Judgment Analysis on Streaming Data

Dissertation submitted in partial fulfilment for the award of the degree

Master of Technology in Computer Science

by

DEBPRASAD KUNDU

Roll No.: CS2102
M.Tech, 2nd year

Under the supervision of
Dr. Malay Bhattacharyya

Computer and Communication Sciences Division
INDIAN STATISTICAL INSTITUTE

June, 2023

CERTIFICATE

This is to certify that the work presented in this dissertation titled "Judgment Analysis on Streaming Data", submitted by Debprasad Kundu, having the roll number CS2102, has been carried out under my supervision in partial fulfilment for the award of the degree of Master of Technology in Computer Science during the session 2021-22 in the Computer and Communication Sciences Division, Indian Statistical Institute. The contents of this dissertation, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

_____

Dr. Malay Bhattacharyya
Associate Professor, Machine Intelligence Unit
Associate Member, Centre for Artificial Intelligence and Machine Learning
Associate Member, Technology Innovation Hub on Data Science, Big Data Analytics, and Data Curation
Indian Statistical Institute, Kolkata

Acknowledgements

First and foremost, I take this opportunity to express my sincere thankfulness and deep regard to *Dr. Malay Bhattacharyya,* for the impeccable guidance, nurturing and constant encouragement that he had provided me during my post-graduate studies. Words seem insufficient to utter my gratitude to him for his supervision in my dissertation work. Working under him was an extremely knowledgeable experience for a young researcher like me.

I want to personally thank *Dr. Sujoy Chatterjee, Assistant Professor, Amity University, Kolkata, India* for providing me the source of datasets which are employed in Chapter 1 of this dissertation. I also wants to thank all the researchers and faculties of ISI, Kolkata, and others who helped me to accumulate the data used in Chapter 2 of this dissertation through a survey.

I also thank the CSSC and ISI Library for extending their supports in different ways in my urgent need.

I shall forever remain indebted to my parents, teachers and friends for supporting me at every stage of my life. It is their constant encouragement and support that has helped me throughout my academic career and especially during the research work carried out in the last one year.

Date: 10-06-2023

———————————————

Debprasad Kundu
Roll No.: CS2102
M.Tech, 2nd year
Indian Statistical Institute

Abstract

Crowd-powered systems have drawn attention in a variety of applications, primarily because of their distributed but unified problem-solving power. Recent research has shown that crowd-based models can be used to make opinion-based judgments better. For the tasks involving image processing, natural language processing (NLP), and other areas, aggregating multiple opinions can predict the correct judgment. Unfortunately, these models do not demonstrate acceptable performance in reaching a suitable consensus judgment due to the participation of non-experts as crowd workers. For problems such as these, it is not a viable option to choose majority voting due to limitations in the volume of available annotators. Unfortunately, all the existing algorithms are based on a setting where the number of annotators and their opinions are previously provided. In this dissertation, we provide the very first algorithm that can perform judgment analysis on crowdsourced opinions available in streams. As spammers may present as annotators and consistent annotators or annotators with more confidence levels will also be present, in the proposed algorithm, we take care of both of these. As a result, the proposed algorithm is influenced by consistent annotators and less influenced by spammers. In the proposed algorithm, first, we assume that the number of options for each question is two and implement the proposed algorithm on 3 datasets: one small and two large. In the WVSCM dataset (small dataset) we got 73.1250% accuracy whereas the Majority Voting algorithm provides 75.00% accuracy and the Weighted Majority Voting algorithm provides 73.1250% accuracy. But in the Fact Evaluation dataset (large dataset) we get 94.5455% accuracy and in the Sentiment Analysis dataset (large dataset), we obtain 93.0023% accuracy, which is higher than the results from Majority Voting or Weighted Majority Voting. We have also proposed a direction to derive the judgment where the number of options for a question can go beyond two.

Contents

List of Figures

List of Tables

CHAPTER 1

Judgment Analysis on Single Dimensional Streaming Opinions

1 Judgment Analysis on Single Dimensional Streaming Opinions

In this chapter, we will discuss Judgment Analysis on Single Dimensional Streaming Opinions. Here we assume that annotators can provide only single-dimensional opinions i.e., yes, no, skip, etc as available for that question.

1.1 Introduction

In today's interconnected world, the power of crowd has emerged as a transformative force driving collaboration, innovation, and problem-solving. Crowdsourcing, a concept that harnesses the collective intelligence and diverse skills of a large group of individuals, has revolutionized the way we approach tasks, projects, and challenges [7]. By leveraging the collective wisdom and expertise of a crowd, organizations and individuals can tap into a vast pool of resources, ideas, and perspectives that were previously inaccessible [48]. Crowdsourcing goes beyond traditional hierarchical structures and embraces the democratization of knowledge and participation. It empowers people from different backgrounds, regions, and skill sets to come together, collaborate, and contribute their unique insights to a common goal. The rise of digital platforms and online communities has further facilitated the process, making it easier than ever to connect and engage with diverse individuals across the globe.

One of the key benefits of crowdsourcing is its ability to generate innovative and creative solutions. By bringing together a multitude of perspectives, individuals can build upon each other's ideas, challenge existing assumptions, and uncover groundbreaking approaches. This collective intelligence often leads to out-of-the-box thinking and breakthrough innovations that may not have been possible through traditional means. Additionally, crowdsourcing allows for the efficient allocation of resources. Instead of relying solely on a limited set of experts, organizations can tap into a much larger talent pool and distribute tasks to those best suited for them. This not only accelerates the pace of work but also ensures that projects benefit from a wide range of expertise and experiences.

Even though crowd-powered systems have only been in the spotlight for a decade, they have roots that go back several centuries [6]. These models are distributed yet the unified ability to address a range of issues makes them intriguing [23]. In 2005, Amazon Mechanical Turk (MTurk) [30] debuted as an official crowd-powered system (Wikipedia). This offers a fresh opportunity to gather relevant judgments for building Information Retrieval (IR) test collections in a scalable and cost-effective man-

ner [1]. As a result, numerous crowdsourcing platforms have appeared in the market, including the biotechnology firm 23andMe, WikiProjects that enable crowd-powered learning, the crowdfunding platform Kickstarter, Crowdfynd, etc. From a larger standpoint, based on the skills of the crowd workers they use, these crowdsourced platforms can be divided into two categories –collaborative and competitive [42]. Competitive crowdsourcing refers to the scenario where a significant number of individuals tackle a particular problem independently while having conflicting interests [8]. Collaborative crowdsourcing refers to a situation where a group of individuals works together to solve the same problem [8]. Ipeirotis in 2010 [21] studied the behavior of collaborative crowdsourcing for the first time and Boudreau et al. [4] studied Competitive crowdsourcing for the first time in 2011.

There are diverse crowdsourcing structures [5, 28] that necessitate different strategies for matching work to agents [15]. Contest-based platforms, for instance, TopCoder and InnoCentive, issue open calls for participation, and the most exceptional submissions are awarded prizes [14]. On the other hand, microtask platforms like Amazon Mechanical Turk assign simple tasks to available crowd agents on a first-come-first-served basis. However, when dealing with platforms that involve skilled crowds and specialized work, such as oDesk (now Upwork) [5], IBM's Application Assembly Optimization platform [43], and to some extent Samasource's SamaHub platform [16], it becomes imperative to employ efficient allocation algorithms. Crowdsourcing platforms can be classified into two distinct models based on the delivery method they employ: a distributed micro-task model and a contest-based model[Source]. For collaborative tasks, they either use experts or non-experts. Past experiences showed the involvement of expert crowds mainly for specialized tasks like DARPA's Red Balloon contest [42], Netflix Prize, etc. Researchers have begun to gain insights into the real-world behaviors observed on collaborative crowd-powered platforms [22, 32]. These attempts highlight several limitations and invoke the necessity for remodeling many such systems powered by the crowd.

Opinion-based judgments have achieved success with crowd-based systems [38, 39]. Multiple crowdsourced opinions may be aggregated in predicting the "gold" judgment for a particular question (after being combined). Unfortunately, these collaborative models do not demonstrate satisfactory performance in getting the appropriate judgment due to the participation of non-experts as crowd workers. According to Sorokin and Forsyth's research, inaccurate annotation contributes to a portion of the errors observed [40]. When there are limited opinions available but multiple possible outcomes for a judgment, the situation becomes worse. It is crucial to create algorithms that can automatically identify the truths from potentially conflicting and

noisy claims made by various information sources in order to enable reliable crowd-sourcing applications. Desirable truth-finding algorithms must be efficient in order to handle crowdsourcing applications involving streaming data.

In the current interconnected digital world, an immense volume of data is being generated in the form of data streams from various sources, including social networks, online shops, military surveillance, and sensors, to name a few. Having a reliable and efficient (both in terms of time and space) method for classifying data streams is therefore becoming a challenging task. Unlike conventional models, big data models operate under the assumption that the complete dataset cannot be stored in its entirety and should be analyzed in real-time or on-the-fly [29]. Due to the limited memory, we must therefore analyze the data in a few passes (or even just one). Again, as a basic requirement, It is anticipated that the storage space required for data streams and the processing time per item will exhibit sublinear growth [34]. The emergence of this challenge in computing, which goes beyond approximation and randomization, necessitates the development of novel kind of algorithms for many existing problems encountered in a streaming setting. Judgment analysis has never been done in a streaming scenario. In this dissertation, we propose the first-of-its-kind approach for performing judgment analysis on the data received in streams.

The rest of this dissertation is organized as follows. The motivation is described in section 1.2. Section 1.3 describes the problem taken for this dissertation. Basic terminologies are defined in section 1.4. Section 1.5 discusses our proposed method, its time complexity, and space complexity In section 1.6 we show some experiments on three datasets by our proposed approach. Section 1.8 shows some extension of our work. Finally, section 1.9 concludes this chapter of this dissertation.

1.2 Motivation

Streaming data analysis is commonly used to handle the influx of big data, which is often in the form of continuous data streams. Many real-life situations require the analysis of such streaming data to gain valuable insights and make timely decisions. Judgment analysis on streaming data is an emerging problem. At any time instant, billions of online users get engaged in different activities. So, it is not possible to get the opinions of different uses at a time. Most of the time the crowd-workers provide their opinions at different times. Moreover, it is not possible to store all the responses of the crowd-workers to make judgments. Keeping all of these in mind, we propose the first-ever algorithm to make a judgment on streaming data. The main model pursuing which we develop the judgment analysis approach is portrayed in Fig. 1.1.

