Web Science Exam Report

Andreas Bjerregaard
fvk220@alumni.ku.dk

Abstract

This project report investigates a variety of recommender systems and approaches for making offline Amazon software product recommendations. These approaches are subsequently evaluated and discussed along with the specific data and setting.

1 Introduction and data

This project is based on the 5-core Software subset of Amazon Review Data (2018), see Ni et al. (2019). The data is cleaned by dropping missing ratings and duplicates (by keeping the latest review in cases where a user has reviewed the same item multiple times). A test set is formed by extracting the latest positively rated item (rating ≥ 4) by each user. Lastly, users who do not appear in the training set are removed from the test set. After this processing, the 12805 original rows form a train/test split with 10171 and 1711 rows, respectively — with columns being rating, user ID, item ID, and review time. A collection of metadata compliments this review data, describing each product and includes, e.g., product title, description, price and brand. The metadata is filtered for items which do not occur in either the train or test split.

Since the test data only contains a single review (of which the goal is to include in a top-k recommendation cutoff), the real recommendation performance will be underestimated; essentially we predict the *next* relevant item rather than just *a* relevant item. Nonetheless, this split is more reasonable w.r.t. the temporal aspect of user preferences compared to hold-out strategies.

Figure 1 visualizes the distributions of mean rating by user and by item in the training set, which gives some intuition about the data. The mean rating per user has a median at 4.0 and mean 3.8 ± 2.2 (mean ± 2 SD) which shows more variance, while there generally is displayed some more agreement for means by item with a median at 3.7 and mean 3.6 ± 1.6.

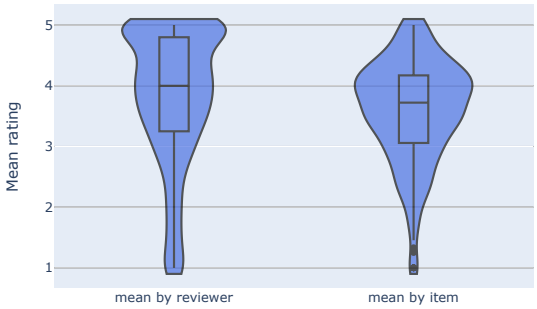

Figure 1: Violin plots with quartiles showing the distribution of mean rating by user and item, respectively, of the training data.

Figure 2 shows a histogram over the number of ratings per item in the training set ordered by decreasing frequency. Notably, it is a long tailed distribution with few high-frequently rated products; the most popular 40 items (5%) here corresponds to 21.6% of reviews. Because of this property, the ratings of the high-frequent items tend to define neighbourhoods, and providing robust predictions in the long tail becomes difficult; this is problematic since items in the long tail are especially important to recommender systems (Aggarwal et al., 2016, p. 32), typically having larger profit margins among other properties. Further, since users in general try to buy things they think they will like, selection bias is included (ratings are *missing not at random*) (Aggarwal et al., 2016, p. 251). While a review is not identical to a purchase, the available data entails this assumption and may skew the real user preferences. Further, fake reviews may be included.

2 Collaborative filtering systems

Using off-the-shelf `scikit-surprise` implementations of common Collaborative Filtering (CF) algorithms, a user-based neighbourhood model (kNNWithMeans) and a latent factor model (SVD) is fit to the training data. Hyperparameters for both

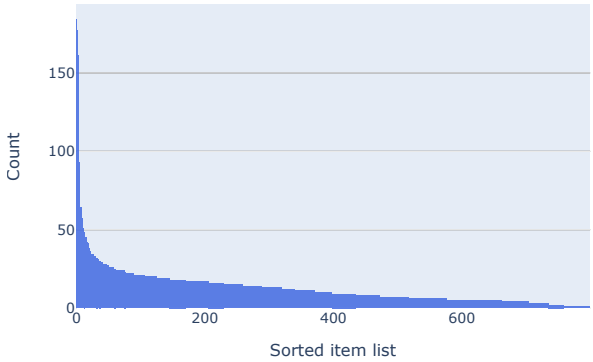

Figure 2: Histogram of item ratings with bins sorted by decreasing frequency. Each bin represents one item, and its size its count of ratings.

