A Lens Model Based Judge-Participant Benchmark Repository and Its Application to Select Ranking Methods

BY

NIHARIKA RAJENDRA HUBLI
B.E., Visvesvaraya Technological University, 2015

THESIS

Submitted as a partial fulfillment of the requirements for the degree of Master of Science in Industrial Engineering at the Graduate College of the University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

Houshang Darabi, Chair and Advisor
Michael J. Scott, Mechanical and Industrial Engineering
Ashkan Sharabiani, Exelon Group
This thesis is dedicated to my grandfather, S. S. Hubli whom I lost this year. He has and will be my strongest support and inspiration.
ACKNOWLEDGMENT

I would like to thank my advisor and chair of my defense committee, Prof. Houshang Darabi for his support and guidance to complete my thesis. His unwavering belief in my abilities has been a source of constant encouragement. I would also like to acknowledge the other members in my committee, Prof. Michael Scott and Ashkan Sharabiani for being a part of my final defense and playing a major role in the completion of this work.

I would like to wholeheartedly thank my family back home, for their constant support all through my difficult times and for always encouraging me. My father, Rajendra Hubli and my mother Sheela Hubli are together responsible for where I stand today. My friends have also played an important role in this work and I thank them with all my heart –Fazle Karim at Prominent Labs and Nishanth Bhat, for their constant help and support.
# TABLE OF CONTENTS

1. INTRODUCTION ........................................................................................................................................ 1

2. BACKGROUND ........................................................................................................................................ 7
   2.1 The Lens Model ................................................................................................................................. 7
   2.2 Score Aggregation and Ranking of Participants ............................................................................. 8
       2.2.1 Average Score Based Ranking ............................................................................................... 10
       2.2.2 Median Score Based Ranking ................................................................................................. 10
       2.2.3 Social Judgment Scheme (SJS) .............................................................................................. 11
       2.2.4 Z-score based ranking ............................................................................................................. 12
       2.2.5 Matrix Factorization .............................................................................................................. 13
   2.3 Multi-label Classification .................................................................................................................. 15

3. METHODOLOGY .................................................................................................................................... 17
   3.1 Simulating Full Score Matrices ....................................................................................................... 18
   3.2 Compare Performance of Ranking Methods .................................................................................. 23
   3.3 Build Method Recommender ........................................................................................................ 25

4. RESULT AND DISCUSSION .................................................................................................................. 29
   4.1 Behavior of Ranking Methods ....................................................................................................... 29
   4.2 Performance of the Method Recommender and Its Working ..................................................... 31

5. CONCLUSION AND FUTURE WORK ................................................................................................. 34

6. CASE STUDY: EXPO 2017 .................................................................................................................. 32
<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Summary of Ranking Methods</td>
<td>9</td>
</tr>
<tr>
<td>2. Evaluation of Model Performances</td>
<td>27</td>
</tr>
<tr>
<td>3. Features of the score matrix at EXPO 2017</td>
<td>33</td>
</tr>
<tr>
<td>4. Recommended Ranking Methods for EXPO 2017</td>
<td>33</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Three Phases to Implement the Proposed Framework</td>
<td>17</td>
</tr>
<tr>
<td>2. Structure of the judging system simulator based on the lens model framework</td>
<td>18</td>
</tr>
<tr>
<td>3. Steps to Generate Sparse Matrices</td>
<td>22</td>
</tr>
<tr>
<td>4. Procedure to Compare Ranking Methods</td>
<td>24</td>
</tr>
<tr>
<td>5. Calculation of Range of Acceptance</td>
<td>26</td>
</tr>
<tr>
<td>6. Variation of $\tau$ with matrix size and level of sparsity</td>
<td>30</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

SJS  Social Judgment Scheme
MF   Matrix Factorization
SUMMARY

There exist competitions, where multiple participants are evaluated by judges or field experts. These evaluations are further used to rank and award those participants with the best performance. While several methods exist to do so, each competition introduces its own set of unique features like, number of participants, number of judges involved, range of scores obtained, number of judges assigned to each team etc. These features along with inherent problems introduced by human error and inconsistencies raises a question of the most appropriate ranking method to be used.

A study of these ranking methods has been carried out to identify and find a solution to present the winner. This has been accomplished by building a method recommending model that studies these features of a judging system and suggests the most suitable ranking method. The method recommending model is based on a machine learning approach called multi-label classification that has been built on a simulation bed. The simulation bed consists of score matrices mimicking real world competitions, built using the Lens Model framework.

The final model suggests ranking methods with an accuracy of 95%, hamming loss of 0.01 and F1 score of 0.99. This fairly accurate method recommending model can be used in the future to decide the best ranking method for a given competition. This framework has been implemented on the scores received at the annual senior design competition organized by the College of Engineering, University of Illinois at Chicago.
1. INTRODUCTION

Consider a setting where there are multiple participants that have to be assessed by multiple human evaluators. These evaluations are combined to reach a final consensus by ranking the participants or decide the best participant. Such a setting closely resembles a group decision making problem that can be found in various real world applications like assessing academic articles and journals (Cook, Golany, Penn, & Raviv, 2007; Cook, Raviv, & Richardson, 2010), customer review based ranking of advertising models (Chen, Cheng, & Huang, 2013), web based rating and rank aggregation (Beg & Ahmad, 2003), selecting the best alternative for water resources (Morais & De Almeida, 2012), selecting the most advantageous tender after expert evaluation (Tsai, Wang, & Lin, 2007), obtaining situational expertise in emergency situations from experts of varied backgrounds (Wang, Wang, & Martínez, 2017) and many more.

In this work we are inferring from group consensus methods and applications to solve a real world problem. It centers around competitions where participants compete for a final winning position. These participants are evaluated by multiple judges, assumed to be experts in their respective fields. Their evaluation is based on various criteria either previously stated by the organizer or those of personal importance to the judge and in most cases a combination of both. Let us call this a ‘judging system’ (Athalye, 2015), comprising of \( n \) participants who are evaluated by \( m \) judges. An example of such a judging system is the EXPO conducted by the College of Engineering at the University of Illinois at Chicago (“EXPO: Designing Our World”, n.d). Teams of students in their senior year present their senior design projects to judges from varied backgrounds from the industry and academia. The judges are assigned to each of the participating teams by matching the major of the students in the team with the judge’s area of research or work. Judge’s evaluation is
recorded in numerical ratings on a scale of 1 to 100. The judges are instructed to evaluate teams based on two main parameters; technical quality and presentation quality. Once the judges complete their evaluations, these scores are recorded into a single score matrix. The score matrix is then used to find the average score of each team and this score is then used to rank and finally award the best teams in multiple categories. In the sections that follow, we explain the details of the judging system, review the related literature and state the problem statement.

