Chapter 5

Temporal Diversity in Recommender Systems

As we explored in the previous chapter, CF algorithms are often evaluated according to how *accurately* they predict user ratings [HKTR04]. However, as recommender systems grow dynamically, a problem arises: current evaluation techniques do not investigate the temporal characteristics of the produced *recommendations*. Researchers have no means of knowing whether, for example, the system recommends *the same items* to users over and over again, or whether the most *novel* content is finding its way into recommendations. The danger here is that, as results may begin to stagnate, users may lose interest in interacting with the recommender system.

In this chapter, we investigate one dimension of temporal recommendations: the diversity of recommendation lists over time. We first examine why temporal diversity may be important in recommender system research (Section 5.1) by considering temporal *rating patterns* and the results of an extensive user survey. Based on these observations, we evaluate three CF algorithms' temporal diversity from 3 perspectives (Section 5.2): by comparing the intersection of sequential top-N lists, by examining how diversity is affected by the number of ratings that users input, and by weighting-in the trade-off between accuracy and diversity over time. We finally *design* and *evaluate* a mechanism to promote temporal diversity (Section 5.3), comparing its performance to a range of baseline techniques. We conclude in Section 5.4 by discussing future research directions.

5.1 Why Temporal Diversity?

We explore the importance of temporal diversity from two perspectives: changes that CF data undergoes over time (Section 5.1.1) and how surveyed users responded to recommendations with varying levels of diversity (Section 5.1.2).

5.1.1 Changes Over Time

In Chapter 3, we performed an extensive analysis of how three rating datasets changed over time. In this section, we briefly summarise the main conclusions of this analysis, and how they relate to the direction we examine in this chapter.

1. **Data Growth.** Recommender systems grow over time: new users join the system and new content is added as it is released. Pre-existing users can update their profiles by rating previously unrated content; the overall volume of *data* thus grows over time. Section 3.2.1 shows the movie and user

growth across rating datasets; from these we see that there is a continuous arrival of both new users and movies.

2. **Summary Statistics Change.** As a consequence of the continuous influx of ratings, any summary *statistics* related to the recommender system's content may also change. These changes affect the ratings' summary statistics: in [Kor09a], Koren shows how *global* summary statistics vary over time. Similarly, Section 3.2.2 looks at how changes are further reflected in the global rating mean, median and mode. All the summary values fluctuate over time, reflecting how the overall distribution of ratings shifts as more users interact with the system.
3. **User Interaction.** Lastly, Section 3.2.3 looked at the rating frequency: these plots show the high variability in how users interact with the recommender system. Some users appear more frequently than others, and there is a large variance in the volume of rated items.

What do we learn from observing these changes? The datasets do not only remain incredibly sparse, but they also do not stabilise; recommender systems continuously have to make decisions based on *incomplete* and *changing* data, and the range of the changes we observe in the Netflix data have a strong impact on the predictability of ratings. Furthermore, the continuous rating of content means that the data that an algorithm will be trained with at any particular time is likely to be different than data it trained with previously. The question we explore in this chapter is: does the influx of new data translate to new content being recommended?

5.1.2 User Survey

In order to determine whether temporal diversity is important for recommender system *users*, we designed three surveys that *simulate* systems that produce *popular movie* recommendations over the course of five "weeks." We opted to recommend popular movies in order to avoid a variety of confounds that would emerge had we selected a personalised CF algorithm (e.g., the quality of the algorithm itself and the cumbersome process of asking users to rate films). Survey 1 (S1) and Survey 2 (S2) both recommended popular movies drawn from a list of the 100 all time most profitable box office movies[1]. S1, however, had *no diversity*: it consistently recommended the top-10 box office hits. S2's recommendations, instead, did change over time. Each week, approximately seven of the previous week's ten recommendations would be replaced by other movies in the top-100 box office list. Lastly, Survey 3 (S3) recommended movies that were randomly selected from the Netflix dataset: the recommendation process included full diversity, but was very unlikely to recommend popular movies, given the size of the dataset.

Each survey was structured as follows[2]. The users were first queried for demographic data. They were then offered the first week's recommendations, represented as a list of ten movie titles (and the relative DVD covers and IMDB links) and asked to rate these top-10 recommendations on a 1-5 star scale. After submitting their rating, they were presented with a buffer screen containing thirty DVD covers, and had to click to continue to the subsequent week; this aimed at diverting users' attention before

[1]http://www.imdb.com/boxoffice/alltimegross

[2]Full description and reproduction of the surveys is found in Appendix A

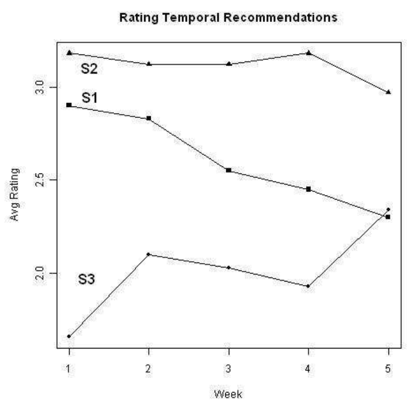


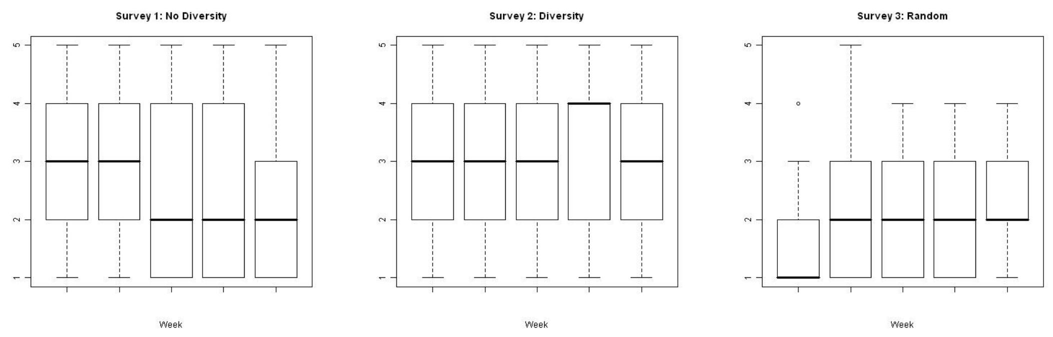
Figure 5.1: Survey Results for (S1) Popular Movies With No Diversity (S2) Popular Movies With Diversity and (S3) Randomly Selected Movies

