197* movies from *2,113* users [4]. In addition, all users' k-nearest neighbors' data are also prepared in advance.

The experiment began by initializing the database and makes the supplement for information about directors of all the movies via a web crawler. Bearing in mind that some movies have more than one director, and there are no rules of distinction which are recognized by the public, only the first director was chosen during this process. After completion of the data preparation, ***CHESTNUT*** with different correction levels was run through each user in the database in turn.

Since ***CHESTNUT*** is a memory-based collaborative filtering system, to examine overall performances, we chose mainstream memory-based collaborative filtering techniques, namely: *item-based* and *user-based collaborative filtering* from ***Mahout*** as the benchmark [22].

All the implementations were conducted in Java and all the experiments were run on a Windows 10 Pro for Workstations based workstation Dell Precision 3620 with Inter Xeon E3-1225 processor (Quad Core 3.3GHz, 3.7Ghz Turbo, 8MB, w/ HD Graphics P530) and 32GB of RAM (4X8GB, 2400MHz, DDR4).

Our experimental study aimed to answer the following three questions:

(1) How much performance improvement can be achieved with ***CHESTNUT***, compared with mainstream memory-based collaborative filtering techniques?

(2) How many performance benefits have been gained with ***CHESTNUT***, when different optimization levels are deployed?

(3) What tradeoffs are caused if ***CHESTNUT*** is optimized with **significance weighting**?

### 6.1 Recommendation Performance

We first demonstrated that ***CHESTNUT*** can significantly improve the **unexpectedness** of recommendation results and while maintaining its scalability. For this purpose, we varied the number of items in the recommendation lists from 5 to 1000, and each time increased the number by 5. As shown in Figure 5, ***CHESTNUT*** could perform unexpectedness between 0.9 and 1.0. However, *item-based* and *user-based collaborative filtering* could only perform unexpectedness within the ranges 0.75 to 0.8 and 0.43 to 0.6 respectively. This is because unexpectedness was one of the major goals set during the design and development of ***CHESTNUT***.

