**Selection of Similarity Function for Context-Aware Recommendation Systems**

# Kunal Gusain1 and Aditya Gupta1

1 Computer Science Department, Bharati Vidyapeeth’s College of Engineering, Guru Gobind Singh Indraprastha University, Delhi, India

kunalgusain1995@gmail.com and adityag95@gmail.com

**Abstract.** Earlier recommendation engines used to work on just the User and Item details, however more recently Users’ specific contexts have found an equally significant place as a metric in finding recommendations. The addition of contexts to the mix makes for more personalized suggestions and search for a truly efficient Context-Aware Recommendation System (CARS) continues. Differential Context Weighting (DCW) based CARS need to compute similarities between different users to give recommendations. Our objective is to analyse, compare and contrast various similarity functions, not only to find the best suited one, but also to implement an efficacious and economic CARS. To optimize the weights in DCW we use a metaheuristic approach.

**Keywords:** recommender system; context; context-aware recommendation; similarity functions; differential context weighting;

# 1 Introduction

Recommendations guide many choices in our daily lives, and thus the process of recommending is one which merits time and attention. Traditionally recommendation engines were built using the features of *Users x Items*. Over time a third input called ‘context’ has also come into the picture, which is nothing but the users’ specific data about the choice they made [1]. For a particular user it could be the time recommendation was made in, the day or even the weather. Contexts could be collected either implicitly by the website/app/device or explicitly by asking for users input. Simple collaborative filtering techniques for not able to handle this increased dimension, *Users x Items x Contexts*. The increased metric was not the only issue, since contexts could be different for the same user at different times, and the collection of all the contexts was not possible, the data often obtained was sparse. Even in dense data though we needed a metric to find ways to implement a successful Context-Aware Recommendation System (CARS). The pioneering work into CARS was first done by Adomavicius et al. [2] and since then there has been no looking back.

There are primarily two problems with using the contextual dataset, Firstly, the selection of contexts that will be used and Secondly, increasing sparsity and dimensionality in the data. Zheng et al. has worked an compared many methodologies which can be used to solve the above problems, like Context-Aware Matrix Factorisation (CAMF) [3], Contextual Sparse Linear Method (CSLIM) [4] and Differential Context Relaxation (DCR) [5, 6] and Unger et al. has done significant work in using auto encoders to tackle the problem of dimensionality [7]. More recently an improved variant of DCR, which is the Differential Context Weighting (DCW) has also been proposed [8]. Context-Aware Recommendation System (CARS) although significantly better than traditional engines is still a relatively recent entrant and thus a fascinating research topic, and much more work is needed to be done to arrive at a truly efficient CARS. Out of all the rating models DCR has proved to be quite efficient however, it property of taking into account only selective contexts or ‘relaxing’ of contexts gave rise to DCW, which now gave weights to all the contexts [8]. In our paper we analyse the working of DCW in the aim to improve its ratings and further the research in CARS. Depending upon different similarity functions, substantially different results can be obtained and thus various functions need to be analysed to arrive at the best possible one. To the best of our knowledge no work has been done in this regard.

The objective of this paper to study, compare and contrast various similarity functions in the aim to find the best one that can be employed in DCW to obtain the recommendations. The most prominent and relevant functions like, Jaccard, Euclidean, Cosine, Manhattan, Dice and Minkowski are hereby studied. We further find the Root Mean Squared Error (RMSE) feature and entry rich contextual dataset available, which is the LDOS-CoMoDa Movie Dataset. To optimize the weights required for DCW, we use the swarm based metaheuristic [9, 10] technique called Particle Swarm Optimization (PSO) [11, 12]. Experimental results are studied and recommendations using the best similarity metric is obtained.

# 2 Methodology

## **2.1 Rating Model**

To find the recommendations we need to assign ratings based on the data, and this is the core of any CARS. We employ Differential Context Weighting (DCW) [8] to find the ratings because out of the filtering and modelling techniques [2], giving weights to all the contexts improves the recommendation and makes sure all contexts play a role in finding of the ratings, thus increasing the relevance of the CARS.

In DCW, we do not ‘relax’ or set aside weights that are not of immediate importance (like in DCR) instead the aim is assign weights to all the features depending upon their importance or relevance in the data. Weights are between numeric 0 and 1, and depend upon the contribution of that feature. Higher weigh simply means that the contribution of that feature is more than the one with a comparatively lower weight.

