A SCALABLE MODEL FOR COLLABORATIVE FILTERING USING SELF-ORGANIZING MAPS AND EXTENDED DENSITY-BASED INSTANCE SELECTION

Jerome, Otobong B.
Obafemi Awolowo University
NIGERIA
jeromeblacq@gmail.com

ABSTRACT

Increase volume of input data poses a problem of scalability to traditional memory-based recommender systems, In this paper, we propose a recommendation system model that is both efficient and highly scalable. The model is also very intuitive which makes it easy to implement and debug on a large scale and commercially. This model is an ensemble of several algorithms. The recommendation here is done in two stages, in the first stage we build a condensed representation of the input space using self-organising maps to cluster the input then further extracting the most relevant instances based on their neighbourhood densities ordering. The final case base carries the overall topology of the original big-data but is considerably reduced in size. In the second stage, the online recommendation is done using the similarity of the user to the users in the condensed case base. We achieved a recommendation accuracy of 96.1%, at a very high speed online. Certain system constants play an important role in the system"s accuracy, parameters such as SOM matrix size, the neighbourhood function as well as the similarity metric, were determined here experimentally using different accuracy measures. This showed that the accuracy of recommendation depends more on the ability of the data set to capture the nature of the users" space than on the size of the data set. In most cases, the data set contains lots of redundant data points and as such it is useful to extract such points as have more information to represent the neighbourhood than to make use of the entire neighbourhood.

Keywords: Collaborative Filtering, Instance Selection, Recommendation.

1.0 INTRODUCTION

The frustration of combing through millions of contents in search for one most appealing to a user is one of the most common hindrances to optimal user experience in modern times.
Users are especially vulnerable to the consequences of the paradox of choice, in most practical cases, it is an enormous boost to the user experience where the system sieves through its content based on some information and provides the user with insightful suggestions. To this end, we build recommenders, also known as recommender systems.

Recommender systems are ubiquitous, applied in systems like Netflix on movie recommendation, Spotify for personalized playlists, e-commerce platforms for product recommendations and even the financial sector to recommend financial instruments based on customers' risk aversion and other factors.

Recommendation systems are in ever-increasing demand in recent times because of the corresponding increase in the volume of content users have to choose. There is a plethora of literature on the subject, proposing various algorithms. In general, approaches towards recommendation are either from the users' perspective or that of the recommended items. The former termed collaborative filtering and the later content-based filtering. The implementation of either is generally either memory-based, model-based or a hybrid. Some recommendation systems, namely knowledge-based recommenders, make use of an explicit rule-based approach, capturing these rules explicitly from users' preferences. In either case, the trade-off between the quality of recommendation and the scalability of the model seems to be a common concern. Collaborative filtering is the most commonly used of the approaches, used by popular systems such as the book recommendations on Amazon [2].

This approach assumes that individuals with shared preferences in the past will share similar tastes in the future [1].

As stated by [3], collaborative filtering provides three additional advantages over content-based filtering: 1. It allows support for filtering items whose content is not easily analyzed by automated processes. 2. It can filter items based on quality and taste. 3. It can provide a serendipitous recommendation.

Collaborative filtering can also adjust appropriately to changes in user preferences. However, there is a danger of Matthew's effect with collaborative filtering systems. But, that is out of the scope of this paper.

The various collaborative filtering algorithms developed over the last few decades target the problem from different perspectives. [7] proposes the use of a noise-resilient matrix, namely NORMA, to achieve less biased matrix approximation and thus more accurate collaborative filtering with noisy user-item ratings. [4] proposes an approach where the information-theoretic measure known as entropy estimates the notion of selective predictability.[6] presents an algorithm for modelling the collaborative filtering task using quaternion algebra. These algorithms have different motivations and as such different evaluation metric. K-Nearest Neighbour based approaches are traditional and intuitive. However, they pose an enormous challenge in terms of scalability also in most practical cases user-rating sets do not have classes assigned to them.

In this paper, we propose a simplified, intuitive and easy to implement algorithm aimed at increasing online recommendation speed, quality of recommendation and scalability. The proposed technique uses case-based reasoning. But in other improve scalability the dataset is preprocessed using a topography-retaining instance selection algorithm.

