Exploring Contextual Paradigms in Context-Aware Recommendations

Conor Morgan
School of Computing Science
University of Glasgow
Glasgow, Scotland, UK
2199772m@student.gla.ac.uk

Iulia Paun
School of Computing Science
University of Glasgow
Glasgow, Scotland, UK
iulia.paun@glasgow.ac.uk

Nikos Ntarmos
School of Computing Science
University of Glasgow
Glasgow, Scotland, UK
nikos.ntarmos@glasgow.ac.uk

***Abstract*—Traditional recommendation systems utilise past users' preferences to predict unknown ratings and recommend unseen items. However, as the number of choices from content providers increases, additional information, such as context, has to be included in the recommendation process to improve users' satisfaction. Context-aware recommendation systems exploit the users' contextual information (e.g., location, mood, company, etc.) using three main paradigms: contextual pre-filtering, contextual post-filtering, and contextual modelling. In this work, we explore these three ways of incorporating context in the recommendation pipeline, and compare them on context-aware datasets with different characteristics. The experimental evaluation showed that contextual pre-filtering and contextual modelling yield similar performance, while the post-filtering approach achieved poorer accuracy, emphasising the importance of context in producing good recommendations.**

***Index Terms*—recommendation systems; context-aware recommendations; contextual paradigms; user-centred information models.**

I. Introduction

Recommendation systems, a sub-type of information filtering systems, have been actively used to address the information overload problem by highlighting relevant items to the users based on their tastes and previous interaction patterns. As the popularity of recommendation systems kept growing, the number of items available for recommendation also increased. This led to the development of more complex user-item models, which often capture underlying information about users or items to improve the quality of the recommendations and ultimately, the users' satisfaction and engagement. One of the dimensions that was explored as part of this process was *context*.

Context in recommendation systems is the general situation that the user is in when interacting with the item; it can take many different forms like the mood a user is in or where they are when using the item. Context affects decisions that we make in our everyday lives, from what to eat to where to go context is involved in the decision-making process. The reason for introducing context into recommendation systems comes from this idea of context affecting other decisions, and so if a model can utilise context, then the recommendations will be more accurate and appropriate. A non-contextual recommendation algorithm may suggest a horror movie to watch, which may normally be a good recommendation, but if the user is with a young child, then given the context, a horror movie is not a good recommendation.

There are many different ways of including context in a recommendation system, each with their own advantages and disadvantages within their methodology. This work investigates the three main paradigms within context-aware models, providing a detailed evaluation of how each approach can be implemented under various recommendation tasks. This also opens the stage for use cases where humans and machines disagree, and how to model randomness in human decision making. Furthermore, context-aware recommendation systems could support users in the decision making process by creating useful summaries from unstructured data.

A. Problem Formulation

As with traditional recommendation algorithms, there are a range of methods and approaches to produce context-aware recommendations. Therefore, given the abundance of options, it is often difficult to know which model to select for each use case. One of the main differences across the recommendation systems is how the context is utilised; is the context used at the start of the process on the data, is it used afterwards to alter the recommendations made or is it best to use the context directly with the system to produce recommendations? These are some of the key questions that we address as part of this study.

Datasets for models that contain contextual information are not as common as datasets without the context, and those that contain the users' context may be of unknown quality. Collecting context in datasets often relies on the user submitting their ratings and data to use which can result in inaccurate or missing data. Therefore, another problem investigated in this work is whether the dataset used, and its characteristics (e.g., sparsity), affect the recommendations made, as well as the model.

B. Contributions

As part of this work, we investigated how the use and modelling of the context can impact the quality of the rec-

This work is supported by the Erasmus+ Programme of the European Union under the PRIMES project (no. 2016-1-UK01-KA201-024631) and by the UK Government through an EPSRC grant (no. 509668).

ommendations. To this end, two context-rich datasets were sourced and used as part of the experimental evaluation. Our methodology is comprised of the three main paradigms found in context-aware recommendations, and led to the following contributions: *(i)* built a pre-filtering recommendation system that modifies the contextual data to be used with a traditional model to produce recommendations; *(ii)* investigated a heuristic-based contextual modelling recommendation algorithm, which incorporates the contextual data with a collaborative filtering nearest neighbour model to create recommendations; and *(iii)* implemented a post-filtering recommendation system that takes rating predictions from a traditional model and incorporates context with two different methods to make recommendations.

