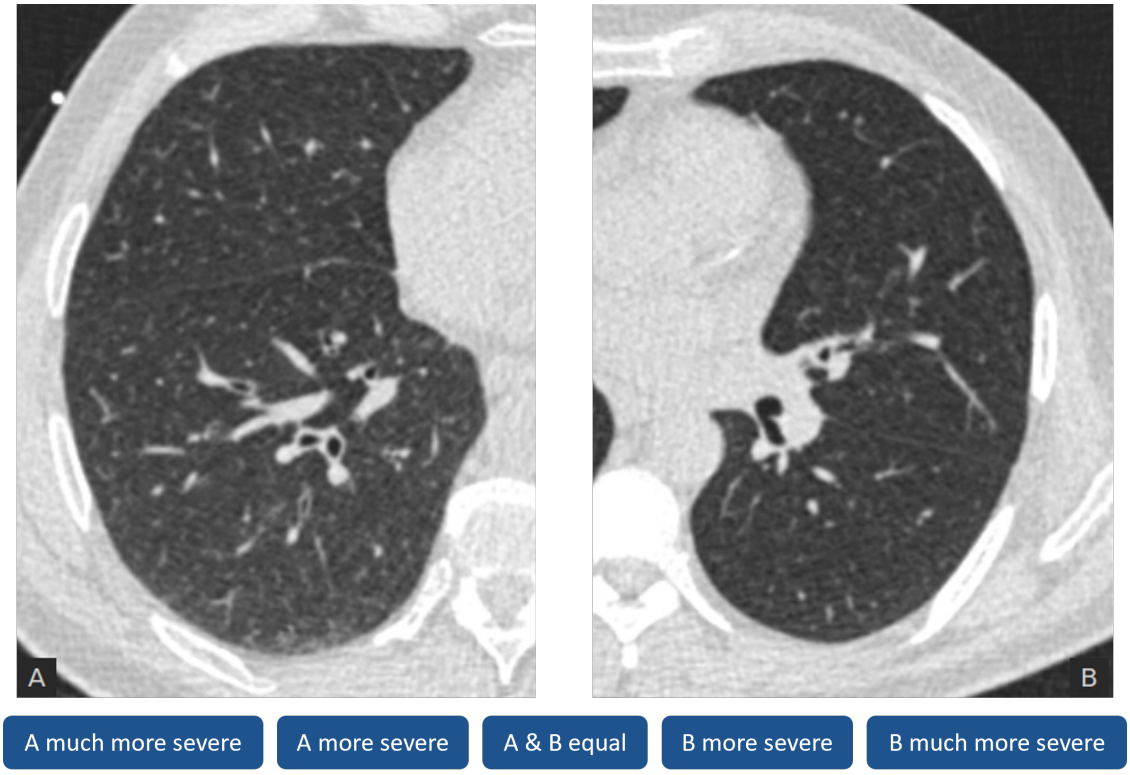


Rank-based annotation system for subjective assessments in supervised learning

Application in computed tomography of the lungs

Master's thesis in Computer science and engineering

Herman Bergström

Hanna Tärnåsen

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Rank-based annotation system for subjective assessments in supervised learning

Application in computed tomography of the lungs

Herman Bergström
Hanna Tärnåsen

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Rank-based annotation system for subjective assessments in supervised learning
Application in computed tomography of the lungs
Herman Bergström, Hanna Tärnåsen

Supervisor: Ida Häggström, Department of Electrical Engineering
Supervisor: Mats Lidén, School of Medical Sciences Örebro
Examiner: Ida Häggström, Department of Electrical Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of the rank-based annotation system created in this thesis.

Typeset in LaTeX
Gothenburg, Sweden 2023

Rank-based annotation system for subjective assessments in supervised learning
Application in computed tomography of the lungs
Herman Bergström, Hanna Tärnåsen
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Supervised learning has become a common approach for extracting information from images. To effectively train a model, a large amount of labeled data is required. While some image annotation tasks are objective and well-defined, others require the annotators to make a subjective assessment. The difficulty and subjective nature of these annotation tasks cause the standard rating-based annotation techniques to suffer from inconsistencies between annotators, implying that two different annotators could assign highly differing labels based on their personal biases. This thesis' overarching goal is to provide an alternate rank-based system for annotating subjective data that could be applied to supervised learning, with the hope of increasing the quality of labels.

The target application for this project is the annotation of the degree of bronchial wall thickening seen in CT scans of the lungs in patients with chronic obstructive pulmonary disease (COPD). Four potential implementations are compared, and consistency, as well as resource demands, are evaluated in several parts. These include imitating the annotation process with simulation, user evaluation with arbitrary subjective assessments, and lastly evaluating bronchial wall thickenings with radiologists.

After evaluation, it is observed that the implementation showing the most potential is one based on the TrueSkill algorithm, which employs Bayesian inference and assumes that underlying scores are not definite but instead follows a normal distribution. The findings presented in this thesis indicate a clear increase in inter-annotator agreement for this rank-based system and the study demonstrates that the indirect approach of evaluating images creates more reliable labels than the direct rating-based method.

Keywords: Approximate Sorting, Noisy Sorting, TrueSkill, Rank-Based Annotation, Subjective Annotations, Supervised Machine Learning,

Acknowledgements

We would first and foremost like to thank our supervisor, Mats Lidén, for his continued guidance and engagement. This thesis would not be possible without the expertise you offered in the field of medical imaging, and the efforts you made to gather annotated data. We look forward to working with you in the future. Further, we would also like to thank our examiner and Chalmers supervisor Ida Häggström. We greatly appreciate the feedback you offered during the initial stages of this thesis, as well as the opportunity to reach out to you with any and all questions.

Finally, we would like to offer our sincerest gratitude to all the people who volunteered to annotate data for this project. This includes the 13 friends and acquaintances who offered their time during the initial user evaluation. It also includes the four experts at Örebro University Hospital who provided the annotations of bronchial wall thickening, essential for this thesis. The data each of you helped us gather has been invaluable when attempting to achieve the goals of this project.

Herman Bergström and Hanna Tärnåsen, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1

Introduction

Since the rise of convolutional neural networks (CNN), the approach of supervised learning has become increasingly popular for solving image analysis tasks [1]. Common applications include image classification, object detection, and segmentation. However, in order to train a CNN, a large amount of labeled data is required. The process of creating labeled data is referred to as annotation, and the exact nature of the annotation task can vary depending on the target application. Many of these tasks have an objective ground truth and do not entail much cognitive workload, such as drawing bounding boxes around cars or stating if an image contains a human or not. In contrast, some annotations tasks instead have participants make a subjective assessment. Such assessments could for example be establishing the quality of a video [2] or determining the emotional state of a person [3].