Our data set consist of 14,116 ratings from users on 100 jokes from the jester dataset.
We split the model into two processes:
i.) Offline Model Building: Offline our goal is to reduce the enormous user cases to a subset, the least subset that captures all its properties and label such clusters. We made use of Self-Organizing Maps SOM, then performed instance selection on the dataset.
ii.) Online recommendation: Online we use an instance-based learning algorithm to compare users with other users in the generalized case set and recommend using this similarity. We considered Locally weighted regression and K-Nearest Neighbours, KNN had better performance, although in theory any instance-based approach a can be used in this phase
We judge the performance of the model using its time complexity and the error derived from the various stages.

1.1.0. BACKGROUND

1.1.1 Self-Organizing Maps

A Self-Organizing Maps SOM is an unsupervised learning algorithm and type of artificial neural net (ANN) that produces a low-dimensional, discretized representation of the input space of the training samples. SOM applies competitive learning as opposed to error-correction [8].

The following stages characterize the training of a SOM:

1 **Initialization:** We initialized all the connection weights with small random values.

2 **Competition:** Given inputs *x*:

$x = \{x_i: i = 1, ..., K\}$ … … … … ... equation (4)

and the connection weights w_j between the input units *i* and a competitive neuron *j*

$w_j = \{w_{ji} : j = 1, ..., N; i = 1, ..., K\}\}$ … … … … ... equation (5)

where N is the total number of neurons. We used distance function *D(x,w)* to determine the closeness of the input to each competitive neuron. The neuron with the greater similarity becomes the winner neuron, formally the winner neuron is the neuron for which

$\| D(x,w) \| = \min_i\{\| D(x,w_i)\|\}$ … … … … ... equation (6)

3 **Cooperation:** We define a topological neighbourhood around the winning neuron, and the weight of the winning neuron as well as other neurons in its 'neighbourhood' is updated.

The general form of the update is:

$m_j (t + 1) = m_j (t) + h_{cj}(t) [x(t) m_j (t)]$ … … … … ... equation (7).

where:

t denotes time; $x(t)$ the input vector drawn from the input data set at time *t*.

h_{cj} (t) is known as the neighbourhood function. As $t \to 1$; h_{cj} $(t) \to 0$,

The neighbourhood function determines the rate of change around the winner unit.

A commonly used neighbourhood function is the gaussian neighbourhood given as

$h_{cj}(t) = exp(-\frac{{d_{c,j}}^2}{2\sigma^2}).$ … … … … ... equation (8)

where:

d_{cj} represents the distance between the winning neuron *c* and a neuron *j in* the neighbourhood.

$\sigma(t)$ =. σ_0 $exp({}^{t}/_{\tau_\sigma}).$ … … … … ... equation (9)

$\sigma(t)$ is the width of the topological neighbourhood, it determines the amount of participation of each topological neighbour in the learning process, it decreases monotonically to zero as the distance goes to infinity, and its a system constant.

The Gaussian function is computationally expensive and often approximated to the bubble function, other commonly used neighbourhood functions include Mexican hat and triangle function.

4 **Synaptic Adaption:** The competitive neurons in the neighbourhood decrease their values of the discriminant function for the input pattern by adjusting the connection weights, such that the response of the winning neuron is made more similar to the input pattern.

1.1.2 Quantization error:

Quantization error QE is a widely used measure of the quality of a Self-Organizing Map, [9]. The squared distance between an observed data ***xi*** and its corresponding centroid ***G*** is a quantization error[10]. The sum of all these quantization errors overall training is the measure of the distortion,[9][10]. Formally:

$$QE = \frac{1}{N}\sum \left\| \overrightarrow{x_i} - \boldsymbol{G}_{\overrightarrow{x_i}} \right\| \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \text{equation (10).}$$

where N is the number of data-vectors
This error evaluates the fitting of the neural map to the data. Thus, the is we expect the optimal SOM to yield the smallest average quantization error[9].

1.1.3 Topographic error:

Topographic error TE measures the topology preservation in the map or how well the SOM modelled the structure of the input space. *The function c*ompares each best matching unit $b(x_i)$ to the second-best matching unit $b'(x_i)$ and *counts* an error if they are not next to each other. The total number of errors divided by the total number of data points gives the topographic error of the map. Formally:

$$TE = \frac{1}{N}\sum_{i=1}^{N} t(x_i) \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \text{equation (11)}$$

$$t(x_i) = \begin{cases} 0, & b(x_i) \text{ is next to } b'(x_i) \\ 1, & otherwise \end{cases} \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \text{equation (12)}$$

The topographic error focuses on how well each data point is mapped and not the entire distortion [11].