presenting them with the next week's recommendations. After rating all five week's worth of recommendations, they were asked to comment on the recommendations themselves and answer a number of questions relating to diversity over time. Users were invited to participate in one or more of the surveys via departmental mailing lists and posts on social networks: S1 was completed 41 times, S2 had 34 responses, and S3 was completed 29 times. Due to the surveys' anonymity, we do not know how many users completed more than one survey. We therefore treat each completed survey individually. Of the 104 total responses, 74% of the users were male, 10% were 18-21 years old, 66% were 22-30 years old, and 24% were between 31 and 50 years of age. On average, the users claimed to watch 6.01 ± 6.12 movies per month, and while 61% of them said they were familiar with recommender systems, over half of them claimed they used them less than once a month. On the other hand, 29% use recommender systems weekly or daily: our respondents therefore include a wide variety of movie enthusiasts and both people who do and do not use recommender systems.

We averaged the users' ratings for each week's recommendations and plot the results in Figure 5.1. The S2 results (popular movies with diversity) achieve the highest scores: on average, these five weeks of recommendations were rated 3.11 ± 0.08 stars. The low temporal standard deviation reflects the fact that the rating trend remains relatively flat; the average for each week is about 3 stars. S3's results (randomly selected movies), were consistently disliked: the average rating peaks at 2.34 stars for week 5. In fact, some users commented on the fact that recommendations "appeared to be very random," "varied wildly" and the system "avoid[ed] box office hits." The main result of our surveys is reflected in S1's results (popular movies with no diversity): as the same recommendations are offered week after week, the average ratings *monotonically* decrease. The average for week 1 was 2.9, which falls within the range of values measured in S2, while by week 5 the average score is 2.3, which is lower than the average score for the random movies that same week. Not all users commented on the lack of recommendations diversity; however, most of them modified their ratings for the recommendations as the lack of diversity persisted. This shows that when users rate they are not only expressing their tastes or preferences; they are also responding to the impression they have of the recommender system. In the

Week	P-Value	S1 vs. S2	S1 vs. S3	S2 vs. S3
1	3.209e-6	0.32	7.4e-5	5.7e-6
2	0.003699	0.3003	0.0277	0.0033
3	0.002241	0.0881	0.0881	0.0015
4	0.0006937	0.0302	0.0943	0.0005
5	0.04879	0.07	0.88	0.10

Table 5.1: ANOVA P-Values and Pairwise T-Test Values For The 5 Weeks



(a) Survey 1 (b) Survey 2 (c) Survey 3

Figure 5.2: Boxplots of Each Week's Ratings for the Three Surveys

case of S1, users commented on the fact that the algorithm was "too naive" or "not working," and the lack of diversity "decreased [the respondent's] interest."

The final part of the surveys asked users about qualities they sought in recommendations: they had to give a rating reflecting how important they believed that accuracy, diversity, and novelty are in their recommendations. Overall, 74% said it was important for recommender systems to provide results that *accurately* matched their taste (23% selected the 'neutral' option to this question). 86% said it is important for recommendations to *change over time*; in fact, 95% stated it is important that they are offered new recommendations. It thus quickly becomes apparent that temporal diversity is a highly important facet of recommender systems, both in terms of the direct responses and rating behaviour of the surveyed users. In the following sections, we evaluate the temporal diversity of three state of the art CF algorithms.

Analysis of Variance

In order to test the statistical significance of the three different survey's results, we performed an analysis of variance (ANOVA): in this case, the null hypothesis is that the ratings for each survey are of the same distribution. We can reject the null hypothesis with 99% confidence with p-values less than 0.01: the p-value we measured for the three methods is 9.72e -14. A pairwise t-test between each survey further shows that the ratings input for each survey cannot be attributed to the sampling of the study.

We also performed an ANOVA on each week's data, in order to observe the change that S1 went through as it was rated. The null hypothesis, in this case, would state that the differences in ratings are consistent throughout all weeks; the p-value tells us what the chances are of randomly sampled users providing the ratings we have collected. The p-values are shown in Table 5.1; they all remain lower than the 95% confidence threshold. The table also shows pairwise t-test p-value results, week by week. We expected to observe three patterns: (a) a sequence of values that reflected S1 and S2 *diverging* from each other (as users punish S1 for having no diversity), (b) a sequence of values reflecting a convergence between S2 and S3, as the survey with no diversity becomes rated as highly as the random recommendations, and (c) values depicting a continuing difference between S2 and S3. In other words, given the rating distributions of S2 and S3, that are different from each other, we expected to observe S1's distribution change from being similar to S2's to being similar to S3's. The boxplots in Figure 5.2 reflect these changes: they show the per-week distribution of ratings for each survey. Most notably, the distributions in S2 (Figure 5.2(b)) remain relatively consistent over the weeks and S1 (Figure 5.2(a)) decreases over time.