An issue that can be encountered when applying the standard rating-based method to these tasks, where the annotator simply assigns a number or label to an image, is that results vary heavily depending on the annotator. It not only becomes an issue of how the annotator perceives the image but also of how they perceive the ratings or labels of the annotation task. Furthermore, assessments do not necessarily have an underlying ground truth, implying that results will vary regardless of the competence of annotators. A proposed solution that has shown promise is to instead use a rank-based approach [4], [5]. This involves creating an ordering of the images by comparing them, rather than assigning labels directly.

Although a rank-based system has the potential to increase annotator consistency, it also demands a different amount of resources compared to a rating-based one. As the set of data points that are to be ordered increases in size, the amount of possible pairwise comparisons increases quadratically. Consequently, if annotation resources are limited, exhaustive pairwise comparisons will quickly become infeasible. One field where this is the case is medical imaging, where annotation requires trained radiologists. While there have been studies within this field to limit the number of comparisons required to rank Computed Tomography (CT) images [6], these have mostly concerned individual radiologists sorting a list of images by themselves. As the comparisons are subjective, the opinions of radiologists will differ, inducing noise into the annotations.

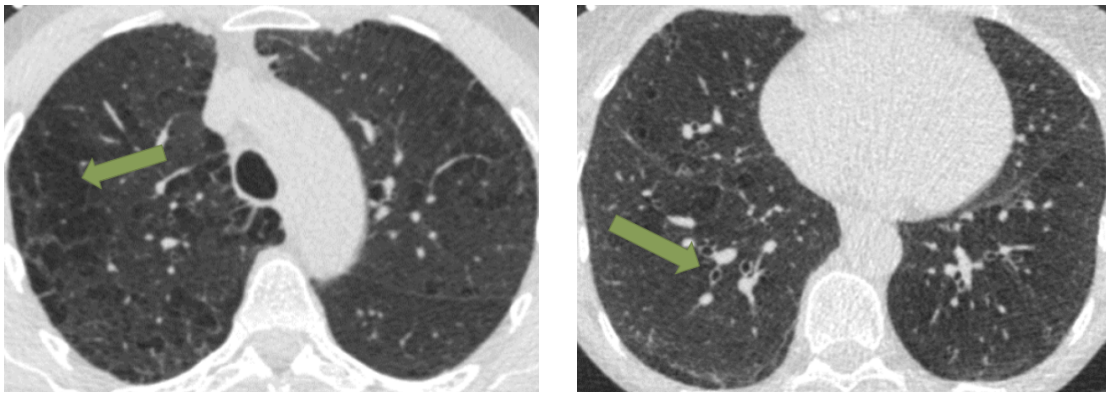
The objective of this thesis is to construct a novel rank-based method for subjective image annotation which can manage noisy comparisons. The system should be able to account for the opinions of multiple annotators while handling disagreements.

With a target application in medical imaging, the desired system should be able to create a reasonable ordering of elements given only a sample of noisy comparisons. Further, the system should allow radiologists to produce more consistent labels than a rating-based system would.

1.1 Chronic Obstructive Pulmonary Disease

Chronic Obstructive Pulmonary Disease (COPD) is a respiratory disease generally caused by smoking [7]. Typical symptoms are shortness of breath and cough. According to the World Health Organization, it is the third leading cause of death worldwide [8]. COPD is diagnosed with a pulmonary function test (spirometry) revealing chronic airway obstruction [9], [10]. The disorder causes a variable combination of emphysema (i.e. destruction of the alveoli) and small airway disease, and usually, the larger airways are also affected. The relative amount of emphysema and airway involvement in a patient defines different COPD subtypes - phenotypes - that cannot readily be distinguished with the pulmonary function test.

Computed tomography is the best non-invasive method for visualizing the lungs, making a visual assessment of emphysema and bronchial involvement in COPD possible [11]. Emphysema causes dark cystic air spaces in CT images of the lungs, an example of which can be seen in Figure 1.1a. Affected airways can be visible as a thickening of the bronchial walls, highlighted in Figure 1.1b.



(a) An example of the emphysema phenotype. The arrow highlights the destroyed lung tissue.

(b) An example of bronchial wall thickening. The arrow highlights the swollen wall.

Figure 1.1: CT image demonstrations of phenotypes prominent in COPD.

While spirometry is used for diagnosing COPD, quantifying the degree of bronchial wall thickening and emphysema in CT images can differentiate COPD phenotypes [12]. Phenotyping is important in research and clinical medicine, for example when investigating therapies directed towards one of the disease components. Furthermore, visual and quantitative CT evaluation has been shown to add prognostic information in COPD, even when spirometry data is present [13].

One project that aims to predict and prevent pulmonary disease is the Swedish cardiopulmonary bioimage study (SCAPIS) [14], [15]. The SCAPIS study is a Swedish population-based cohort for research in Cardiovascular Disease and COPD. A total of 30,000 randomly chosen Swedish individuals between the ages of 50 and 64 have received thorough medical examinations as part of the study, including blood tests, CT imaging, and spirometry. The dataset provides a large amount of CT scans of the lungs in individuals of different health, including people suffering from COPD where the aforementioned phenotypes can be observed. Quantifying the degree of bronchial wall thickening among subjects in SCAPIS could help lead to insights and further understanding of the pathology of COPD.

Nevertheless, extracting reliable airway measurements from CT scan slices has proven to be a difficult task, with most research having been focused on using classical image analysis methods, such as segmentation, to measure the airway wall [16], [17]. This makes it difficult to create an automated process for quantifying the degree of bronchial wall thickening visible in an image. One of the main issues is that the size of small airways is around the size of most CT scan resolutions, meaning that it is just barely measurable [18]. Despite this, radiologists are able to get an impression and make assessments when looking at these images. In an attempt to mimic this, an interesting new approach would be to instead train a CNN to predict the degree of bronchial wall thickening, as a number in an arbitrary range. However, as this method is a case of supervised machine learning, it consequently requires labeled data.

1.2 Objective and Method

The primary objective of this thesis is to implement and evaluate a rank-based annotation system where images are ordered through comparisons. The long-term goal is to create a system that would ultimately enable more consistent and reliable annotation of subjective data. The target application, used to evaluate the performance of the system, is the annotation of bronchial wall thickening visible in CT images. However, the developed system should:

- Allow for any type of comparison where a subjective assessment is made.
- Produce an approximate ordering of the elements given a sample of potentially noisy comparisons.
- Support groups of annotators ordering data together by managing disagreements.

The approach taken has been to look into the area of sorting algorithms in order to compare various solutions and their applicability and improve on those that were recognized as relevant. This will be covered in more detail in Section 3.1.

1.3 Contributions

These specific advancements in the field are incorporated within this thesis work:

- Experimental evaluation of the performance of different sorting algorithms in the case of noisy comparisons.
- A novel method for subjective annotation of data.
- Empirical evidence that annotation via pairwise comparison can increase consistency and inter-annotator agreement compared to a rating-based system when annotating bronchial wall thickening.