1.1 Collecting Responses

Every judge performs a thorough evaluation of every participant assigned to him/her. If to be treated as ecology of human behavior, it is safe to say that the participant’s performance is the stimuli and the final evaluation of the judge is the response to this stimulus which is to be recorded. In such applications, paired comparisons, ranked ordering and numerical ratings are the most common methods of collecting responses (Brown & Daniel, 1990; Chen et al., 2013). Paired comparison provides the easiest mechanism to collect these reactions as the judge will have to choose between two participants at once and indicate the better performing participant over the other. Rank ordering procedure involves placing a set of participants in the order of best performance to the worst performance. To summarize the functioning of these procedures, according to (Brown & Daniel, 1990), “At their most basic level, these two procedures produce ordinal data, based on the proportion of times each stimulus is preferred in the paired-comparison case, and on the assigned ranks in the rank ordering procedure”. The most popular form of collecting judge’s responses would be the numerical rating system where the judges are simply asked to provide a single score based on the performance of the participant on a continuous scale within a predetermined range. Low scores mean poor performance and high scores mean higher levels of performance. The definitive scope of this research is to use only the collected matrix of scores to rank participants.
Numerical ratings possess the property of conveying maximum amount of information by providing the opportunity for judges to numerically indicate the differences in their perception of the participant’s performance (Brown & Daniel, 1990; Moore, 1975). These ratings allow judges to record their responses without directly comparing with other participants at every step (Conklin & Sutherland, 1923) and to clearly differentiate between similar performing participants (Maio, Roese, Seligman, & Katz, 1996). This work attends to those applications that collects responses of evaluation in the form of ratings, as it gives access to the intricate variabilities in a judging system. It is only advantageous to keep the collected scores in the rating format than to convert it to different forms as it would risk reduction of information and validity (Johnson, Sallis, & Hovell, 1999; Maio et al., 1996) for further analysis.

1.2 Challenges to choose a suitable ranking method

The numerical score that a judge assigns to a participant consists of two parts, (1) the true score of the participant (2) the judgment criterion of the evaluating judge (Brown & Daniel, 1987). Assume that matrix $S$ consists of the scores provided by the judges to all the participants. As judges enter the judging system, they bring a certain criterion (also addressed as judgment policy (Hastie & Kameda, 2005)) with them that is inherent to the nature of the judge. This is called judgment criterion (Brown & Daniel, 1990; Cooksey, 1996a). While the true score is solely dependent on the performance of the participant, the judgment criteria depends upon the factors analogous to the judge, like toughness, knowledge about the topic in discussion, time of the day when the evaluation is conducted etc. The judgment criteria of various judges us subjected to many inconsistencies caused due to human nature. The end-point problem causing the judge to perceive the scoring scale at unequal intervals, lack of consistency when the same judge is asked to evaluate the same participant at a different time or order and the perceptual and criterion shift caused due to the change in order of evaluation are some sources of human error (Brown & Daniel, 1990). When judges are assigned a
large number of participants for evaluation, they are subjected to making repeated judgments over a long period of time. This results in a phenomenon called decision fatigue causing a reduction in decision quality (Pignatiello, Martin, & Hickman, 2018). Although these are valid inconsistencies, they are not discussed in this work as these are problems related to the set up and organization of the judging system.

A major challenge to select the most suitable ranking, along with the inconsistencies caused by human nature is due to unknown information. In most judging systems, all judges do not evaluate all the participants. This leads to a sparse score matrix, $S$ where the scores of judges not assigned to evaluate certain participants are unknown. In such a case, the participants are not subjected to the judgment criterion of all the judges. While some participants experience judges with a particular judgment criterion and human inconsistencies, the other participants might experience a different nature of evaluation all together. For example, consider a tough judge, (whose scores are generally lower than the average scores in the score matrix) who evaluates participants A. Even if A has been awarded the highest score in the score matrix by most of the judges, the tough judge’s evaluation could possibly affect the final ranking considerably. If the tough judge had evaluated all the participants, the final outcome would have complete information and an average of these scores would contain the same judge’s criterion through all the participants.

Comparison of alternatives by assigning numerical preferences comes with caveats and peculiarities, especially because of its close resemblance to a social choice problem (Scott & Antonsson, 1999). Without contradicting existing theories of preference aggregation, this work compares the behavior of existing ranking methods to varied levels of sparsity of score matrix, $S$. It is accomplished using a simulation bed containing complete information and the establishment of a true rank condition explained in the sections that follow.
1.3 Ranking of Participants and the Machine Learning Approach

There are multiple ways in which the matrix $S$, can be used to determine the ranking of participants. The scores are initially aggregated into a single value for every participant and ranked where the participant with the highest aggregated score is given the lowest rank value and the participant with the lowest aggregated score receives the highest rank value. The ranking methods under examination in this work have been divided in three categories. Two of them use scores as they are recorded: (1) average ; (2) median. Two more methods transform these scores by considering the centrality of judgment: (3) Social Judgment Scheme Model (Davis, 2014); (4) Z-Score Ranking (“LISEF Judges - Z-Score Ranking,” n.d.). Another classification of methods could be attributed to those that use statistical tools to predict unknown scores in the score matrix and further rank participants: (5) Matrix Factorization (Koren, Bell, & Volinsky, 2009).

The defined matrix $S$ contains several disparities that occur due to varying judgment criteria and procedural complexities as mentioned in section 1.2. It is important to note that every score matrix form a judging system has inherent features like the average and median of scores awarded, range of scores, number of participants and judges etc. There is no existing mechanism that allows a scientific comparison between methods to facilitate an informed choice of the most suitable ranking method for a given score matrix. Limited work has been accomplished in the area of benchmarking mechanisms that can compare the performance of these ranking methods. This work creates a framework where the comparative performances of these ranking methods are established using an association metric called Kendall’s $\tau$. The framework uses the concept of a true rank phenomenon recorded using a simulation bed. The simulation bed, called the judge-participant benchmark repository, consists of 70,000 score matrices that have been built to mimic the interaction between judges and participants as in a real world judging system. Each score matrix in the repository has
been simulated using the Lens Model framework (Cooksey, 1996b). The repository plays a central role to study, compare and select suitable ranking methods for a given score matrix without having to implement them individually. A training dataset created using this repository has been used to recommend suitable ranking methods for a given score matrix by building a multi-label classifier, called the method recommender.