Over the first four weeks of data, we observed the pattern we expected. The p-value between S1 and S2 begins high (0.32) and monotonically decreases until week four (0.03). The p-value between S1 and S3, instead, begins low (7.4e-5) and monotonically increases to 0.09. Lastly, the p-value shared between S2 and S3 remains consistently below 0.01 (the 99% confidence threshold). However, in week five there is an abrupt change in our results: not only is the overall p-value now found at the border of the 95% confidence threshold, but all t-test values become very high. Most worryingly, the divergence between S1 and S2 is no more, and S2 and S3 now share a very high p-value. There are a number of factors that may have skewed our experimental results. The first relates to why S3 may have been rated more positively in week 5:

- **Random Recommendations.** Survey 3 recommended random movies; however, as the surveys were pregenerated, all users were given the same (albeit random) recommendations in S3. One of the potential problems here is that this particular set of selected movies may elicit the same response from many users. In other words, the factors that influence users' responses (i.e., selection of movies) remained constant to all users. In week five, we noted that some of the randomly selected movies were less unknown than movies selected in previous weeks (for example, the first recommendation was the movie Crash[3]). The problem here is that S3's distribution in week five (as can be observed in Figure 5.2(c)) changes, thus impacting any comparisons between the distributions of the three surveys.
- **Edge Effect.** The survey instructions told users that they would be rating five weeks' recommendations. By the time they reached week 5, they thus knew that they had nearly completed the survey—their rating behaviour may have changed at this point. Similarly, they may also be responding more positively in week 5 due to the heavily negative response that was given for week 4. If we consider the rating mode of each week, then S3's results show that the weeks that receive a majority of 1* ratings (week 1 and week 4) are always followed by a week with a higher mode.
- **Lack of Data.** One of the reasons we may be seeing these results is the fact that our surveys have not been answered by many people. In fact, one of the problems that we noted was that

[3]http://www.imdb.com/title/tt0375679/

many users would abandon the surveys, possibly since, as other users noted, they did not like the recommendations (or thought the system was not working). S1 was visisted 122 times, yet only completed 41 times: overall, all the surveys were fully completed by fewer than 1 in 2 visitors.

There are thus a number of factors that influence our ability to observe a five week trend when comparing the three surveys. However, these do not detract from the main result of the survey: users who are faced with non-changing recommendations tend not to only lose interest, but also to reflect their impatience with the system in the ratings that they input. Temporal diversity is therefore an important quality that all recommender systems should provide.

5.2 Evaluating for Diversity

Given the above, we now aim to examine how diverse CF algorithms are over time. We focus on CF algorithms; a *baseline*, where a prediction for an item is that item's mean rating, the *item-based k-Nearest Neighbour (kNN)* algorithm, and a *matrix factorisation* approach based on Singular Value Decomposition (SVD), as reviewed in Chapter 2. We chose these algorithms since they not only reflect state-of-the-art CF, but also each manipulate the rating data in a different way and may thus produce varying recommendations.

5.2.1 From Predictions to Rankings

All of the algorithms share a common theme: they produce predicted ratings that can then be used to recommend content. The idea is to use the predictions in order to generate a personalised ranking of the system's content for each user. However, it may be the case that items share the same predicted rating. For example, a number of items may all have 5-star predictions. In this case, the predicted rating alone is not conducive to a meaningful ranking. We solve this problem by introducing a *scoring function* to rank items, regardless of the model used to generate predicted ratings. The scoring function uses two pieces of information: the predicted rating, and the *confidence* in the prediction (i.e., number of data points used to derive it) as used in [MLG$^+$03]. Assuming a 5-star rating scale, we first subtract the scale mid-point (3 stars) from the prediction and then multiply by the confidence:

$$s_{u,i} = (\hat{r}_{u,i} - 3.0) \times confidence(\hat{r}_{u,i}) \tag{5.1}$$

This scoring function ensures that items with high prediction and confidence are promoted, and low prediction with high confidence are demoted (i.e., we use it on all predictions and not simply as a tie-breaker). For example, an item with a predicted 5 star rating, derived from 2 ratings, will be ranked lower than another item with a 4 star prediction based on 50 ratings. If two items had the same score, then we differentiated them based on their respective average rating date: the item that had been rated more recently is ranked higher. The greatest advantage of this method, as detailed in [MLG$^+$03], is the heightened *explainability* of recommendations.

5.2.2 Methodology

In order to examine the sequence of recommendations produced by a system, we explore CF algorithms that iteratively re-train on a *growing dataset*. Given a dataset at time t and a window size μ (how often the system will be updated), we train the algorithm with any data input prior to t and then *predict* and *rank all* of the unrated items for each user. The t variable is then incremented by μ, and the entire process is repeated, except that now the latest ratings become incorporated into the training data. In other words, at time t we generate a set of top-N lists—corresponding to the top-N recommendations each user would receive—in order to examine how the sequence of ranked items that we produce will vary as the system is updated. The main difference between this process and the one we used in Chapter 4 is that we predict *all* unrated items for each user (rather than only predicting items that will be rated in the next time window); this allows us to produce ranked lists of recommendations. This method includes a number of advantages: we test the algorithms as data grows (and view more than a single iteration of this process), making predictions based only on ratings that are currently available. We simulate the iterative update of deployed systems, and stay true to the order users input ratings.

Since users do not necessarily log-in consistently to the system, we cannot be certain that each top-N list would have been viewed by each user. We therefore only generate a top-N list for the users who will rate at least one item in time $(t + \mu)$; we assume that if the user is rating an item then they have logged into the system and are likely to have seen their recommendations. The benefit of this is that we compare the current recommendations to those that users are likely to have seen before. It remains possible that users viewed their recommendations without rating any content; however, given this uncertainty in the data, we only consider the scenario where there is evidence that the users have interacted with the system. In this chapter, we continue using the same 5 subsamples of the Netflix dataset (as used in Chapter 4) for cross-validation purposes.