Fixed set of questions
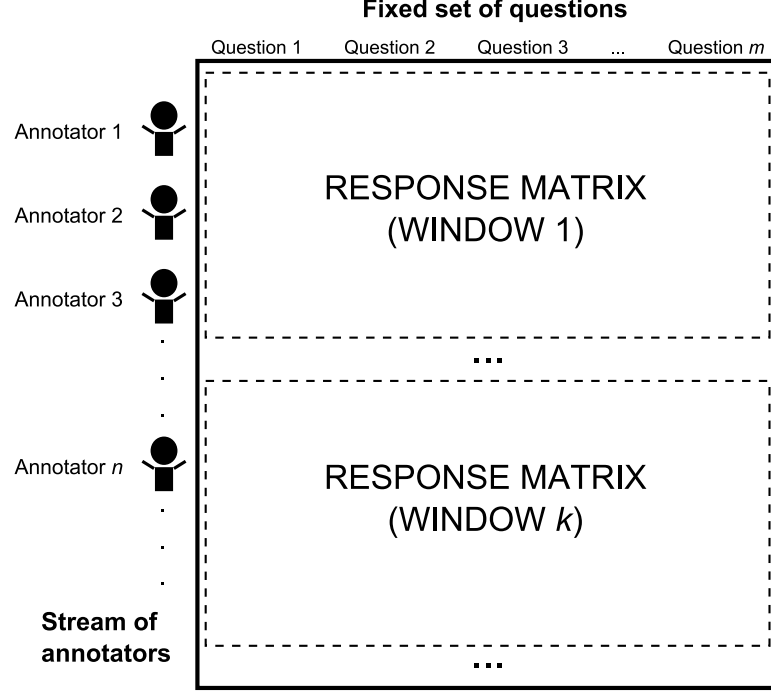
Figure 1.1: The basic model of judgment analysis to be performed on the streaming opinions. The response matrix has fixed number of columns (corresponding to fixed set of questions) but it keeps growing alongside the rows (due to streaming annotators).

1.3 Problem Formulation

Now we formalize the problem of judgment analysis for the streaming data [29]. We assume that the data item $s_l \in [m] \times [n]$ arrives as an input stream $T = <T_1, T_2, \cdots, T_k>$. It is not possible to get all the elements of the input stream at a time because the input data stream arrives sequentially in a streaming setting. So, we process them by pursuing a window-based model.

Definition 1.1 (**Data stream model** [29]). *A data stream model defines an input stream $\mathscr{S} = <s_1, s_2, \cdots>$ arriving sequentially, item by item, and describes an underlying signal S, where $S : [1 \ldots N] \rightarrow \mathbb{R}$ is a one-dimensional function.*

In streaming algorithms, the data is available as a stream and we like:

- the per-item processing time
- storage and
- overall computing time

to be simultaneously $O(N, t)$, preferably $O(polylog(N, t))$, at any time instant t in the data stream.

7

We consider a set of questions $Q = \{q_1, q_2, \ldots, q_m\}$ that are the labeling tasks. The set of annotators be $A_l^i = \{A_{1n}^i, A_{2n}^i, \ldots, A_{ln}^i\}$ whose opinions came at timestamp T_i. The set of opinion values $O = \{o_1, o_2, \ldots, o_k\}$ can take either some integer values (e.g., $0, 1, 2, etc.$) or some categorical values (e.g., Yes, No, Skip, etc.) for any particular question.

An annotation process in a streaming setting consists of the following:

1. Set of questions $Q = \{q_1, q_2, \ldots, q_m\}$

2. Set of options $O = \{o_1, o_2, \ldots, o_k\}$

3. An annotator A_{pl}^i or a set of annotators A_l^i arriving at timestamp T_i

4. $\tau : (Q \times A_{pl}^i) \rightarrow O$ is a mapping function over the time points $T = \{T_1, T_2, \cdots\}$

After any timestamp, we have to obtain the final judgment for all the questions in Q with the constraint mentioned above. It is to be noted that the cardinalities of the sets Q, A_n^i and O (m, n and k, respectively) are not necessarily the same at any timestamp. In fact, the sizes of A_l^i may also be different for different i. For each timestamp, we define a response matrix R as a matrix of dimension $n \times m$ whose elements R_{ij} denote the opinion provided by the ith annotator of that timestamp for the jth question such that $R_{ij} \in O \; \forall i, j$. We further denote a judgment matrix J of dimension $m \times k$ whose elements J_{ij} denote the weight of the jth opinion for the ith question. Note that, there can be multiple entries in a row of the judgment matrix, however, the maximum value denotes the most appropriate opinion for a particular question. A demo of the response matrix for some timestamps is shown in Fig 1.2.

It is considered that the annotator chooses only one option for a particular question from that set of multiple opinions. Also, all the annotators do not provide their opinion at the same time. So, the responses of the annotators come in a streaming way. Depending on the mass opinions in that context we have to find the best option for each question. In reality, most of the annotators does not provide the maximum number of question, and hence R is a sparse matrix.

1.4 Preliminary Details

1.4.1 Basic Terminologies

Throughout this dissertation, we use the term response vector to symbolize the responses of a single annotator or more specifically one row of the response matrix. We consider the responses of each annotator streaming in as a vector. The other notations

$$
T_1 :: \begin{pmatrix}
2 & 0 & 1 & 0 & -1 & -1 & \cdots & 0 \\
-1 & 0 & -1 & 1 & 1 & 1 & \cdots & -1 \\
1 & 1 & -1 & 0 & 0 & -1 & \cdots & 1 \\
-1 & 0 & 2 & -1 & -1 & 1 & \cdots & 1 \\
1 & -1 & 1 & 0 & -1 & 0 & \cdots & 0
\end{pmatrix}
$$

$$
T_2 :: \begin{pmatrix}
-1 & 0 & 1 & 0 & -1 & -1 & \cdots & 0 \\
1 & -1 & -1 & 1 & 1 & 1 & \cdots & 1 \\
1 & 2 & 0 & 1 & 0 & -1 & \cdots & -1 \\
-1 & 0 & 1 & 1 & -1 & 1 & \cdots & 1 \\
1 & -1 & 1 & 0 & -1 & 0 & \cdots & 1
\end{pmatrix}
$$

$$
\vdots
$$

$$
T_n :: \begin{pmatrix}
1 & -1 & 1 & 0 & 1 & -1 & \cdots & 1 \\
0 & 0 & -1 & 0 & 2 & 0 & \cdots & 1 \\
1 & 1 & 0 & 0 & 0 & -1 & \cdots & 1 \\
-1 & 0 & 1 & 1 & -1 & 1 & \cdots & 1 \\
1 & 1 & 2 & -1 & -1 & 1 & \cdots & 1 \\
1 & -1 & 1 & 0 & -1 & 0 & \cdots & 1
\end{pmatrix}
$$

Figure 1.2: A snapshot of the response matrices at different timestamps has been shown. Each row represents the opinions provided by a particular annotator and each column represents the opinions for a particular question given by different annotators at a particular timestamp. Here, the number of options is three, namely, 1(Yes), 0(No), and 2(Skip). Here, $R_{ij} = -1$ means that the ith annotator does not provide his opinion for the jth question. Also, we see that for T_1, T_2 timestamp, the number of annotators is 5 whereas, for T_n timestamp, the number of annotators is 6 i.e, the number of annotators is not the same for each timestamp

have their usual meaning unless specified otherwise. There are some basic terminologies that are fundamental to judgment analysis. We introduce these terminologies and a few related notions that will be used throughout the dissertation.

- **Annotator:** An annotator is a crowd worker from the online community who provides their judgment over a specific question.

  There is very less chance to guess a label at random by a good annotator. An annotator may be good at some aspects of a task but may be worse at others.

- **Spammers:** The annotators who provide their opinions without taking into account their knowledge of the available options.

- **Opinion:** Opinion is an annotation that is marked by an annotator. There is a finite set of annotations for a given problem.

  Annotation is a function that returns one of the available opinions according to some procedure.

- **Domain of Opinions:** The possible set of opinion.

  For example, let there be four types of opinions for a set of questions. These opinions are 'Yes', 'No', 'Neutral' and 'Skip'. So here the annotator should choose any one option from the opinion set. So the domain of the opinion is 'Yes', 'No', 'Neutral', and 'Skip' and here the cardinality of the opinion is four.

  When questions are distributed among a crowd to solicit answers, the aggregation process takes place after collecting the answers from the crowd. By aggregating these individual answers, a prediction or label for each question can be determined.

- **Gold judgment:** The true labeling for each question.

- **Question difficulty:** The question difficulty is the level of hardness of a question.

- **Annotator accuracy:** The accuracy of an annotator means how reliable the annotator is.

  A competent annotator is capable of providing accurate judgments, while an unskilled worker may offer inconsistent judgments. Various models employ different approaches to calculate worker accuracy. Some algorithms assess worker accuracy using metrics such as the confusion matrix, F1 score, likelihood or sensitivity measures, or other suitable methods. These metrics help evaluate and quantify the reliability of annotators' judgments.

1.4.2 Related Work

Majority Voting:

The majority voting algorithm is a simple yet effective approach used in judgment analysis to aggregate the opinions or judgments of multiple individuals or classifiers. It is commonly employed in various fields such as data science, machine learning, and decision-making processes. The algorithm works by combining the individual judgments or predictions of multiple sources and determining the final decision based on the majority opinion.

- **Advantages:**

  - *Simplicity*: This algorithm is easy to understand and implement.
  - *Robustness*: This can handle a wide range of judgment types, including binary decisions, categorical predictions, and numerical estimates.
  - *Low Computational Cost*: This is computationally efficient since it does not involve complex calculations or optimization procedures.

- **Disdvantages:**

  - *Equal Weighting*: algorithm assumes that each source or classifier's judgment is equally reliable. However, in reality, different sources may have varying levels of expertise or accuracy. Treating all sources equally can lead to suboptimal results when some sources have a higher quality or are more reliable than others.
  - *Correlated Judgments*: If the judgments provided by different sources are highly correlated, the majority voting algorithm may not effectively capture the full range of perspectives.
  - *Lack of Consensus Resolution*: In cases where there is no clear majority among the judgments, the majority voting algorithm does not provide a mechanism for resolving conflicts or reaching a consensus. This can result in an ambiguous or inconclusive final decision.

Weighted Majority Voting:

Weighted Majority Voting (WMV) is an algorithm used in judgment analysis to aggregate the opinions or judgments of multiple experts or classifiers. It aims to combine individual judgments and produce a collective decision that represents the consensus or majority viewpoint. In the context of judgment analysis, the WMV algorithm assigns weights to each expert or classifier based on their perceived reliability or competence. Experts who have demonstrated higher accuracy or expertise in the past are

assigned higher weights, indicating that their judgments carry more influence in the final decision.