The following sections explain the various aspects and stages involved in accomplishing the above stated objective. Section 2 provides a brief background of the Lens Model framework, ranking methods under examination and the multi-label classifier. Section 3 discusses the methodology, explaining the various steps involved in building judge-participant benchmark repository, comparison of ranking method performances and construction of the method recommender. Section 4 explains the behavior of the ranking methods in response to the intrinsic features of the score matrices along with the model selection and evaluation of the method recommender. Section 5 concludes this topic of discussion along with insight for future work.
2. BACKGROUND

As established earlier, the score assigned by every judge consists of two parts: a true score and a judgment criterion where the true value is related to the nature of the stimuli (participant) and judgment criteria is related to the nature of the judge. Given that we will be taking this theoretical position established in section 1.2, consider the matrix $S$ of shape $(m \times n)$ consisting of $m'$ judges evaluating $n'$ participants. Due to the components of every score entered in this matrix, it is expected that the rows and columns follow particular trends. The trend along the each row reveals the judgment criteria of the judge and the trend across each column reveals the true scores of participants. It is important that the simulated matrices maintain these patterns. For such a requirement, Brunswick’s lens model provides a framework that can be used to systematically generate interactions between judges and the participants, referred to as the decision maker and its decision task environment in social judgment theory (Cooksey, 1996b).

2.1 The Lens Model

The Lens Model provides a clear representative framework to study human judgment. It represents human judgment as a linear combination of available information from the environment and the judgment criterion. It has been widely used in complex decision making applications as in clinical psychology (Cooper & Werner, 1990; Ruben & Hall, 2016), decision making in education and medical applications (Chung, Wong, & Yang, 2000; Shulman & Elstein, 1975), study of lie detectors (Hartwig & Bond, 2011) and social welfare (Brehmer, 1976; Dalgleish, 1988). The approach of generating a simulation bed and further using it to test the score aggregation and ranking methods has been inspired by the works of (Bisantz et al., 2000; Hastie & Kameda, 2005).
The simulation used in this work is a straightforward implementation of Egon Brunswick’s probabilistic functionalism (Brunswick, 1955). The initial theory of probabilistic functionalism sowed the seed of research in human judgment by studying the interactions between ecology (environmental factors) and the decision maker together using parallel concepts (Cooksey, 1996a) in conditions of uncertainty. The Lens Model has been created as a framework of this theory by Hammond, Stewart, Brehmer & Steinmann (1986). The ecology and the decision maker are considered as two end nodes of the lens model (Cooksey, 1996b) and they interact with each other using proximal cues to produce a functional response to a specific stimulus. These cues are probabilistically linked to the ecology in the form of a distal criterion. On the other end, the organism uses its judgment criterion to provide weights to the perceived cues and linearly combine them (cue utilization) to produce the final response. The name “probabilistic functionalism” comes from the theory that the ecological validities and the cue utilization validities are only probabilistically linked to the proximal cue (Cooksey, 1996a, 1996b). The Lens Model facilitates a clear framework of this theory by depicting all the above mentioned interactions. As described by Cooksey (1996b), lens model acts as a “conceptual vehicle” (p. 141) to the Brunswick’s probabilistic functionalism. Section 3.1 illustrates its use specifically in the judging system environment.

2.2 Score Aggregation and Ranking of Participants

Consider matrix $S$ of size $(m, n)$ with $m$ judges and $n$ participants. Each judge $i$, assigned to participant $j$ provides an evaluation score, $s_{ij}$. Since all judges do not evaluate all participants, matrix $S$ is not full ie., it does not contain all scores $s_{ij}$. Using the score matrix $S$, each ranking method combines $s_{ij}$ to a single value named $G_j$ for every participant. The value of $G_j$ is then used to rank these participants. The procedure followed by the five ranking methods under examination to combine scores $s_{ij}$ to $G_j$ has been summarized in Table 1.
Table 1 Summary of Ranking Methods

<table>
<thead>
<tr>
<th>Ranking Method</th>
<th>Working of the Ranking Method</th>
<th>Transformation of Collected Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>Each participant is evaluated by the assigned judges and each participant’s average score is used for ranking.</td>
<td>No change</td>
</tr>
<tr>
<td>Median</td>
<td>Each participant is evaluated by the assigned judges and each participant’s median score is used for ranking.</td>
<td>No change</td>
</tr>
<tr>
<td>Social Judgment Scheme</td>
<td>Each participant is evaluated by the assigned judges and each judge is assigned a weight with respect to the discrepancy magnitude between judges. The weighted average of each participant is used for ranking.</td>
<td>No change</td>
</tr>
<tr>
<td>Z-Score</td>
<td>Each participant is evaluated by the assigned judges and each score is subtracted by the mean and divided by the standard deviation of all the scores awarded by the respective judge to convert to a $z$-score. The participant’s average $z$-score is used for ranking.</td>
<td>Transformed to $Z$-scores</td>
</tr>
<tr>
<td>Matrix Factorization</td>
<td>Each participant is evaluated by the assigned judges and if there exists unknown scores, these are calculated using stochastic gradient descent to minimize error. The participant’s average scores are used for ranking.</td>
<td>Depends on the performance of stochastic gradient descent</td>
</tr>
</tbody>
</table>
2.2.1 Average Score Based Ranking

The most commonly used method to calculate the final score for competing participants is by simply finding the average of score provided by evaluators for each participant:

$$\forall j \ G_j = \frac{1}{m'} \sum_{i=1}^{m'} s_{ij} \quad (1)$$

The value of $s_{ij}$ is given by the score that judge $i$ awards participant $j$ and $m'$ is the number of judges out of $m$ judges who evaluated participant $j$. The aggregated score $G_j$ is then used to rank participants. The procedure followed is simple and computationally efficient. While it provides certain information about how different judges rate participants, it also generalizes the behavior of judges in cases of anomaly. It is an ongoing discussion as to the level of overgeneralization caused by averaging scores in judging systems where the evaluations given by judges are not nearly the same. Although, certain works show that when scores are averaged over a reasonable number of judges, the final score is robust to varying scale interval configurations (discussed in section 1.2) and individual judging criterion (Brown & Daniel, 1990).

2.2.2 Median Score Based Ranking

The method of median score based ranking involves selecting the midpoint in an ordered set of scores for every participant that has received evaluations. The median rating for each participant becomes its aggregated score to be ranked. The underlying procedure involves arranging scores in ordinal format and treating them as ranks instead of considering numeric degree of difference between scores (Brown & Daniel, 1990). This method, like average based ranking fails to capture those evaluations that are abnormal as there is no numerical difference measure observed between judge’s scores.
2.2.3 Social Judgment Scheme (SJS)

Social Judgment Scheme came into existence to solve challenges related to group decision making applications where a group of individuals express their preferences in terms of continuous quantitative judgments to reach a final consensus (same as the judging system in discussion). This consensus model addresses the cognitive process involved, by considering interpersonal influences between decision makers with respect to the final outcome (Davis et al., 1997). The work of (Davis, 2014) introduces the concept of \textit{discrepancy magnitude} as the distance between the positions that judges take during this process. As the discrepancy between judges increases, there is an exponential decay in \textit{interpersonal influence} (Davis et al., 1997). This exponential decay is considered in the final group decision process where a weight is assigned to every judge, as explained by Davis et al, (1997), “..is a function of the total discrepancies between that person and everyone else” (p. 141) as shown in equations (2) and (3). Intuitively, each judge \(i\)’s influence on the final outcome of the competition depends upon the centrality of the score that he/she assigns with respect to other judge’s scores for the same participant.