II. Background

Context-Aware recommendation systems extend the regular 2-dimensional recommendation systems which only use users and items, to take into account context when generating the recommendations. Context is a multifaceted concept and has many different meanings depending on the area of study it is being used from computer science to linguistics, philosophy, psychology and other sciences and as such it has been defined many times [3]. Context in recommendation systems can be said to be any information signal - e.g. location, time, social companion, device, and mood– regarding the situation in which a user experiences or interacts with an item. [5].

Context can be added to the Rating function (eq. 1) that takes users and items as dimensions, to use context and extend the function:

$$R: \; Users \; \times \; Items \; \times \; Context \rightarrow Ratings \qquad (1)$$

where $Users$ and $Items$ are the complete sets of the users and items involved, and $Context$ is a set of the different contextual attributes.

With multiple contexts, there would be multiple contextual dimensions to the model. This is the multidimensional approach. In this approach the rating function is shown as a mapping of a set of w different dimensional values to a rating [4], as shown in eq. 2.

$$R: \; D_1 \; \times \; D_2 \; \times \; \ldots \; D_w \rightarrow Ratings \qquad (2)$$

As the traditional 2-dimensional ratings can be mapped to a user-item matrix, the multi-dimensional approach is mapped to a w-dimensional cube. Two of the dimensions will always be for the user and item dimensions. If d_1 and d_2 are the user and item dimensions and the remaining dimensions are for context, we can we can define context as an attribute $c \in C$ where C is the set of contexts [13]. If there are multiple contexts, $c_1 \ldots c_n$, this function can rewritten as follows:

$$R: \; Users \; \times \; Items \; \times \; c_1 \ldots \; c_n \rightarrow Ratings \qquad (3)$$

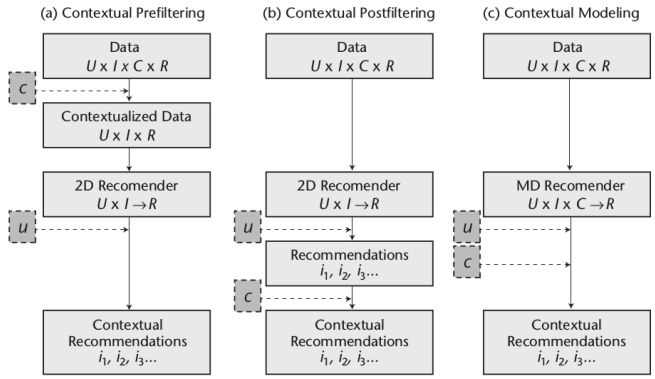

Fig. 1. The three methods for using context in recommendation systems: (a) Contextual Pre-filtering (b) Contextual Post-filtering (c) Contextual Modelling

A. Contextual Paradigms

There are three paradigms for making contextual recommendations (Figure 1). As context aware systems contain an extra dimension in the rating function (User x Item x Context), eq. 1, the system can include the context at different points during the process and each looks slightly different from the traditional method. The point at which context is included in the process is the defining characteristic of the three different paradigms.

Pre-filtering - The context is used to select only the data points that are the most relevant for the chosen context. The ratings are pre-filtered for the relevant context and the filtered set of ratings are used with a 2-dimensional recommendation system to make predictions.

Post-filtering – The context is ignored at first, recommendations are made using a 2-dimensional model on the whole dataset, discarding the context dimension. The recommendations are then filtered or adjusted with an additional step using the context to create the final predictions.

Modelling – The context is used directly with the recommendation technique, not before or after as with previous methods. In pre- and post-filtering, traditional recommendation systems are employed with context being used to alter the data before or the results after; modelling is the only paradigm that provides the most integrated results, however at times, it can be complex and computationally intensive.