Now we use the similarity function to find the similarities between different users based on their weighted contexts as given by Equation (1). Here ‘σ’ denotes the weights, and ‘c’ as well as ‘d’, are the contexts for the user.

 (1)

As in [8], the DCW rating model, if ‘P’ depicts the predicted rating and user is ‘a’, then the rating assigned by user to an item ‘i’ is shown in Equation (2). Rating is given by ‘r’ and ‘’ is the rating function.

(2)

To start, we must first find the neighbours (‘N’) that have rated the item ‘i’ with the context ‘c’. Since for the same user the contexts change over a period of time, thus giving rise to multiple ratings over different contexts, we fuse the maximally similar ones to their neighbours. Formula for selection is given in Equation (3).

 (3)

Once the neighbours have been selected, we must now find their individual contributions. For this we have given weights to all the contexts, and thus use the mathematical average of all the ratings given by that neighbour (Equation (4)).

(4)

DCW rating model is implemented the equation above.

## **2.2 Similarity Function**

Similarity function is the metric used to calculate the similarity between two objects. They usually take two objects as input, and give a large positive value for the objects with high degree of similarity and either negative value or zero for the objects with low degree of similarity. It can also be defined as the distance between the dimensions or features representing the objects. The similarity functions find applications in recommender systems, clustering, sequence alignment etc.

Similarity is generally measured in the range from 0 to 1. Similarity equals ‘1’ if two objects are identical and ‘0’ if two objects are completely different. There are various similarity functions which are being used these days, few of them which we considered for our experiment are:

1. Euclidean Distance
2. Manhattan Similarity
3. Minkowski Similarity
4. Cosine Similarity
5. Jaccard Similarity
6. Dice Similarity

**Euclidean Distance**

Euclidean Distance is described as the distance between two objects in the Euclidean space. Euclidean distance works well on when data is dense and continuous. It can also be referred as the length of path connecting two objects. This distance is calculated using Pythagoras theorem. The function for Euclidean distance is given in Equation (5).

 (5)

**Manhattan Distance**

Manhattan Distance is a metric to calculate absolute sum of differences between the coordinates of two objects. In this we are tasked with finding the sum of absolute difference between the coordinates of objects under review. The function for Manhattan distance is given in Equation (6).

 (6)

**Minkowski Distance**

Minkowski Distance can be considered as the generalized metric form of Manhattan Distance and Euclidean Distance. The function for Minkowski distance is given in Equation (7).

 (7)

The most important parameter in the above Equation is p. If p = 1, then the Equation behaves as Manhattan distance and if p = 2, then the Equation behaves as Euclidean distance. Similarly, when p equals infinity the resulting Equation is called Chebyshev distance.

**Cosine Similarity**

Cosine similarity is the measure of similarity between two vectors which finds the normalized dot product of two vectors. By trying to calculate the cosine similarity we are effectively trying to figure out the cosine of angle between two objects. If two vectors have the same orientation then their cosine similarity is 1, if two vectors are at right angle then their cosine similarity is 0. The cosine similarity is generally used in positive vector space, where the result is bounded by [0, 1]. The function Cosine Similarity is given in Equation (8).

 (8)

**Jaccard Similarity**

Jaccard Similarity or Jaccard index is a similarity measure which is used to calculate the similarity between two finite sample sets. It can be defined as the cardinality of intersection of two set divided by cardinality of union of two sets. The function Jaccard Similarity is given in Equation (9).

 (9)

**Dice Similarity**

Dice similarity is also known with several other names like Sorensen index and Dice coefficient. It is used to calculate the similarity between two samples. Dice similarity is very similar to Jaccard Similarity. It also doesn’t satisfy the triangle inequality property just like Jaccard Index. Dice similarity works well with heterogeneous data and gives less weight to outliers hence reducing the effect of outliers on the final output. The function Dice Similarity is given in Equation (10).