1.1.4 Instance Selection:

In supervised learning, a machine learning algorithm iterates through a training set, T, which is a collection of training examples called instances[12].
Instance selection is the selection of a subset from a dataset while maintaining the underlying distribution**,** such that the sampled subset has all the characteristics of the overall data population.

According to [13], from the statistical point of view, reduction of the training set size will not affect prediction accuracy of the final classifier when the conditional probability *P(c|x)* of predicting class *c* for given vector x remains unchanged when estimated from the original training set T and the set of prototypes P obtained from the data filtering process. There are several algorithms used for this selection task the one applied here is the Extended Local Density-based Instance Selection [14]).

1.1.4.1 Extended Local Density-based Instance Selection (XLDIS)

The XLDIS algorithm is an extension of an earlier proposed algorithm Local Density-based Instance Selection (LDIS) [14], the LDIS algorithm selects cases that have a high concentration of points near to them [15]. The algorithm assumes that the *denser* instance within a given neighbourhood can represent more information about that neighbourhood than its less dense neighbours.

The main difference between XLDIS and LDIS is that XLDIS selects the instances that have a higher *local density ordering* (LDO) within its neighbourhood.

The notion of density used here adapted from[16-19], is a measure of the spatial concentration of other instances within P around the instance *x*[14].

Formally:

$$D(x,P) = -\frac{1}{|P|}\sum_{y \in P} d(x,y) \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \qquad \text{Equation (13).}$$

When P is a subset of the whole dataset, Dens (x, P) represents the local density of x, considering the set P[14]. The LDO of a given instance ***x*** represents how dense x is, in comparison to the other instances within ***c(l(x)),*** the subset of P consisting of cases in the same class as ***x.***
As shown by [14], the algorithm for XLDIS is summarized below:

```
Algorithm 1: XLDIS algorithm
  Input: A set instances $T$ and the number $k$ of neighbors.
  Output: A set $S$, such as $S \subseteq T$.
  begin
      $S \leftarrow \emptyset$;
      foreach $l \in L$ do
          $toAvoid_x \leftarrow false$, for every $x \in c(l)$;
          Sorting $c(l)$ in a descending order, according to the $LDO$ of its instances;
          foreach $x \in c(l)$ do
              if $toAvoid_x = false$ then
                  $foundHigher \leftarrow false$;
                  foreach $y \in pkn(x,k)$ do
                      if $LDO(x) < LDO(y)$ then
                          $foundHigher \leftarrow true$;
                  if $\neg foundHigher$ then
                      $S \leftarrow S \cup \{x\}$;
                      foreach $y \in pkn(x,k)$ do
                          if $LDO(x) > LDO(y)$ then
                              $toAvoid_y \leftarrow true$;
      return $S$;
```

Figure (1) The XLDIS Algorithm

1.1.5 Distance or Similarity Metric:

The metric is the property of any set of elements characterized by another function called the distance between all pair of items [20].

An issue in KNN is the choice of an appropriate measure of distance or similarity for training and classification [21]. Thus, our implementation parameterizes this.
As stated by [22],
formally, a distance is a function D with nonnegative real values, defined on the Cartesian product ***X*** × ***X*** of a set ***X***. It is called a metric on *X* if for every $x, y, z \in X$:

- $D(x, y) = 0$ ***iff*** $x = y$ (the identity axiom)
- $D(x, y) + D(y, z) \geq D(x, z)$ (the triangle inequality);
- $D(x, y) = D(y, x)$ (the symmetry axiom).

A set *X* provided with a metric is called a metric space.
For example, the Hamming distance, a very intuitive and commonly used distance metric, represents the number of symbols needed to change a bitmap to another [23].
Hamming Distance: $D(x, y) = \#\{i: X_i \neq Y_i, i=1,\dots,n\}$ … … … Equation (14).