5.2.3 Measuring Diversity Over Time

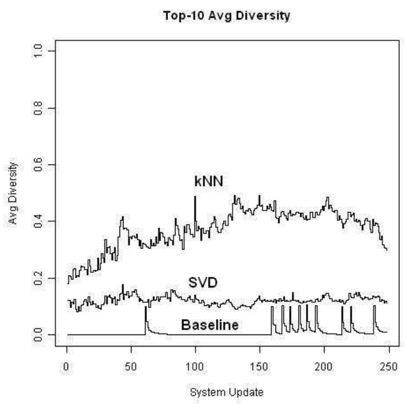
We define a way to measure the diversity between two ranked lists as follows. Assume that, at time t, a user is offered a set of 10 recommendations. The next time the user interacts with the system only 1 of the 10 recommendations is different. Therefore, the diversity between the two lists is $\frac{1}{10} = 0.1$. More formally, given two sets L_1 and L_2, the set theoretic difference (or relative complement) of the sets denotes the members of L_2 that are not in L_1:

$$L_2 \backslash L_1 = \{x \in L_2 | x \notin L_1\} \tag{5.2}$$

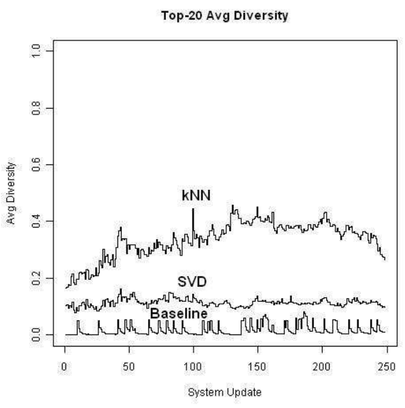
In our example above, only 1 of the 10 recommendations was not the same: the set theoretic difference of the two recommendation lists has size 1. We thus define the diversity between two lists (at depth N) as the size of their set theoretic difference divided by N:

$$diversity(L_1, L_2, N) = \frac{|L_2 \backslash L_1|}{N} \tag{5.3}$$

If L_1 and L_2 are exactly the same, there is no diversity: $diversity(L_1, L_2, N) = 0$. If the lists are completely different, then $diversity(L_1, L_2, N) = 1$. This measure disregards the actual ordering of the items: if a pair of lists are re-shuffled copies of each other, then there continues to be no diversity. However, we can measure the extent that recommendations change as a result of the same content being promoted or demoted by measuring diversity at varying depths (N).

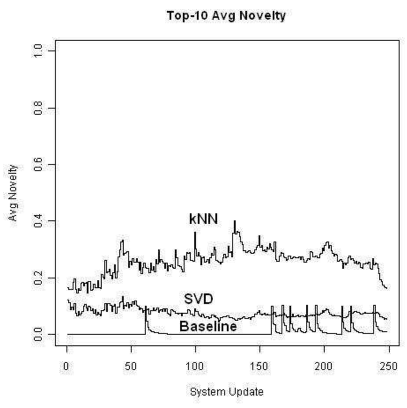


(a) Top-10 Diversity

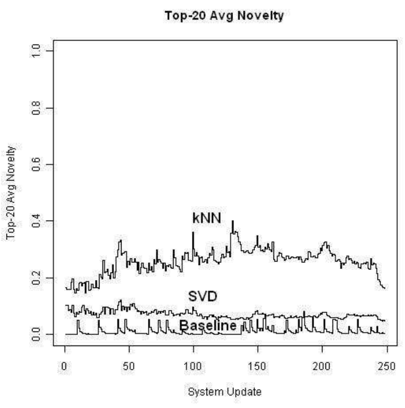


(b) Top-20 Diversity

Figure 5.3: Top-10 and 20 Temporal Diversity for Baseline, kNN and SVD CF



(a) Top-10 Novelty



(b) Top-20 Novelty

Figure 5.4: Top-10 and 20 Temporal Novelty for Baseline, kNN and SVD CF

One of the limitations of this metric is that it measures the diversity between two lists only; it thus highlights the extent that users are being *sequentially* offerered the same recommendations. In order to see how recommendations change, in terms of *new* items appearing in the lists, we define a top-N list's *novelty*. Rather than, as above, comparing the current list L_2 to the previous list L_1, we compare it to the set of *all* items that have been recommended to date (A_t):

$$novelty(L_1, N) = \frac{|L_1 \backslash A_t|}{N} \tag{5.4}$$

In this case, a list's novelty will be high if all of the items have *never* been recommended before, and low if all of the items have been recommended at some point in the past (not just in the last update). We further define the *average diversity* δ_t and *average novelty* η_t that is generated by a given CF algorithm (at time t) as the average of the values computed between all the current top-N lists and the respective previous list for all users.

5.2.4 Results and Analysis

We computed δ_t and η_t for each of the 3 algorithms over all 249 simulated system updates outlined in Section 5.2.2, and plotted the results for the top-10 and top-20 recommendations in Figure 5.3. These results provide a number of insights into recommender system temporal diversity. As expected, the baseline algorithm produces little to no diversity. On average, users' top-10 recommendations differ by (at most) one item compared to the previous recommendations. Both the factorisation and nearest neighbour approaches increment diversity; furthermore, the kNN algorithm is, on average, consistently more diverse than the sequence of recommendations produced by the SVD.