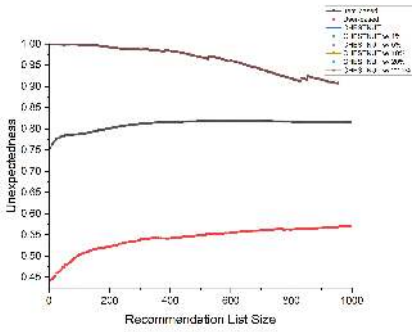


**Fig. 5.** Levels of Unexpectedness in ***CHESTNUT*** and for benchmarks

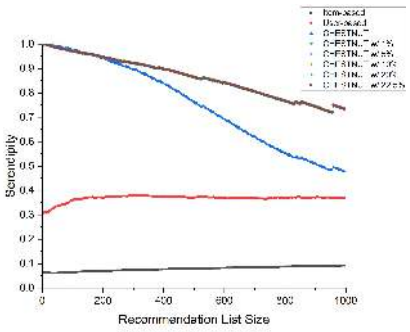


**Fig. 6.** Levels of Serendipity in ***CHESTNUT*** and for benchmarks

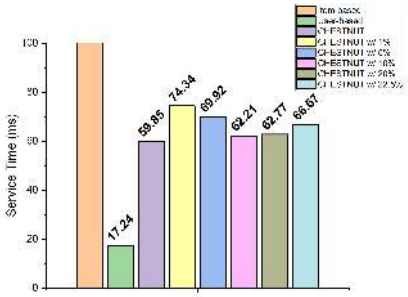


**Fig. 7.** Service Time of ***CHESTNUT*** and for benchmarks

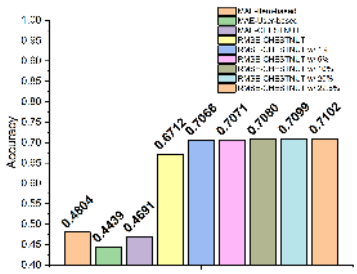


**Fig. 8.** Levels of Accuracy in ***CHESTNUT*** and for benchmarks

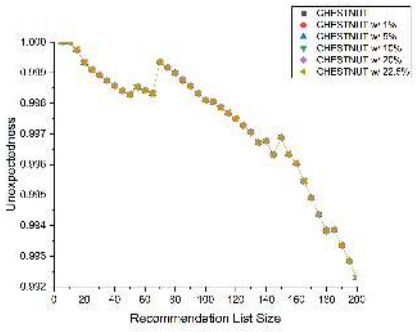


**Fig. 9.** Unexpectedness Breakdown within ***CHESTNUT***

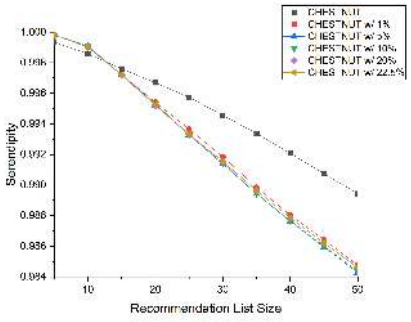


**Fig. 10.** Serendipity Breakdown within ***CHESTNUT***

Figure 6 shows that ***CHESTNUT*** could continue its dominant performance in serendipity, which follows the same experiment settings. As benchmark systems, *item-based* and *user-based* systems perform serendipity within the ranges of 0.05 to 0.08 and 0.3 to 0.4, respectively. Nevertheless, ***CHESTNUT*** still outperformed these conventional systems in serendipity performance. There are two interesting observations within this series of experiments. One is that, although the *item-based* approach could produce more unexpected results than the *user-based*, the *user-based* approach provided more serendipitous recommendations.

The other interesting fact is that serendipity performance degraded gradually, when applying ***CHESTNUT*** without optimization. However, optimized versions of ***CHESTNUT*** performed better scalability. More details of this observation, will be discussed in Section 6.2.

As for time consumption, more details are provided in Figure 9. It is necessary to highlight that, in the *item-based case*, approximately 10,000 milliseconds were required, on average. However, the *user-based* approach did achieve very good performance, by consuming 17.24 milliseconds on average. As for ***CHESTNUT***, although it is slightly slower than the user-based approach, it is still much faster than *item-based* implementation. All versions of ***CHESTNUT*** could finish the service between 59.85 and 74.34 milliseconds on average, which supports the assertion that ***CHESTNUT's*** performance is very competitive.

Finally, yet importantly, we have explored the accuracy of the recommendation results among the three systems. As their design goals, *item-based* and *user-based* approaches achieved 0.4804 and 0.4439 in **MAE**, which implies that they produce quite accurate results. However, for ***CHESTNUT***, the results, irrespective of whether they are with or without the optimization, are less accurate than the benchmark systems.

### 6.2 Performance Breakdown

Based on Section 6.1, we observed the necessity to explore a performance breakdown analysis. We first examined the **unexpectedness** evaluations in detail. Different from previous settings, we took a closer view of **unexpectedness** performance, by narrowing the recommendation list size from 5 to 1,000 to 5 to 200. The most interesting observation is that, unexpected results were irrelevant to the optimization levels of ***CHESTNUT***. As Figure 9 shows, although there are variations in this metric, **unexpectedness** still remains over 0.992. However, we have found that **significance weighting** did not affect the **unexpectedness** performance at all, which indicates that the levels of optimization did not affect the performance of *cInsight*. This is because the threshold in *cInsight* served as the lower bound[11], and our optimization mainly aims to correct any extremely high similarities, which are caused by too small an intersection size between users.

However, optimizations do play a role in *cUsefulness*. To examine this in more detail, we maintained a very narrow view by setting the recommendation

[11] When the value is less than it, *making connections* terminates

list size from 5 to 50. It has been observed that when a recommendation list size is smaller than 15, all optimized versions produce more serendipitous results than in the original version, although they were already very serendipitous. When the size is between 15 to 50, the situation was reversed. However, if we combined Figure 10 with Figure 6, the overall scalability of ***CHESTNUT*** is much weaker than the optimized versions.

This performance variation could be explained from two aspects. Since ***CHESTNUT*** could only make predictions within a small group compared to the other systems, and when there was no optimization, the predictions could be virtually high and this led to an obvious drift, as illustrated in Figure 6 (the blue line). We believe that the most important benefit of optimization is that, it **stabilizes serendipity performance and improves the scalability of the whole system**, by improving the reliability of the similarity values.

### 6.3 The Tradeoff Caused by Optimization

Here, we have mainly focused on the tradeoff caused by Similarity Correction, since the other optimization aims to make ***CHESTNUT*** runnable. There are two main tradeoffs to discuss about.

First, there are some runtime overheads when values are corrected. As Figure 9 shows, all optimized versions have a slight increase in the service time. As for the variations within these optimized versions, this is because if the correction rate were too high or too low, it would increase the computation difficultly and then cause overheads.

Second, we observed a very interesting situation. In the early investigations of **significance weighting**, researchers claimed that this approach was able to improve the accuracy of recommendations, and further investigation has supported that this very setting is effective [9, 8, 7]. However, optimized versions of ***CHESTNUT*** has conflicted with this. Figure 8 reveals a slight trend of accuracy loss, when the optimized levels were increased. We believe this is because of ***CHESTNUT's*** characteristics. What has been improved, via this optimization, is the trustworthiness of the similarity values. Unlike accuracy-oriented systems, it cannot be equal to the accuracy in serendipitous systems.