1.4 Thesis Outline

Chapter 2 will present the theoretical background for understanding the thesis. It will cover basic concepts such as sorting regular and noisy data. It will also explain some probabilistic topics as well as offer some more in-depth explanations of parts relevant to the implemented systems. Furthermore, evaluation methods, including inter-annotator agreement and Spearman's footrule distance, will also be discussed.

Chapter 3 will detail the methodology used when implementing the sorting algorithms, including Merge Sort, Borda Count, Hamming LUCB, and TrueSkill, and the overall evaluation structure used in this project.

Chapter 4 will present the results of the simulation experiment, including the outcomes for the various sorting algorithms. The chapter will also provide a conclusion that summarizes the findings which will then be further evaluated in the next chapter.

Chapter 5 will detail the evaluation of the system on arbitrary subjective annotations. The results will include group and individual testing, and their setup will be explained. The chapter will conclude with information that can be passed down to the next evaluation step.

Chapter 6 will present the final evaluation of the thesis, the annotation of bronchial wall thickening in CT scans. Here the results and conclusions from chapters four and five will be used for the final assessment and testing.

Finally, Chapter 7 will conclude with a summary of the main findings and their implications, limitations of the study, and suggestions for future research. This conclusion will tie together the various sections of the thesis and provide a comprehensive overview of the study.

2

Theory

This chapter covers the technical background essential to comprehend the key concepts presented in this thesis. We start by defining time complexity and addressing how it can be used to estimate the efficiency of generic algorithms. Thereafter, the theoretical sorting of a list with merge sort is described using a more concrete example. The idea of noisy sorting, or sorting data with noise, is then introduced. Following that, a probabilistic perspective is thoroughly explored, beginning with an explanation of Bayesian inference and proceeding with Gaussian distribution. To better understand some of the implementations in this thesis, the two major topics, TrueSkill and confidence bounds in multi-armed bandits, are also covered. Finally, this chapter will discuss various evaluation techniques that will be used in the report.

2.1 Time Complexity

The efficiency of sorting algorithms can be measured by their time complexity, which is defined as the number of operations required to sort a given set of data. Time complexity is often expressed in terms of the input size n and is used to estimate how the algorithm's performance would scale with the increase or decrease in the input data [19]. The big Oh notation is a commonly used metric in computer science to describe the time and space complexity of an algorithm. It is written as $O(f(n))$ and provides a way to categorize an algorithm's growth rate as the input size increases.

There exist some primary types of complexities that are used to evaluate the efficiency of algorithms based on their time complexity. These include constant time complexity ($O(1)$), logarithmic time complexity ($O(\log n)$), linear time complexity ($O(n)$), linearithmic time complexity ($O(n \log n)$), quadratic time complexity ($O(n^2)$), exponential time complexity ($O(2^n)$), and factorial time complexity ($O(n!)$) [20]. The complexities can be ordered from the best to the worst, with constant time complexity being the most efficient and factorial time complexity being the least efficient ($1 \ll \log n \ll n \ll n \log n \ll n^2 \ll 2^n \ll n!$) [19].

2.2 Merge Sort

Merge Sort is a well-known sorting algorithm with the expected time complexity of $O(n \log n)$ [19]. As the name suggests, the core of the algorithm is centered around merging already sorted sublists. This is done by first splitting the list into sublists

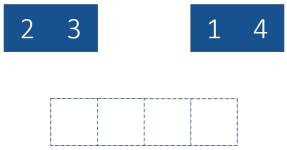
only containing one element. When merging sublists, they are merged two and two until completion where elements are put in the correct order for each merge, the process of merging two lists can be seen in Algorithm 1. The amount of comparisons this requires is reduced as a consequence of the sublists themselves already being sorted.

Algorithm 1 General process for merging two stored lists

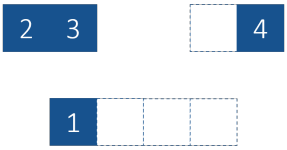
function MERGETWOSORTEDARRAYS($leftSubArray, rightSubArray$)
  $i \leftarrow 0$, $j \leftarrow 0$
  $merged \leftarrow$ initialize an empty list
  while $i <$ len($leftSubArray$) and $j <$ len($rightSubArray$) **do**
    if $leftSubArray[i] \leq rightSubArray[j]$ **then**
      Append $leftSubArray[i]$ to $merged$
      $i \leftarrow i + 1$
    else
      Append $rightSubArray[j]$ to $merged$
      $j \leftarrow j + 1$
    end if
  end while
  Append the remaining elements of $leftSubArray$ to $merged$
  Append the remaining elements of $rightSubArray$ to $merged$
  return $merged$
end function

To further grasp the concept the process is exemplified in Figure 2.1 and Figure 2.2.

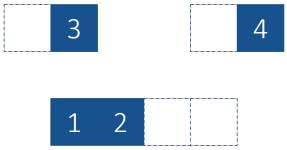
Figure 2.1: In this image a list with the elements 2,3,4, and 1 has been divided into sublists only containing one element. Thereafter the first two sublists are compared with the conclusion that $2 < 3$, merging them to the sublist $\{2, 3\}$. Similarly, $\{4\}$ and $\{1\}$ are merged to the sublist $\{1, 4\}$.



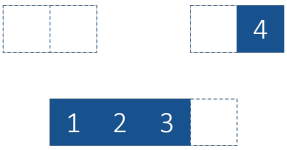
(a) Merging setup with the two first sublists that are to be merged. Where in this case these are the only 2 sublists available $[2, 3]$ and $[1, 4]$



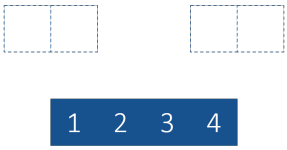
(b) Comparing the first two elements in the sublists; 2 and 1. Since $1 < 2$, it can be deduced that 1 will be the smallest element. It is subsequently moved from the sublist to the sorted merged list.



(c) Comparing the first two elements in the current sublists; 2 and 4. Concludes in 2 entering the merged list as the second smallest element.



(d) Comparing again the first two elements in the current sublists; 3 and 4. Concludes in 3 entering the merged list as the third smallest element.



(e) With elements only remaining in one of the sublists, the rest of this list can simply be appended and subsequently complete the merging.

Figure 2.2: This image visualizes the process after the creation of the two sorted subsets created in Figure 2.1. It shows step by step how the lists $[2, 3]$ and $[1, 4]$ merges into the final sorted list $[1, 2, 3, 4]$.

While not constructed to manage noisy comparisons but rather assumes deterministic data, merge sort still makes for an interesting benchmark as it does have the goal of minimizing the number of comparisons required.