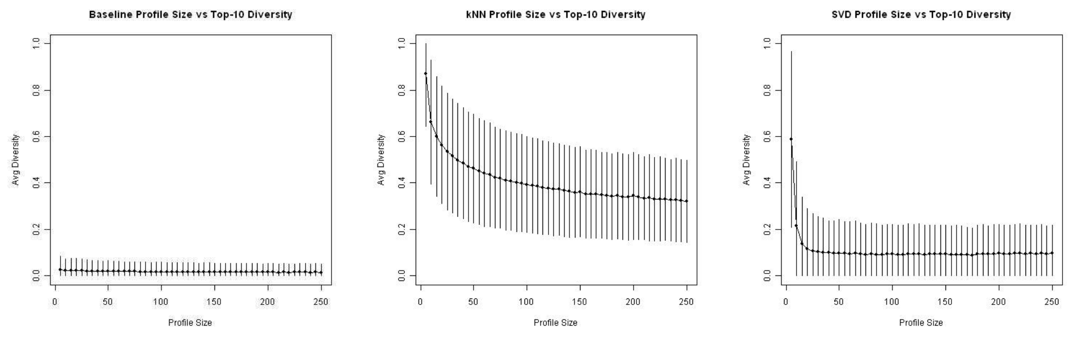
The novelty values (Figures 5.4(a) and 5.4(b)) are lower than the average diversity values. This means that, when a different recommendation appears, it is more often a recommendation that has appeared at some point in the past, rather than something that has not appeared before. There are a variety of factors that may cause this; for example, new items may not be recommended because they lack sufficient ratings: the CF algorithm cannot *confidently* recommend them. However, this metric does not tell us whether the new recommendations are new items to the system, or simply content that has (to date) not been recommended. A full analysis of the novelty of recommendation warrants a closer inspection of when items join the system and when they are recommened. In order to focus our analysis, we thus separate the problems of recommending *new content* from that of *diversifying* sequential recommendations: in this chapter, we focus on the latter.

Both Figure 5.3(a) and 5.3(b) also look very similar: the diversity values for the top-10 and top-20 recommendations are nearly the same. In order for this to happen (i.e., for a comparison between two top-10 lists and two top-20 lists to produce the same value) there must be *more* diversity between the larger lists. For example, if only 1 item changes in the top-10, the diversity is $\frac{1}{10} = 0.1$, and the pair of top-20 lists will only produce this diversity value if 2 items have changed, $\frac{2}{20}$. What this means is that not all of the changes in the item rankings are occuring in the top-10: new items are also being ranked between the $11th$ and $20th$ positions.

At the broadest level, we thus observe that (a) both the baseline and SVD produce less temporal diversity than the kNN approach, and (b) across all CF algorithms, diversity is never higher than approximately 0.4. However, these are averaged results across many users, who may be each behaving in very different ways: we now perform a finer grained analysis of temporal diversity to explore the relation between users and the diversity they experience.

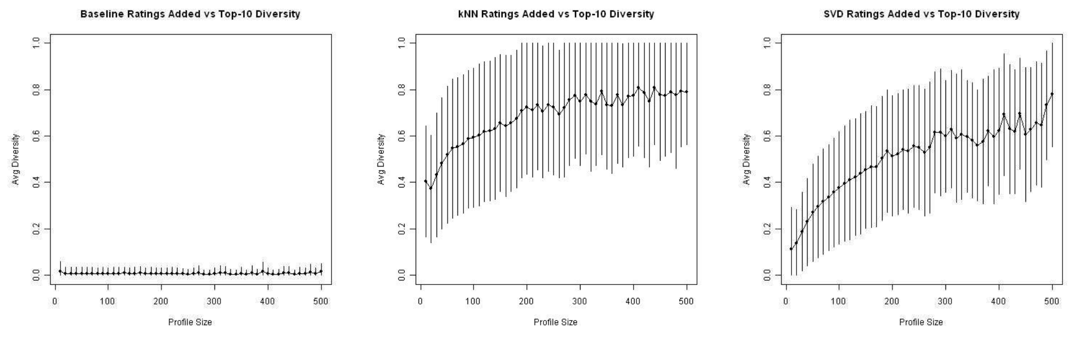
5.2.5 Diversity vs. Profile Size

The metric in Section 5.2.3 does not factor in the fact that the distribution of ratings per user is not uniform. Some users have rated a lot of items, while others have very sparse profiles. Users' *profile size* (i.e., the number of ratings per user) may affect their recommendation diversity. We thus binned the above temporal results according to users' current profile size and then averaged the diversity of each group. We plot the results in Figure 5.5. The baseline (Figure 5.5(a)) continues to show next to no diversity, regardless of how many items users have rated. The rationale behind this is that the only profile information that the baseline factors in when it computes recommendations is whether the user



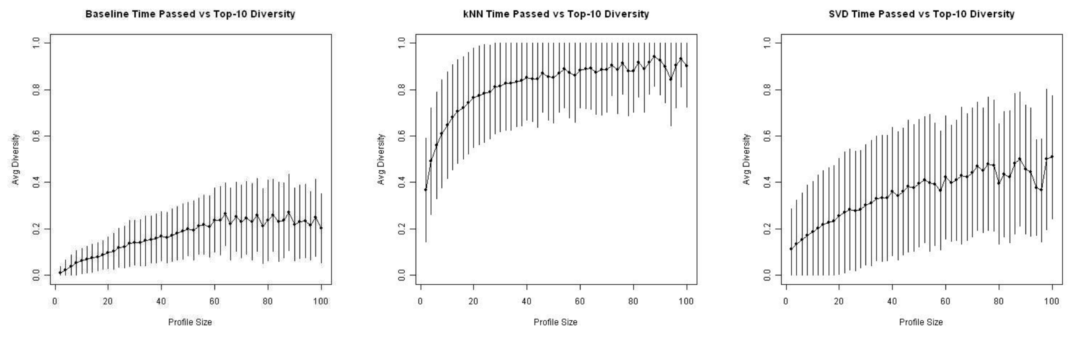
(a) Baseline (b) kNN (c) SVD

Figure 5.5: Profile Size vs. Top-10 Temporal Diversity for Baseline, kNN and SVD CF



(a) Baseline (b) kNN (c) SVD

Figure 5.6: Ratings Added vs. Top-10 Temporal Diversity for Baseline, kNN and SVD CF



(a) Baseline (b) kNN (c) SVD

Figure 5.7: Time Passed vs. Top-10 Temporal Diversity for Baseline, kNN and SVD CF

has rated one of the popular items; results will only be diverse if the user rates all the popular content. The kNN (Figure 5.5(b)) and SVD (Figure 5.5(c)) results, instead, show a negative trend: diversity tends to reduce as users' profile size increases. These results can be interpreted as follows: as users augment the set of ratings that represent their tastes, the breadth of items that are recommended to them via CF reduces, and they will be exposed to less and less new content.