Other Metrics Include

- Cosine Distance
- Minkowski
- Euclidean
- City Block

1.1.6 K-Nearest Neighbor: This is an instance-based machine learning paradigm that gives an output Y to an input vector X by drawing the mode of outputs gotten from K instances drawn from the case base. This algorithm can also be called case-based reasoning [24].
According to (Oliver Sutton 2012), the purpose of the k Nearest Neighbours (kNN) algorithm is to use a database in which the data points are separated into several separate classes to predict the classification of a new sample point.
The Algorithm:

```
//Case = class representing each instance vector;
List< Case > caseBase = new ArrayList< Case > ( ); //Case base
Integer k;
List<Case> nearestNeigbours = new ArrayList<Case > ( ); //Case base
Case newCase = new Case()
for (Case case : caseBase) {
  //calculate distance(case, newCase) upon descriptive attributes
}
//select into nearest neighbours, k cases according to their distances to n;
//n.setClass(majority class in nearestNeigbours);
```

2.0 Literature Review:

The following are some existing recommendation systems leveraging collaborative filtering considered in this paper:

2.1 Entropy-Based Collaborative Filtering Algorithm: Personalized Recommender System Using Collaborative Entropy-Based Collaborative Filtering Technique, was proposed by [4], here the information-theoretic measure known as entropy to is used to estimate the notion of selective predictability, the term predictability used in this approach describes the level of consistency of the relationship between a pair of users over the commonly rated items of the two [4].
For each ordered pair of users *(m, a)*, at each of the rating levels *"x"* of *"m"*, the Subjective Predictability (*SP)* of each of the rating levels *"r"* is computed, for user *"m"* and recorded in a user-user matrix[4].

$$SP(r)^{x}{}_{m\to a} = \begin{cases} \frac{Y_{x,r}}{Y_x} if\ Y_x > 0 \\ \frac{1}{n}\ if\ Y_x = 0 \end{cases} \quad \dots \quad \dots \quad \dots equation(15)$$

Y = number of items commonly rated by *"m"* and *"a"*, $Y_{x,r}$ = number of items out of the commonly rated items of *"m"* and *"a"* where *"m"* and *"a"* have rated the item as *"x"* and *"r"* respectively and Y_x = number of items out of the commonly rated items of *"m"* and *"a"* where *"m"* has rated the item as "*x*",

$$Y_x = \sum_{r=1}^{n} Y_{x,r} \quad \ldots \quad \ldots \quad \ldots \quad equation(16)$$

Also, the predictability index *PI* at each of the rating levels *"x"* of *"m"* towards the active user *"a"* is computed and recorded in the same user-user matrix. To determine the rating of a user on an item a set of *"Fully Qualified Mentors"* is drawn from the dataset using the *PI*, then the subjective probabilities assigned by each of the fully qualified mentors to each of the rating levels are weighted by their respective predictability indices and aggregated to obtain the overall probability of each of the rating levels for the active user "*a*" for the target item *"v"* (denoted by $P(r)^{v}_{a}$) as shown[4]:

$$P(r)^{v}_{a} = \frac{\sum_{m=1}^{|FQM^{v}_{a}|}(PI^{x}_{m\to a} * SP(r)^{x}{}_{m\to a})}{\sum_{m=1}^{|FQM^{v}_{a}|} PI^{x}_{m\to a}} \quad \ldots \quad \ldots \quad \ldots \quad equation(17)$$

2.2 Quaternion Collaborative Filtering for Recommendation: The application of quaternion representations based on hypercomplex numbers in recommendation models was also proposed [6]. Here the user is represented by a quaternion:

$$H_u = U_u + V_u\, i + X_u\, j + Y_u\, k \ldots \quad \ldots \quad \ldots \quad equation(18)$$

And items are represented as:

$$H_i = P_i + Q_i\, i + S_i\, j + T_i\, k \ldots \quad \ldots \quad \ldots \quad equation(19)$$

Where $U_u, V_u, X_u, Y_u, P_i, Q_i, S_i, T_i \in \mathbb{R}_d$

and *i, j, k* satisfies the relation:

$$i^2 = j^2 = k^2 = ij\,k = -1 \ldots \quad \ldots \quad \ldots \quad equation(20)$$

We model the user-item item relationship with the Hamilton product $H_u \otimes H_u$ [6].
In the approach proposed by [6], Suppose that we have ***N*** items and ***M*** users, the preferences of users over items formulate a user-item interaction matrix $\Upsilon \in \mathbb{R}^{M \times N}$

We then model the collaborative filtering problem as the task of filling in the missing values in the interaction matrix.

2.3 K-Nearest Neighbor:
When applied to the task of collaborative filtering KNN, recommendation is computed first, by computing the similarities between the target user and all other users who have rated the target item using a suitable metric.

Then the prediction for the target item is computed using at k-closest users found from step one, and by applying a weighted average of deviations from the selected users' means.

This algorithm can also be applied by comparing all items to items the active user has rated,

Then recommending items most similar to ones the user favors in his ratings.