2.3 Noisy Sorting

In contrast to the deterministic sorting setting, noisy sorting assumes that the outcome of a comparison will vary according to some probability. For example, in the model introduced by Braverman et al. [21], there is an unknown true order, and the probability of a correct comparison is $\frac{1}{2} + \lambda$, where $\lambda > 0$ is a constant. As a proposed solution, the authors apply maximum likelihood estimation to estimate an ordering of the elements given observed noisy comparisons. However, there are additional approaches to address the noisy sorting problem. One such approach is to view the issue as a multi-armed bandit problem [22], which refers to a hypothetical scenario in which an individual chooses between actions that have different rewards.

In contrast, another approach is attempting to minimize a sorting cost function, such as that produced by a probabilistic interpretation of the least squares function [23].

While there are potentially many different attributes the same set of data can be ordered according to, the problem can be generalized. Similar for all orderings is that the data is being assessed based on some quality. To simplify the explanations in this thesis, if a comparison between two elements i and j determines that i is of greater quality (e.g. the comparison implies $i > j$), this thesis will refer to this as element i winning a comparison against element j.

2.4 Bayesian Inference

Bayesian inference is a framework that revolves around updating the beliefs about a distribution according to some observed data [24]. Let $y \in \mathbf{Y}$ be the observed data, and $\theta \in \Theta$ be the parameters that describe a likelihood function $P(y \mid \theta)$. Here, both θ and y could potentially be multi-dimensional. After having made observations of y, and given a prior distribution $P(\theta)$, the distribution of θ can be updated using Bayes' rule, which states that

$$P(\theta \mid y) = \frac{P(y \mid \theta)P(\theta)}{P(y)} = \frac{P(y \mid \theta)P(\theta)}{\int_\theta P(y \mid \theta)P(\theta)}.$$

In this case, $P(\theta \mid y)$ is referred to as the posterior distribution. As the marginal distribution $P(y)$ can become very costly to calculate, it is common to utilize the notion of conjugate priors. That is, if the posterior $P(\theta \mid y)$ and the prior $P(\theta)$ are in the same distribution family, then the posterior can be calculated using a closed-form expression, avoiding integration. It is then said that $P(\theta)$ is a conjugate prior to $P(y \mid \theta)$. For example, let the likelihood function $P(y \mid \theta)$ follow a binomial distribution such that $y \mid \theta \sim Bin(n, \theta)$. Further, assume that the parameter θ follows a beta distribution with parameters α and β, that is $\theta \sim Beta(\alpha, \beta)$. After observing a sample y, the posterior will also follow a beta distribution. More specifically, $\theta \mid y \sim Beta(\alpha+y, \beta+n-y)$. For more information regarding conjugate families, we refer to the book by Johnson et al. [25].

2.5 Gaussian Distribution

The Gaussian distribution (often called normal distribution) is a prevalent continuous distribution family [26]. It is defined by two parameters; the mean μ and the standard deviation σ. For simplicity, the distribution is most commonly denoted $\mathcal{N}(\mu, \sigma^2)$ and the probability density function is defined as

$$pdf(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

With a bell-shaped curve, the distribution is symmetric around its mean. This makes it commonly used for the estimation of values where the positive deviation should be about the same as the negative deviation.

2.6 TrueSkill

The TrueSkill algorithm is a skill rating system originally developed by Microsoft for Xbox Live online matchmaking [27]. The algorithm is often seen as a generalization of the Elo system, which was originally introduced to rank chess players [28]. However, TrueSkill also utilizes Bayesian Inference and factor graphs to estimate the underlying skills of players, while also keeping track of the uncertainty of its estimations. The main idea is that the skill s of a player follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, modeling the uncertainty of the true skill of a player. As matches are played, the current expected skill μ, as well as the confidence σ, will be updated. For a match with n players, let $\mathbf{s} \coloneqq (s_1, ..., s_n)$ be the vector containing the skills of each player. Further, team assignments are defined by k non-overlapping subsets $A_j \subset \{1, ..., n\}$, and A denotes the set of all teams $\{A_1, ..., A_k\}$. Subsequently, the vector $\mathbf{r} \coloneqq (r_1, ..., r_k)$ represents the outcome (rank) of each team. The posterior update of a game is given by the equation

$$P(\mathbf{s} \mid \mathbf{r}, A) = \frac{P(\mathbf{r} \mid \mathbf{s}, A)\, P(\mathbf{s})}{P(\mathbf{r} \mid A)}.$$

For the details on how this posterior is calculated, we refer to the original paper by Herbrich et al. [27]. Apart from updating the estimation of a player's skill level, the system also predicts the quality of a match, given the players on each team, defined as the probability of a draw.

While originally designed for video game matchmaking, the TrueSkill algorithm can be applied to any system where comparisons between two or more items are being made, and the outcomes of the comparisons are inconsistent (e.g. noisy). For example, one can imagine that instead of comparing players, we compare CT scans of bronchial wall thickening. Furthermore, instead of attempting to estimate an underlying skill of a player, one can attempt to estimate the degree of bronchial wall thickening present in an image. The problem can be reformulated to fit any annotation task where a subjective assessment is made.

2.7 Borda Count

Borda Count is a simple, yet robust, noise-resistant sorting algorithm [29]. The idea is that every element in a list has an associated score. After a pairwise comparison between two elements, the scores of both are updated. The element that wins the comparison has its score increased by 1, and conversely, the element that loses has its score decreased by 1. Which two elements to compare are chosen at random,

meaning that the score of every element will reflect its ability to win against another random element in the collection. The final estimated sorting is produced by ordering the elements according to their scores.

2.8 Confidence Bounds in Multi-Armed Bandits

Multi-armed bandits' is a field of research where algorithms make decisions under uncertainty [30]. An algorithm collects rewards by selecting one arm (action) a every discrete time step. The reward an arm returns is probabilistic, with each arm having an unknown expected reward $\mu(a)$. The goal of the algorithm is to maximize the total reward over a set amount of rounds, by leveraging exploration and exploitation. To do this, the algorithms keep track of the currently estimated expected reward $\bar{\mu}(a)$ of the arms, as well as how confident they are in that estimation. To represent this, confidence intervals, a notion common in statistics [31], is used. More specifically, a confidence radius $r(a)$ is introduced for each arm. The main idea is that the probability that $\bar{\mu}(a)$ and $\mu(a)$ differ by more than $r(a)$ is small. More formally,

$$P(|\bar{\mu}(a) - \mu(a)| \leq r(a)) \geq 1 - \lambda,$$

where λ is some small constant derived from the definition of r. Following this, an Upper Confidence Bound (UCB), as well as a Lower Confidence Bound (LCB), can be defined as

$$UCB(a) = \bar{\mu}(a) + r(a) \qquad \text{and} \qquad LCB(a) = \bar{\mu}(a) - r(a)$$

respectively. The interval $[LCB(a), UCB(a)]$ is referred to as the confidence interval. Within the multi-armed bandits' field, algorithms can utilize these bounds when deciding which arm to pull next. For example, given two arms a and a', if $UCB(a) < LCB(a')$, then it can with high probability be assumed that $\mu(a) < \mu(a')$. One algorithm of interest for this project which utilizes confidence intervals is the Hamming LUCB sorting algorithm.