Let the aggregated score for each participant \(j\) be \(G_j\) (Davis, 2014; Davis et al., 1997),

\[
\forall j \quad G_j = \sum_{i=1}^{m'} c_i s_{ij} \quad (2)
\]

where \(c_i\) is the weight assigned to each judge with respect to the scores assigned by all judges for each participant \(j\) and \(s_{ij}\) is the score judge \(i\) assigned to participant \(j\) in matrix \(S\). The weight \(c_i\) is given by,

\[
\forall j \quad c_i = \frac{\sum_{t=1}^{m'} f(|s_{ij} - s_{ij}'|)}{\sum_{i=1}^{m'} \sum_{t=1}^{m'} f(|s_{ij} - s_{ij}'|)} \quad (3)
\]
The function $f(\left|s_{ij} - s_{i'j}\right|)$ is called the social influence function, as mentioned before is given by Equation (4). The value of $\theta$ in this experiment has been assigned a value of 1.00 as in the work of Davis (2014) and Davis et al., (1997). The final values of $G_j$ are then used to rank participants in the matrix.

$$\forall j \ f(\left|s_{ij} - s_{i'j}\right|) = e^{-\theta(\left|s_{ij} - s_{i'j}\right|)}$$  \hspace{1cm} (4)

### 2.2.4 Z-score based ranking

The method of z-score transformation finds application in various contexts in machine learning (Al Shalabi & Shaaban, 2007; Al Shalabi, Shaaban, & Kasasbeh, 2006) as a procedure followed for data pre-processing. Its advantage includes simplicity and robustness to scales that are not uniform across data features. It has also been used in many applications similar to that in this work, where competitors are to ranked based on evaluations provided by judges in a science and engineering fair (“LISEF Judges - Z-Score Ranking,” n.d.)

The scores $s_{ij}$ assigned to participants are first transformed into a z-score using the standard formula,

$$z_{ij} = \frac{(s_{ij} - \mu_i)}{\sigma_i}$$  \hspace{1cm} (5)

where $z_{ij}$ is the new z-score for participant $j$, $\mu_i$ is the mean and $\sigma_i$ is the standard deviation of scores assigned by judge $i$. The average z-scores obtained are treated as final aggregate scores for each team and further ranked:

$$\forall j \ G_j = \frac{1}{m'} \sum_{i=1}^{m'} z_{ij}$$  \hspace{1cm} (6)
The z-score method gives a clear picture of the centrality of every judge’s score with respect to all scores received by the participant. As observed by Brown & Daniel (1990), “Two sets of ratings will necessarily produce identical mean Z-scores only if the sets of ratings are perfectly correlated (if the ratings of each observer of one set are linearly related to all the other observers of that set and to all observers of the other set)” (p. 9).

2.2.5 Matrix Factorization

The method of matrix factorization gained popularity and proved to be advantageous in applications similar to the one stated in this work after its use in the famous Netflix Prize Problem (“Netflix Prize,” n.d.). In the year 2006, Netflix (a DVD renting company at the time) challenged the machine learning community to build a model that can beat their current recommender system (Bennett & Lanning, 2007). Using the ratings provided by users for movies that they watched, teams were formed to build a model that can most accurately predict unknown ratings and further recommend new movies to the user. Similar to our problem, the percentage of user ratings available was extremely small, compared to the number of items (movies) that can be recommended. Matrix factorization was a widely used approach by the winning teams in the competition to build an advanced recommender system. This method has been used in many applications due to proven levels of flexibility for large, real life problems along with good scalability and predictive accuracy (Koren, Bell, & Volinsky, 2009). A simple adaptation of this method has been used in the discussed problem. This particular variation of the method is called ‘Regularized Incremental Simultaneous Matrix Factorization (RISMF)’ and has been explained briefly in this section from the works of Takács, Pilászy, Németh, & Tikk, (2008) ¹

¹ With changes in notations in context to the problem in discussion
The idea of matrix factorization is to use the sparse score matrix $S$ and find two latent ($K = 2$) matrices $A$ and $B$, whose dot product gives a matrix $\hat{S}$, containing all scores, in other words, a full matrix. Matrices $A$ and $B$ are derived as a result of ‘implicit feedback’ (Koren et al., 2009) or as latent features that consist of inherent patterns of judge’s scores. In other words, each row in $A$ represents the strength of the association between a judge and the features and each row of $B$ would represent the strength of association between a participant and features. Therefore, their dot product is a valid output as it maintains these patterns along with predicting unknown scores. Consider matrices $A$ (of size $m \times 2$) and $B$ (of size $n \times 2$) whose dot product approximately equals to $\hat{S}$,

$$\hat{S} \approx A \times B^T$$  \hspace{1cm} (7)

In order to find these matrices $A$ and $B$, they are first initialized to random numbers and iteratively updated until the difference between $\hat{S}$ and $S$ attains a value below a predefined minimal threshold, initialized to 0.001. This method is called gradient descent. The values in the matrices of $A$ and $B$ are updated using the below equation,

$$a_{ik}' = a_{ik} + \alpha(2e_{ij}b_{kj} - \beta a_{ik})$$

$$b_{kj}' = b_{kj} + \alpha(2e_{ij}a_{ik} - \beta b_{kj})$$  \hspace{1cm} (8)

where $a_{ik}'$, $b_{kj}'$ are the new updated values, $\alpha$ is the learning rate and $\beta$ is the regularization parameter, introduced to avoid overfitting. The values of $\alpha$ and $\beta$ has been initialized to 0.0002 and 0.002 in this work after analyzing best correlation measures with respect to the true rank benchmark (explained in section 3.2). The matrix factorization function has been implemented using the Python
code formulated by Yeung (2010) that performs the same set of steps as explained above and as shown in (Koren et al., 2009; Takács et al., 2008).

Once the final full matrix $\hat{S}$ is obtained, the aggregated score, $G_j$ is calculated using equation (9) and further ranked,

$$\forall j \quad G_j = \frac{1}{m'} \sum_{i=1}^{m'} \hat{s}_{ij} \quad (9)$$

### 2.3 Multi-label Classification

The process of classification in machine learning is to learn from a training set to classify objects into predefined labels. Each of these objects has its own features as input variables. A single label classification algorithm assigns a single label to each object resulting in a single output variable. Whereas in multi-label classification problems, the classifier uses features as input and classifies the object into more than one label. This results in more than one output variable (Tsoumakas & Katakis, 2007). It is widely used in applications like classifying objects in an image (Zhou & Zhang, 2007), ensemble based integration of classifiers to predict protein function (Yu, G., Rangwala, H., Domeniconi, C., Yu, 2013) and classification of music into more than one emotion (Trohidis & Kalliris, 2008).