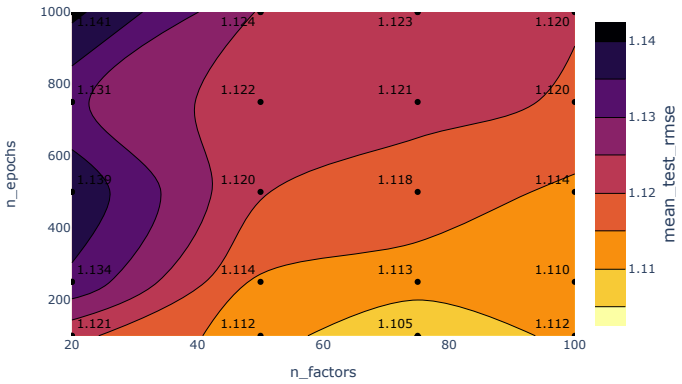

Figure 3: Contour map visualising mean validation RMSE over the SVD hyperparameter grid search.

models are tuned via 5-fold cross-validation[1] using Root Mean Square Error (RMSE). The optimal hyperparameters and corresponding validation errors are listed on Table 1. Appendix A shows the grid search for kNNWithMeans, and Figure 3 the grid search for SVD. In contrast to the kNN grid search, where similarity measures are discrete and we wish to avoid the sense of interpolation on the hyperparameters, the SVD grid search is here shown as a contour map, giving a sense of the distances between sampling points and how the error sizes relate to each other. Along with the dimensions shown on the contour map, different learning rates are also considered; a 3D scatter visualization is given in Appendix B. While the objective of the grid search is to obtain a generalizable model minimizing RMSE, training time is not considered. Note however — while not an issue here — that training time can be a serious concern in real applications with very large user databases. Appendix B additionally encodes training time with marker size.

In Section 5, the SVD model is found to outperform the neighbourhood-based model on all accuracy metrics. However, with the memory-based kNN approach, we have a simple, intuitive, and interpretable[2] model which is further relatively stable to the addition of new data. On the other hand, SVD uses dimensionality reduction to estimate the complete user-item matrix; it is more robust, efficient, and has a high level of coverage with similarities being computed in the reduced space (Aggarwal et al., 2016, p. 9). Nonetheless, latent factors are far more difficult to interpret — and hence the model as a whole.

2.1 Error analysis for neighbourhood-based CF

Next, an error analysis is done on two example users u_1, u_2; the first and latest in the test set sorted by time of review.[3] The model does not manage to get a hit on either user at cutoff 15. The rating histories of their 10 nearest neighbours (by Pearson correlation) are listed in Appendix C. For u_1, note how the 10 neighbours have no reviews for item `B0014KJ6EQ`, while they all have for the other item, `B0017I8NQM`. In the full training data, these items appear with frequencies 4 and 17, respectively. In this case, perhaps a better neighbourhood-assignment would attribute more importance to the rarer item. For such users with very few ratings, it is difficult to identify meaningful neighbours and the community-concept of CF becomes weak (Aggarwal et al., 2016, p. 24). Section 4 treats this issue by building a hybrid recommender system based on a switching strategy. Further, the two ratings given are equal (= 3), so not much information is available regarding user preference.

User u_2 has reviewed 7 items in the training set all during 2015–2016 yet his/her test item in 2018. The rating history of the 10 closest neighbours appear to largely share this temporal aspect (listed in Appendix C) — hence likely making it harder to make accurate predictions matching late entries in a time-sorted test set.

Next, we seek to get an idea about how user/item statistics relate to model performance. Instead of

[1] Using only three folds wrongly encourages fewer neighbours for kNN and, more generally, provides an unrealistic bad setting for CF.

[2] Note that since it is user-based rather than item-based, predictions are still not easily interpretable for the end user.

[3] IDs 'A2G0O4Y8QE10AE' & 'A2SACTIFMC5DXO'

Model	Val RMSE	Test RMSE	Optimal hyperparameters
kNNWithMeans	1.1971	1.1497	`k_neighbours: 4, measure: 'pearson'`
SVD	1.1048	0.9972	`n_epochs: 100, n_factors: 75, lr_all: 0.15`

Table 1: Results from grid search using 5-fold cross-validation. RMSE is averaged on the hold-out CV validation folds.

detecting users where $RR = 0$ (true for 1691 of 1711 users in the test set), users are instead found where $RR \neq 0$; this is meaningful as the NB-based CF algorithm performs rather badly (see Section 5) s.t. only a small subset of users is found. These users have on average rated 7.45 items (mean was 5.58 for all users), and the subset of items they have rated received on average 18.64 ratings (mean was 12.71 for all items). This follows the intuition that the power of communities increases as more ratings are given. Naturally, a consequence of this is that the rating counts are statistically smaller when the model performs badly. This effect is however biased as the popularity of the items should also be taken into account.

3 Content-based systems

3.1 Item representations

TF-IDF vector space Item representations are first made by vectorizing product titles using the `sklearn` implementation of the TF-IDF weighting scheme. Prior to this, a preprocessing pipeline is applied: removing HTML, transforming to lowercase, removing characters not in a–z, tokenizing, removing stopwords, stemming tokenized words, and removing single-letter words. The last step is otherwise done by the TF-IDF vectorizer. Applying this to the `title` column of the metadata (which has been filtered for items not present in either training- or test data, see Section 1) gives an array of size $(801, 646)$; this corresponds to 801 unique item IDs and 646 words in title vocabulary.