The performance of this system increases with the case base, consequently it is very time and memory expensive.

2.4 Single Value Decomposition: Another approach towards collaborative filtering is the use of single value decomposition, this a matrix factorization technique. A limitation being that typically collaborative filtering datasets are sparse, yet SVD requires a complete metric

2.5 Contribution

Compared to these algorithms SOM-XLDIS, provides an enormous data reduction while preserving accuracy. It is also extremely intuitive

3.0 Methodology

3.1 Data Collection:

The data used in this paper is the jester dataset made up initially of 100 rows representing movies and 36711 users. The rating is a real value between **-10.0** and **+10.0**. The dataset is associated with the below research by [25].

3.2 Preprocessing:

We transposed the dataset into that of 100 features and 36711 cases. We seek to limit the scope of the proposed model to just scalability of a recommendation system, not dealing with missing data hence we took out users with missing ratings resulting in a dataset that consisting of 14,116 ratings from users on 100 jokes.
The data is further normalized using min-max normalization, given as:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \text{ where } x \in X \quad \dots \quad \dots \quad \dots \quad equation(21)$$

The purpose of normalization is to provide a uniform scale across the different feature in the dataset. There are various types of normalization such as Min-Max, Z-score & Decimal scaling. Min-max produces values between 0 - 1 and maintains the linear relationship in the pre-normalized dataset.

3.3 Data Splitting and Cross-Validation

According to [26], Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to test the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being examined. The basic form of cross-validation is k-fold cross-validation.

We used an arbitrary K value of 5 and implemented as follows:
· The dataset is randomized to preserve its distribution during the split
· The dataset is split into five groups
· For each of the groups:

1. The group is set aside as test set
2. We develop the model using the four other groups
3. We test the model against the test set
4. The error is stored

The cumulative error of the model is computed as the average error from all the iterations.

3.4 Model Building:

- Initialize an *N* x *M* Self-Organizing Maps SOM.
- Cluster the dataset using the SOM training routine as follows [27]:
    a. Calculate the number of neurons in the competitive layer $\boldsymbol{M} = m_1 * m_2$.
    b. Initialize the $\boldsymbol{M}$ centroids w_i (0), $i = 1, \ldots, \boldsymbol{M}$.
    c. Initialize learning rate $\boldsymbol{\eta(0)}$**,** parameter $\boldsymbol{\sigma(0)}$, epochs counter $\boldsymbol{\kappa = 0}$ and repetitions counter $\boldsymbol{k = 0}$.
    d. For each epoch $\boldsymbol{\kappa}$ do the following steps for $\boldsymbol{n} = 1, \ldots \boldsymbol{N}$:
        i. We set vector $\boldsymbol{X^n}$ as the neural network input.
        ii. Select the winner neuron $\boldsymbol{m}$.
        iii. Update the weight vectors for all neurons in the neighbour of the winning neuron.
        iv. $\boldsymbol{k} = \boldsymbol{k}+1$.
    e. Gradually decrease the learning rate $\boldsymbol{\eta(k)}$.
    f. Gradually decrease the width $\boldsymbol{\sigma(k)}$.
    g. Check for termination. If not set $\boldsymbol{\kappa = \kappa + 1}$ and return to step d.
- We use the Extended Local Density-based Instance Selection algorithm to select Surrogate instances to summarize the dataset.

3.5 Prediction:

We perform prediction using KNN. The model above returns the concise form of the initial dataset, to predict the rating $\boldsymbol{x_{i,j}}$ of user $\boldsymbol{i}$ on a joke $\boldsymbol{j}$ **:**

- Define the vector, $\boldsymbol{X^i}$ from other jokes rated by ***i.***
- For each user in the dataset $\boldsymbol{n}$:
    - Fetch users with complete ratings for jokes in $\boldsymbol{X^i}$ and the joke ***j,***
    - Define the vector, $\boldsymbol{X^n}$ from jokes in $\boldsymbol{X^i}$ rated by user ***n.***
    - Find the similarity between $\boldsymbol{X^n}$ and $\boldsymbol{X^i}$ **and** record the value
- Select the $\boldsymbol{k}$ closest vectors to $\boldsymbol{X^i}$ ***(k is a system variable).***
- For each selected vector
    - Get the value $\boldsymbol{x_{n,j}}$ of user $\boldsymbol{n}$ on a joke $\boldsymbol{j}$ and record the value
- $\boldsymbol{x_{i,j} = mode(x_{n,j})}$