2.9 Hamming LUCB

The Hamming LUCB algorithm is similar to Borda Count, but it utilizes a few parameters to determine the next comparison. The algorithm was initially developed by the authors Heckel et al. for an approximate ranking task [22]. The primary parameter that differentiates Hamming LUCB from Borda Count is the number of subsets that the list is split into, as well as their respective sizes. Rather than estimating the exact ordering of elements, the algorithm aims to split the list into n subsets in such a way that we can be confident enough that the elements in subset i are larger than those in subset $i - 1$. If n is equal to the size of the list, this gives the exact ordering.

In addition to keeping track of an estimated score, each element also has an associated confidence radius. By ensuring that the lower and upper confidence bounds of two elements chosen from two neighboring subsets do not overlap, the approximate sorting is produced. The permitted error margin between the subgroups is the second parameter. This is necessary because it would require numerous comparisons before the confidence intervals of two adjacent elements stopped overlapping. Instead, the number of necessary comparisons can be drastically decreased by adding a gap between the subsets and making sure they dont overlap. The elements within the error margin may, however, not be correctly assigned to the subset according to this method.

2.10 Evaluation Methods

The following sections offer background on the evaluation methods that are utilized in this thesis.

2.10.1 Krippendorff's Alpha

A measure of the degree of agreement or consensus among annotators who have independently annotated the same data is called inter-annotator agreement (IAA) [32] (also referred to under various similar names such as inter-rater reliability, inter-observer reliability etc.). The degree of the annotators' agreement beyond chance is often expressed as a statistical measure. These measurements adjust for the possibility of random agreement and take into consideration both the annotators' agreement and disagreement. A high level of IAA is favored since it shows that the annotations are dependable and consistent, which boosts the trust in the data and the models that are based on it. In contrast, a low level of IAA indicates inconsistent or ambiguous annotations.

Krippendorff's Alpha [33] is one statistical measurement that uses a variety of techniques to produce an IAA measurement. The fundamental form of Krippendorff's alpha is

$$1 - \frac{D_o}{D_e},$$

where D_o denotes the observed difference and D_e is the difference anticipated by chance. The outcome α equating to 0 would essentially be the result of the observed annotations being indistinguishable from annotations selected at random. In contrast, an α equal to 1 would be indicative of full agreement among annotators and subsequently imply perfect reliability. There is no definite distinction as to what alpha could be considered reliable. However, alphas above 0.8 are generally thought to be entirely reliable, while alphas above 0.667 are commonly considered acceptable to make educated assumptions [33]. In essence, depending on the circumstances, each situation needs to be interpreted differently.

While there are many other measurements of IAA, such as Fleiss' Kappa [34] and Cohen's Kappa [35], we decided to use Krippendorff's Alpha due to its flexible nature as it can handle more than two annotators, missing values, as well as many different

types of labels (including ordinal). Although this is not a standardized metric in the medical field, it has been utilized in some recent literature including the studies by Vikgren et al. [36] and Lidén et al. [37].

To measure the discrepancy between values assigned to data points by different annotators, numerous so-called difference functions can be used. These essentially quantify the difference between two potential value assignments; c and k. The functions are used to define D_o and D_e as

$$D_o = \frac{1}{n} \sum_c \sum_k o_{ck}\, \delta_{ck}^2 \qquad \text{and} \qquad D_e = \frac{1}{n(n-1)} \sum_c \sum_k n_c\, n_k\, \delta_{ck}^2,$$

where o_{ck}, n, n_c, and n_k are the frequencies in the respective coincidence matrix, which keep track of agreements and disagreements amongst annotators, and δ_{ck}^2 represents the selected appropriate difference function [38].

The simplest difference function implementation is the nominal metric approach. An observed pair of annotations is essentially assessed as either the same or different. This means that if two annotators are in disagreement, the difference is the same regardless of what labels they provided. The formula for the nominal difference function is

$$\delta_{ck}^2 = \begin{cases} 0 & \text{iff } c = k \\ 1 & \text{iff } c \neq k \end{cases}.$$

Another function more suitable for annotations where there is an underlying ordering is the ordinal difference function. This approach takes into account both the frequencies in the ranks as well as the deviations in annotators' ranks. The formula for Krippendorff's Alpha Ordinal metric differences is

$$\delta_{ck}^2 = \left(\sum_{g=c}^{g=k} n_g - \frac{n_c + n_k}{2} \right)^2,$$

where c and k are different ranks. For more information regarding the available difference functions we refer to the paper by K. Krippendorff [38].

2.10.2 Spearman's Footrule Distance

Spearmans' footrule distance is a method of estimating how much one list differs from another through the sum of the absolute differences between the indexes of each element in the two lists [39]. Thereafter it is possible to divide the sum by the overall number of list entries to obtain the distance normalized. The Spearman's footrule distance essentially calculates the degree to which the entries in two ranked lists differ from one another. Because it is non-parametric, it can be used to compare lists of any size without making any assumptions about how the data are distributed in general. The distance is more formally defined as

$$D(\pi, \sigma) = \sum_{i=1}^{n} |\pi(i) - \sigma(i)|,$$

where $\pi(i)$ and $\sigma(i)$ represents the rank of element i in list π and σ respectively.

3

Method

This chapter presents the comprehensive methodology adopted in this thesis, which is applicable to all the evaluation segments. The evaluation segments encompass the simulation evaluation, arbitrary subjective annotation evaluation, and bronchial wall thickening annotation evaluation. The methodology details the implementation of sorting algorithms, including the necessary components every algorithm should incorporate. It also provides an in-depth description of the individual sorting algorithms developed. Furthermore, the chapter outlines the various steps that were made when evaluating the algorithms, providing a holistic view of the project's structure.

3.1 Implementing Sorting Algorithms

To achieve the primary objective of this thesis, it is imperative to devise a system that can be used with simulated annotations, as well as with real annotations provided through a graphical user interface (GUI). Furthermore, it was of interest to explore the possibility for users to compare more than two elements at a time. This was explored as it could potentially increase annotation speed, and subsequently reduce the resources required. As such, an essential attribute of the algorithms was that annotations could not only be submitted in a pairwise manner but also as a list of data points that had been ordered. To ensure this, an interface that could be used across different algorithms was designed. The first step was to identify the essential components that every sorting algorithm should possess. The main components identified were the ability to retrieve the next desired comparison, and the ability to infer the current sorting of the data based on the results of that comparison. The algorithms were also required to hold the data which was represented with corresponding keys that are unique for all data points.

To facilitate the development of these algorithms, an abstract class was created to encapsulate the fundamental components required for each algorithm, including the sorting mechanism and the necessary data. The class included two primary methods that every instance of the algorithm should implement; *get_comparison()* and *inference()*.