- **Advantages:**

  - *Aggregates Diverse Perspectives:* WMV allows for the integration of judgments from multiple experts or classifiers, each with their own unique perspectives and approaches.

  - *Handles Expert Heterogeneity:* In many cases, not all experts have the same level of expertise or accuracy. WMV addresses this by assigning different weights to individual experts based on their historical performance. This approach ensures that the more reliable and accurate experts have a greater influence on the final decision, while still considering the contributions of other experts.

  - *Improved Accuracy:* By combining the judgments of multiple experts, WMV has the potential to improve the overall accuracy of the decision-making process. When the individual experts have varying strengths and weaknesses, the aggregated decision can leverage the strengths and compensate for the weaknesses, resulting in a more reliable outcome.

- **Disdvantages:**

  - *Weight Assignment Challenges:* Assigning appropriate weights to experts can be a challenging task. Determining the true reliability or expertise of each expert is not always straightforward, and subjective biases may influence the weight assignment process. Inaccurate weight assignments can lead to biased or suboptimal aggregated decisions.

  - *Lack of Error Estimation:* WMV does not provide an explicit estimation of the uncertainty or error associated with the aggregated decision. While the algorithm can capture the diversity of opinions, it does not quantify the confidence or variability in the aggregated result.

  - *Complexity and Interpretability:* The weighted aggregation process in WMV can introduce complexity in understanding the decision-making process. Interpreting the weights assigned to experts and the influence of each judgment can be challenging.

1.5   Proposed method

In this section, we present a detailed description of the proposed method.

1.5.1 Judgment Analysis

There are lots of well-known clustering algorithms like k-mean, k-medoid, etc. In each of these cases, we need to predefine the value of k. Moreover, all the data points have to store for all these clustering algorithms which are very inefficient for streaming data. For the DBScan clustering algorithm, we need to predefine the frequency i.e, the number of data points present in a cluster, which is also very inefficient for streaming data.

As we are working with data coming in a streaming way, we cannot store all the data. Also, we can think that similar(nearer) data(response vector) can be stored in one place and dissimilar(farther) data(response vector) can be stored in a different place. Each of these places will be called a cluster. But, we try to find a way in which instead of storing all the data points we just store one data point in each cluster which we will call the *core point* or *representative* of that cluster. We keep only the number of data points that are in that cluster. We say that a data point is in a cluster if the distance between the data point and the core point of the cluster is less than a positive integer hyp i.e, for two data points v_1 and v_2, v_1 and v_2 are in the same cluster if $d(v_1, v_2) \leq hyp$ where d is the distance metric and hyp is an integer value. This hyp can be a predefined integer or it can be obtained using some standard method like the elbow method. For the application of this algorithm, we can define the distance metric in the following way:

Let $v = (v_1, v_2, \ldots, v_n)$ and $u = (u_1, u_2, \ldots, u_n)$ be two data points. Then

$$d(u, v) = \sum_{i=0}^{n} \#(u_i \neq v_i) \tag{1}$$

Here, $\#(u_i \neq v_i)$ indicates the number of i for which u_i and v_i are not same. It is easy to keep the trace of clusters instead of all the data points as we can say that the vectors lie in a cluster containing similar or nearer opinions.

Now, *these clusters are kept in a special type of data structure* which is a collection of key-value pairs where values can be of any data type more specifically, values are the non-overlapping clusters, and for each cluster, a unique key will be assigned. So, keys can not be repeated and must be immutable. We term this data structure as a *dictionary*. For our experiments, we denote the key corresponding to a cluster as the ID of that cluster. The value corresponding to each key contains a *core point*, the number of response vectors assigns to that cluster which is termed as *frequency* here. So, in short, we can say that *each cluster is a dictionary and these clusters are kept in another dictionary.*

The motivation for using the dictionary is to save the storage as well as provide a good judgment for the data coming in a streaming way. Whenever a response vector or

a window of response vectors comes into the model we first assign the response vector to the proper cluster. Now for each response vector of the coming window(which may contain a single vector also) and for each question, in the corresponding position of the judgment matrix, we just increase the value by the accuracy of the annotator corresponding to that vector and a portion of the ratio of the question attempted by the representative vector and the number of annotators present in the corresponding cluster i.e, if 1 is the response of a question q_i by any annotator a_j and response vector of a_j lies in cluster c_k then we will increase the value of ith row and 2nd column in the judgment matrix by the accuracy a_j and a portion of the ratio of the question attempted by the representative vector of c_k and the number of annotators present in c_k(In some approach we add another term as question difficulties which is explained later). Finally, the index of the maximum value for each question will be the final judgment of that question.

1.5.2 Proposed algorithm

The suggested approach is predicated on the notion that, unlike the majority voting algorithm, every annotator cannot be regarded as an equal expert. Also, relying on trolls could introduce noise into the verdict. Keep in mind that not every annotator offers their thoughts on every topic. The response matrix or block R_i represents the annotators' thoughts on each question at timestamp T_i. The annotators are indicated in the response matrix's rows, and the columns are marked with the associated questions. Every one of the potential choices should be replaced by a distinct integer if the answer matrix includes categorical annotation for some datasets. This grid is undoubtedly sparse.

Let us consider an easy illustration. Let's say a query is answered with the choices 'Yes', 'No', and 'Skip'. As the response is categorical, each potential opinion will be substituted by the corresponding specific integers 1, 0, and 2. If -1 appears in any cell, the corresponding annotator has not tried that specific question. Let us consider c is a constant, annotator a_j is in the cluster c_k, $|\{a_i : a_i, a_p \in c_k\}|$ and $|Q_k|$ are the number of annotators in c_k and the number of responses of the representative of c_k respectively. Also, J is the judgement matrix of order (# question × # option of each question)

The complete algorithm is formally presented in the Algorithm 1. Here we use a dictionary namely *dict* which contains the clusters. The word *block* is used to represent the window of the response vectors coming at a particular timestamp. J is a matrix of order (*the number of question × the number of options*) which contains the weight of each option corresponding to each question. hyp is an integer which is explained in the previous subsection. We assume that before providing the opinions

each annotator has to provide the opinion for some gold result by which we can find the accuracy of each annotator.

Algorithm 1 The approach of Judgment Analysis

Input: The judgment matrix J, the clusters in a dictionary form $dict$ in time T_i, response matrix $block$ coming at the timestamp T_i, the accuracy of all the annotators present in the current window, hyp, and the control parameter c.
Output: The final judgment (derived opinions) $\gamma(q_i)$ for each question q_i.
Algorithmic Steps:

1: $dict \leftarrow \text{CLUSTER}(dict, block, hyp)$
2: weight $\leftarrow$ []
3: **for** $a_p \in block$ **do**
4:      $Acc(a_p) \leftarrow$ Accuracy of a_p
5:      $Q_k \leftarrow \{q_k : a_p \in c_k, q_k \in c_k \ with \ q_k \neq -1\} \setminus\setminus Q_k$ is the set of questions answered by the representative of the cluster c_k
6:      $A_k \leftarrow \{a_i : a_i, a_p \in c_k\} \setminus\setminus A_k$ is the set of annotators present in c_k
7:      $w \leftarrow Acc(a_p) + \text{c} * \frac{|Q_k|}{|A_k|}$
8:      weight.append(w)
9: **end for**
10: **for** $q_i \in block$ **do**
11:      **for** $a_j \in block$ **do**
12:          J[q_i][response of a_j] $\leftarrow$ J[q_i][response of a_j] + weight of a_j
13:      **end for**
14: **end for**
15: **for** 0 to J.shape[0] **do**
16:      $\gamma(q_i) \leftarrow \arg max_k J_i$
17: **end for**

Here we use a subroutine *CLUSTER* in the first line of the algorithm 1 which helps us to build the dictionary containing the clusters, in short, it helps to find the clusters containing the core points and the number of annotators who provide their opinion in a similar manner. Here, by using *dict[i]* denotes the value of the key i in the dictionary *dict* i.e., the cluster with ID i and by *dict[i] = DIRECTORY(block[j])*, we assign the value of the key i as *DIRECTORY(block[j])* in the dictionary *dict*. *DIRECTORY(v)* is a data structure of a cluster whose representative is v.

Let us discuss some of the justifications for the algorithm's stages in this context. We have taken into account the clusters' ability to combine the effects of various groups of questions answered by various but related sets of annotators. In order to give greater priority to the annotators who have provided opinions for a larger set of questions, we consider the value of both the number of questions and the annotators when calculating the weight of the opinions. Because spammers may attempt a large number of queries, accuracy is also regarded as a crucial factor. Therefore, we must eliminate

The CLUSTER() subroutine

CLUSTER($dict, block, hyp$){

 1: $i \leftarrow$ number of keys in dict

 2: **for** $j \in (0, number\ of\ annotators\ in\ block)$ **do**

 3:     **if** number of key in $dict = 0$ **then**

 4:         $i \leftarrow i + 1$

 5:         $dict[i] \leftarrow$ DIRECTORY($block[j]$) *directory()* is a function to make a cluster

 6:     **else**

 7:         $l \leftarrow 1$

 8:         $k \leftarrow i$

 9:     **while** $l \leq k$ **do**

10:         $distance \leftarrow d(block[j], dict[l].representative)$

11:         **if** $distance \leq hyp$ **then**

12:             frequency of $dict[l] \leftarrow$ frequency of $dict[l] + 1$

13:         **else**

14:             $l \leftarrow l + 1$

15:         **end if**

16:         **if** $l = k + 1$ **then**

17:             $i \leftarrow l$

18:             $dict[i] \leftarrow DIRECTORY(block[j])$

19:         **end if**

20:     **end while**

21:     **end if**

22: **end for**

23: return $dict$

}

The DIRECTORY() data structure

DIRECTORY($block[j]$){

 1: $CLUST \leftarrow$ {} *CLUST* is an empty dictionary

 2: $CLUST.representative \leftarrow block[j]$

 3: $CLUST.frequency \leftarrow 1$

 4: return $CLUST$

}

those instances. For the purposes of this study, the control parameter c is a fractional value that aids in determining the order of importance within a bounded value.