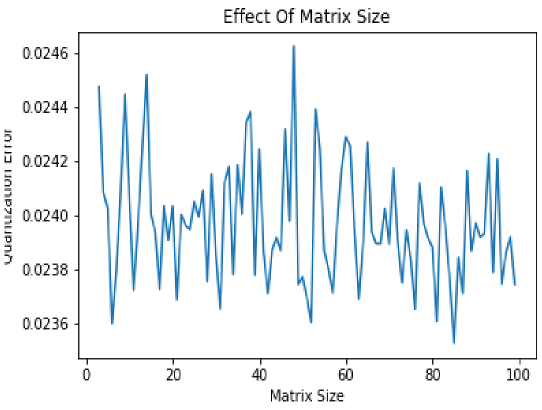
4.0 Evaluation:

We evaluate this model in both stages, during the offline stages the deciding matrices are the quantization error and the topographic error, during prediction we make use of a percentage error metric defined as;

$$E = \frac{number\ of\ misclassifications}{total\ number\ of\ classifications} * 100 \ \ldots \quad \ldots \quad \ldots \quad equation(22)$$

4.1 Effect of matrix size:

The choice of matrix size is of enormous importance. We obtain these values experimentally.



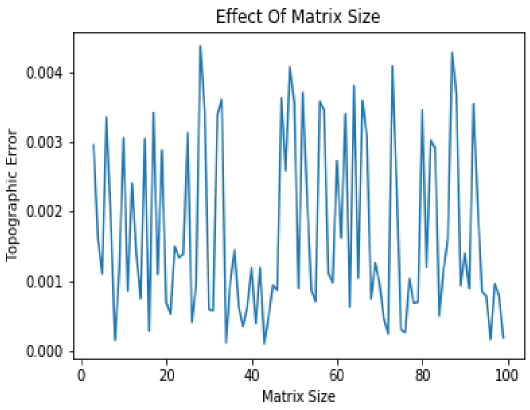


Figure (2) The effect of matrix size on Quantization error Figure (3) The effect of matrix size on Topographic error

The figures above show the variation between the topographic and quantization error and the size of the matrix ***(n*** x ***n)***. Our priority is the overall preservation and quality, we used the matrix size of (84 x 84), as this minimizes the quantization error.

4.2 Effect of the neighbourhood function:

The neighbourhood function mentioned earlier determines the rate of change of the neighbourhood around the winner neuron. We measured the consequence of the neighbourhood function across various matrix sizes.

Function	**Average QE**	**Min QE**	**Max QE**
Triangle	0.02294	0.0227	0.02326
Mexican Hat	0.02412	0.0234	0.02483
Bubble	0.022830	0.022571	0.023643
Gaussian	0.02192	0.02161	0.02413

Table (1.0) The effect of varying neighbourhood functions

As seen above, the classification error appears comparatively constant while varying the neighbourhood function types.

4.3 Classification Error:

Using a gaussian neighbourhood function and (84 x 84) sized matrix, we built the model. During online classification, we achieved an average classification error of 3.9% implies a classification accuracy of 96.1.

4.4 Effect of Similarity Metric And Neighbourhood Size:

Distance Function	**Neighborhood Size**					
	5	**10**	**15**	**20**	**25**	
Euclidean	15	11	20	16	12	**Error %**
Hamming	25	42	21	27	18	
Correlation	4	3	6	5	7	
Cosine	13	20	23	19	10	

Table (2.0) The effect of varying similarity metrics against various neighborhood sizes.
The table above shows that the correlation distance measure performed comparatively better on the model.

5.0 CONCLUSION

In this paper, we developed a scalable model for collaborative filtering using Self-Organizing Maps, Local Density-Based Instance Selection and the Nearest Neighbor algorithm. SOM-XLDIS is an efficient tool in condensing and providing high fidelity recommendation in minimal time. We established that the most significant parameter in preserving the neighbourhood topology is the SOM matrix size which we determined experimentally, and also that the similarity metric as well as the neighborhood size plays a significant role in the online recommendation.