The former method, *get_comparison(user_id)*, returns the list *keys*, which effectively contains the next elements the algorithm suggests should be compared. The amount of elements returned directly reflects the size of comparisons. For example,

two elements indicate a pairwise comparison, while four elements imply that the user is supposed to order this list of four elements. The size is determined through a parameter that is provided upon initialization of the sorting algorithm. The user who is presently annotating is passed as a parameter since in some circumstances it is considered relevant to offer comparisons on an individual basis.

inference(user_id, keys, diff_lvls), the second function, entails appropriately processing the user's annotations and incorporating them into the corresponding algorithms. The user's response is represented as the lists *sorted_keys* and *diff_lvls*, with the latter being a representation of the level of difference between the elements.

Furthermore, there are three more required functions; *get_result()*, *is_finished()*, and *comparison_is_available()*. These can essentially return the current sorted list, check if the sorting can be considered finished, and check on an individual basis if there are any possible comparisons to be made. The following pseudocode elucidates this process.

Algorithm 2 General process for ordering data with a sorting algorithm

Initiate $sort\_alg$ with *data*

while not $sort\_alg.is\_finished()$ **do**

$keys \leftarrow sort\_alg.get\_comparison()$

$sorted\_keys,\ diff\_lvls \leftarrow$ response from user when given $keys$

$sort\_alg.inference(user\_id,\ sorted\_keys,\ diff\_lvls)$

end while

The *sorted_keys* provided by the user represent a sorted list that indicates the order they believe to be correct. Additionally, the *diff_lvls* list represents the perceived difference level between adjacent elements in the list. Since there are n *sorted_keys* in the list, there are n-1 perceived differences in the *diff_lvls* list. For instance, if $sorted\_keys = [x_1, x_2, x_3]$, then $diff\_lvls =$ [difference between $x_1 \& x_2$, difference between $x_2 \& x_3$]. Representing the difference levels involves using an enum with three values:

- *None*, corresponding to no perceived difference.
- *Normal*, corresponding to a perceived difference.
- *Large*, corresponding to a large perceived difference.

These values indicate the extent of difference between adjacent elements in the list. We decided upon these levels as we reasoned that there would be scenarios where a user thinks that there is no difference between two elements, as well as scenarios where the user thinks the difference is very clear. By representing the difference levels in this way, more information can be gained from the user's sorting preferences.

When sorting lists with more than two elements, it is important to demonstrate how the algorithm can be traversed. To do this, all $(n-1)!$ comparisons must be

conducted, with each comparison being treated as a pairwise comparison. The maximum difference levels between the respective comparisons can then be determined and stated as their difference level. For example, let us consider a list of sorted elements $[x_1, x_2, x_3, x_4]$ and their corresponding difference levels [*None*, *Large*, *Normal*]. When sorting this list, there are actually $3! = 6$ pairwise comparisons. By adopting the largest difference level between the two compared data points for each pairwise comparison the result becomes (x_1, x_2) : *None*, (x_1, x_3) : *Large*, (x_1, x_4) : *Large*, (x_2, x_3) : *Large*, (x_2, x_4) : *Large*, and (x_3, x_4) : *Normal*. As such in the individual algorithms, it is possible to represent the original sorted list as these multiple pairwise comparisons.

The following sections will cover the algorithms which have been implemented following this interface. When choosing which algorithms to evaluate in the project, we have attempted to take a multitude of approaches, utilizing different fields of theory, into account. The four algorithms which have been implemented are Merge Sort, Borda Count, Hamming LUCB, and TrueSkill. Firstly, as Merge Sort is one of the more popular algorithms used in the deterministic sorting setting, we wanted to include it as a benchmark algorithm which does not take noise into consideration. Secondly, Borda Count was implemented due to its combined simplicity and robustness, making it an excellent benchmark for noisy sorting algorithms. Thirdly, Hamming LUCB was of interest as it took inspiration from the multi-armed bandits literature while also offering termination criteria for when the list is approximately sorted. Lastly, the TrueSkill algorithm was integrated as it is able to choose the next comparison in a more dynamic fashion than the aforementioned algorithms, while utilizing the Bayesian inference framework.

3.1.1 Merge Sort

The Merge Sort algorithm follows a simple and intuitive approach. The developed algorithm instantiates by splitting the data into sub-lists, each containing one element. These sub-lists are then merged two at a time, with each element being appended in the correct order, eventually resulting in the complete list being sorted. For instance, let us consider an unsorted list of elements $[x_1, x_2, ..., x_{n-1}, x_n]$. When Merge Sort is instantiated with regards to this data, it will create a list of sub-lists $[[x_1], [x_2], ..., [x_{n-1}], [x_n]]$.

The *get_comparison()* function is straightforward as it returns the first key in the first two sub-lists. However, the inference step, *inference()*, of this algorithm can be quite complex, as it requires determining which elements to merge and in what order. When merging two sub-lists only consisting of two elements the merged list is quite clear. However, when the sub-lists are larger it is not quite as intuitive. Essentially the chosen approach is to compare the two sub-lists' first elements to determine the element that is the smallest in both subsets. After having determined the smallest element, it is simply moved to the end of the new list and the process is repeated until both of the sub-lists are emptied.

Unfortunately, Merge Sort has a significant flaw that has yet to be addressed: it can only manage comparisons of exactly two elements at a time. As such, the algorithm

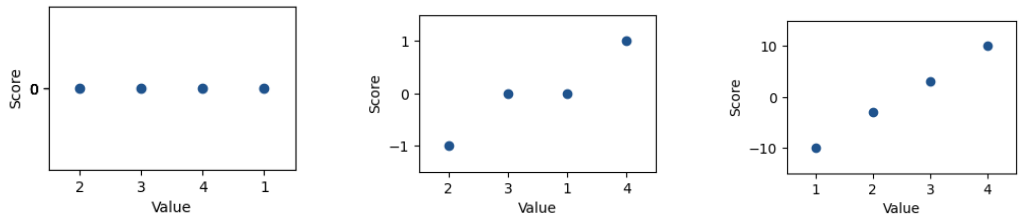
will in practice only be used with pairwise comparisons.

3.1.2 Borda Count

The Borda Count algorithm takes a different approach compared to the Merge Sort algorithm. Borda Count method includes assessing the plausibility of a win for a random element rather than assuming full certainty. At the start of the algorithm, a list of data points is created with corresponding scores, initially set to 0. The primary objective is to generate random comparisons that update the corresponding score for each data point. A win increases the score, while a loss decreases it. The final ordering of the list is achieved by sorting the elements based on their acquired scores.