B. Related Work

In context-aware recommendations, there are two main ways approaches towards pre-filtering: exact pre-filtering and generalised pre-filtering. Exact pre-filtering (EPF) is the method of only selecting the data points that have the exact context as the relevant context for the prediction. The system only selects the ratings from the initial full set of all ratings that have the relevant context c. The system then uses the chosen recommendation model on this reduced dataset and creates recommendations for c [11]. In sparse datasets, exact pre-filtering can be problematic and make a dataset even sparser

than it was before. The data will be completely relevant to the context which is an advantage however, there may not be enough data to make reliable, accurate recommendations.

Generalised pre-filtering (GPF) was introduced by [1] and permits a more general filter for selecting relevant data. Instead of choosing the exact context, generalised filtering allows similar contextual values to the relevant context to be chosen. This is to increase the size of the dataset and keep the principle of the pre-filtering method – to select the data points that have the most relevant contexts. An example of the method is if the recommendations are for a movie, the context is "Day of the week" and the relevant selection is "Wednesday" – $R(\ Eleanor,\ The\ Avengers,\ Wednesday)$. There may not be enough data with the exact filtering method for the same movie on Wednesday. To generalise it, the context could be opened up to select "Weekdays" instead of only Wednesday, giving five possible days for the context instead of just one. A drawback of this method is that sometimes irrelevant contexts may still be selected as it is not known exactly how each context affects the predictions at the time of pre-filtering – in the example "Monday" may be irrelevant for a movie on Wednesday but gets selected because of the size of the generalisation. The scope of the generalisation may need to be refined through trial and error which can be time consuming and inefficient.

In post-filtering, the contextual information is ignored, and a 2-dimensional recommendation model can use all the 2-dimensional data to create a set of recommendations. Context is then used to filter and adjust the output of the 2D model. When removing the context, if a user has multiple ratings for the same item in different contexts, the average rating for the item over each context is used e.g a user has rated an item in 3 different contexts: Morning, Afternoon, Evening, the average of each rating in the 3 contexts will be used as the rating for the user and item combination in the 2D recommendation algorithm [4]. Post-filtering, like pre-filtering, allows the use of any traditional recommendation system to create the predictions and post-filtering uses the context of the data to alter the recommendations.

Panniello et al. [12] show 2 methods which can be used for post-filtering: weight and filter. Both methods rely on creating a model that analyses the original, contextual data to find usage patterns and use this to contextualise the recommendations. The contextual probability $P_k(u, i)$ that a user u selects item i in context k is calculated and this probability is used to weight or filter the predictions from the 2D recommendation system. Contextual probability is calculated by the number of neighbours (customers similar to u) who purchased the same item i in the same context k, divided by the total number of neighbours. Both approaches use the contextual probability however in slightly different ways as shown in [11].

There have been a number of heuristic based methods developed for context-aware recommendations developed from the turn of the century. [2] proposed to extend the traditional neighbourhood-based approach to the multidimensional case with context. The extension was to use a multidimensional distance metric in the algorithm when calculating similarity metrics, instead of the standard user-user similarity. [10] presented a new method called Contextual Neighbours which is based on collaborative filtering. The method involves creating a contextual profile for each user u in context c, $Prof(u, c)$ and using the profiles of each user to find the nearest neighbours for user u. The profiles are used to define similarity between users and find the nearest neighbours. The similarity is calculated by using the cosine measure as this is a popular collaborative-filtering approach.

The study in [16] proposed 2 contextual modelling methods in Differential Context Relaxation (DCR) and Differential Context Weighting (DCW). DCR treats the recommendation algorithm as a collection of functional components and applies context relaxation differently in each component. An example of relaxation is if there is a context element Time with values {Morning, Afternoon, Weekend} and Location with {Home, Cinema}, and the relevant context is the {Morning, Home}. If the data with context {Morning, Home} was too sparse, relaxation would allow other similar contexts to be included as relevant even though they are not a direct match. This could allow, depending on the relaxation, {Morning, Home} and {Morning, Cinema} to be classed as relevant context. This is similar to the method of generalised pre-filtering in [1] but this context relaxation is happening directly with the recommendation system and the context is being used by the system too. DCR extends Resnick's algorithm for k-Nearest Neighbour user-based collaborative filtering [14] and Pearson correlation as a similarity measure.