In essence, Step 7 of the Judgment algorithm(1) says that an annotator who is reliable and has a high accuracy number should be given more weight. We refer to the annotators who have tried a lot of questions as consistent annotators. In general, there are lots of annotators who attempt very few numbers of questions. We want to decrease their significance. Mainly, this step contains two controlling factors. The first factor denotes the priority value assigned to the annotator with respect to the other annotators in the corresponding cluster c_i. The second factor is the ratio between the dimension of the response question size by the representative and the annotator size of the cluster. A question size is higher means that the annotator is part of a cluster that might be consistent over many questions. Therefore, when combining the effects of numerous clusters, it is important to give the relevant opinion high priority and weight. Another crucial point is that if a question is answered by a small number of annotators, either the question is difficult or the annotator's level of trust is excessive. As a result, their opinions ought to be granted more weight. This is the reason for keeping the annotator size of the cluster in the denominator part of the aforementioned equation. So by balancing these two factors, the weighted judgment of a specific option for a question is ultimately determined.

1.5.3 Time Complexity Analysis

First, we see the time complexity for the *CLUSTER* subroutine which is the time complexity for step 1 in the algorithm 1. The time complexity for the while loop at step 9 of the subroutine will take $O(k)$ time where k is the number of clusters present in the dictionary $dict$ at that time. The *for* loop at step 2 of the subroutine will run for $O(n)$ times where n is the number of annotators in the window coming at the timestamp T_i. So as a whole, the time complexity for the subroutine is $O(nk)$ where n is the number of annotators in the window coming at the timestamp T_i and k is the number of clusters.

So, the time complexity of step 1 in algorithm 1 is $O(nk)$ where n is the number of annotators coming at a time and k is the number of clusters. The time complexity of steps 3-9 is $O(m)$ where m is the number of questions. The time complexity of steps 10-14 is $O(mn)$. The time complexity of steps 15-17 is $O(m)$.

So as a whole, the time complexity to make the judgment is $O(nk) + O(m) + O(mn)$, i.e., $O(mn)$ as $k \ll m$.

1.5.4 Space Complexity Analysis

Every cluster contains the core point or representative and the number of annotators present in that cluster. So, each cluster needs $O(m)$ space where m is the number of questions. Hence, for the *CLUSTER subroutine*, the only required space is $O(mk)$ for storing the dictionary $dict$ where k is the number of clusters in $dict$.

So, the space complexity for Step 1 is $O(mk)$ where m is the number of questions and k is the number of clusters in $dict$. The space complexity for Steps 2-17 is $O(m)$ where m is the number of questions as we need to store the judgment matrix J.

As a whole for the judgment for streaming data, we need $O(mk) + O(m)$ i.e., $O(mk)$ storage for our proposed algorithm where k is the number of clusters in $dict$ and m is the number of questions.

1.6   Empirical Analysis

We demonstrate the performance of the proposed algorithm on the following datasets.

1.6.1   Dataset I – WVSCM

Duchenne experiment data accompanying the paper by Whitehill, et al. [46].
   Dataset Details:
   The task of the Mechanical Turk involves assigning labels to facial images that depict smiles, categorizing them as either *Duchenne* or *Non-Duchenne*. A Duchenne smile, also known as an "enjoyment" smile, can be differentiated from a Non-Duchenne smile, also called a "social" smile, by the presence of activation in the *Orbicularis Oculi* muscle surrounding the eyes. The differentiation between Duchenne and Non-Duchenne smiles holds significance in multiple fields such as psychology experiments, human-computer interaction, and marketing research. Even certified experts in the Facial Action Coding System find it challenging to accurately identify and code Duchenne smiles.

   In the file containing ground truth, each row contains two columns. The first column contains the image ID; the second column contains the ground truth Duchenne label (0 or 1). Here, out of the total 160 images, 58 of them displayed Duchenne smiles. There are 64 annotators who provide their opinion. Out of these 64 annotators, 14 annotators have accuracy more than or equal to 50%, 13 annotators have accuracy more than or equal to 60%, 7 annotators have accuracy more than or equal to 70% and only 1 annotator has an accuracy more than or equal to 80%. In the response file, each row contains three columns. The first column contains the image ID; the second col-

umn contains the labeler ID; the third column contains the label (0: non-Duchenne, 1: Duchenne).

Design of Experiment:

Any annotator can provide only one opinion between 0 and 1. Also, we assume that each annotator has to provide their opinion on some of the images whose ground truth values are given. Also, we use the elbow method to find the threshold hyp.

Results:

We consider the window size as 16 for each timestamp. We use distance metric d defined in the equation 1.

Firstly, we take the control parameter c as 0.001. we got 73.1250 % accuracy after the judgment of 64 annotators in a streaming way. The accuracy after various windows is shown in Table 1.

Table 1: The performance accuracy obtained for the WVSCM in a streaming setting with window size 19 with $c = 0.001$.

Window	Number of raters	Accuracy
1	16	70.0000 %
2	32	74.3750 %
3	48	70.6250 %
4	64	73.1250 %

Now, we take the control parameter c as 0.001. we got 73.1250 % accuracy after the judgment of 64 annotators in a streaming way. The accuracy after various windows is shown in the table 2

Table 2: The performance accuracy obtained for the WVSCM in a streaming setting with window size 19 with $c = \frac{1}{|Q|}$, where Q denotes the set of questions.

Window	Number of raters	Accuracy
1	16	70.0000 %
2	32	74.3750 %
3	48	70.6250 %
4	64	73.1250 %

Now, we compare our final accuracy with various traditional methods.

The accuracy on Majority Voting is 75.00%

The accuracy of our proposed algorithm is 73.1250%
So, less accuracy using the proposed algorithm is

$$(75.00 - 73.1250)/75.00 * 100 = 1.8750/75.00 * 100 = 2.5\%$$

Since our method is based on a streaming setting, it is acceptable to obtain a slightly lesser (not statistically significant) accuracy value.

Table 3: The performance accuracy obtained for the WVSCM. The best accuracy over the column is shown in bold.

Algorithm	Accuracy(%)	Data Setting
Approach 3 (Proposed Best)	73.1250	Streaming
Majority Voting	**75.0000**	Batch / Streaming
Weighted Majority Voting	73.1250	Batch / Streaming

1.6.2 Dataset II – Fact Evaluation

The dataset contains the judgments of relations from people about public figures on Wikipedia. It contains 42,623 examples of "attended or graduated from an institution" example. Each of these was judged by a minimum of 5 annotators, resulting in a total of 216,725 judgments from the annotators who were already trained for this task, and as a whole, 57 annotators provide their opinions.

Dataset Details:

The full version of the dataset contains relations in the form of a triplet: the relation in question, called a predicate; the subject of the relation; and the object of the relation. Subjects and objects are represented by their Freebase MIDs, and the relation is defined as a Freebase property. The evidence for the relations is also included in the form of a URL and an excerpt from the web page that our raters judged. The format of the files is JSON. Each line consists of the following fields: predicate, subject, object, an array of evidence, the web page from which this evidence was obtained, a short piece of text supporting the triple, an array of judgments from human annotators, hash code of the identity of the annotator, and judgment of the annotator (0/1/2 denoting no/yes/skip respectively). Answers to 576 facts are available as the gold data.

The basic version of the dataset contains two columns: one is question id and the other is metadata. This metadata contains a JSON-encoded dictionary containing the judgments of all the raters' for this question and other pieces of relevant data described below in the description section. The dataset is provided under the Creative

Commons Attribution-ShareAlike 3.0 license by Google Inc. We work on this dataset.

Design of Experiment:

The labels of the items are largely unbalanced: 92.19% of the gold items have the true label of 'yes', while only 3.30% have the label 'no" and 4.51% have the label 'skip. Here the option 'skip' may have different significance. If we find that more annotators choose 'skip' for a particular question then that question can be considered as difficult. We assume that each annotator has to provide their opinion on some of the questions whose ground truth values are given. We use the elbow method to find the threshold hyp. Keeping this in mind we proceed with the following approaches.

- **Approach 1:** We first consider the 'skip' option as an independent option as well as to find the question difficulties.

- **Approach 2:** We exclude the 'skip' option from an independent option but consider the 'skip' option in finding the question difficulties.

- **Approach 3:** We fully discard the 'skip' option from the data and then use our algorithm for judgment.

Results:

We consider the window size as 19 for each timestamp. We use distance metric d defined in the equation 1.

Firstly, we take the control parameter c as 0.001. After the judgment of 57 annotators in a streaming way, we got 89.7569 % accuracy for Approach 1, 94.0000 % for Approach 2, and 94.3636 % accuracy for Approach 3. A comparison of results in the different conditions of our proposed methods is shown in Table 4.

Table 4: The performance accuracy obtained for the fact evaluation dataset of Google in a streaming setting with window size 19 with $c = 0.001$. The best accuracy in each row is shown in bold.

Window	Number of raters	Approach 1	Approach 2	Approach 3
1	19	83.6806 %	87.6364 %	**87.8182** %
2	38	90.7986 %	95.0909 %	**95.4545** %
3	57	89.7569 %	94.0000 %	**94.3636** %

Now, we take the control parameter c as the inverse of the total number of questions available to the annotators for providing their opinion. After the judgment of 57 annotators in a streaming way, we got 89.9306 % accuracy for Approach 1, 94.3636 %

21

for Approach 2, and 94.5455 % accuracy for Approach 3. A comparison of results in the different conditions of our proposed methods is shown in Table 5.

Table 5: The performance accuracy obtained for the fact evaluation dataset of Google in a streaming setting with window size 19 with $c = \frac{1}{|Q|}$, where Q denotes the set of questions. The best accuracy in each row is shown in bold.

Window	Number of raters	Approach 1	Approach 2	Approach 3
1	19	83.6806 %	87.6364 %	**88.0000** %
2	38	90.4514 %	94.9091 %	**95.6364** %
3	57	89.9306 %	94.3636 %	**94.5455** %

From Table 5, one can see that the best accuracy is obtained when we fully discard the 'skip' option from the data (Approach 3). Hence, Approach 3 is the best form of the proposed algorithm for the fact evaluation dataset. Table 6 shows the comparison of our approach with some predefined approach for this data. For other pre-defined approaches, we fully discard the 'skip' option from the data. Here we can see that our approach gives more accuracy than other approaches.

Table 6: The performance accuracy obtained for the fact evaluation dataset of Google. The best accuracy over the column is shown in bold.

Algorithm	Accuracy (%)	Data Setting
Approach 3 (Proposed Best)	**94.5455**	Streaming
Majority Voting	94.3636	Batch / Streaming
Weighted Majority Voting	94.3636	Batch / Streaming
Algorithm proposed by Liu, Peng, Ihler [26]	94.1818	Batch

1.6.3 Dataset III – Sentiment Analysis

The sentiment analysis dataset is provided by the crowd-powered company Crowd-Flower. The CrowdFlower sentiment data inquires the annotators to judge the sentiment of a tweet that discusses the weather. The data comprises 98,979 tweets. Each tweet was evaluated by at least 5 annotators, for a total of approximately 569,375 answers.