REFERENCES

[1] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich Recommender Systems: An Introduction, Cambridge University Press, 2011.
[2] S. K. Gorakala, and M. Usuelli, Building a Recommendation System with R, Birmingham: Packt Publishing Ltd, 2015.
[3] Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999* (pp. 230-237). (Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999). Association for Computing Machinery, Inc. https://doi.org/10.1145/312624.312682.
[4] H. Chandrashekhar and B. Bhasker, Personalized Recommender System Using Entropy-Based Collaborative Filtering Technique, Journal of Electronic Commerce Research, 2011.
[5] M. A. Ali, K. B. Jumari, S. A. Samad Learning Vector Quantization algorithm as classifier for Arabic handwritten characters recognition. 2007. Proceedings of the 6th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 15-17
[6] S. Zhang, L. Yao, L. V. Tran, A. Zhang and Y.Tay Quaternion Collaborative Filtering for Recommendation Place: Press, 2019.
[7] D. Li, C. Chen, Z. Gong, T. Lu, S. M. Chu, N. Gu Collaborative Filtering with Noisy Rating Proceedings of the 2019 SIAM International Conference on Data Mining,2019.
[8] T. Kohonen, and T. Honkela, Kohonen Network, http://www.scholarpedia.org/article/Kohonen_network (retrieved 3/11/2019), 2007.
[9] E. A. Uriarte, and F. D. Martin, Topology Preservation in SOM World Academy of Science, Engineering and Technology 21, 2008.
[10] E. Bodt, M. Cottrell, M. Verleysen, Statistical tools to assess the reliability of self-organizing maps 2002.

[11] G.T. Breard," Evaluating Self-Organizing Map Quality Measures as Convergence Criteria".Open Access Master's Theses. Paper 1033. https://digitalcommons.uri.edu/theses/1033 (2017).
[12] D. R. Wilson, and T. R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, Netherlands: Kluwer Academic Publishers, 2000.
[13] M. Blachnik, and M.Kordos, Comparison of Instance Selection and ConstructionMethods with Various Classifiers. *Applied Sciences* (ISSN 2076-3417; CODEN: ASPCC7) 2020.
[14] J. L. Carbonera, An efficient approach, for instance-selection. Conference: International Conference on Big Data Analytics and Knowledge Discovery, 2017.
[15] J. L. Carbonera and M. Abel, A density-based approach for instance selection, IEEE 27th International Conference on Tools with Artificial Intelligence, 2015.
[16] Carbonera, J.L., Abel, M.: A cognition-inspired knowledge representation approach for knowledge-based interpretation systems. In: Proceedings of 17th ICEIS. pp. 644–649 (2015)
[17] Carbonera, J.L., Abel, M.: A cognitively inspired approach for knowledge representation and reasoning in knowledge-based systems. In: Proceedings of the 24th International joint conference on Artificial Intelligence (IJCAI 2015) (2015)
[18] Carbonera, J.L., Abel, M.: Extended ontologies: a cognitively inspired approach. In: Proceedings of the 7th Ontology Research Seminar in Brazil (Ontobras) (2015)
[19] Carbonera, J.L., Abel, M.: A novel density-based approach for instance selection. In: Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on. pp. 549–556. IEEE (2016)
[20] T. Kohonen, Self-Organizing Maps, Springer Science & Business Media, 2001.
[21] P. Schneider,M.Biehl, and B. Hammer, Adaptive Relevance Matrices in Learning Vector Quantization, Netherlands: *Neural computation*, *21*(12), 3532–3561. https://doi.org/10.1162/neco.2009.11-08-908, 2009
[22] M. Li, X. Chen, X. Li, B. Ma, and P. M.B. Vit´anyi, The Similarity Metric, ieee transactions on information theory, vol. 50, no. 12, 2014.
[23]A. Bookstein, V. A. Kulyukin and T. Raita, Generalized Hamming Distance, https://link.springer.com/article/10.1023%2FA%3A1020499411651, 2002.
[24] C.P.Ezenkwu, E.H Johnson and O.B. Jerome, Automated Career Guidance Expert System Using Case-Based Reasoning Technique ,Computing, Information Systems, Development Informatics & Allied Research Journal, 2017.
[25] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: A Constant Time Collaborative Filtering Algorithm, *Information Retrieval* 4, 133–151 (2001). https://doi.org/10.1023/A:1011419012209, 2001.
[26] P. Refaeilzadeh, L.Tang, H. Liu Cross-Validation. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, 2009.
[27] J. Salatas, Implementation of Competitive Learning Networks for WEKA, https://jsalatas.ictpro.gr/implementation-of-competitive-learning-networks-for-weka/ network (retrieved 3/11/2019), 2011.