There are several approaches to build a multi-label classification model. They are classified as problem transformation, adapted algorithm and ensemble methods (Madjarov, Kocev, Gjorgjevikj, & Džeroski, 2012; Szymański & Kajdanowicz, 2017). Due to its efficiency and simple interpretation, this work mainly uses the problem transformation approach where the multi-label classification problem is converted into one or more single label classification problems (Alazaidah, Thabtah, & Al-Radaideh, 2015). Once, transformed a base classifier is used in conjunction to
perform the classification task. While the works of Alazaidah et al., (2015) and Tsoumakas & Katakis (2007) explain each of the problem transformation methods with an example, here is a brief description:

1. Binary Relevance (Boutell, Luo, Shen, & Brown, 2004) method uses a one-against-all strategy where the original dataset is transformed into separate datasets based on the number of labels. It further implements the underlying base classification algorithm by treating these datasets independently (Alazaidah et al., 2015; Sorower, 2010) which can be an advantage in some while a disadvantage in cases with label dependencies.

2. Classifier Chains (Read, Pfahringer, Holmes, & Frank, 2011) is an extension of the binary relevance method where the classification problem is transformed into a single label classifier by considering the predictions of the previous binary relevance transformation as input for the next binary relevance classification (Zhang & Zhou, 2014). This approach considers label dependencies in the transformation process.

3. Label Powerset considers each unique label vector as one class during transformation. The base classifier uses this transformed data and treats each class independently. A proven disadvantage of this method is that a dataset consisting of a large number of classes with fewer samples could lead to class imbalance (Sorower, 2010).

All of the above problem transformation methods along with certain base classifiers have been implemented in this work to arrive at the best performing method recommender. Multi-label classification, due to its comparatively complex nature, requires a different set of metrics from classical classification problems. The performance of the classifier is determined using accuracy, hamming loss and F1 score as metrics for comparison (Alazaidah et al., 2015; Madjarov et al., 2012; Tsoumakas et al., 2007).
3. METHODOLOGY

The framework to compare ranking methods and select the most suitable method for a given score matrix containing numerical ratings is accomplished in three phases as shown in figure 1. The first step is to simulate a full matrix $S$ to achieve a state of complete information. In matrices where all judges do not evaluate all participants, only a small percentage of the total scores are available, leading to a sparse matrix $S$. In such a case, it is seen that the participants are not subjected to the judgment criterion of all the judges. For example, consider a score matrix where the average of scores is 50 on a scale of 100. If there exists a judge whose average score is only 30 across all the participants that he/she evaluated, any participant evaluated by this judge receives a low aggregated score. But if all the participants are subjected to the low scoring behavior of this particular judge, we now have complete information. This full matrix, with complete information is used as benchmark to compare the discussed ranking methods.

![Figure 1: Three Phases to Implement the Proposed Framework](image)

Figure 1 Three Phases to Implement the Proposed Framework
3.1 Simulating Full Score Matrices

The full score matrices contain evaluations of all judge-participant interactions. For the purpose of simulation and further analysis, it is assumed that the behavior of each judge remains the same throughout the course of the judging process, across all participants. This would result in the full score matrix $S$, showing column-wise trend of the participant’s performance and a row-wise trend of the judge’s judgment criterion. The true score of each participant $P_j$, is used to generate score $s_{ij}$ as the evaluation score provided by judge $i$ to participant $j$ in the matrix $S$. Proximal cues, $C_{jk}$ for participant $j$ and cue $k$ are generated using error values $e_{jk}$ and recorded in matrix $V$. The judgment criterion of a judge $i$ for cue $k$ is given by $w_{ik}$ and is recorded in matrix $W$. The perceptual error for a judge $i$ while perceiving cue $k$ to evaluate participant $j$ is given by $e'_{ijk}$.

![Figure 2: Structure of the judging system simulator based on the lens model framework adapted from “The Robust Beauty of Majority Rules in Group Decisions” by R. Hastie and T. Kameda, 2005, Psychological Review, 112(2), p. 499](image-url)
The process begins by generating the true scores of the participants, $P_j$. It is generated as a random number from a normal distribution $N(50,15)$. The use of normal distribution ensures that all scores and uniformly distributed around the mean, generating low and high values. The value of the true score is then used to generate each of the proximal cues $C_{jk}$ using equations (1), (2) and (3) for $k = 1,2,3$ respectively. While cues are probabilistically linked to the true score for each team as shown in figure 2, the true scores by themselves are in no way directly exposed to the judge. The error values, $e_{jk}$ are generated from normal distributions $N(10,3)$, $N(10,6)$ and $N(10,10)$ respectively for each cue.

\begin{align*}
C_{j1} &= P_j + e_{j1} \quad (1) \\
C_{j2} &= P_j + e_{j2} \quad (2) \\
C_{j3} &= P_j + e_{j3} \quad (3)
\end{align*}

The simulator in this work uses three cue values. Owing to the basic structure and probabilistic nature of the Lens Model, the specific number of cues chosen does not affect the behavior of simulation in this application. The values of proximal cues for each participant results in the formation of matrix $V$ of size $(n,3)$. This process depicts the left side of the Lens Model in figure 2. To simulate the evaluation of the judges, the judgement criterion of each judge is generated as weights given to respective proximal cues. It is generated a set of three random numbers that are standardized such that they added up to 1 (Hastie & Kameda, 2005; Dawes, 1979) as shown in equation (4). These weights are recorded in the matrix $W$ of size $(m,3)$ and this process depicts the right side of the Lens Model in figure 2.

\[ w_{i1} + w_{i2} + w_{i3} = 1 \quad (4) \]

The matrices $V$ and $W$ are combined to generate the score matrix $S$. In other words, each judge perceives the performance of the participant using the proximal cues, combines them by using
his/her own judgment criterion to provide an evaluation score. In the process, different judges make
different errors in perceiving these cue values (Hastie & Kameda, 2005). This error $e'_{ijk}$ has been
added to the value of $C_{j1}$. As shown in equation (5), to simulate the score $s_{ij}$, the $i^{th}$ row in matrix $V$
is combined with the $j^{th}$ row in matrix $W$ with respect to each combination of proximal cue value
and judgment criterion.

$$s_{ij} = w_{i1} (C_{j1}) + w_{i2} (C_{j2}) + w_{i3} (C_{j3}) \quad (5)$$

Various components of the Lens Model have been generated using random numbers from a
normal distribution. This causes some values of the final score to be negative at the end of all
computation. At the end of simulation, less than 1% of the matrices consisted of negative scores. As
the aim of generating these matrices is to mimic score matrices in the real world, those score matrices
with negative scores have been eliminated and further regenerated to ensure only positive values.