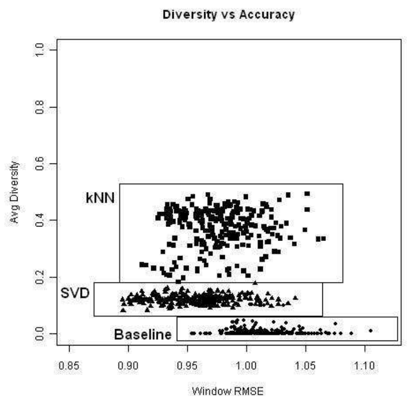


Figure 5.8: Comparing Accuracy with Diversity

5.2.6 Diversity vs. Ratings Input

Our temporal diversity metric is based on *pairwise* comparisons; we compare each sequential pair of top-N lists. One factor that may thus play an important role when determining how diverse a pair of lists will be from one another is *how much* the user rates in a given session. For example, one user may log in and rate two items while another may log in and rate fifty; the temporal diversity that each user subsequently experiences may be affected by these new ratings. We therefore binned users according to how many new ratings they input, and plot the results in Figure 5.6. As before, the baseline remains unaffected by how many new ratings each user inputs. The kNN (Figure 5.6(b)) and SVD (Figure 5.6(c)), instead, show a positive trend. These results can be interpreted as follows: the more you rate now, the more diverse your *next* recommendations will be.

5.2.7 Diversity and Time Between Sessions

The previous analysis was concerned with how diversity is influenced by a *single* user rating content. However, users do not rate alone: an entire community of users rate content over extended periods of time. We highlight this point with an example: some users may consistently log in and rate items every week; others may rate a few items now and not return for another month (and, in their absence, other users will have continued rating). In other words, diversity may be subject to the *time* that has passed from when one list and the next are served to the user. In order to verify this, we binned our diversity results according to the number of weeks that had passed between each pair of lists, and plot the results in Figure 5.7. In this case, all three of our algorithms show a positive trend: the longer the user does not return to the system, the more diversity increases. Even the baseline diversity increases: if a user does not enter the system for a protracted period of time, the popular content will have changed. However, web businesses tend to use recommender systems to *increase* user engagement and activity (e.g. clickthroughs, rented movies), and the natural diversification of recommendations because of time will only be useful for the least active members of the system.

5.2.8 Lessons Learned

Overall, *average temporal diversity* is low. Ranking content based on popularity offers next to no diversity, while the kNN method produces the largest average temporal diversity. Larger *profile sizes* negatively affect diversity; it seems that users who have already rated extensively will see the least diverse recommendations over time. Pairwise diversity between sequential lists is largest when users rate many items before receiving their next recommendations; users should be encouraged to rate in order to change what they will be recommended next. Diversity will naturally improve as users extend the *time between* sessions when they interact with the system (even popular content eventually changes).

A fundamental question to ask is how diversity relates to accuracy, the metric of choice in traditional CF research. To do so, we take the predictions we made at each update, and compute the Root Mean Squared Error (RMSE) between them and the ratings the visiting users will input. We then plot RMSE against average diversity in Figure 5.8. A plot of this kind has four distinct regions: low accuracy with low diversity (bottom right), high accuracy with low diversity (bottom left), low accuracy with high diversity (top right), and high accuracy with high diversity (top left). We find that the results for each algorithm cluster into different regions of the plot, corresponding to the different diversity results that they obtain. In terms of RMSE, different algorithms often overlap; for example, the kNN results sit between the two others—in terms of accuracy—and above them when considering diversity. However, kNN CF is sometimes less accurate than the baseline. The baseline sits toward the bottom right of the plot: it offers neither accuracy nor diversity. The SVD, on the other hand, tends to be more accurate than the baseline, although there is little diversity gain.

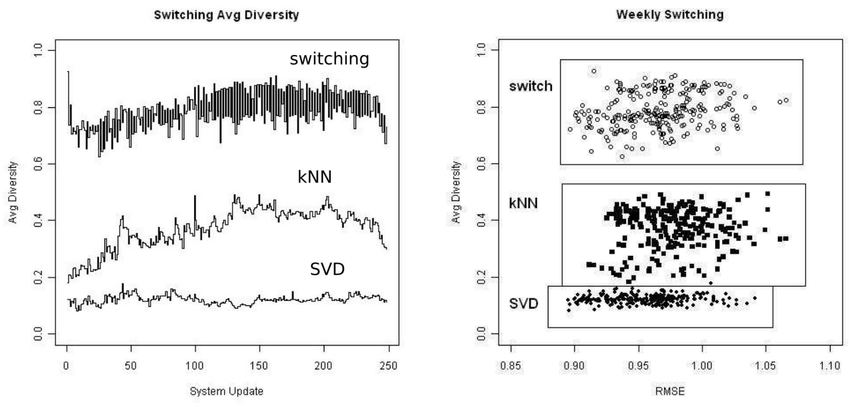
Coupling the low diversity that we have observed in CF algorithms and the high importance users place on temporally diverse recommendations implies that improving the temporal diversity of a recommender system is an important task for system developers. In the following section, we describe and evaluate a number of techniques that meet this goal: they increase temporally diversity, without significantly impacting recommendation accuracy. We then discuss the potential implications that modifying top-N lists may have to promote diversity.