Word2Vec vector space Next, item encodings are made with a Word2Vec-based approach. Using the Python `gensim` library, a model pre-trained on a subset of the Google News dataset (about 100 billion words) is used, which subsequently contains 300-dimensional vectors for 3 million words and phrases. By consideration of the vocabulary entries, lemmatization is applied instead of stemming; at the end of cleaning, this results in 543 Word2Vec-indexable words. In using the pretrained model, the context of words do not have an influence on their

	ASIN	Title
Item1	B001DSGXFY	Acronis True Image Home 2009 [OLD VERSION]
Item2	B01HAP47PQ	Pinnacle Studio 20 Ultimate (Old Version)
Item3	B01HAP3NUG	Pinnacle Studio 20 Plus (Old Version)

Table 2: Three example products for illustration of similarity using different text representations.

	TF-IDF			Word2Vec		
	Item1	**Item2**	**Item3**	**Item1**	**Item2**	**Item3**
Item1	1.00	0.06	0.06	1.00	0.66	0.64
Item2	0.06	1.00	0.74	0.66	1.00	0.92
Item3	0.06	0.74	1.00	0.64	0.92	1.00

Table 3: Cosine similarity based on TF-IDF and Word2Vec representations of item titles from Table 2.

Word2Vec-embedding, and item representations are simply made as the mean word embedding of a title. This decision is grounded in that other well-performing document embedders (like Doc2VecC, see Chen (2017)) are based on learning the mean word embedding to describe documents.

Now, even if two product titles do not have an overlap in vocabulary, similarity can still be measured in a meaningful way. Denoting `f(x)` as the vector representation of `x`, then notice that for similarity of short titles, Word2Vec's property that e.g. `f('king') - f('man')` $\approx$ `f('queen') - f('woman')` (Mikolov et al., 2013) gives rise to far better embeddings. This is however not always an issue, as only few items are presented to the user, and a vocabulary overlap is more likely for such items.

To illustrate strengths and shortcomings of the representation techniques, three products are now picked from the metadata; these are listed in Table 2. From Table 3, it is seen that the two Pinnacle Studio products are similar in the vector spaces of both TF-IDF and Word2Vec. However, the Acronis product shares virtually no similarity to the others based on TF-IDF — even though both are photography editors and similar products overall. On the other hand, Word2Vec captures this similarity through word embeddings, and attributes a higher similarity score — but interpretability of the text

representation is largely lost.

Figure 4 shows the title representation matrices of all 801 unique items. Clearly, the TF-IDF representation is sparse. This is further a means to qualitatively assess the text preprocessing step. From the TF-IDF image, the dark vertical lines (at $x \in \{360, 600\}$) appear at features 'old' and 'version', respectively, which are common in software-jargon and could mislead similarity measures. For terms such as 'version', concatenating the following number may lead to an increase in accuracy, but is not explored further.

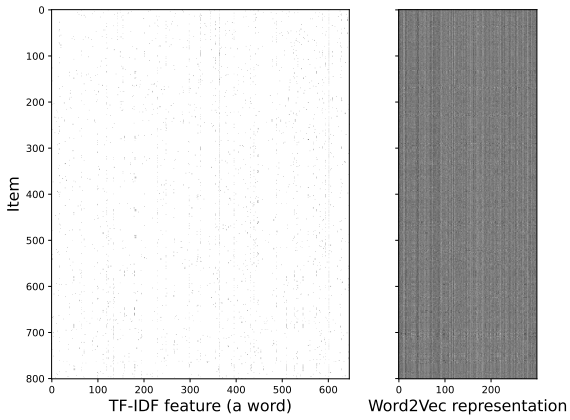

Figure 4: Representations (x-axis) of the titles of items (y-axis). The darker an entry is, the larger it is.

All text Both representations are also made on the feature `all_text`, which contains text appended from the title, brand, description, and category columns of the metadata. This text is cleaned using the same text preprocessing pipelines as earlier described.

Additional features Further, three numerical features are added: count of words in the `all_text` column, their average word length, and item price. For word counts and lengths, no stemming or stopword removal is applied. As not all prices are filled, the remaining prices are given a default value: the mean price $67.02. These features are normalized to a 0–1 range. Other arbitrary features could be made — e.g., binary labels for different price thresholds, LIX readability of the description, etc. As the *available* metadata is based on text, the text representations techniques are very powerful, and (while not shown here) the additional features result in a slight decrease in performance.