Dataset Details:

The full version of the dataset is available at a large scale with many details. It provides information about the question (the question ID), rater (the rater ID), judgment (the answer of the annotators from the set {0, 1, 2, 3, 4}), tweet_text (the tweet content itself), country (the citizenship of the annotator), region (the region from where the annotator belong), city (the city from where the annotator belong), started_at (the time of when the annotator started working for this annotation) and created_at (the timestamp of when the annotator finished working for this annotation). The possible answers to the opinions are as follows. The '0' denotes Negative, '1' denotes Neutral/author is just sharing information, '2' denotes Positive, '3' denotes tweet not related to the weather condition, and '4' denotes I can't tell.

We use a basic version of the data for our analysis. It contains three columns providing information about the question (the question ID), the rater (the rater ID), and the judgment (the rater's answer, from 0 through 4). For our experiment, we take the opinion as 0 for 0(Negative), 1 for 2(Positive), and 2 for 4(I can't tell) and the rest opinions are neglected.

Design of Experiment:

We assume that each annotator has to provide their opinion on some of the questions whose ground truth values are given. We use the elbow method to find the threshold hyp. Here the option 'I can't tell' may have a different significance. If we find that more annotators choose 'I can't tell' for a particular question then that question can be considered as difficult. Keeping this in mind, we proceed with the following approaches.

- **Approach 1:** We first consider the 'I can't tell' option as an independent option as well as to find the question difficulties.

- **Approach 2:** We exclude the 'skip' option from an independent option but consider the 'I can't tell' option in finding the question difficulties.

- **Approach 3:** We fully discard the 'I can't tell' option from the data and then use our algorithm for judgment.

Results:

We consider a window size of 50 for each timestamp. We use distance metric d defined in the equation 1.

Firstly, we take the control parameter c as 0.001. After the judgment of 1950 annotators in a streaming way, we got 91.6479 % accuracy for Approach 1, 92.0993 % for Approach 2, and 92.0993 % accuracy for Approach 3. A comparison of results in the different conditions of our proposed methods is shown in Table 7.

Table 7: The performance accuracy obtained for the Sentiment Analysis dataset of CrowdFlower in a streaming setting with window size 50 with $c = 0.001$. The best accuracy in each row is shown in bold.

Window	Number of raters	Approach 1	Approach 2	Approach 3
1	50	71.5576 %	**71.7833** %	**71.7833** %
2	100	**80.8126** %	**80.8126** %	**80.8126** %
3	150	**84.4244** %	84.1986 %	83.9729 %
4	200	86.9074 %	**87.1332** %	**87.1332** %
5	250	**88.9391** %	**88.9391** %	**88.9391** %
6	300	**89.8420** %	**89.8420** %	**89.8420** %
7	350	**90.2935** %	**90.2935** %	**90.2935** %
8	400	**90.0677** %	**90.0677** %	**90.0677** %
9	450	90.2935 %	**90.5192** %	**90.5192** %
10	500	90.2935 %	**90.5192** %	**90.5192** %
11	550	90.5192 %	**90.7449** %	**90.7449** %
12	600	90.5192 %	**90.9707** %	**90.9707** %
13	650	90.5192 %	**90.9707** %	**90.9707** %
14	700	90.5192 %	**90.9707** %	**90.9707** %
15	750	90.5192 %	**90.9707** %	**90.9707** %
16	800	90.7449 %	**90.9707** %	**90.9707** %
17	850	90.7449 %	**90.9707** %	**90.9707** %
18	900	90.7449 %	**91.1964** %	**91.1964** %
19	950	90.7449 %	**91.1964** %	**91.1964** %
20	1000	90.9707 %	**91.4221** %	**91.4221** %
21	1050	90.9707 %	**91.4221** %	**91.4221** %
22	1100	90.9707 %	**91.4221** %	**91.4221** %
23	1150	90.7449 %	**91.4221** %	**91.4221** %
24	1200	90.9707 %	**91.6479** %	**91.6479** %
25	1250	90.9707 %	**91.6479** %	**91.6479** %
26	1300	90.9707 %	**91.6479** %	**91.6479** %
27	1350	90.9707 %	**91.6479** %	**91.6479** %
28	1400	90.9707 %	**91.6479** %	**91.6479** %
29	1450	90.9707 %	**91.6479** %	**91.6479** %
30	1500	90.9707 %	**91.6479** %	**91.6479** %
31	1550	91.6479 %	**92.0993** %	**92.0993** %
32	1600	91.6479 %	**92.0993** %	**92.0993** %
33	1650	91.6479 %	**92.0993** %	**92.0993** %
34	1700	91.6479 %	**92.0993** %	**92.0993** %
35	1750	91.6479 %	**92.0993** %	**92.0993** %
36	1800	91.6479 %	**92.0993** %	**92.0993** %
37	1850	91.6479 %	**92.0993** %	**92.0993** %
38	1900	91.6479 %	**92.0993** %	**92.0993** %
39	1950	91.6479 %	**92.0993** %	**92.0993** %

Now, we take the control parameter c as the inverse of the total number of questions available to the annotators for providing their opinion. After the judgment of 1950 annotators in a streaming way, we got 92.0993 % accuracy for Approach 1, 92.3251 % for Approach 2, and 93.0023 % accuracy for Approach 3. A comparison of results in the different conditions of our proposed methods is shown in Table 8

From Table 8 we can see that we get the best accuracy using our proposed approach when we exclude the 'I can't tell' option as an independent option but consider finding the question difficulties i.e., Approach 2 and when we fully discard the 'I can't tell' option from the data i.e, the Approach 3. Taking similarity with the Fact Evaluation dataset, we say that Approach 3 is the best approach of the proposed algorithm for the Sentiment evaluation dataset. Table 9 shows the comparison of our approach with some predefined approach for this data. For other predefined approaches, we fully discard the 'I can't tell' option from the data. Here we can see that our approach gives more accuracy than other approaches.

1.7 Necessity for being two options for each question

In the previous section, we see that we use only two options for each question. There is a reason behind this assumption. We see that the distance function takes an important role to make the cluster. As each question has only two options, the responses can be thought of as 1-dimensional responses. So, if the opinion of two annotators for the same question is different, it increases the distance by 1. On the other hand, let us consider the responses of two annotators for the same question as 0(Negative) and 3(Neutral). If 1(Positive) is the opinion of the core point then for each case the distance will be increased by 1 whereas their interpretation is different. Actually, if the number of options for a question is d, then the responses can be thought of as $(d-1)$ dimensional responses. As a result, our distance function will not work properly.

1.8 Further Experimentation on Sentiment Analysis Dataset by Changing Distance Metric

In the previous version, we take the distance metric defined by the equation 1 and we use only 2 options from 5 options for each question. Now we are trying to make a direction to work with all the options provided in the dataset. Clearly, we cannot use the distance metric defined in equation 1.

Design of Experiment:

In the given dataset, we have 5 options for each question. So, we can convert every option to a 4-dimensional vector in the following way:

Table 8: The performance accuracy obtained for the Sentiment Analysis dataset of CrowdFlower in a streaming setting with window size 50 with $c = \frac{1}{|Q|}$, where Q denotes the set of questions. The best accuracy in each row is shown in bold.

Window	Number of raters	Approach 1	Approach 2	Approach 3
1	50	71.5576 %	71.7833 %	**72.0090** %
2	100	81.0384 %	80.8126 %	**81.2641** %
3	150	**84.6501** %	84.4244 %	**84.6501** %
4	200	86.9074 %	87.3589 %	**87.5847** %
5	250	88.9391 %	88.9391 %	**89.1648** %
6	300	90.2935 %	**90.0677** %	**90.0677** %
7	350	**90.5192** %	**90.5192** %	**90.5192** %
8	400	**90.5192** %	**90.5192** %	**90.5192** %
9	450	90.7449 %	**90.9707** %	**90.9707** %
10	500	90.7449 %	90.9707 %	**91.1964** %
11	550	90.9707 %	91.1964 %	**91.4221** %
12	600	90.7449 %	91.1964 %	**91.4221** %
13	650	90.7449 %	91.1964 %	**91.4221** %
14	700	90.7449 %	91.1964 %	**91.4221** %
15	750	90.7449 %	91.1964 %	**91.4221** %
16	800	91.1964 %	91.1964 %	**91.4221** %
17	850	91.1964 %	91.1964 %	**91.4221** %
18	900	91.1964 %	91.4221 %	**91.6479** %
19	950	91.1964 %	91.4221 %	**91.6479** %
20	1000	91.4221 %	91.6479 %	**91.8736** %
21	1050	91.4221 %	91.6479 %	**91.8736** %
22	1100	91.4221 %	91.6479 %	**91.8736** %
23	1150	91.4221 %	91.6479 %	**91.8736** %
24	1200	91.4221 %	91.8736 %	**92.0993** %
25	1250	91.1964 %	91.6479 %	**92.0993** %
26	1300	91.1964 %	91.6479 %	**92.0993** %
27	1350	91.1964 %	91.6479 %	**92.0993** %
28	1400	91.1964 %	91.6479 %	**92.0993** %
29	1450	91.1964 %	91.6479 %	**92.0993** %
30	1500	91.4221 %	91.6479 %	**92.3251** %
31	1550	92.0993 %	91.8736 %	**92.7765** %
32	1600	92.0993 %	92.3251 %	**92.7765** %
33	1650	92.0993 %	92.3251 %	**92.7765** %
34	1700	92.0993 %	92.3251 %	**92.7765** %
35	1750	92.0993 %	92.3251 %	**92.7765** %
36	1800	92.0993 %	92.3251 %	**93.0023** %
37	1850	92.0993 %	92.3251 %	**93.0023** %
38	1900	92.0993 %	92.3251 %	**93.0023** %
39	1950	92.0993 %	92.3251 %	**93.0023** %

Table 9: The performance accuracy obtained for the Sentiment Analysis Judgment Dataset of CrowdFlower. The best accuracy over the column is shown in bold.

Algorithm	Accuracy(%)	Data Setting
Approach 3 (Proposed Best)	**93.0023**	Streaming
Majority Voting	92.3251	Batch / Streaming
Weighted Majority Voting	91.6479	Batch / Streaming

- 0 (*Negative*) converted to $(1,0,0,0)$

- 1 (*Neutral*) converted to $(1,1,0,0)$

- 2 (*Positive*) converted to $(-1,0,0,0)$

- 3 (*Tweet not related*) converted to $(1,1,1,0)$

- 4 (*I can't tell*) converted to $(1,1,1,1)$

Here we use the *Euclidean Distance Metric* as the distance metric d which is defined in the equation 2.