This framework has been used to generate a single matrix $S$ with $m$ judges and $n$
participants. The judge-participant benchmark repository consists matrices of various sizes. Hence,
this simulation is repeated for different matrix sizes to generate a repository that will be used in the
next phase to compare performances of ranking methods. The smallest matrix size in this repository
is a 5 by 10 matrix (5 judges evaluating 10 participants) and the largest size is a 141 by 146 matrix
totaling to 35 different sizes. The size of possible interactions between judges and participants range
from 50 (as in a 5 by 10 matrix) to 20586 (as in a 141 by 146) matrix. Starting from a 5 by 10 matrix,
4 judges and 4 participants are added at the same time at every step of matrix size progression. A
progression of 4 has been used as a small enough step size without loss of generality.
The primary challenge in this work is to compare existing ranking methods for sparse matrices. In other words simulate conditions where all judges do not evaluate all the participants. Level of sparsity is defined as the percentage of available scores in the score matrix. For example, if a score matrix of size \((50,100)\) contains 300 scores in total, the percentage of available scores will be 6\% (i.e., \(300/(50*100)\)). Therefore, high level of sparsity results in a sparse matrix containing a low percentage of available scores out of the total number of scores. On the other hand, low level of sparsity results in a fuller matrix containing high percentage of available scores out of the total number of scores. The lowest level of sparsity is initialized at 0 where the score matrix \(S\) contains all the scores and this remains the same for all matrix sizes.

Each full matrix in the judge-participant benchmark repository is subjected to 20 different levels of sparsity, starting from the lowest level of 0. At the level of maximum sparsity, every judge and every participant has to contain at least one score. If this condition is not accomplished, the size of the matrix will no longer be the same, affecting the consistency of data in the repository. This causes each matrix size having a different percentage of scores at their maximum level of sparsity. For matrix \(S\) of size \((m,n)\), the percentage of the total number of scores at the maximum level of sparsity is given by equation (6)

\[
\text{Percentage of Scores at Maximum Level of Sparsity} = \frac{\max(m,n)}{mn}
\]  

(6)

For each matrix size, starting from sparsity level of 0, an increase of 5\% of the total number of scores is created at every level until sparsity level of 19. This adds up to a total of 20 different conditions of sparsity for each matrix size including the full matrix at sparsity level 0. Starting from the full matrix \(S\) at sparsity level 0, \(S(0)\), scores at random locations in the matrix are removed until
the maximum level of sparsity is reached at $S(19)$. The matrix at $S(19)$ satisfies the condition of at least one score for every single judge and participant. Once, this condition is satisfied, 5% of the total number of scores are added in the next level, $l$ generating the matrix $S(18)$. The added scores are the original scores from the full matrix. This step of adding scores is repeated until $l = 18$. The creation of matrices with different sparsity is shown in figure 3. Therefore, we now have a total of 20 levels of sparsity for a particular matrix of a given size. For each matrix size, 100 such matrices have been generated, subjected to 20 levels of sparsity thereby, generating a judge-participant benchmark repository consisting of 70,000 matrices. This repository is made available and can be used for comparison and benchmarking purposes for future research (Hubli & Darabi, 2018)

Figure 3: Steps to Generate Sparse Matrices
3.2 Compare Performance of Ranking Methods

The use of full matrices in this work plays a primary role to compare existing ranking methods. By comparing the known with the unknown, the comparative performance of ranking methods has been achieved. As we mentioned earlier, most score matrices from real judging systems are sparse. Benchmarking of ranking methods is a challenge without knowing the true rank of participants. In this section, we define a true rank to establish the comparative performance of ranking methods.

A “true rank” is defined as the ranking obtained using the average based ranking method when all judges evaluate all participants. In other words, the rankings obtained when the matrix $S$ is full (i.e., $S(0)$) is called the true rank. Based on its definition, we can say that the true rank has been obtained using complete information of patterns in judgment criterion of judges and the performance of participants. With the known ranking established, any comparison with the true rank can be used as a performance metric of a given ranking method. The procedure followed for this systematic comparison is shown in figure 4. Every matrix $S(0)$ is compared with its own subset of sparse matrices, $S(l)$ where $l = 1,2,3 \ldots 19$. Every sparse matrix is used to rank participants using each of the five ranking methods in discussion. The ranking obtained is then compared with the true rank, $T$ using a “measure of association” for variables with a monotonic relationship called Kendall’s $\tau$ (“Kendall tau metric,” n.d.). The value of $\tau$ is derived by finding the number of concordant and discordant pairs (Kendall, 1938; “Kendall tau metric,” n.d.) and is therefore, sensitive to positional differences in rankings. The degree of association ranges from +1 to -1 and like other correlation coefficients, value of +1 signifies positive correlation, -1 signifies negative correlation and a value of 0 signifies no correlation between the rankings in comparison. Every time a rank obtained using a given ranking method is compared to the true rank, the value of $\tau$ is diligently recorded for each of
the 35 different matrix sizes, across 20 different sparsity levels and 100 trials for each combination. The value of $\tau$ is regarded as the comparative performance measure of ranking methods.

![Figure 4: Procedure to Compare Ranking Methods](image)

In the process of recording the values of $\tau$, several features of the score matrices have also been recorded for the next step of building the method recommender. These features are unique to every score matrix irrespective of its corresponding full matrix, $S(0)$. They include (a) number of judges; (b) number of participants; (c) total size of the score matrix ($mn$); (d) percentage of available at the level of sparsity; (e) maximum score; (f) minimum score; (g) average of scores; (h) median of scores; (i) average number of participants evaluated by each judge; (j) average number of judges evaluating each participant. The resulting training dataset consists of 70,000 rows with all the listed
features and the value of $\tau$ for each of the five ranking methods as columns. This training dataset is used to build the method recommender in the next phase.

3.3 Build Method Recommender

As shown in figure 1, the method recommender is now built using multi-label classification with the training dataset. The training dataset consisting of features of the score matrices and the values of $\tau$ for each ranking method. The process of multi-label classification uses the features of the score matrices in the judge-participant benchmark repository as input to produce an output variable communicating suitable ranking methods. The suitability of a given ranking method is decided by its measure of association with the true rank. In order to do so, the output variable is converted to a binary form where a ‘1’ indicates that the method is suitable and a ‘0’ indicates that a particular method is not suitable for the given score matrix. Therefore, the resulting vector of length 5 consists of 0s and 1s in the order of suitability with respect to each of the methods as shown,

[Average, Median, Social Judgment Scheme, Z-Score, Matrix Factorization]