3.2 Making predictions

For each user, a user profile is made by computing the weighted mean of the vectors of user-rated items. Here, the weight corresponds to the overall rating of that item. For *all* items except already rated ones, the model computes item scores as the cosine similarity between user and item profile in vector space. This inference is simply done for each user in the test set, resulting in $1,360,957$ scores based on 1711 unique users. Note that when measuring similarity, the two text representation schemes can be handled in different ways and weightings, taking differing magnitudes into account. Here, cosine similarity is simply computed directly on the concatenated array.

4 Hybrid systems

This section treats the combination of approaches from the different classes of recommender systems introduced. Here, the CF-based scores (from SVD) will be combined with predictions from the content-based model (using TF-IDF and Word2Vec on titles, and the three additional scraped features; see the previous section). As the different models have clear up- and downsides, this section seeks to balance them against each other.

The predictions are first merged, keeping all scores of the content-based model via an outer join. Consequently, users who are not present in the test set are removed from the SVD predictions. As there is one item in the test set which is not present in the training set, the CF algorithms cannot cover this item, and a default value of the mean user rating is given.

4.1 Weighted strategy

First, the model scores are combined used a weighted sum on the SVD and content-based scores.[4] For this, the scores are first normalized to have mean 0 and standard deviation 1. With α and β denoting the weight used for the CF- and content-based scores respectively, two sets are applied: $(\alpha, \beta) = \{(1,1); (1,2)\}$. During evaluation (Section 5 and Table 4), the results are denoted Normalized Sum and Weighted Sum.

Further, the model rankings are combined through the Reciprocal Rank Fusion method (Cormack et al., 2009) with $k = 20$. This hybrid model is denoted RRF.

[4] Alternatively, the rankings could be used, but there would be more ties.

		P@5	MAP@5	MRR@5	HR@5	P@15	MAP@15	MRR@15	HR@15
CF	KNNWithMeans	0.0023	0.0070	0.0070	0.0117	0.0016	0.0085	0.0085	0.0245
	SVD	0.0153	0.0172	0.0172	0.0766	0.0092	0.0264	0.0264	0.1373
Content	TFIDF	0.0289	0.1039	0.1039	0.1444	0.0150	0.1133	0.1133	0.2250
	Word2Vec	0.0289	0.0808	0.0808	0.1444	0.0139	0.0879	0.0879	0.2092
	TFIDF-W2V	0.0309	0.0779	0.0779	0.1543	0.0153	0.0863	0.0863	0.2297
	TFIDF-W2V-all-text	**0.0393**	**0.1441**	**0.1441**	**0.1964**	**0.0176**	**0.1514**	**0.1514**	**0.2636**
Hybrid	RRF	0.0275	0.1117	0.1117	0.1373	0.0146	0.1202	0.1202	0.2192
	Normalized Sum	0.0330	0.1244	0.1244	0.1648	0.0159	0.1325	0.1325	0.2390
	Weighted Sum	0.0328	0.1254	0.1254	0.1642	0.0161	0.1337	0.1337	0.2414
	USwitch at 25	0.0367	0.1541	0.1541	0.1835	0.0161	0.1602	0.1602	0.2420
	ISwitch at 50	**0.0386**	**0.1592**	**0.1592**	**0.1929**	**0.0175**	**0.1670**	**0.1670**	**0.2624**
	ISwitch at 100	**0.0385**	**0.1592**	**0.1592**	**0.1923**	**0.0178**	**0.1674**	**0.1674**	**0.2665**
	Meta	0.0131	0.0408	0.0408	0.0655	0.0076	0.0448	0.0448	0.1146

Table 4: All implemented models and their ranking-based accuracy scores. The first three content-based models use just the item title, while the last one uses all the metadata text categories. The two last ones further use the three additional numerical features described in Section 3.1. Except for Meta, the hybrid models are all combinations on the SVD and TFIDF-W2V user-item scores.

4.2 Switching strategy

As discussed by Aggarwal et al. (2016, p. 212), cold-start issues can be mitigated using switching mechanisms to combine content-based and collaborative recommenders. For any item, the CF score is here chosen below a threshold on item ocurrences in the training data, and the content-based score above (in both cases the normalized scores, see the previous section). By observing the actual rating frequencies — the user support — of all items, two thresholds are chosen: 50 and 100. The results are listed and discussed in Section 5 as ISwitch. While this is item-based switching, an analogous user-based switching strategy is defined, here applied at a threshold of 25 ratings. Alternatively, combination strategies could be attempted.

4.3 Meta-level strategy

Lastly, a meta-level strategy is implemented, which follows the approach described on pages 216–217, Aggarwal et al. (2016). This sequential model uses user profiles from the content-based model as input to a neighbourhood-based CF algorithm. Using the Word2Vec user profile representation (where a user is the rating-weighted mean of their rated item vectors) gives rise to more robust similarities compared to the more sparse TF-IDF user profiles. For each user/item pair we wish to infer a rating for, a close peer group is found taking the 3 most similar users which have rated the item in question. The result is then simply the similarity-weighted mean of ratings.