DCW is evolved from DCR and has many similarities. DCW introduces weighting vectors, a collection of weights, which are used to scale the contribution of each context element to the algorithm. The weights are used with a similarity metric to calculate how similar different ratings contexts are and depending on the similarity, the context is given a score. This score is used to weight the influence the rating has on the prediction. This system allows more of the dataset to be used to create results with only contexts with similarities below a threshold not being used.

Latent Factorisation models can but used for context-aware recommendations. Matrix factorisation is based on the 2-dimensional user-item matrix and so to include the contextual dimensions, a multidimensional cube is used for each additional dimension of context. Tensor factorisation is then used to factorise the multidimensional cube in the same way a matrix factorisation model would the 2D matrix, and so tensor factorisation methods can be thought of as contextual versions of matrix factorisation methods [4]. An example of tensor factorisation is the Multiverse Recommendation model by [8] which uses different types of context for additional dimensions in the tensor, leading to a compact model of the data to provide context aware recommendations. Tensor factorisation models can be complex and resource intensive when the data and dimensions get large.

III. Experimental Methods

When context needs to be incorporated into the recommendation systems and there are three methods for how to include it: the context can be used at the start of the process to filter the dataset and select the relevant ratings with the same context before being passed to a 2D recommendation model; the context can be used at the end of the process to modify ratings after a 2D recommendation system makes predictions based on the data with context values removed; the context can be used directly with the recommendation algorithm to create a multi-dimensional recommendation system and produce recommendations.

A. Pre-filtering Paradigm

For this study, EPF was used to filter the dataset. EPF was chosen over GPF to use because all the data chosen is known to be exactly relevant to the user's context; GPF has the possibility of adding noise to the recommendation system with some contexts that are not completely relevant. This project is focused on comparing different methods of using context in recommendation systems and EPF is the purest pre-filtering approach and it was decided that it would be the best representation of the advantages and disadvantages of the approach.

The design for EPF meant the dataset was split into different sets, with each set containing the exact same set of context attributes as the other data points in the set. Each context set had their context removed and the users, items and ratings were passed to the model. The recommendation algorithm was a pre-packaged system from Surprise [7]. Each set was split into train and test sets and the recommendation model was trained on the training set and used the test set to output a set of predictions. The model predicted a score for each user-item in the test set and created a set of predictions for each group of contexts. The predicted ratings were compared against the value of the true ratings from the test set.

B. Post-filtering Paradigm

To generate the ratings for this work, the context variables were removed from the dataset before the whole dataset was split into train and test sets. The full train set was used to train the recommendation model and then the test set was used to generate a set of predictions. These predictions are the focus for the post-filtering stage and the original dataset with the context values are used too. It was decided to implement both Weight and Filter post-filtering methods as both methods rely on the calculation of the Contextual Probability($P_k(u, i)$). Once the contextual probability is calculated then it is trivial to implement either of the methods so both were chosen to compare the different versions of the same model.

Contextual Probability($P_k(u, i)$) is defined as the number of neighbours (customers similar to u) who purchased the same item i in the same context k, divided by the total number of neighbours. Every prediction contains the user and item the prediction is made for, plus the predicted rating. For each prediction, $P_k(u, i)$ is calculated for the user and item and the context the rating was made in (taken from the original dataset). $P_k(u, i)$ refers to neighbours in its definition and so each user has a set of neighbours that needs to be found. The method in [12] was followed to identify the neighbourhood. The neighbourhood was found by using the cosine similarity on the context values of ratings in the dataset and the target context k. The users that had the most similar contexts were added to the neighbourhood. Each neighbourhood was limited to a certain size afterwards, with the exact size depending on performance.

C. Modelling Paradigm

Differential Contextual Weighting (DCW) from [16] was used as the model to create context-aware recommendations. DCW was chosen as the model to use because of it's interesting concept of being able to assign weightings to contexts, meaning contexts can be chosen to have more of an effect on the model. Assigning weightings to features in recommedantion systems is not a new idea, however applying the weighting to contexts in the system is a novel approach. Each context variable will affect the overall rating in different manners, with some having a greater effect than others and having the ability to weight these ratings provides the chance to find the best combination of contexts for the most accurate results. The weighting system also has the advantage of using all the ratings in the dataset and assigning each a score based on similarity, not filtering out or removing data which could negatively affect sparse datasets.