For two vector $u = (u_1, u_2, \ldots, u_4), v = (v_1, v_2, \ldots, v_4)$

$$d = \sqrt{\sum_{i=1}^{4} (u_i - v_i)^2} \tag{2}$$

Results:

Here we show the result using the distance metric mentioned in equation 2 for example. For Table 10, we consider all the options present in the dataset and also consider the question difficulties to make judgments about the questions.

Table 11 shows the comparison of our approach with some predefined approach for this data. All approaches are implemented on a full set of data. Here we see that our approach gives more accuracy than the Weighted Majority Voting Algorithm but a little less than the Majority Voting Algorithm.

1.9 Conclusion and Future Work

Here, we employ a data structure to keep the opinions of a subset of the annotators. Using this data structure, our proposed algorithm derives the judgment for each question. We demonstrate the effectiveness of our method on three datasets, namely the WVSCM dataset[46], the fact evaluation dataset of Google, and the judgment analysis

Table 10: The performance accuracy obtained for the Sentiment Analysis dataset of CrowdFlower in a streaming setting with window size 50 with $c = \frac{1}{|Q|}$, where Q denotes the set of questions.

Window	Number of raters	Our Approach
1	50	55.1000 %
2	100	71.7000 %
3	150	77.5000 %
4	200	81.4000 %
5	250	84.0000 %
6	300	84.9000 %
7	350	85.4000 %
8	400	86.0000 %
9	450	86.3000 %
10	500	86.6000 %
11	550	87.3000 %
12	600	87.3000 %
13	650	87.3000 %
14	700	87.3000 %
15	750	87.2000 %
16	800	87.5000 %
17	850	87.6000 %
18	900	87.3000 %
19	950	87.3000 %
20	1000	87.3000 %
21	1050	87.4000 %
22	1100	87.4000 %
23	1150	87.5000 %
24	1200	87.7000 %
25	1250	87.8000 %
26	1300	87.8000 %
27	1350	87.8000 %
28	1400	87.9000 %
29	1450	87.9000 %
30	1500	87.9000 %
31	1550	88.0000 %
32	1600	88.1000 %
33	1650	88.1000 %
34	1700	88.1000 %
35	1750	88.1000 %
36	1800	88.1000 %
37	1850	88.1000 %
38	1900	88.1000 %
39	1950	88.2000 %

Table 11: The performance accuracy obtained for the Sentiment Analysis dataset of CrowdFlower. The best accuracy over the column is shown in bold.

Algorithm	Accuracy(%)	Data Setting
Our Approach	88.2000	Streaming
Majority Voting	**90.2000**	Batch / Streaming
Weighted Majority Voting	83.0000	Batch / Streaming

dataset of CrowdFlower. We obtain more than 73% accuracy for the WVSCM dataset, more than 90% accuracy in the fact evaluation dataset of Google, and the judgment analysis dataset of CrowdFlower and show that further improvements can be achieved through a recursive process.

In this dissertation, a new type of approach is proposed for judgment analysis when the data comes in a streaming way. The proposed approach is the first attempt at judgment analysis. This approach is efficient irrespective of the number of annotators but the number of options for each question should be fixed or more specifically 2 i.e., 'Yes'-'No' or 'Positive'-'Negative, etc.

For the proposed algorithm, we need to know the accuracy of each annotator. We find that accuracy for each dataset using a small portion of gold data. Basically, getting the final judgment depends greatly on an annotator's consistency and accuracy across all of the questions. Along with the well-known majority voting algorithm, other extant algorithms have been compared to the suggested strategy. Though other algorithms are based on batch data and we are working on streaming data, we compare our results with that algorithm at the final stage. Question difficulty, annotator biasness, spammers, annotator confidence, etc. are significant characteristics in the judgment analysis issue. The proposed model can also integrate fresh feature selection techniques to produce more precise labelings. Also, our proposed algorithm is working on such a scenario when the number of options is 2. But in reality, the number of options may be more than 2 in many question sets. In this dissertation, we also show that the proposed approach is effective by taking the appropriate distance metric. So, finding the appropriate distance metric needs to pay further attention.

Availability

The codes are available at the following GitHub repositories:

1. Codes applied on the WVSCM dataset (TABLE 1, 2, 3)

2. Codes applied on the Fact Evalution dataset(TABLE 4, 5, 6)

29

CHAPTER 2

Judgment Analysis on the Stream of Multi-dimensional Opinions

2 Judgment Analysis on the Stream of Multi-dimensional Opinions

In the previous chapter, we discussed on the problem of Judgment analysis on single-dimensional streaming opinions. Now we address the problem of judgment analysis on the stream of multidimensional opinions. Instead of the options available for the questions, here annotators can provide multidimensional opinions like the location of ATM machines, positions of sewage lines, etc.

2.1 Introduction

Our first work is dedicated to single-dimensional opinions (e.g., options like yes, no, maybe, etc.). In this work, we extend this to multidimensional opinions wherein a single opinion comprises multiple dimensions (e.g., coordinates of the position of sewage line, etc. instead of the yes, no option). We showcase the effectiveness of this method in addressing a practical issue, namely aggregating the opinions for the position of sewage lines in some areas when the opinions are coming in a streaming way. This can be used for smart city planning.

In the past decade, there has been a clear observation that optimized use of human resources [20] can effectively solve various real-life problems in a very efficient way. The idea of crowdsourcing a large task has gained immense popularity as a highly promising method to efficiently complete the job [8, 33, 45, 46]. Furthermore, in real-life applications such as positioning of sewage lines, etc., accurately understanding the specific public demand in various areas of the city/village is essential. Besides, annotators should have in-depth demography and locality-based population density of the city/village. Moreover, collecting all the information at a time or storing all the responses together is a major challenge as most cases the data came in a streaming setting.

The establishment of an efficient sewage system is crucial for the smooth functioning and sustainable development of any community or city. A well-designed sewage line network ensures the safe and efficient disposal of wastewater, protecting public health and preserving the environment [35]. However, the successful implementation of a sewage line project relies heavily on careful evaluation and judgment of various factors to determine the optimal placement of the sewage lines. One such example is demonstrated in Fig. 2.1.

Though the problem to construct sewage lines is not new, implementing the lines by taking the opinions of several people is a new kind of problem. At the time of imple-
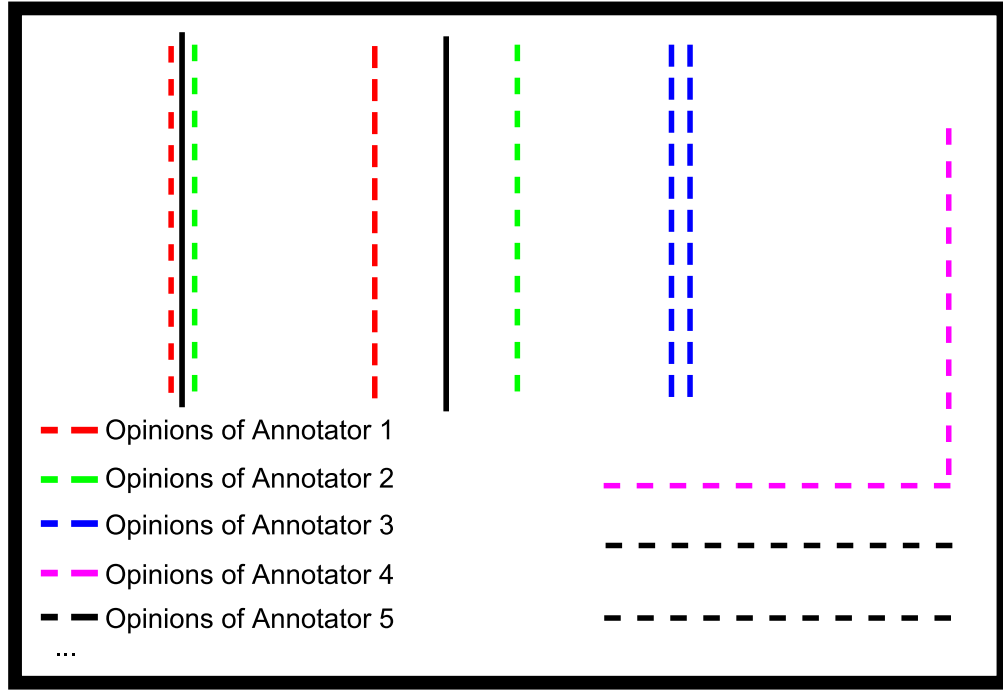
Figure 2.1: An example scenario that aims to construct sewage lines by taking opinions from the crowd (citizens taking part as the annotators). The boundary area denotes the smart city to be designed. The dashed lines represent the opinions (the suggested paths along which the sewage lines are requested to be constructed) of annotators and the solid black line (the consensus path along which the sewage lines will go) represents the judgment obtained from the valid opinions coming from the annotators upto time T_i. We assume that each sewage line should have a maximum length and the distance between two sewage lines requested for should satisfy some minimum value. Clearly, not all the opinions received from each crowd are acceptable. Here, the opinions of Annotator 3 and Annotator 4 are not accepted as their opinions are too close, which violates the constraints. The multiple opinions received from the crowd constitute the multi-dimensional judgment analysis problem to be addressed.

mentation, some constraints should be followed. In some cases, some sewage lines are already present. But for betterment, some more sewage lines need to be added. In this case, the number of constraints should be increased. We consider the simple case for our problem i.e., no sewage line is present in the place. Also, all the annotators who provide their opinion, are not equally efficient or have equal knowledge to provide the opinions. This problem opens a new research direction of judgment analysis on streaming data and a simple example is illustrated here.

To delve deeper into this concept, let's explore a scenario where an organization with limited knowledge about a city aims to build several sewage lines (For example the number is 3). Again, before implementation, we need to keep in mind that there is some maximum length for each sewage line and any two-sewage line cannot be placed very closely. Therefore, to gather information about the actual public demand for sewage lines, this issue can be delegated to individuals through an open forum, allowing them to share their opinions and insights. So, every annotator should have to provide 6 points (i.e., two endpoints for each sewage line). In this case, applying a simple majority voting method for aggregated decision-making is not suitable. Component-wise majority voting does not ensure the preservation of constraints. Here, we propose the very first approach to solve the sewage line problem when opinions came from different annotators.

The rest of this dissertation is organized as follows. Section 2.2 discusses our proposed method, its time complexity, and space complexity. In section 2.3 we show some experiments on three datasets by our proposed approach. Finally, section 2.4 concludes the dissertation.