Computing all user profile similarities beforehand and simply indexing this gave considerably faster computation time. It is however still slow to compute all 1.4 million required for our evaluation purposes (approx. 30 min.); using a heap-based structure could help.

5 Experimental evaluation

Table 4 lists ranking-based utility scores for all implemented models at rank cutoffs 5 and 15. From the table, notice first that MAP@k equals MRR@k in this setting — when at most one item can be marked as relevant. Further, the mean hit rate at k is simply P@k $\times$ k. Both of these properties are easily verified from the definitions of the metrics. A plethora of other evaluation criteria and measures exists, but the included ones give a good overview of the expected recommendation accuracy. In this concrete setting, it would make sense to drop P@k and MRR@k seeing the two others are more intuitive with a single relevant item and provide the same information; they are however kept along with their actual implementations as a means to verify each other. A limitation of these measures could be if one wanted to drop the binary relevance criteria, and instead assign higher importance to items where the user rated 5 rather than 4. For such multi-graded relevance, a cumulated gain-based metric could be applied (Järvelin and Kekäläinen, 2002). For the listed metrics, the neighbourhood-based CF model fares the absolute worst, followed by the meta-level strategy which in essense is sim-

ilar. Recall Table 1 also lists test set RMSE for the CF-based approaches; approximately 1.15 for kNN, and 1.0 for SVD. Likewise, RMSE for META gives 1.20. Interestingly, even though META scores higher in ranking-based metrics than kNN, the regressed values are worse. The latent factor model (SVD) scores higher across both types of metrics.

While RMSE may be a good measure for making robust predictions (due to squaring erroneous outliers), it is still heavily influenced by ratings of popular items (much alike the other listed metrics). One way to deal with this could be weighing errors according to the item popularity. More importantly, the recommender problem is not *only* a regression task. Since users usually only interact with a top-k ranking, we are especially interested whether relevant items appear in this small subset. Because of this, ranking-based measures often give a more realistic view of model usefulness (Aggarwal et al., 2016, p. 240).

After the META model, RRF with $k = 20$ was found as the worst-performing hybrid. Weighted sums of normalized scores worked surprisingly well, with 16.5% hit rate @5. These are perhaps the most simple and intuitive kinds of hybrid models, and work well as baselines to compare with for other hybrids.

As for switching strategies, the item- and user-based approaches perform similarly.For neighbourhood models, a slightly more intricate strategy could be to consider whether the neighbours are relevant based on similarity or some other arbitrary feature. For instance, if a user only has few ratings (perhaps even on rare items) but they are identical to another user, they may very likely share taste. With the implemented approaches, they are simply dropped no matter similarity. Alternative to switching using just a priori knowledge about strengths and shortcomings, a bucket-of-models approach could also be applied, where a fraction of the training data is held out to evaluate the performance of the different models before selecting a score.

To further investigate the temporal aspect of recommendation accuracy probed in Section 2.1, the moving average of HR@5 for test users sorted by review time is shown on Figure 5. By including CF-based predictions when inferring ratings for popular items, the accuracy grows — mainly when the test set review was far back. By approximating item introduction date by an item's first review, one could explore this topic further. This aspect could,

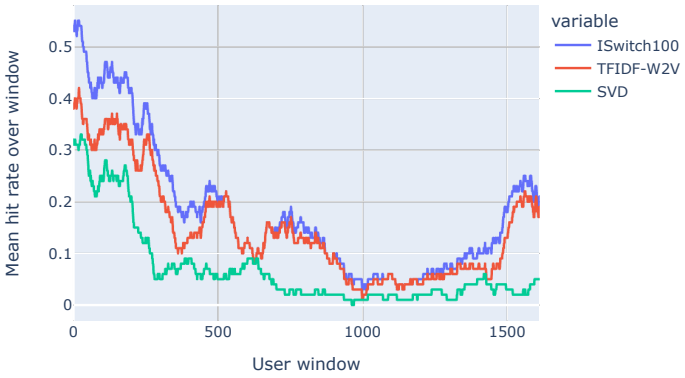

Figure 5: Moving average over mean hit rate for three recommender systems, window size 100.

e.g., assist switching behaviour. In isolation, the lines also suggest that only the purely CF-based system breaks down when having to predict more recent items — however, even for the content-based system, the earlier items are still easier to predict.

While additional fine-tuning (choice of hyperparameters and minor model-specific changes) could improve the accuracy measures, it is wrong to do so based on the test scores — this is also why results are presented for a somewhat large selection of approaches. Ideally, cross-validation or a separate validation set is made use of. Nonetheless, this collection of models and metrics gives a broad and sensible idea about the crude range of accuracy to expect for different kinds of recommender systems.