The results by DCW in papers [6], [15] are promising and are showing low error scores compared to other contextual models and traditional recommendation systems; one paper showed a comparison of two related differential context modelling techniques and DCW provided the best results [16].

DCW is a heuristic, neighbourhood-based collaborative filtering model that incorporates weighting vectors when comparing similarity of contexts to control the contribution of each contextual feature to the algorithm. The weighting vectors are a collection of weights and are used to scale the contribution of each context element to the algorithm. They calculate the similarity of contexts and assign a score to all ratings based on context, instead of filtering out ratings. The similarity of a context and the given context is calculated with the weighted Jaccard Metric. If the similarities are above a certain threshold then the context is relevant, and the rating can be used. Also, the score is used to weight the ratings, in that a higher similarity score gives a higher weighting as it is thought that the rating will be more valuable to the system.

The weighting is the main focus of this algorithm and so a weighting vector is predefined for the algorithm. Each contextual attribute gets assigned a weighting in the range 0-1 and this is used to weight the attribute, for example if there are 2 contextual attributes {Time, Social} with a set of values that each context can have. The weighting vector could be [0.75, 0.25] assigning a weight of 0.75 to the Time attribute and 0.25 to the Social attribute when calculating similarity between the contexts.

IV. Evaluation

A. Datasets and Evaluation Metrics

The first dataset used during implementation was the LDOS-CoMoDa dataset [9]. This is a context-rich movie dataset and contains ratings for movies and 12 pieces of contextual information plus 18 other variables about the movies. The ratings and the contextual information were acquired explicitly from the users directly after watching the movie. The contextual information describes the situation on how the user watched the movie and is based on real interaction with the movie and not any hypothetical situations. The values in the dataset are all integers and any missing data was given a value of -1. There were unique integer user IDs and item IDs and ratings were on a scale of 1-5 with 5 being the best.

The DePaul Movie dataset is another context-rich dataset and was collected by researchers at DePaul University [17]. It has ratings for movies and three context variables – Time, Location, and Companion. The ratings and context were acquired explicitly by students from the university after watching the movie. The context is based on the situation the user is watching the movie in and not off any hypothetical situations or from memory. Time was based on what part of the week the movie was watched and the values for time were Weekday, Weekend; Location gave information on where the movie was watched with two options of Cinema, Home; Companion gave information on who the user was with when watching the movie and had possible values of Alone, Family, Partner. The ratings are on a scale from 1-5 with 5 being the best, missing values were represented as "NaN" and the context values were the text of the value unlike the first dataset. The DePaul dataset was chosen as it is a much denser dataset than the LDOS-CoMoDa with a density of 65.8% and should provide more reliable results.

There were five metrics in total that were used for evaluation of the models in the work and these were split into error-based metrics (i.e, RMSE, MAE) and ranking-based metrics (i.e., precision, recall, F1 score). Error and ranking metrics were used in this study as they both give indications of different properties the results have. Error-based metrics are focused on the accuracy of the prediction for a specific item. Ranking-based metrics are only focused on ranking the items in order of top recommendation to worst recommendation. Ranking metrics do not consider the value of the prediction when calculating the metric, the items in the recommended list are compared with the true recommended list to give a score.

B. Results

From the results shown (Table I), it is not clear as to which method is the best. Not one method completely dominates the scores across the set of results in either datasets. With RMSE and MAE, pre-filtering and contextual modelling have the best scores with similar scores on LDOS dataset and pre-filtering the best in DePaul. The ranking metrics tell a different story however with modelling having better F1 scores for both datasets, though the scores are again very similar.

Post-filtering methods are the worst across the board with highest error values and the lowest F1 scores; recall is one metric where post-filtering does well in and even better than some other methods, so even describing post-filtering as the worst is not clear-cut either.