2.2 Proposed Method

In this section, we detail on the proposed method of judgment analysis for multi-dimensional opinions received in the form of streaming data.

2.2.1 Judgment Analysis

If we draw the sewage lines on the map of that location, then we can see sewage lines as line segments. So, for simplicity, we can consider each sewage line as a straight-line segment. A line segment is a finite portion of a line that connects two distinct points. These points, known as endpoints, define the boundaries of the segment.

To analyze and evaluate line segments effectively, we often need to make judgments about different points along the segment. These judgments help us understand the relative positions, distances, and relationships between these points. By examin-

ing the properties of line segments, we can gain insights into geometric shapes, angles, and other elements of spatial reasoning.

Firstly, each annotator does not provide his opinion in the same way. For example, let A and B be two annotators who provide their opinion. Let, the opinion of A be $\{(x_{11}, y_{11}), (x_{12}, y_{12}); (x_{21}, y_{21}), (x_{22}, y_{22}); (x_{31}, y_{31}), (x_{32}, y_{32})\}$ and the opinion of B be $\{(x_{21}, y_{21}), (x_{22}, y_{22}); (x_{31}, y_{31}), (x_{32}, y_{32}; (x_{11}, y_{11}), (x_{12}, y_{12})\}$. Though the sequence of the opinion of A and B is not the same but their responses are the same. So both responses should be treated as same after sorting. So we need to sort the sequence in such a way that the resultant sequence is exactly the same. Also, we need to take care of the fact that if two responses for a particular line be $\{(x_{11}, y_{11}), (x_{12}, y_{12})$ and $\{(x_{12}, y_{12}), (x_{11}, y_{11})$ then they are also same after sorting. Also, every annotator does have not equal expertise in demography and locality-based population density. Moreover, there are some annotators who do not follow the constraints. For taking care of this problem we increase the weightage of the response of the annotators who provide more valid responses. Finally, we calculate the mean for each endpoint of the line segment which will be the final endpoints for the line segments.

2.2.2 Proposed Algorithm

Here we consider two cases. Firstly, we need to draw one single-line segment. Here, we give equal importance to every annotator who provides valid judgment. The algorithm, in this case, is mentioned in the algorithm 2. Here we use SWAP(u, v) which swaps the two inputs.

The DIV() subroutine

DIV(A, Tot){

  1: $P.start[1] \leftarrow \frac{A.start[1]}{Tot}$

  2: $P.start[2] \leftarrow \frac{A.start[2]}{Tot}$

  3: $P.end[1] \leftarrow \frac{A.end[1]}{Tot}$

  4: $P.end[2] \leftarrow \frac{A.end[2]}{Tot}$

  5: return P

}

Now we consider that we need to make more than one sewage line. So, we need to update the algorithm 2 in some sense so that it will work. Also, we now consider the weightage of the response provided by the annotators. This algorithm is mentioned in the algorithm 3

In line number 10 of algorithm 2 *number of valid response by a_i* is added with Tot_{index} to increase the weightage of the responses of the corresponding annotator. Line num-

Algorithm 2 Judgment of getting two endpoints of a line segment

Input: Co-ordinates of two endpoints of a line segment provided by an annotator or a window of annotators containing the coordinates of the two endpoints, a special type of data structure A containing previously stored the mean of co-ordinates coming upto T_i timestamp i.e., the mean of all the co-ordinates of start points $A.start$ and the mean of all the co-ordinates of endpoints $A.end$, the total number of points Tot already came up to T_{i-1} timestamp.

Output: Co-ordinates of two endpoints of the final line segment at T_i timestamp i.e., the updated A, the total number of points Tot already came up to T_i timestamp.

Algorithmic Steps:

1: **for** the response $R_i = \{(x_i, y_i), (u_i, v_i)\}$ of each annotator a_i **do**
2:      $Tot \leftarrow Tot + 1$
3:      **if** (x_i, y_i) lies in the right of (u_i, v_i) **then**
4:          $(x_i, y_i), (u_i, v_i) \leftarrow$ SWAP$((x_i, y_i), (u_i, v_i))$
5:      **else if** (x_i, y_i) not lies in the left of (u_i, v_i) and lies in the above of (u_i, v_i) **then**
6:          $(x_i, y_i), (u_i, v_i) \leftarrow$ SWAP$((x_i, y_i), (u_i, v_i))$
7:      **end if**
8:      $A \leftarrow A + DIV(R_i, Tot)$
9: **end for**

Algorithm 3 Judgment of getting two endpoints of multiple line segment

Input: Co-ordinates of endpoints for m line segments provided by an annotator a_i be $R_i = \{R_{i1}, R_{i2}, ..., R_{im}\}$ where $R_{ik} = \{(x_{ik}, y_{ik}), (u_{ik}, v_{ik})\}$, a special type of array data structure A whose each element contains previously stored the corresponding mean of co-ordinates coming up to T_i timestamp, an array Tot stored the weightage of each line up to T_{i-1} timestamp, the minimum distance between two line segment dis, the maximum length of any line segment len

Output: Co-ordinates of endpoints of the line segments after judgment of T_i timestamp, i.e., the updated array A, the array Tot stored the weightage of each line up to T_i timestamp

Algorithmic Steps:

1: **for** the responses R_i of each annotator a_i **do**
2:      $R_i \leftarrow$ POSITIONING(R_i)
3:      Index $\leftarrow$ []
4:      **for** each line k in R_i **do**
5:          Index.append(k)
6:      **end for**
7:      R_i, Index $\leftarrow$ LENGTH$(R_i,$ Index$)$
8:      R_i, Index $\leftarrow$ DISTANCE$(R_i,$ Index$)$
9:      **for** each index in Index **do**
10:         $freq \leftarrow$ number of valid response by a_i
11:         $Tot_{index} \leftarrow Tot_{index} + freq$
12:         $A_{index} \leftarrow A_{index} + freq * DIV(R_{index}, Tot_{index})$
13:      **end for**
14: **end for**

The POSITIONING() subroutine

POSITIONING(R_i){

1: **for** each line k **do**
2:     **if** (x_{ik}, y_{ik}) lies in the right of (u_{ik}, v_{ik}) **then**
3:         $(x_{ik}, y_{ik}), (u_{ik}, v_{ik}) \leftarrow$ SWAP$((x_{ik}, y_{ik}), (u_{ik}, v_{ik}))$
4:     **else if** (x_{ik}, y_{ik}) not lies in the left of (u_{ik}, v_{ik}) and lies in the above of (u_{ik}, v_{ik}) **then**
5:         $(x_{ik}, y_{ik}), (u_{ik}, v_{ik}) \leftarrow$ SWAP$((x_{ik}, y_{ik}), (u_{ik}, v_{ik}))$
6:     **end if**
7: **end for**
8: **for** each two-line j and k when $j < k$ **do**
9:     **if** (x_{ij}, y_{ij}) lies in the right of (x_{ik}, y_{ik}) **then**
10:         $(x_{ij}, y_{ij}), (x_{ik}, y_{ik}) \leftarrow$ SWAP$((x_{ij}, y_{ij}), (x_{ik}, y_{ik}))$
11:     **else if** (x_{ij}, y_{ij}) not lies in the left of (x_{ik}, y_{ik}) and lies in the above of (x_{ik}, y_{ik}) **then**
12:         $(x_{ij}, y_{ij}), (x_{ik}, y_{ik}) \leftarrow$ SWAP$((x_{ij}, y_{ij}), (x_{ik}, y_{ik}))$
13:     **end if**
14: **end for**
15: return R_i

}

The LENGTH() subroutine

LENGTH(R_i, Index){

1: **for** each line k in R_i **do**
2:     $length \leftarrow l_2((x_{1k}, y_{1k}), (u_{1k}, v_{1k}))$ $\backslash\backslash l_2(u, v)$ denotes l_2 norm of u and v
3:     **if** $length > len$ **then**
4:         delete endpoints $(x_{1k}, y_{1k}), (u_{1k}, v_{1k})$ from R_i and its index from Index
5:     **end if**
6: **end for**
7: return R_i, Index

}

The DISTANCE() subroutine

DISTANCE(R_i, Index){

1: **for** each two-line j and k in R_i when $j < k$ **do**
2:     $d \leftarrow l_2(line_j, line_k)$ $\backslash\backslash\ l_2(line_j, line_k)$ denotes distance between $line_j$ and $line_k$
3:     **if** $d < dis$ **then**
4:         delete endpoints $(x_{1k}, y_{1k}), (u_{1k}, v_{1k})$ from R_i and its index from Index
5:     **end if**
6: **end for**
7: return R_i, Index

}

ber 11 finds the mean of the endpoints to obtain the final judgment up to the response of the annotator a_i.

2.2.3 Justification of Line 11-12 in Algorithm 3

In this algorithm, we propose the first approach to find the best position of the sewage lines after taking opinions from the annotators by increasing the weightage of consistent annotators and then finding the mean of all responses up to the timestamp T_i and reporting the position. This is proposed in lines 10-12 in Algorithm 3. So, justification of these lines is necessary. In line 11, we increase the weightage of the response of a_i by the number of valid responses. Line 12 finds the mean whose justification is given below.

Let $x_1, x_2, ..., x_s$ be the points whose mean is stored in A_j for some j, and not all points are necessarily distinct. Now a new point x_p came with weightage $freq$. Let, B_j be the new mean of points and we get B_j after adding u with A_j.

$A_j = \frac{1}{s} \sum_{i=1}^{s} x_i$

$B_j = \frac{1}{s+freq} (\sum_{i=1}^{s} x_i + freq * x_p)$

Also, $B_j = A_j + u \implies u = B_j - A_j$

So, $u = \frac{1}{s+freq} (\sum_{i=1}^{s} x_i + freq * x_p) - \frac{1}{s} \sum_{i=1}^{s} x_i$

$\implies u = \frac{s(\sum_{i=1}^{s} x_i + freq * x_p) - (s+freq) \sum_{i=1}^{s} x_i}{s(s+freq)}$

Hence, $u = \frac{freq * (x_p - A_j)}{s+freq}$

So, $B_j = A_j + \frac{freq * (x_p - A_j)}{s+freq}$

2.2.4 Time Complexity Analysis

Clearly, the *SWAP subroutine* takes $O(1)$ times and the *DIV subroutine* takes $O(1)$ times. As a result, the time complexity for the algorithm 2 is $O(1)$ for each annotator.