For the algorithmic details, the selection of which elements to compare is arbitrary, and in our implementation, it is done by randomly pairing the elements in the list. This ensures that all data points are compared equally, as each element occurs once in the list of planned comparisons (except for a few elements in the cases where the size of the data is not divisible by the comparison size). When the algorithm has compared all paired data points, it randomly pairs elements once more. For inference, all $(n-1)!$ comparisons and their respective maximum difference levels are obtained. If there is no observed difference between the two elements, no score is updated for either element. For a win, a value of $+1$ is added to the element that scored higher, and a value of -1 is added to the element that scored lower. If the difference level is *Large*, the values added are $+2$ and -2, respectively.

Consider the simple case where the objective is to sort a list of elements, each with an individual element, where the underlying values are the discrete numbers 1 through 4. This process is visualized in Figure 3.1 where the x-axis represents the underlying true values of the elements. At first, as can be seen in Figure 3.1a, all values would be initialized to the same value, in this case 0. After one iteration, meaning outputting an arbitrary comparison and inputting the result of said comparison, the updated scores can be seen in Figure 3.1b. It is here clear that the *get_comparison()* function yielded the element pair with the true values 2 and 4, and that the user returned the information that 4 was perceived as larger than 2, meaning the elements with values 2 and for 4 got their scores updated with -1 and +1 respectively. With the same basic concept the algorithm can continue, and after 4 iterations the sorting, as seen in Figure 3.1c, is seemingly correct where the lowest value also has the lowest score.



(a) The different elements are initialized to the same value 0

(b) An arbitrary comparison is chosen, 2 and 4, and their individual scores are subsequently updated by +1 for win and -1 for loss.

(c) After 20 comparisons the scores for the elements and their respective value could look like this.

Figure 3.1: A visualization of how sorting the elements with the values [1,2,3,4] could be performed with Borda Count

One significant detail about Borda Count is that Borda Count does not have a finishing condition, and consequently requires a maximum comparison amount. This means that the algorithm will continue to make comparisons and update scores until it is explicitly stopped.

3.1.3 Hamming LUCB

As previously covered, Hamming LUCB attempts to split a list into a predetermined amount of sets with a given error margin, while keeping track of an estimated score as well as a confidence radius for each element. The radius used in this project is the same as introduced in the original paper by Heckel et al. [22]. Let $u(a)$ be the number of times element a has been compared. Further, let s be the size of the list and $d \in (0, 1)$ be a tolerance parameter, where larger values imply a higher error tolerance. The radius is then defined as

$$r(a) = \sqrt{\frac{\beta(a, d/s)}{2u(a)}},$$

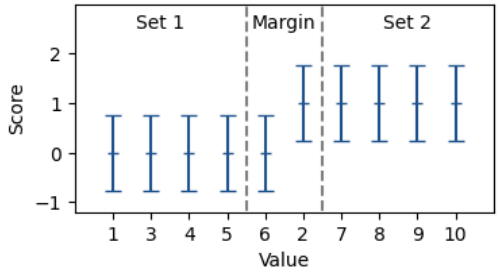
where β is defined as

$$\beta(a, d') = \log\left(\frac{1}{d'}\right) + 0.75\log\left(\log\left(\frac{1}{d'}\right)\right) + 1.5\log\left(1 + \log\left(\frac{u(a)}{2}\right)\right).$$

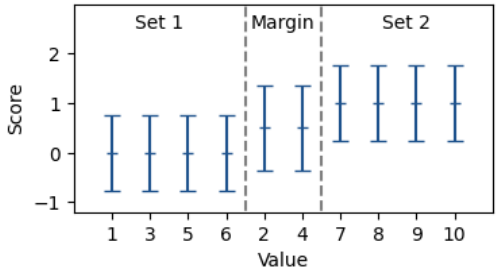
When selecting the next comparison, the algorithm considers the elements from each subset where the confidence interval is furthest outside the set, as well as each element that is currently within an error margin. Out of these, the algorithm selects the element with the largest confidence radius and compares it with a random element from the list. This approach prioritizes elements that are near the boundaries of a subset or within one of the error margins. As for the inference, it is quite similar to Borda Count as the score of an element is increased for a given win and decreased for

a loss, where a comparison labeled as *Large* would also here indicate 2 wins/losses. The formula used is $\tau_i \leftarrow \frac{T_i-1}{T_i}\tau_i + \frac{1}{T_i}1\{\text{i wins}\}$, where τ_i is the approximated score of data point i, and T_i represents the amount of times i has been compared [22]. With each inference, T_i for all received elements at the certain step is simply incremented, and their respective scores, τ_i, are subsequently updated based on the new T_i and the previous score.

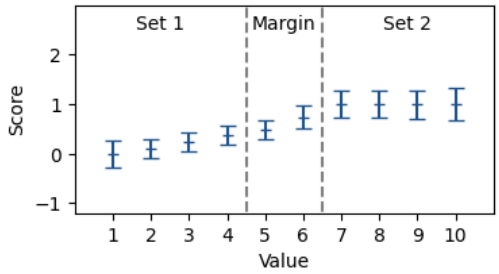
Figure 3.2 shows how the underlying representation of data points is updated during an example run. The list consists of discrete values 1 through 10. The parameters supplied imply that the algorithm should split the list into two sets of equal size, with a margin of 2 elements between them. During initialization of the algorithm, each element is compared once and given a score of 1 if it wins, or 0 if it loses, as can be seen in Figure 3.2a. During the next step, the results of which are depicted in Figure 3.2b, 2 is compared against 4 and loses. The exact choice of elements to compare here was arbitrary as each previously had the same radius. Lastly, Figure 3.2c shows the underlying representation of the algorithm after the termination criteria have been met.



(a) During the initial round, each element is compared once.



(b) The value 2 is compared to 4 and loses.



(c) The final sorting of the algorithm before it terminates. Note that no element in Set 1 overlaps with an element in Set 2. Further, note that elements closer to the edges of the sets have been compared more and subsequently have a smaller radius.

Figure 3.2: A visualization of how sorting the elements with integer values 1 through 10 could be performed with Hamming LUCB.

3.1.4 TrueSkill

The final algorithm that was implemented was TrueSkill. At its core, the algorithm uses the TrueSkill ranking system which assumes that the score of every element follows a normal distribution. This method allows the ranking of the data points with a Bayesian statistical perspective. A significant difference from Borda Count and Hamming LUCB is foremost that the certainty of a data point can not be directly derived from its number of comparisons, but rather, the TrueSkill measurement is dynamic and adaptive depending on the results of the comparisons.