6 Conclusion

By applying a variety of recommender systems to the Amazon Review Data (Ni et al., 2019), both limitations and advantages of different models and measures have surfaced. In particular, hybrid systems leveraging the strengths of both conventional CF algorithms and content-based techniques have resulted in recommendations with mean hit rate @5: 0.19, and MAP@5: 0.16 on a small test set with a single relevant item. To this end, predictions from an SVD model were combined with predictions stemming from TF-IDF and Word2Vec item representations. Other meta-level strategies, contextual word embeddings, and fine-tunings seem to be the next steps towards improving utility — or perhaps other criteria.

References

Charu C Aggarwal et al. 2016. *Recommender systems*, volume 1. Springer.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197.

A kNN grid search

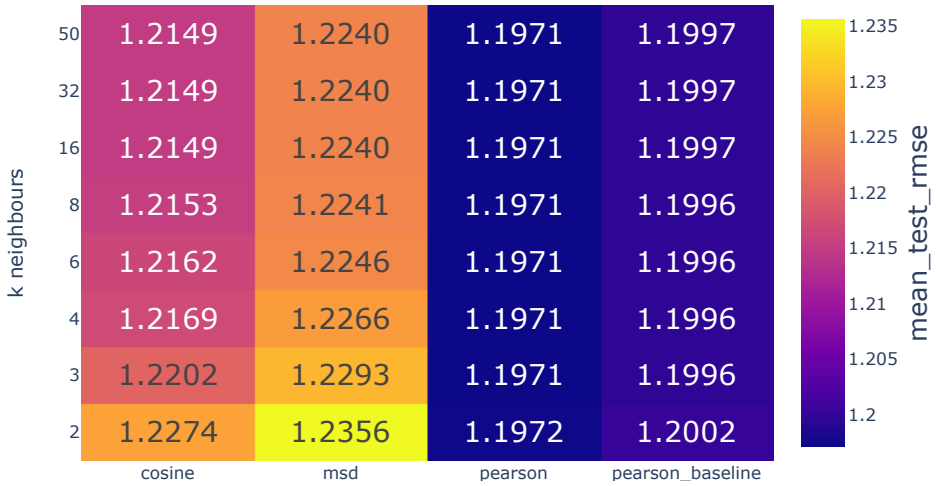

Figure 6: Discrete heatmap visualising the kNN hyperparameter grid search.

B SVD grid search scatter

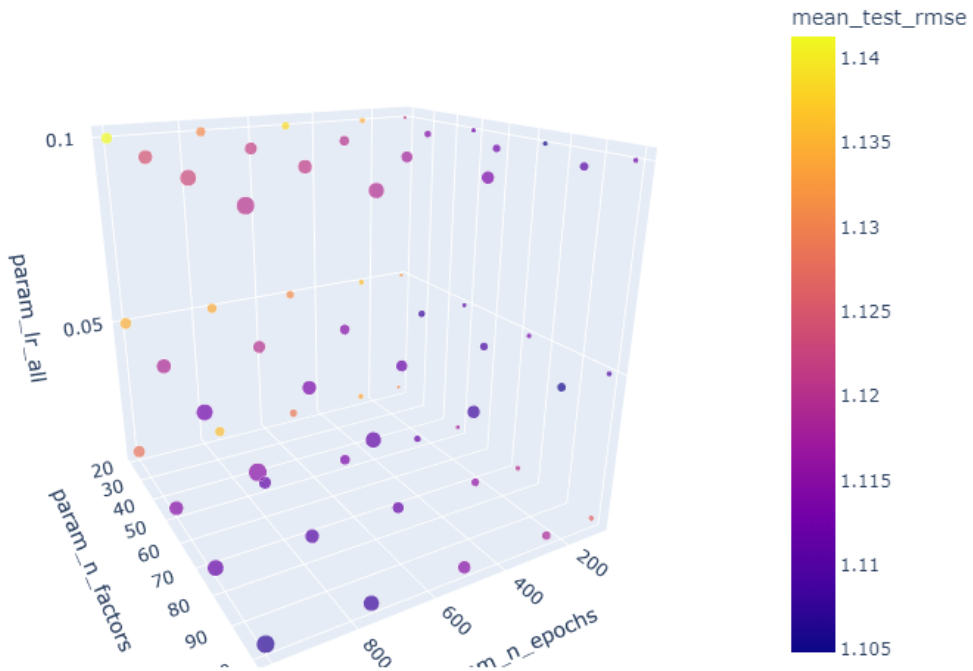

Figure 7: Scatter plot visualising the SVD hyperparameter grid search. Marker size proportional to training time.