5.3 Promoting Temporal Diversity

The easiest way of ensuring the recommendations will be diverse is to do away with predicted ratings and simply rank items randomly. However, *diversity* then comes at the cost of *accuracy*: recommendations are no longer personalised to users' tastes. The random survey (Section 5.1.2) showed that this is not a viable option, since the recommendations were rated very low. We can thus anticipate that, when promoting diversity, we must continue to take into account users' preferences. We do so with two methods: temporal hybrid switching, from a system (Section 5.3.1) and user (Section 5.3.2) perspective, and re-ranking individual users' recommendations (Section 5.3.3).

5.3.1 Temporal Switching

Many state of the art approaches to CF combine a variety of algorithms in order to bolster prediction accuracy [AT05]. However, as described by Burke [Bur02], another approach to building hybrid CF



(a) Switching Temporal Diversity (b) Switching RMSE vs Diversity

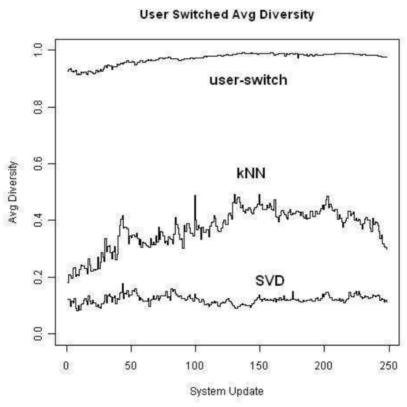
Figure 5.9: Diversity (a) and Accuracy (b) of Temporal Switching Method

algorithms is to *switch* between them. Instead of combining prediction output, a mechanism is defined to select one of them. The rationale behind this approach is as follows: a given a set of CF algorithms, that each operate in a different way, are likely to produce *different* recommendations for the same user; the top-N produced by a kNN may not be the same as that produced by an SVD. We thus switch between the two algorithms: we cycle between giving users kNN-based recommendations one week, and SVD-based recommendations the following week.

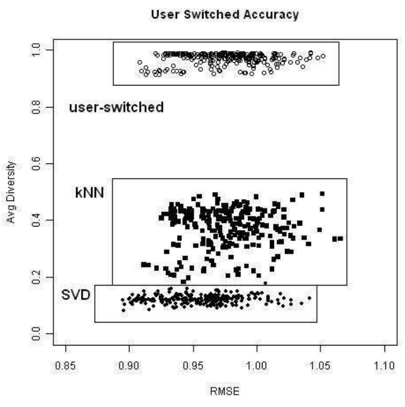
We plot the top-10 diversity over time for this switching method in Figure 5.9(a). Diversity has now been incremented to approximately 0.8: on average, 8 of the top-10 recommendations ranked for each user is something that was not recommended the week before. How does this affect accuracy? Intuitively, the overall accuracy that the system will achieve will be somewhere between the accuracy of each individual algorithm. We compare the accuracy and diversity of our switching technique in Figure 5.9(b). The results for the switching method now cluster into two groups; each group lies *above* the candidate algorithms we selected. In other words, accuracy fluctuates between the values we reported for kNN and SVD CF, but the fact that we are switching between these two techniques ensures that diversity has been greatly increased.

5.3.2 Temporal User-Based Switching

The method described in the previous section is very straightforward: the system changes the CF algorithm that is used from one week to the next in order to favour diversity. However, this method does not take into account how users behave; in particular, we previously noted that not all users have *regular* sessions with the recommender system. In fact, if their sessions were every other week, then the switching technique described in the previous section would be of no use at all. We therefore also tested a user-based switching algorithm. It works as follows: the system keeps track of *when* a user last appeared, and *what* algorithm was used to recommend content to that user during the last session. When the user reappears, the system simply picks a different algorithm to that which it used previously. As before, we switched between using an item-based kNN and an SVD-based approach in our experiments. The results are shown in Figure 5.10.

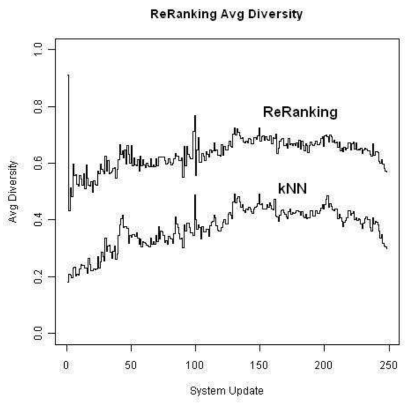


(a) Temporal Diversity

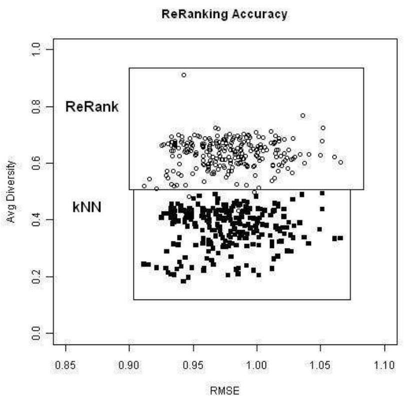


(b) RMSE vs. Diversity

Figure 5.10: Temporal Diversity and Accuracy vs. Diversity With User-Based Temporal Switching



(a) Temporal Diversity



(b) RMSE vs. Diversity

Figure 5.11: Temporal Diversity and Accuracy vs. Diversity When Re-Ranking Frequent Visitors' Lists

The temporal diversity (Figure 5.10(a)) is now near 1: on average, users are being offered different recommendations to those that they were shown the last time they interacted with the system. On the other hand, accuracy (Figure 5.10(b)) now falls between the kNN and SVD results. In other words, we sacrifice the low-RMSE of the SVD, but still do better than simply using the kNN approach: in return, the average diversity has been greatly amplified.

The only overhead imposed by user-based switching is a single value per user that identifies which algorithm was last used to compute recommendations; however, unlike the temporal switching method in the previous section, we are now required to compute both kNN and SVD at every update, albeit for a subset of users. We do not consider this to be an unsurmountable overhead, given that state of the art algorithms already tend to ensemble the results of multiple CF algorithms.