 (10)

## **2.3 Optimization**

Once the weighed values have been obtained, DCW finally has to optimize these weights to find the ratings. The optimization technique we employ for this step is Particle Swarm Optimization (PSO). This is a swarm intelligence methodology which is based on the behaviour of birds and bees in nature [11, 12]. A nature based metaheuristic technique, the idea is that just like in nature, bees swarm together, and similarly we can make the particles swarm together for optimization. All the particles are plotted in what is called the target space, and then each particle is allowed to swarm or move forward in the space to find the best possible location it can find. Each particle has a definite velocity that is it has a defined speed and direction. In PSO each particle has a fascinating characteristic called the Social component of that particle, due to which it tends to attach itself to the best location in its neighbourhood. As the particle moves through target space looking for best possible value, and if passes an optimal value, and then cannot find a value better than this one, it can also come back to this previous location. These values that PSO looks for is nothing but the value closest to the target value, such that all the particles in the target space can be optimized. Its working in a step by step is a characteristic feature of any metaheuristic algorithm. PSO is a knowledgeable algorithm, in the sense that at each particular location, all the particles are aware about their environments and have knowledge regarding the path they have taken, their direction, and what speed are they moving by. It has also been shown to have a high convergence rate and is a highly optimal technique for optimization problems like ours in DCW.

## 

# 3 Experimental Setup

We used publicly available LDOS CoMoDa movie context aware dataset for the experimental purpose. This dataset served the best purpose for our experiment because it has lot of contexts and is dense as compared to some other datasets available on the internet. It contains around 1000 entries and more than 15 contexts like age, mood, emo, location etc.

We first calculate the weights which are to be assigned to each context using particle swarm optimisation which is a swarm based optimisation technique. After calculating the weights for all the contexts, the next step is to run the DCW with different similarity functions on our dataset.

The accuracy of the similarity functions is measured using Root Mean Squared Value RMSE. We measured the accuracy. We calculated the RMSE for all the similarity twice, first time on the test data and second time on the overall complete data. The results for both are show in below in Figure 1 and Figure 2.

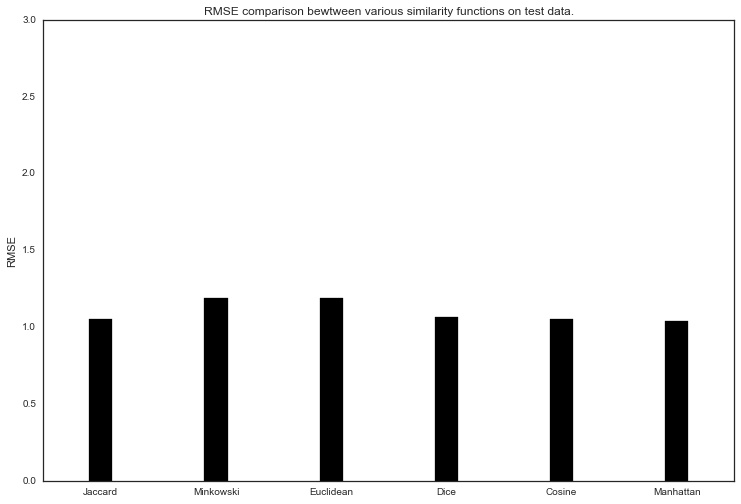


Figure 1. RMSE comparison between various similarity functions on test data

It can be seen from the Figure 1 that Dice similarity performs slightly better than the other 5 functions and Minkowski Similarity functions performs the worst.

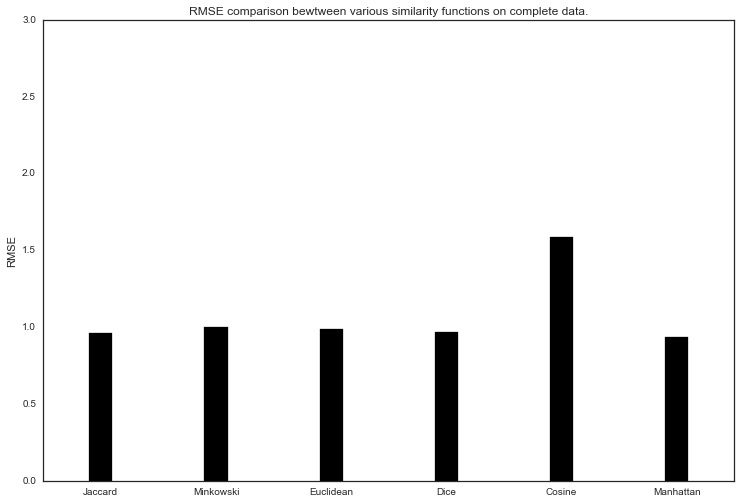


Figure 2. RMSE comparison between various similarity functions on total data

Figure 2 shows that Dice Similarity function again performs the best among all the other similarity measures even on the total data and this time cosine similarity function performs worst among them. From both of these above observations we can conclude that Dice similarity function is the best algorithm for Differential Context Weighting recommendation engine.