C Rating histories

C.1 First user

First user (displayed: training entries) in time-ordered test set, and the top 10 neighbours' rating histories.

	overall	reviewTime	reviewerID	asin	reviewText
2492	3.0	08 8, 2008	A2G0O4Y8QE10AE	B0017I8NQM	Unfortunately Corel stopped ...
2407	3.0	10 5, 2010	A2G0O4Y8QE10AE	B0014KJ6EQ	I bought MacDictate after ...

	overall	reviewerID	asin	reviewTime
0	3.0	A3FY1GXS48WR8B	B0013O54OE	06 24, 2008
1	4.0	A3FY1GXS48WR8B	**B0017I8NQM**	10 1, 2008
2	5.0	A37MH7ICH80QOX	B000WCQCE4	02 19, 2010
3	5.0	A37MH7ICH80QOX	**B0017I8NQM**	02 19, 2010
4	4.0	A11JU33HMT5XPU	**B0017I8NQM**	07 24, 2008
5	4.0	A11JU33HMT5XPU	B001AFFYSW	02 9, 2009
6	3.0	A11JU33HMT5XPU	B005FIWT74	05 18, 2012
7	3.0	ACQYIC13JXAOI	B001EJU9ZM	10 1, 2008
8	5.0	ACQYIC13JXAOI	**B0017I8NQM**	02 19, 2010
9	5.0	ACQYIC13JXAOI	B003YJ5ESM	03 13, 2011
10	3.0	ACQYIC13JXAOI	B00MV9EL6M	10 4, 2014
11	2.0	ADY836HK6QSYR	B000TME1K4	07 10, 2008
12	3.0	ADY836HK6QSYR	**B0017I8NQM**	09 18, 2008
13	2.0	ADY836HK6QSYR	B009YYZJDQ	05 26, 2013
14	3.0	ADY836HK6QSYR	B00OW2PJ5I	12 29, 2014
15	5.0	A2E1EFNIZL2FVA	**B0017I8NQM**	07 24, 2008
16	5.0	A2E1EFNIZL2FVA	B00EZPXOTA	12 18, 2013
17	5.0	A2E1EFNIZL2FVA	B00HRQB28Y	05 12, 2014
18	3.0	A1JMB7RDVEMN71	B000Y7Y6IQ	05 15, 2008
19	5.0	A1JMB7RDVEMN71	B001AFD20E	08 21, 2008
20	4.0	A1JMB7RDVEMN71	**B0017I8NQM**	11 20, 2008
21	4.0	A3L8XRYMLZZES6	**B0017I8NQM**	07 24, 2008
22	1.0	A3L8XRYMLZZES6	B000TMHZX4	01 15, 2009
23	3.0	A3L8XRYMLZZES6	B001C31OZY	03 21, 2013
24	5.0	A3L8XRYMLZZES6	B003CIPI9A	03 28, 2013
25	3.0	A3L8XRYMLZZES6	B01AC3ZAHW	03 31, 2016
26	4.0	A1GNYV0RA0EQSS	B000Y7Y6IQ	04 6, 2008
27	5.0	A1GNYV0RA0EQSS	**B0017I8NQM**	12 21, 2008
28	4.0	A1GNYV0RA0EQSS	1413313728	02 9, 2012
29	2.0	A1GNYV0RA0EQSS	B0041DVMZE	03 12, 2012
30	4.0	A1GNYV0RA0EQSS	B009066EOG	04 5, 2013
31	4.0	A1GNYV0RA0EQSS	B009716GWE	04 18, 2013
32	1.0	A1GNYV0RA0EQSS	B01617VNYC	12 15, 2015
33	3.0	A2D66KSHQQHOSD	B000TME1K4	06 14, 2008
34	4.0	A2D66KSHQQHOSD	B000V6YPAY	07 17, 2008
35	3.0	A2D66KSHQQHOSD	**B0017I8NQM**	11 23, 2009
36	1.0	A2D66KSHQQHOSD	B00OMQLT6C	12 26, 2014

C.2 Last user

Last user (displayed: training entries) in time-ordered test set, and the top 10 neighbours' rating histories.

	overall	reviewTime	reviewerID	asin	reviewText
7861	3.0	06 29, 2015	A2SACTIFMC5DXO	B00EDSI8HW	It works
12661	2.0	06 29, 2015	A2SACTIFMC5DXO	B00EFRMECQ	not usable
7506	5.0	08 19, 2015	A2SACTIFMC5DXO	B00CTTEKJW	I am a Prime Member and it is the best thing g...
8747	5.0	07 3, 2016	A2SACTIFMC5DXO	B00FZ0E0HE	Doubled ordered . I am so lame.
8732	5.0	07 3, 2016	A2SACTIFMC5DXO	B00FZ0FETC	Doubled ordered . I am so lame.
10851	5.0	07 3, 2016	A2SACTIFMC5DXO	B01019T6O0	Works as intended.
11569	4.0	07 3, 2016	A2SACTIFMC5DXO	B015IHWAZW	Works as intended.