Considering the complexity of the method to create and use to get predictions from as well as the results, pre-filtering appears to be the recommended method to use. The pre-filtering results are very similar to, if not better than those of contextual modelling, and the complexity of the method is much lower and more straightforward to get results. If a strictly ranking only recommendation system is being created, then the DCW algorithm and contextual modelling may be worth implementing as it has the best F1 scores across both datasets.

TABLE I
The results for the LDOS-CoMoDa dataset and the DePaul dataset. Exact Pre-Filtering (EPF), Differential Contextual Weighting (DCW), Post-Filtering Weight (PFW) and Post-Filtering (PFF) are evaluated against RMSE, MAE, Precision@k, Recall@k and F1@k.

LDOS-CoMoDa								
	RMSE	*MAE*	*Precision@k*		*Recall@k*		*F1@k*	
k =	-	-	5	10	5	10	5	10
EPF	0.985	0.855	0.486	0.375	0.829	0.903	0.544	0.446
DCW	1.059	0.768	0.606	0.525	0.737	0.862	0.589	0.557
PFW	2.849	2.652	0.424	0.329	0.822	0.899	0.482	0.398
PFF	2.188	1.729	0.432	0.333	0.825	0.898	0.487	0.399
DePaul								
	RMSE	*MAE*	*Precision@k*		*Recall@k*		*F1@k*	
k =	-	-	5	10	5	10	5	10
EPF	1.158	0.968	0.719	0.696	0.648	0.839	0.636	0.706
DCW	1.383	1.172	0.674	0.695	0.636	0.876	0.642	0.749
PFW	2.512	2.141	0.592	0.584	0.595	0.798	0.559	0.635
PFF	1.904	1.420	0.623	0.602	0.623	0.816	0.588	0.653

The ranking scores for DePaul dataset are better than the equivalent results in LDOS for precision and F1 whereas LDOS has better recall scores. When comparing ranking scores from the same paper [12], the F1 scores in the paper are values between 0.2-0.5 and are lower than the scores on both datasets, with Post-filtering having the lowest F1 scores but still as good as the results from the paper.

There appears to be a trend across both datasets, but mainly LDOS, that the precision and F1 scores are best when k=5 rather than 10 and the opposite for recall. This trend is due to the combination of the sparsity of the datasets, and how precision and recall are calculated. Ranking metrics are evaluated on the predictions made for each user in the test set, predictions are made for the whole test set and then the predictions are split by user to calculate the rankings. Each user will have a predicted top-k list and an actual top-k list based off their actual ratings. Depending on the density of the dataset, users in the test set will have different amounts of predictions made and this can cause unexpected behaviour. Some users will have few predictions made for them and so for some values of k, users may not have enough predictions for a top-k list.

V. Conclusions and Future Work

This work comprised an evaluation of various methods to include contextual information into recommendation systems, namely Contextual Pre-Filtering, Post-Filtering and Modelling methods were implemented, as well as Exact Pre-Filtering, Post-Filtering Weighting and Filter methods and Differential Contextual Weighting (DCW). Two datasets with contextual information were used to test how the methods performed on datasets with different data densities and with different contextual set-ups.

Each method was evaluated with each dataset and on a series of metrics, both error and ranking-based metrics. From the evaluation, the best performing method was not clear – both EPF and DCW had similar results. It was concluded that the best method to use depends on the situation and type of recommendations being made. The relative simplicity and good performance of EPF makes it the choice for all round recommendation, however the results can be misleading on sparse datasets. If rankings of items are to be predicted then DCW is the choice as it outperforms EPF on ranking metrics and has consistent results across both sparse and dense datasets, however the algorithm can be complex to implement. Both post-filtering models were the worst performing for the error-based and ranking-based metrics, though they did produce some good recall scores and with the DePaul dataset the ranking metrics were not far off the best.

The two datasets, LDOS-CoMoDa and DePaul both contained ratings of users for items and had contextual values for each rating. LDOS was found to be a much sparser dataset and had fewer ratings to use than DePaul. The two datasets had a noticeable difference in the results of the models, DePaul had similar if not lower error scores and higher ranking scores than the methods with LDOS. This showed that the methods do not produce the best or most accurate results when using a sparser dataset.