## 7 Discussion

Our experimental study revealed two main points for further discussions. First, ***CHESTNUT*** **has been proven that it is applicable to deploy the Information Theory-based algorithm, as an end-to-end recommender system which can induce serendipitous recommendations.** Especially, while the recommendation size is less than 50, ***CHESTNUT*** has dominated the serendipity performance, with close to the upper bound in evaluations. **Second, during the system implementation, it has been observed that** ***CHESTNUT*** **still needs optimizations via value corrections, to improve overall recommendation quality.** Through revisiting and updating

**significance weighting** concepts, ***CHESTNUT*** has been optimized to improve the overall scalability and serendipitous recommendation performance, because of the reliability of similarity values has been improved greatly.

## 8 Conclusion and Future Work

In this paper, we have presented ***CHESTNUT***, a state-of-the-art memory-based collaborative filtering system that aims to improve serendipitous recommendation performance in the context of movie recommendation. We implemented ***CHESTNUT*** as three main functional blocks, corresponding to the three main metrics of serendipity: insight, unexpectedness and usefulness. We optimized ***CHESTNUT*** by revisiting and updating a conventional method significance weighting, which has significantly enhanced the overall performance of ***CHESTNUT***. The experimental study demonstrated that, compared with mainstream memory-based collaborative filtering systems, ***CHESTNUT*** is a fast and scalable system which can provide extremely serendipitous recommendations. To the best of our knowledge, ***CHESTNUT*** is the first collaborative system, rooted with a serendipitous algorithm, which was built on the user-centred understanding from Information Researchers. Source codes of ***CHESTNUT*** is online at https://github.com/unnc-idl-ucc/CHESTNUT/.

The future work of ***CHSETNUT*** will focus on its extendibility. On the one hand, though ***CHESTNUT*** is not parameter-free, it wouldn't be difficult to extend into different usage context (e.g. shopping, mailing and etc.) since parameters of ***CHESTNUT*** could be obtained through our previous implementation experiences. On the other hand, as mentioned in Section 4, the *levels of connection-making* still rely on our previous experience and function as thresholds, which is the major limitation for system extension. We would further study ***CHESTNUT's*** effectiveness and its extendibility through a series of large-scale user studies and experiments.

## 9 Acknowledgement

We thank for valuable feedback and suggestions from our group members and anonymous reviewers, which have substantially improved the overall quality of this paper. This research is generously supported by National Natural Science Foundation of China Grant No. 71301085 and Hefeng Creative Industrial Park in Ningbo, China.

## References

1. Zeinab Abbassi, Sihem Amer-Yahia, Laks V. S. Lakshmanan, Sergei Vassilvitskii, and Cong Yu. Getting recommender systems to think outside the box. In *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pages 285–288, 2009.
2. Panagiotis Adamopoulos and Alexander Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM TIST*, 5(4):54:1–54:32, 2014.
3. Upasna Bhandari, Kazunari Sugiyama, Anindya Datta, and Rajni Jindal. Serendipitous recommendation for mobile apps using item-item similarity graph. In *Information Retrieval Technology - 9th Asia Information Retrieval Societies Conference, AIRS 2013, Singapore, December 9-11, 2013. Proceedings*, pages 440–451, 2013.
4. Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 387–388, 2011.
5. Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Cataldo Musto. An investigation on the serendipity problem in recommender systems. *Inf. Process. Manage.*, 51(5):695–717, 2015.
6. Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 257–260, 2010.
7. Mustansar Ali Ghazanfar and Adam Prügel-Bennett. Novel significance weighting schemes for collaborative filtering: Generating improved recommendations in sparse environments. In *Proceedings of The 2010 International Conference on Data Mining, DMIN 2010, July 12-15, 2010, Las Vegas, Nevada, USA*, pages 334–342, 2010.
8. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 230–237, 1999.
9. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. *SIGIR Forum*, 51(2):227–234, 2017.
10. Hiroaki Ito, Tomohiro Yoshikawa, and Takeshi Furuhashi. A study on improvement of serendipity in item-based collaborative filtering using association rule. In *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2014, Beijing, China, July 6-11, 2014*, pages 977–981, 2014.