The value of 0 or 1 is derived from the original dataset where the value of $\tau$ for every row is compared with a predetermined tolerance. The maximum value of $\tau$, for a given score matrix (each row of the dataset), $\tau_{\text{max}}$, is considered the best value and a tolerance of 5% from this value is considered as the range of acceptance for the respective ranking method. Difference of 5% in the value of $\tau$ causes less than 5 ranks in the largest pool of 146 participants to not match the true rank. This difference is considered small enough for the leniency of the method recommender. Those methods whose $\tau$ value lies within the range of acceptance is considered to be suitable for the score matrix, as shown in figure 5. Any value that lies in the range of acceptance is given a value of 1 and
those scores outside this zone is given a value of 0, performed for every row independently. The floating value of $\tau_{max}$ ensures that at least one ranking method is recommended for a particular score matrix.

\[
\tau_{max} - 0.05\tau_{max} \quad \tau_{max} \quad \tau_{max} + 0.05\tau_{max}
\]

\[
\text{Range of Acceptance}
\]

Figure 5: Calculation of Range of Acceptance

Note that the maximum value of $\tau$ is 1. If $\tau_{max}$ for a particular judging system is 1, the corresponding ranking method corresponding to it will be unanimously selected along with those methods with values in the range 0.95 and 1.00. The tolerance is one-sided when $\tau$ is at the maximum (1.00) or minimum values (-1.00). The training dataset now consists of all features recorded followed by a binary vector. Therefore, the features are the input variables while the binary vector is now the output or the target variable that ultimately communicates the suitability of a ranking method for the posed scored matrix.

The process of data handling, computations and experimentation in this work has been implemented using Python. The multi-label classifier has been built using the multi-label classification package by scikit-multilearn (Szymański & Kajdanowicz, 2017). The process of feature selection has been performed by initially converting the dataset into a label powerset problem (briefly introduced in section 2.3). With the unique labels converted to particular classes, $\chi^2$ statistic along with its p-value has been used to complete the feature selection process (Doquire & Verleysen, 2011; Trohidis & Kalliris, 2008). Due to its independence from the classification algorithm, this method of feature selection provided a numerical explanation to the most important features affecting
the selection of ranking methods. In this process, it was seen that the average and median of the score matrix do not show significant contribution to the output variable and hence have been eliminated. The maximum and minimum scores as features also showed lower association with the output variable but when transformed into a score range, proved to be more useful with a lower p-value. All other features previously listed have been retained.

With the final set of features, the training dataset has been divided into two parts: 70% for training and 30% for testing by stratified shuffling to ensure that equal number of samples are split between the two sets. The various base classifiers in conjunction with problem transformation methods as listed in the scikit multi-label module (Szymański & Kajdanowicz, 2017) have been implemented on the split data.

Table 2: Evaluation of Model Performances

<table>
<thead>
<tr>
<th>Problem Transformation</th>
<th>Base Classifier</th>
<th>Accuracy</th>
<th>Hamming Loss</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Relevance</td>
<td>Decision Tree</td>
<td>0.9495</td>
<td>0.0113</td>
<td>0.9940</td>
</tr>
<tr>
<td></td>
<td>Stochastic Gradient Descent Classifier</td>
<td>0.8877</td>
<td>0.0238</td>
<td>0.9875</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbors</td>
<td>0.9476</td>
<td>0.0117</td>
<td>0.9938</td>
</tr>
<tr>
<td></td>
<td>Naïve Bayes Classifier</td>
<td>0.8480</td>
<td>0.0685</td>
<td>0.9621</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>0.9499</td>
<td>0.0113</td>
<td>0.9940</td>
</tr>
<tr>
<td>Classifier Chain</td>
<td>Decision Tree</td>
<td>0.9515</td>
<td>0.0110</td>
<td>0.9942</td>
</tr>
<tr>
<td></td>
<td>Stochastic Gradient Descent Classifier</td>
<td>0.8720</td>
<td>0.0273</td>
<td>0.9856</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbors</td>
<td>0.9468</td>
<td>0.0119</td>
<td>0.9937</td>
</tr>
<tr>
<td></td>
<td>Naïve Bayes Classifier</td>
<td>0.8494</td>
<td>0.0706</td>
<td>0.9609</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>0.9504</td>
<td>0.0112</td>
<td>0.9941</td>
</tr>
<tr>
<td>Label Powerset</td>
<td>Decision Tree</td>
<td>0.9517</td>
<td>0.0109</td>
<td>0.9942</td>
</tr>
<tr>
<td></td>
<td>Stochastic Gradient Descent Classifier</td>
<td>0.8400</td>
<td>0.0399</td>
<td>0.9790</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbors</td>
<td>0.9467</td>
<td>0.0119</td>
<td>0.9937</td>
</tr>
<tr>
<td></td>
<td>Naïve Bayes Classifier</td>
<td>0.8454</td>
<td>0.0497</td>
<td>0.9730</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>0.9510</td>
<td>0.0111</td>
<td>0.9941</td>
</tr>
</tbody>
</table>

From Table 2, it can be concluded that decision tree coupled with classifier chain is the best performing model due to its high accuracy, F1 score and low Hamming loss. The decision tree in this model has the following parameters with gini index as the criterion: maximum depth of 5 and
minimum samples split of 2. Although, decision tree coupled with label powerset as the problem transformer has minutely higher levels of accuracy, it has not been chosen as the final model due to proven disadvantages of the label powerset method. Since this method converts a set of unique labels into a single class in the transformation step, it renders classification with very low accuracy when new data is added to the dataset (Tsoumakas & Vlahavas, 2007). With the aim of building a method recommender that is not localized only to the score matrices in the judge-participant benchmark repository the combination of decision tree with classifier chain as a problem transformer has been implemented. This multi-label classifier built as the method recommender is now be used to recommend suitable ranking methods for a given score matrix without having to implement any ranking procedure. Each recommendation is strongly backed by the ranking method’s measure of comparative performance.
4. RESULT AND DISCUSSION

4.1 Behavior of Ranking Methods

The training dataset built in phase 2 provides a clear insight into the behavior of ranking methods. The variation in the value of $\tau$ in comparison to the true rank condition at different levels of sparsity for different matrix sizes is studied using figure 6.