As the *SWAP subroutine* takes $O(1)$ times, the *POSITIONING subroutine* takes $O(m^2)$ times where m is the number of lines that needs to implement. The *LENGTH subroutine* will take $O(m)$ times and the *DISTANCE subroutine* will also take $O(m^2)$ times where m is the number of line that needs to implement. So, in algorithm 3 line 2 takes $O(m^2)$ times, line 4-6 $O(m)$ times, line 7 takes $O(m)$ times, line 8 takes $O(m^2)$ times, line 9-13 takes $O(m^2)$ times. So, this algorithm takes $O(m^2)$ times for each annotator a_i. Now if the window size at any timestamp is k then the time complexity for that window be $O(km^2)$

2.2.5   Space Complexity Analysis

Clearly, the *SWAP subroutine* and the *DIV subroutine* need not require any space. As a result, the space complexity for the algorithm 2 is $O(1)$. This space is required to store the endpoints of the line segment which is always updated after each annotator's response.

As, the *SWAP subroutine*, the *POSITIONING subroutine*, and the *LENGTH subroutine* need not require any space. As a result, the space complexity for the algorithm 3 is $O(1)$. This space is required to store the endpoints of the line segment which is always updated after each annotator's response.

2.3   Empirical Analysis

We try to find the position of the sewage lines by getting opinions in a streaming way from some annotators.

Dataset Details:

The data is collected by taking the opinions of some students, researchers, and faculties of the Indian Statistical Institute, Kolkata, and some of the people who have some prior knowledge of the map of the Indian Statistical Institute, Kolkata. They are requested to provide their opinions for making three sewage lines from Google map provided to them where the length of any sewage line should be a maximum of 0.004 units and the distance between any two line segments should be at least 0.00076 units. This data is stored in a .csv file which contains 8 columns, where the 1st column contains the time of the opinion provided by the annotator, the 2nd column contains the name of the annotator and rest six columns contain the co-ordinate of endpoints of the sewage lines respectively.

Design of Experiment:

We consider that each annotator chooses the points from the exact region of the given map and no one can choose any points outside of the region. Each annotator has prior knowledge about the campus of the Indian Statistical Institute, Kolkata and so they are appropriate people to choose as an annotator for this job.

Results:

Here we applied our proposed method to the dataset. After the responses of each annotator, the endpoints of each sewage line are shown in Table 12. Since the opinions are coming in a streaming way, every time (i.e., the response of each annotator) we store the end-points of the sewage lines which reflect in Table 12. Here we consider that in each window, the response of only one annotator has come.

39

Table 12: The result shows the endpoints of the sewage lines using our approach. Here we consider that the response of only one annotator came in each window.

Annotator	1st of 1st line	2nd of 1st line	1st of 2nd line	2nd of 2nd line	1st of 3rd line	2nd of 3rd line
1	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
2	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
3	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
4	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
5	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
6	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
7	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
8	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
9	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
10	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
11	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)
12	(22.64496919, 88.37635788)	(22.64602119, 88.37642188)	(22.6502146, 88.3770222)	(22.6509818, 88.3768024)	(22.64726633, 88.37679333)	(22.64874267, 88.37639567)

2.4 Conclusion and Future Work

Here we propose the very first method of a particular problem of Judgment Analysis on the Stream of Multidimensional Opinions. We also deploy the method on a dataset where opinions had been collected from some students, researchers, faculties of ISI Kolkata, and some other people who have some prior knowledge about the area. This is the very first approach to this problem in the stream setting. Further attention needs to develop the method. Also, we propose the method when no sewage line is present. But in many cases, new sewage lines need to implement when already some sewage lines are present to develop the drainage system. So, more constraints are present in that case. Some attention is needed to solve this problem. On the other hand, this is one kind of problem in Judgment Analysis on the Stream of Multidimensional Opinions. More problems on this topic can be formulated and solved by proposing some methods.

Availability

The code is available at the following GitHub repository:

- Codes applied on the Sewage lines dataset (Table 12)

References

[1] Omar Alonso, Daniel E Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR forum*, volume 42, pages 9–15. ACM New York, NY, USA, 2008.

[2] M Bhattacharyya. Opinion ensembling: Learning from dependent judgements of the crowd. *Proc. Crowdscale Shared Task Challenge*, page 1, 2013.

[3] Kevin J. Boudreau, Nicola Lacetera, and Karim L. Lakhani. Incentives and Problem Uncertainty in Innovation Contests: An Empirical Analysis. *Management Science*, 57(5):843–863, 2011.

[4] Kevin J Boudreau, Nicola Lacetera, and Karim R Lakhani. Incentives and problem uncertainty in innovation contests: An empirical analysis. *Management science*, 57(5):843–863, 2011.

[5] Kevin J Boudreau and Karim R Lakhani. Using the crowd as an innovation partner. *Harvard business review*, 91(4):60–9, 2013.

[6] Daren C. Brabham. *Crowdsourcing*. MIT Press, 2013.

[7] Thierry Buecheler, Jan Henrik Sieg, Rudolf Marcel Füchslin, and Rolf Pfeifer. Crowdsourcing, open innovation and collective intelligence in the scientific method: a research agenda and operational framework. In *The 12th International Conference on the Synthesis and Simulation of Living Systems, Odense, Denmark, 19-23 August 2010*, pages 679–686. MIT Press, 2010.

[8] S. Chatterjee and M. Bhattacharyya. Judgment analysis of crowdsourced opinions using biclustering. *Information Sciences*, 375:138–154, 2017.

[9] Sujoy Chatterjee, Anirban Mukhopadhyay, and Malay Bhattacharyya. Dependent judgment analysis: A markov chain based approach for aggregating crowdsourced opinions. *Information Sciences*, 396:83–96, 2017.

[10] Sujoy Chatterjee, Anirban Mukhopadhyay, and Malay Bhattacharyya. Smart city planning with constrained crowd judgment analysis. In *2017 AAAI Spring Symposium Series*, 2017.

[11] Sujoy Chatterjee, Anirban Mukhopadhyay, and Malay Bhattacharyya. A review of judgment analysis algorithms for crowdsourced opinions. *IEEE Transactions on Knowledge and Data Engineering*, 32(7):1234–1248, 2019.

[12] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

[13] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proc. WWW*, pages 469–478, 2012.

[14] Dominic DiPalantino and Milan Vojnovic. Crowdsourcing and all-pay auctions. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 119–128, 2009.

[15] Schahram Dustdar and Martin Gaedke. The social routing principle. *IEEE Internet Computing*, 15(4):80–83, 2011.

[16] Francesca Gino and Bradley R Staats. The microwork solution. *Harvard Business Review*, 90(12):92–+, 2012.

[17] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

[18] Ahsanul Haque, Latifur Khan, Michael Baron, Bhavani Thuraisingham, and Charu Aggarwal. Efficient handling of concept drift and concept evolution over stream data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 481–492. IEEE, 2016.

[19] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning Whom to Trust with MACE. In *Proc. NAACL-HLT*, pages 1120–1130, Atlanta, Georgia, 2013.

[20] Jeff Howe et al. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[21] Panagiotis G Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM magazine for students*, 17(2):16–21, 2010.

[22] Panagiotis G. Ipeirotis. Analyzing the Amazon Mechanical Turk Marketplace. *ACM XRDS*, 17(2):16–21, 2010.

[23] Aniket Kittur, Jeffrey V. Nickerson, Michael S. Bernstein, Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Matthew Lease, and John J. Horton. The Future of Crowd Work. In *Proc. CSCW*, pages 1301–1318, 2013.

[24] E. Liberty. Simple and Deterministic Matrix Sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

[25] C. Liu and Y. Wang. Truelabel + confusions: A spectrum of probabilistic models in analyzing multiple ratings. In *Proc. ICML*, pages 225–232, 2012.

[26] Q Liu, J Peng, and A Ihler. Report of crowdscale shared task challenge 2013. *Proc. Crowdscale Shared Task Challenge*, page 2, 2013.

[27] S. Bhattacharya M. Bhattacharyya and S. Bandopadhyay. Estimating Completeness in Streaming Graphs. *Proceedings of the EDBT/ICDT International Workshop on Multimodal Social Data Management*, pages 294–299, 2014.

[28] Thomas W Malone, Robert Laubacher, and Chrysanthos Dellarocas. The collective intelligence genome. *MIT Sloan management review*, 2010.

[29] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[30] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[31] Vikas C. Raykar and Shipeng Yu. Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks. *Journal of Machine Learning Research*, 13:491–518, 2012.

[32] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *Proc. CHI EA*, pages 2863–2872, 2010.

[33] Joel Ross, Lilly Irani, M Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers? shifting demographics in mechanical turk. In *CHI'10 extended abstracts on Human factors in computing systems*, pages 2863–2872. 2010.

[34] R. Rubinfeld and A. Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011.

[35] R Saravanane, Vivek V Ranade, Vinay M Bhandari, and A Seshagiri Rao. Urban wastewater treatment for recycling and reuse in industrial applications: Indian scenario. *Chapter*, 7:283–322, 2014.

[36] Aashish Sheshadri and Matthew Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proc. HCOMP*, 2013.

[37] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems*, 1995.

[38] R. Snow, B. O'Connor, D. Jurafsky, and Andrew Y. Ng. Cheap and fast–but is it good? Evaluating non-expert annotations for natural language tasks. In *Proc. EMNLP*, pages 254–263, 2008.

[39] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Proc. CVPRW*, pages 1–8, 2008.

[40] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *2008 IEEE computer society conference on computer vision and pattern recognition workshops*, pages 1–8. IEEE, 2008.

[41] M. Steyvers, Michael D. Lee, B. Miller, and P. Hemmer. The wisdom of crowds in the recollection of order information. *Advances in Neural Information Processing Systems*, 22:1785–1793, 2009.

[42] John C Tang, Manuel Cebrian, Nicklaus A Giacobe, Hyun-Woo Kim, Taemie Kim, and Douglas" Beaker" Wickert. Reflecting on the darpa red balloon challenge. *Communications of the ACM*, 54(4):78–85, 2011.

[43] Lav R Varshney, Shivali Agarwal, Yi-Min Chee, Renuka R Sindhgatta, Daniel V Oppenheim, Juhnyoung Lee, and Krishna Ratakonda. Cognitive coordination of global service delivery. *arXiv preprint arXiv:1406.0215*, 2014.

[44] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Proc. NIPS*, volume 6, 2010.

[45] Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. The multidimensional wisdom of crowds. *Advances in neural information processing systems*, 23, 2010.

[46] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*, 22, 2009.

[47] Ruvolo P. Wu T. Bergsma J. Whitehill, J. and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043, 2009.

[48] Hua Ye and Atreyi Kankanhalli. Leveraging crowdsourcing for organizational value co-creation. *Communications of the Association for Information Systems,* 33(1):13, 2013.