Context is a broad term and many different things can fall under its definition, so a possible extension to the work would be to research if certain types of context are better for recommendations. For example, "Mood" context is an attribute of the user and "Location" is to do with the space a user interacts with the item. Are contexts are directly concerned with the user, like "Mood", better indications of a rating than external contexts, like "Location". DCW algorithm begins to look at this with giving different weights to contexts, but it could be taken further to focus on the different types of contexts and the effects they have.

Another extension to the study would be to collect a richer dataset of users, items, ratings and contexts for the purpose of comparison across recommendation models. The advantages of collecting a dataset are that the context variables can be chosen to suit the needs of the recommendation task; the rating scale can be chosen beforehand, from rating 1-5 or a simple liked/did not like system could be used; the method of collecting contexts can be specified to be explicit or implicit or a combination of both; and the density of the dataset can be predefined in advance. Sourcing a larger dataset would facilitate the personalisation of the contexts and features to the needs and requirements of the users.

References

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. *Incorporating contextual information in recommender systems using a multidimensional approach*, volume 23. Jan. 2005.

[2] G. Adomavicius and A. Tuzhilin. Incorporating Context into Recommender Systems Using Multidimensional Rating Estimation Methods:. In *Proceedings of the 1st International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces*, pages 3–13, Reading, United Kingdom, 2005. SciTePress - Science and and Technology Publications.

[3] G. Adomavicius and A. Tuzhilin. Context-Aware Recommender Systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 191–226. Springer US, Boston, MA, 2015.

[4] C. C. Aggarwal. Context-Sensitive Recommender Systems. In C. C. Aggarwal, editor, *Recommender Systems: The Textbook*, pages 255–281. Springer International Publishing, Cham, 2016.

[5] P. G. Campos, I. Fernández-Tobías, I. Cantador, and F. Díez. Context-Aware Movie Recommendations: An Empirical Comparison of Pre-filtering, Post-filtering and Contextual Modeling Approaches. In C. Huemer and P. Lops, editors, *E-Commerce and Web Technologies*, Lecture Notes in Business Information Processing, pages 137–149, Berlin, Heidelberg, 2013. Springer.

[6] K. Gusain and A. Gupta. Context-Aware Recommendations Using Differential Context Weighting and Metaheuristics. In H. S. Behera and D. P. Mohapatra, editors, *Computational Intelligence in Data Mining*, Advances in Intelligent Systems and Computing, pages 781–791, Singapore, 2017. Springer.

[7] N. Hug. NicolasHug/Surprise, Mar. 2020. original-date: 2016-10-23T14:59:38Z.

[8] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 79–86, Barcelona, Spain, Sept. 2010. Association for Computing Machinery.

[9] A. Košir, A. Odić, M. Kunaver, M. Tkalčič, and J. Tasič. Database for contextual personalization. 2011. Accepted: 2018-08-01T07:23:35Z.

[10] U. Panniello and M. Gorgoglione. Incorporating context into recommender systems: an empirical comparison of context-based approaches. *Electronic Commerce Research*, 12(1):1–30, Mar. 2012.

[11] U. Panniello, M. Gorgoglione, and C. Palmisano. Comparing Pre-filtering and Post-filtering Approach in a Collaborative Contextual Recommender System: An Application to E-Commerce. In T. Di Noia and F. Buccafurri, editors, *E-Commerce and Web Technologies*, volume 5692, pages 348–359. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[12] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 265–268, New York, New York, USA, Oct. 2009. Association for Computing Machinery.

[13] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 635, Beijing, China, 2011. ACM Press.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, Chapel Hill, North Carolina, USA, Oct. 1994. Association for Computing Machinery.

[15] Y. Zheng, R. Burke, and B. Mobasher. Differential Context Modeling in Collaborative Filtering. May 2013.

[16] Y. Zheng, R. Burke, and B. Mobasher. Recommendation with differential context weighting. In *In The 21st Conference on User Modeling, Adaptation and Personalization (UMAP 2013*, pages 152–164, 2013.

[17] Y. Zheng, B. Mobasher, and R. Burke. Carskit: A java-based context-aware recommendation engine. In *Proceedings of the 15th IEEE International Conference on Data Mining Workshops*. IEEE, 2015.