# 4 Results

The dataset used for comparison of similarity function is LDOS-CoMoDa Movies Dataset with data of over a thousand movies and fourteen different contexts like time, day type, season, location etc. Once the features have been assigned weights, we use all the six similarity functions to find and compare the necessary similarities.

# 5 Conclusions and Future Scope

CARS that work on DCW need to calculate the similarities between different users and thus need a similarity function with the least RMSE. In this paper we have compared and contrasted all the relevant functions that could be used for weighted values and implemented an efficient CARS with the best function found. The function were compared on the basis of

And it was found

Thus it is our recommendation that \_\_\_ should be used for DCW based CARS to obtain the best result.

Future work could focus on finding ways to further reduce the dimensionality of contextual data and further improve the usage of contexts in CARS. Work by Unger et al. [7] has shown that auto encoders can be used to tackle the aforementioned problem, however more research on unsupervised auto encoders is also needed. Although DCW is the currently the most efficient rating model for weighted contexts available to us, but the search for an even better one must go on. Finally, optimization techniques can also be compared and contrasted to find the best way to optimize the weighted data. Work towards hybrids of DCW could provide an interesting perspective.

# References

1. D.B. , Schilit Theimer , C.A. Brunk , C. Evans , B. Gladish , M. Pazzani , Adaptive interfaces for ubiquitous web access, Comm. ACM 45 (5) (2002) 34–38 .
2. G. Adomavicius, T. Alexander, Context-aware recommender systems, in Recommender Systems Handbook, Springer, US, 2011, pp. 217–253.
3. Yong Zheng, Bamshad Mobasher, Robin Burke. “Correlation-Based Context-aware Matrix Factorization“, Proceedings of School of Computing Research Symposium (SOCRS), DePaul University, Chicago, USA, May 2015.
4. Yong Zheng, Bamshad Mobasher, Robin Burke. “CSLIM: A Contextual SLIM Recommendation Algorithm“, Proceedings of the 8th ACM Conference on Recommender Systems (ACM RecSys), pp. 301-304, Silicon Valley, Foster City, CA, USA, Oct 2014.
5. Zheng, Y., Burke, R., Mobasher, B.: Differential context relaxation for context-aware travel recommendation. In: Huemer, C., Lops, P. (eds.) EC-Web 2012. LNBIP, vol. 123, pp. 88–99. Springer, Heidelberg (2012).
6. Zheng, Y., Burke, R., Mobasher, B.: Optimal feature selection for context-aware recommendation using differential relaxation. In: ACM RecSys 2012, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012). ACM (2012).
7. Unger, Moshe, et al. "Towards latent context-aware recommendation systems." Knowledge-Based Systems 104 (2016): 165-178.
8. Zheng, Yong, Robin Burke, and Bamshad Mobasher. "Recommendation with differential context weighting." *International Conference on User Modeling, Adaptation, and Personalization*. Springer Berlin Heidelberg, 2013.
9. X. S. Yang, “Nature-Inspired Metaheuristic Algorithms”, Luniver Press, 2008.
10. Christian Blum, Maria Jos´e Blesa Aguilera, Andrea Roli, Michael Sampels, Hybrid Metaheuristics, An Emerging Approach to Optimization, Springer, 2008 .
11. Xiang-yin Meng, Yu-long Hu, Yuan-hang Hou, Wen-quan Wang, The Analysis of Chaotic Particle Swarm Optimization and the Application in Preliminary Design of Ship”, International Conference on Mechatronics and Automation, August, 2010.
12. J. Kennedy, R. C. Eberhart, “Particle swarm optimization”, IEEE International Conference on Neural Networks, Piscataway, NJ., pp.942-1948, 1995.