11. Junzo Kamahara, Tomofumi Asakawa, Shinji Shimojo, and Hideo Miyahara. A community-based recommendation system to reveal unexpected interests. In *11th International Conference on Multi Media Modeling (MMM 2005), 12-14 January 2005, Melbourne, Australia*, pages 433–438, 2005.
12. Noriaki Kawamae. Serendipitous recommendations via innovators. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 218–225, 2010.
13. Genovefa Kefalidou and Sarah Sharples. Encouraging serendipity in research: Designing technologies to support connection-making. *Int. J. Hum.-Comput. Stud.*, 89:1–23, 2016.
14. Kibeom Lee and Kyogu Lee. Using experts among users for novel movie recommendations. *JCSE*, 7(1):21–29, 2013.
15. Kibeom Lee and Kyogu Lee. Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. *Expert Syst. Appl.*, 42(10):4851–4858, 2015.
16. Jiayi Luo and Rongjun Yu. Follow the heart or the head? the interactive influence model of emotion and cognition. *Frontiers in psychology*, 6:573, 2015.
17. Stephann Makri and Ann Blandford. Coming across information serendipitously - part 1: A process model. *Journal of Documentation*, 68(5):684–705, 2012.
18. Stephann Makri, Ann Blandford, Mel Woods, Sarah Sharples, and Deborah Maxwell. "making my own luck": Serendipity strategies and how to support them in digital information environments. *JASIST*, 65(11):2179–2194, 2014.
19. Lori McCay-Peet and Elaine G. Toms. Investigating serendipity: How it unfolds and what may influence it. *JASIST*, 66(7):1463–1476, 2015.
20. Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 1097–1101, 2006.
21. Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *New Frontiers in Artificial Intelligence, JSAI 2007 Conference and Workshops, Miyazaki, Japan, June 18-22, 2007, Revised Selected Papers*, pages 40–46, 2007.
22. Andrew Musselman. Apache mahout. In *Encyclopedia of Big Data Technologies.* 2019.
23. Kenta Oku and Fumio Hattori. Fusion-based recommender system for improving serendipity. In *Proceedings of the Workshop on Novelty and Diversity in Recommender Systems, DiveRS 2011, at the $5^{th}$ ACM International Conference on Recommender Systems, RecSys 2011, Chicago, Illinois, USA, 23 October 2011*, pages 19–26, 2011.
24. Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. TANGENT: a novel, 'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 657–666, 2009.
25. Sheila Pontis, Genovefa Kefalidou, Ann Blandford, Jamie Forth, Stephann Makri, Sarah Sharples, Geraint A. Wiggins, and Mel Woods. Academics' responses to encountered information: Context matters. *JASIST*, 67(8):1883–1903, 2016.
26. Victoria L. Rubin, Jacquelyn A. Burkell, and Anabel Quan-Haase. Facets of serendipity in everyday chance encounters: a grounded theory approach to blog analysis. *Inf. Res.*, 16(3), 2011.

27. Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295, 2001.
28. Markus Schedl, David Hauger, and Dominik Schnitzer. A model for serendipitous music retrieval. In *Proceedings of the 2nd Workshop on Context-awareness in Retrieval and Recommendation, CaRR 12, 2012, Lisbon, Portugal, February 14-17, 2012*, pages 10–13, 2012.
29. Giovanni Semeraro, Pasquale Lops, Marco de Gemmis, Cataldo Musto, and Fedelucio Narducci. A folksonomy-based recommender system for personalized access to digital artworks. *JOCCH*, 5(3):11:1–11:22, 2012.
30. Tao Sun, Ming Zhang, and Qiaozhu Mei. Unexpected relevance: An empirical study of serendipity in retweets. In *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013.*, 2013.
31. Maria Taramigkou, Efthimios Bothos, Konstantinos Christidis, Dimitris Apostolou, and Gregoris Mentzas. Escape the bubble: guided exploration of music preferences for serendipity and novelty. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 335–338, 2013.
32. Hisaaki Yamaba, Michihito Tanoue, Kayoko Takatsuka, Naonobu Okazaki, and Shigeyuki Tomita. On a serendipity-oriented recommender system based on folksonomy and its evaluation. In *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2013, Kitakyushu, Japan, 9-11 September 2013*, pages 276–284, 2013.
33. Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, February 8-12, 2012*, pages 13–22, 2012.
34. Xiaosong Zhou. *Understanding serendipity and its application in the context of information science and technology*. PhD thesis, University of Nottingham, UK, 2018.
35. Xiaosong Zhou, Xu Sun, Qingfeng Wang, and Sarah Sharples. A context-based study of serendipity in information research among chinese scholars. *Journal of Documentation*, 74(3):526–551, 2018.
36. Xiaosong Zhou, Zhan Xu, Xu Sun, and Qingfeng Wang. A new information theory-based serendipitous algorithm design. In *Human Interface and the Management of Information: Supporting Learning, Decision-Making and Collaboration - 19th International Conference, HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part II*, pages 314–327, 2017.