	overall	reviewerID	asin	reviewTime
0	5.0	A2KIIQX2TH70MR	B00FZ0FK0U	10 30, 2016
1	5.0	A2KIIQX2TH70MR	B0095CATEG	10 30, 2016
2	5.0	A2KIIQX2TH70MR	B00FZ0E0HE	10 30, 2016
3	1.0	A2KIIQX2TH70MR	B0144NYGJY	10 30, 2016
4	5.0	A2KIIQX2TH70MR	B00FZ0FETC	10 30, 2016
5	5.0	A2CHQH95XQUY4E	B00FZ0E0HE	03 8, 2014
6	5.0	A2CHQH95XQUY4E	B00FZ0FK0U	12 16, 2014
7	5.0	A2CHQH95XQUY4E	B00FZ0FETC	12 16, 2014
8	5.0	A2CHQH95XQUY4E	B00PG8FWS6	04 4, 2015
9	4.0	A2CHQH95XQUY4E	B01637ROB6	03 12, 2016
10	4.0	A26808LRG8PLPQ	B00CTTEKJW	04 8, 2016
11	2.0	A26808LRG8PLPQ	B00RKZKFUI	04 8, 2016
12	2.0	A26808LRG8PLPQ	B00P31G9PQ	04 8, 2016
13	2.0	A26808LRG8PLPQ	B00U7LCE6A	12 22, 2017
14	3.0	ASYKUKD5MW46V	B009HBCZPQ	01 20, 2013
15	5.0	ASYKUKD5MW46V	B00CTTEKJW	11 20, 2013
16	3.0	ASYKUKD5MW46V	B00B1TGUMG	01 24, 2014
17	2.0	ASYKUKD5MW46V	B00PG8FFFQ	02 14, 2015
18	1.0	AK8GQ08WTHW4G	B00NG7JVSQ	12 13, 2014
19	5.0	AK8GQ08WTHW4G	0763855553	12 14, 2015
20	1.0	AK8GQ08WTHW4G	B0144NYEY6	03 27, 2016
21	5.0	AK8GQ08WTHW4G	B01019T6O0	06 19, 2017
22	5.0	ARX0KOX5Q01BJ	B009HBCH6S	01 29, 2013
23	2.0	ARX0KOX5Q01BJ	B00CTTEKJW	02 27, 2014
24	5.0	ARX0KOX5Q01BJ	B01019BM7O	08 22, 2015
25	5.0	ARX0KOX5Q01BJ	B01019T6O0	09 7, 2015
26	5.0	ARX0KOX5Q01BJ	B01637RFR4	02 12, 2016
27	1.0	AMQ1MEG334SAP	B002JB3BC2	04 4, 2010
28	1.0	AMQ1MEG334SAP	B002JB1TTO	04 4, 2010
29	1.0	AMQ1MEG334SAP	B0095CAUN6	01 30, 2013
30	1.0	AMQ1MEG334SAP	B0095CATEG	01 30, 2013
31	1.0	AMQ1MEG334SAP	B011XO53WQ	11 29, 2015
32	1.0	AMQ1MEG334SAP	B013CTP6UY	11 29, 2015
33	1.0	AMQ1MEG334SAP	B00FZ0E0HE	09 16, 2016
34	1.0	AMQ1MEG334SAP	B00FZ0FETC	09 16, 2016
35	1.0	AMQ1MEG334SAP	B00FZ0FK0U	09 16, 2016
36	1.0	AX3RYFAA7QHS4	B00E6LJ2SA	01 15, 2014
37	4.0	AX3RYFAA7QHS4	B00EZPXYP4	08 3, 2014
38	4.0	AX3RYFAA7QHS4	B00MHZ71G2	08 14, 2015
39	5.0	AX3RYFAA7QHS4	B00CTTEKJW	01 21, 2016
40	1.0	A3MVMPFBN1RFFV	B002DHGMK0	01 22, 2010
41	1.0	A3MVMPFBN1RFFV	B0094NY3R0	12 25, 2012
42	1.0	A3MVMPFBN1RFFV	B01019T6O0	08 2, 2015
43	5.0	A3VV6SV0Z2HNTA	0763855553	04 1, 2014
44	1.0	A3VV6SV0Z2HNTA	B00P31G9PQ	06 20, 2015
45	5.0	A3VV6SV0Z2HNTA	B01019T6O0	07 31, 2015
46	5.0	A3VV6SV0Z2HNTA	B00L13X6QA	07 31, 2015