It is seen that the average, median and Social Judgment Scheme (SJS) methods behave similarly as the level of sparsity increases. Z-score and matrix factorization methods show a steep fall in the average value of $\tau$ at the maximum level of sparsity. The lowest value of $\tau$ is 0.2786 in comparison with all the methods and occurs with the use of z-score method. This is due to the unavailability of scores to find the capture the mean and standard deviation. Low values of $\tau$ in high levels of sparsity for matrix factorization occurs as it is unable to iteratively learn from existing scores to find unknown scores. Both these methods are heavily dependent on the number of known scores. As the number of scores increases, these two methods show a spike in their performances and even show higher values of maximum $\tau$ value in comparison to median and SJS methods. The rate at which the value of $\tau$ drops in increase in sparsity is also affected by the matrix size. It is observed that this drop is steeper in large matrix sizes and much more gradual in small matrices. While average, median and SJS do not show this behavior distinctly, z-score and matrix factorization methods show a sudden drop in $\tau$ to its lowest value at sparsity level 19.
Average, median and SJS as ranking methods prove to be robust to sparse matrices. With the availability of a certain number of scores, they show consistent behavior. As z-score based ranking method is heavily dependent on the centrality of an individual judge’s score, fewer scores fail to capture the judge’s scoring pattern. On other hand, matrix factorization based ranking performs better in all levels of sparsity except at the maximum level. It is therefore seen that each of these ranking methods behave very differently in specific score matrices and there is no way to tell that a particular method is suitable for a given score matrix. By using the measure of comparative performance, this work is able to accomplish this shortcoming.
4.2 Performance of the Method Recommender and Its Working

As explained in section 2.3, the property of classifier chain as a method of problem transformation is to strongly take into account label dependencies (Zhang & Zhou, 2014). The high levels of performance of this method in recommending ranking methods suggests that the suitability of methods are not completely independent of each other. With a multi-label classifier achieving accuracy of 95%, Hamming Loss as low as 0.0110 and F1 score of 0.9942, this method recommender can be used to find suitable ranking methods for real world applications without having to implement them individually. It can be used in the following steps: (1) Record the features of the score matrix; (2) Feed these features into the preloaded method recommender (3) Interpret the results to implement the recommended ranking method.
The constructed framework has been implemented on a real world judging system to show its capabilities. The College of Engineering, University of Illinois at Chicago organizes an annual competition for engineering students in their final year (“EXPO: Designing Our World”, n.d). The students form teams with their peers and present their senior design projects to judges. Candidates invited as judges came from both, academic and industrial background comprising of tenured professors, lecturers, researchers and executives from all fields of engineering. The teams were evaluated by judges who had the same educational background as the topic of the project. For example, a judge who holds the position of a mechanical engineering at a private firm was only assigned teams that were presenting projects on mechanical engineering concepts. This ensured that all teams received a fair chance to be evaluated by judges who had a basic knowledge of the topic of discussion.

Each judge was given a formulation of the parameters that the teams are to be judged on. These parameters were divided into two main categories: Technical Quality and Presentation Quality. These main categories have sub-categories like project innovation, fulfillment of project objectives, knowledge of background material, group dynamics etc. Evaluations were collected in the form of numerical ratings. Judges were instructed to score each team on a scale of 1 to 5 for each category. Due to this categorization, each team could have a score ranging from a minimum of 12 to a maximum of 60. These scores were then combined by average to rank all teams and finally award the top 48.

The year of 2017 saw the highest participation since the start of the EXPO with 130 participating teams. A total of 107 judges took part in the evaluation process. Table 3 shows all the recorded features of the score matrix collected after collecting all responses.
Table 3: Features of the score matrix at EXPO 2017

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Judges</td>
<td>130</td>
</tr>
<tr>
<td>Number of Teams</td>
<td>107</td>
</tr>
<tr>
<td>Percentage of scores</td>
<td>0.0316</td>
</tr>
<tr>
<td>Range of scores</td>
<td>41</td>
</tr>
<tr>
<td>Average number of teams evaluated by a judge</td>
<td>3.67</td>
</tr>
<tr>
<td>Average number of judges assigned to a team</td>
<td>4.46</td>
</tr>
<tr>
<td>Total Size</td>
<td>15080</td>
</tr>
</tbody>
</table>

Based on the features that were input into the trained model, the following output obtained is shown in Table 4. As shown in the result, due to low sparsity (lower percentage of scores in the matrix), the method recommender suggests against using z-score transformation. This is due to the unavailability of scores capture the mean and standard deviations for each judge. The method recommender model suggests the of average, median, social judgement scheme and matrix factorization methods. These methods can be systematically implemented based on the computation time and hardware available to the organizer(s).

Table 4: Recommended Ranking Methods for EXPO 2017

<table>
<thead>
<tr>
<th>Average</th>
<th>Median</th>
<th>SJS</th>
<th>Z-Score</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
5. CONCLUSION AND FUTURE WORK

Judging systems bring in immense complexities due to evaluations provided by humans. Applications similar to the one discussed in this work pose a challenge of incomplete information. Suitability of a ranking method for a given score matrix could be in terms of computational speed and simplicity, time taken to run the ranking method or even the availability of computing hardware. The nature of a judging system has a direct impact on the features of the score matrix produced. By only using the available score matrix, there is no easy procedure to compare and select the most suitable ranking method. This work builds a framework to attend to this shortcoming. Using the Lens Model framework, full score matrices are generated to mimic the behavior of judges in the decision making environment. With this complete information, the concept of true rank opens doors to the discussion of benchmarking mechanisms for ranking methods. A glimpse of such a comparison can be seen in this work. The use of machine learning strengthens this discussion by the adding the ability of learning from complex real world score matrices to accurately recommend suitable ranking methods.

While this work does not attend to the organizational factors of the judging system, there is immense scope in deducing measures that ensure a consistent ranking of participants by taking into account all variabilities caused due to human heterogeneity. The constructed repository can be used to devise mechanisms that can ensure that all the participants are subjected to the behavior of all judges in real-time. It can also be used to study the threshold features of the score matrix where certain methods are most suitable. The Lens Model framework can be used to build matrices of different size, score ranges and many more variations to study and compare other decision making task environments.
CITED LITERATURE


Hubli, Niharika (2018), “Judge - Participant Benchmark Repository” [Dataset], Mendeley Data, v1 [http://dx.doi.org/10.17632/w7yg54xhjj.1


VITA

NAME: Niharika Rajendra Hubli

EDUCATION: B.E., Industrial and Production Engineering, Visvesvaraya Technological University, Belgaum, 2015

RECENT WORK EXPERIENCE: Graduate Research Assistant, Prominent Lab, Chicago (2016 – 2017)
Business Analyst Intern, LIFT, Huntington Beach (2017)
Design Engineer, General Motors, India (2015 – 2016)

PUBLICATIONS: Niharika Rajendra Hubli, Houshang Darabi, “A Lens Model Based Judge-Participant Benchmark Repository and Its Application to Select Ranking Methods”, to be submitted to the Journal of Decision Support Systems, 2018

PROJECTS UNDERTAKEN: Construction of an automated course scheduler using linear programming on GAMS (CPLEX)
Analyzed historical data for student enrollment at UIC to create an interactive dashboard for reports
Formulation of aircraft dynamic test results into interactive graphs and dashboards
Cost analysis of aircraft interior manufacturing components in the production phase
Design and implementation of customer surveys to analyze aircraft seating comfort followed by analysis
Design and implementation of factorial design to test seat structure in abrasive conditions
Construction of a classification model to label future crimes in Chicago as arrests or no arrest of criminal
Analysis of the effect of climate on ice-cream eating habits using regression and ANOVA