5.3.3 Re-Ranking Frequent Visitors' Lists

An immediate problem with a temporal switching approach is that it requires multiple CF algorithm implementations. In this section, we provide a means of diversifying recommendations to any desired degree of diversity when only a single CF algorithm is used.

One of the observations we made above is that users who have very regular sessions with the recommender system have low top-N temporal diversity. One way of improving overall average temporal diversity thus entails catering to the diversity needs of this group. To do so, we take advantage of the fact that they are *regular* visitors, and only re-rank their top-N recommendations.

The re-ranking works in a very straightforward manner: given a list that we wish to diversify with depth N (e.g., N = 10), we select M, with $N < M$ (e.g., M = 20). Then, in order to introduce diversity d into the top-N, we replace $(d \times N)$ items in the top-N with randomly selected items from positions $[(N+1)...M]$. In the case of $d = 1$, all elements in the first $[1...N]$ positions are replaced with elements from positions $[(N+1)...M]$. This is the method that we used to diversify the recommendations in the user survey S2 (Section 5.1.2); in that case, $N = 10$ and $M = 100$ (the 100 all time box office hits).

In our experiments, we opted to re-rank the top-10 results for any users who had previously visited the system less than two weeks before (recall that our system is updated weekly). The temporal diversity results, shown in Figure 5.11(a), clearly improves the overall average. Furthermore, the accuracy (Figure 5.11(b)) remains the same: the diversity has simply been shifted in the positive direction. However, how does this not hurt accuracy? There are three points to keep in mind: (a) we are only reranking the lists for frequent visitors, others' recommendations are untouched; (b) the items in the top-N are there due to both high prediction value and high confidence (there is a good chance the user will like those items); and (c) we do not promote items that are likely to be disliked by the user (by only re-ranking the top-M).

How do these techniques affect recommendation novelty? Recall that we defined novelty (Section 5.2.3) as proportional to the number of items being recommended that have *never* been recommended before. If we aggregate the temporal results of Figure 5.4(a), we find that the baseline top-10 recommends, on average, 13.53 $\pm$ 2.86 items over time; the SVD top-10 suggests 26.17 $\pm$ 12.51 items over time, and the kNN top-10 recommends the highest number of items over time: 79.86 $\pm$ 59.33. This ensures that kNN will also produce the highest number of *new* recommendations. Weekly switching slightly lowers kNN's average, to 75.36 $\pm$ 53.98 because repeatedly visiting the SVD recommendations reduces the number of total items that can be recommended. However, user based switching maintains the average number of recommended items over time at 79.86 $\pm$ 55.12; it highly promotes temporal diversity without impacting the number of new items that enter the top-10 list over time. However, re-ranking bolsters both the average and standard deviation to 97.93 $\pm$ 78.82; re-ranking thus seems like a promising approach to solving the related problem of temporal novelty in recommendation.

5.4 Discussion

Diversity is a theme that extends beyond recommender systems; for example, Radlinski and Dumais examine how it can be used in the context of personalised search [RD06]. In other cases, diversifying search results is done in order to reduce the risk of query misinterpretation [AGHI09]. Similarly, diversity relates to user satisfaction; more specifically, to users' impatience with duplicate results [HRT09]. We have observed similar 'impatience' in our survey: users who completed the survey with no diversity began to rate recommendations lower as they saw that they were not changing.

It is certainly possible to envisage a finer grained notion of diversity that takes semantic data into account—by measuring, for example, the extent that the same genre or category of items are being recommended. To that end, diversity may also be measured within a *single* top-N list, rather than a pair or sequence of recommendations; such a metric may, for example, take into account the number of highly related items (such as a movie and its sequels, or multiple albums by the same artist) that are being simultaneously recommended. For example, Smyth and McClave [SM01] apply strategies to improve recommender systems based on case-based reasoning; diversity, in this case, is viewed as the complement of similarity. Zhang and Hurley [ZH08] also focus on intra-list diversity, and optimize the trade off between users' preferences and the diversity of the top-N results. In this chapter, we focus on the temporal dimension (inter-list diversity) and whether the exact same items are being offered to users more than once; we do not take semantic relationships between the recommended items into account nor improve the diversity of individual top-N lists. However, both lines of research are not in conflict: ideally, one would like a recommender system that offers diverse results that *change* over time to suit each users' tastes.

There is one limitation to the work we have performed here. We do not know what users were *actually* recommended: in fact, we do not know if users are clicking on their recommendations or are selecting movies to rate by other means. Assuming that Netflix was not simply recommending popular content, this limitation may explain why the baseline results show so little diversity. While the results certainly show what one may expect from providing popularity-based recommendations, it is also possible to envisage higher diversity for the baseline case, if, for example, users dislike their recommendations so much that they are giving them all 1 star (in the next update they will be shown different results). However, this limitation is a widespread problem with CF research; in fact, to date the relationship between what people are *recommended* and what they *rate* remains largely unexplored.

5.5 Summary

This chapter focuses on temporal diversity: how recommendations change over time. In doing so, we have extended how CF can be evaluated over time; in Chapter 4, we focused on the accuracy of predictions; here we added the diversity and novelty of recommendation lists over time. We found that state of the art CF algorithms generally produce low temporal diversity; they repeatedly recommend the same top-N items to a given user. We then defined a metric to measure temporal diversity, based on the set theoretic difference of two sequential top-N lists, and performed a fine-grained analysis of the factors that may influence diversity. We found that, while users with large profiles suffer from lower diversity, those who rate a lot of content in one session are likely to see very diverse results the next time. We also observed that diversity will naturally improve over time. We then designed and evaluated three methods of improving temporal diversity without extensively penalising recommendation accuracy. Two were based on *switching* CF algorithm over time; users are first given recommendations produced with (for example) a kNN approach, and then offered the results of an SVD algorithm. The last method was based on re-ranking the results of frequent visitors to the system.