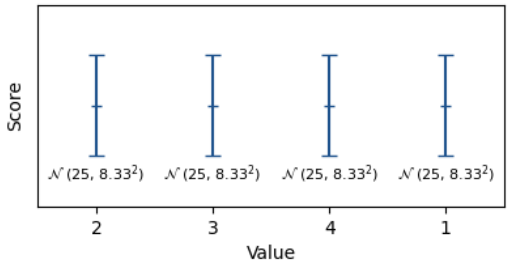
When selecting the next comparison, an adaptation was made of a previously developed algorithm called TSSort [40]. This algorithm considers the combination of elements that have the most overlap in the 95% confidence interval. Furthermore, it also takes into account the width of the widest confidence interval in the set. This is because there may be cases where the overlap between two sets of elements is nearly 100%, but their variance is very small, indicating that the data points are nearly equal in skill. Conversely, there may be another set of elements where the overlap is smaller, but the widest confidence interval is very large, indicating significant uncertainty in the ordering of those elements. By doing so, the algorithm ensures that it makes informed comparisons that take into account not simply the point estimates of the skill levels of the elements but also the level of confidence associated with those estimates. The overlap between two elements i and j can be calculated with the lower and upper limits of the data points with regard to two standard deviations. Then let $a, b := \mu_i \pm 2\sigma_i$, and $c, d := \mu_j \pm 2\sigma_j$ representing the lower and upper limits of the data points, then the formula for the weighted overlap is defined as

$$\text{Overlap}_{i,j} := \frac{\min(b,d) - \max(a,c)}{\max(b,d) - \min(a,c)} \cdot \max(b-a, d-c).$$

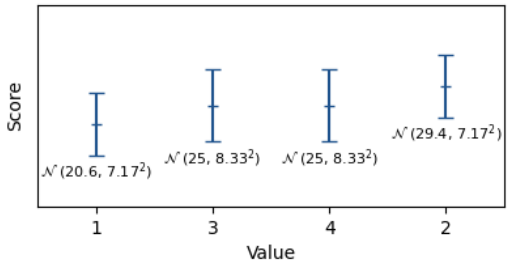
When working with more than two elements, the choice is essentially made by searching for the combination with the highest additive sum. Additionally, modifications were made to prevent the same comparisons from being presented to the same user multiple times. This risk is particularly high in situations where the data points are similar. We reason that presenting the same comparison to a user does not really offer any relevant information for this thesis, while also potentially being frustrating for the annotator. Instead, the algorithm identifies the highest-scored comparison that has not yet been compared by the user and presents that as the next comparison.

Regarding the inference step, TrueSkill uses a Bayesian approach to update the estimated skill levels of the elements. After each comparison, the algorithm updates the mean and standard deviation of the normal distribution representing the skill level of the elements. The update process takes into account the expected outcome of the comparison based on the skill levels of the elements and the uncertainty associated with each skill level. Similar to Hamming LUCB and Borda Count, when a comparison was labeled *Large* it would be treated as 2 consecutive wins. When the difference provided is *None*, TrueSkill counts it as a draw.

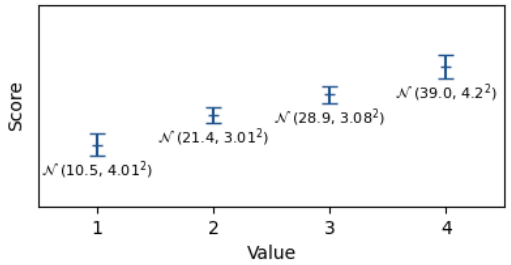
The algorithm's execution when ordering values 1 through 4 can be seen in Figure 3.3. The method instantiates with the default parameters of the TrueSkill algorithm. This means that the score of each element is assumed to follow the Gaussian distribution $\mathcal{N}(25, 8.33^2)$, as shown in Figure 3.3a. The following step, Figure 3.3b, illustrates the scenario where 2 and 1 are compared, yielding new mean and variance values. The comparison can successfully result in higher certainty regarding the points because of the newly acquired knowledge. In essence, these are the values' posterior distributions. The final estimation after 20 comparisons, as shown in Figure 3.3c results in significantly smaller variations, indicating the certainty of the estimate scores, and more exact placements of the values after performing this process.



(a) Each element is initialized with an estimated score of 25, and a variance of 8.33^2.



(b) During the next step, 2 wins over 1. The posterior distributions are calculated for both and the order is updated.



(c) After another 20 comparisons, the current estimations of the distributions could look as above.

Figure 3.3: A visualization of how sorting the list [1,2,3,4] could be performed with TrueSkill.

3.2 Evaluation Structure

The primary goal of this thesis is to compare a rank-based annotation system to a rating-based one when labeling CT scans containing varying degrees of bronchial wall thickening. However, as there are multiple ways to construct a rank-based system, and annotation for this specific task requires trained radiologists, a different form of evaluation will be needed earlier in the development process. For this reason, the project has been split into three stages. These are:

1. Simulation, covered in Section 4.
2. Arbitrary Subjective Annotation, covered in Section 5.
3. Annotation of Bronchial Wall Thickening, covered in Section 6.

The idea was to initially test all the algorithms presented in Section 3.1 using simulated noisy pairwise comparisons. By making this early comparison, algorithms that showed less promise could be excluded before any user tests were performed. Consequently, annotation resources could be more meaningfully distributed during later evaluation. The second step was to test the remaining algorithms on an arbitrary subjective annotation task that did not require any expert knowledge. This way, the results of the algorithms when people with differing opinions annotate could be compared. The best-performing sorting algorithm would then be used as the rank-based system for the final step, where it is compared to a rating-based system. A visualization of this process can be seen in Figure 3.4.

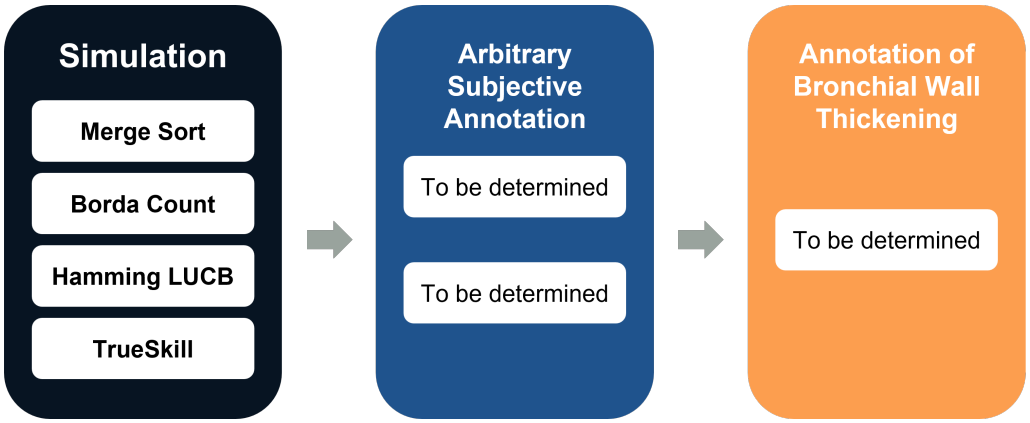


Figure 3.4: Visualization of the evaluation steps that will be made, and how algorithms will be excluded from further testing.

4

Evaluation of Simulation

Each of the four algorithms covered in Section 3.1 (i.e. Merge Sort, Borda Count, Hamming LUCB, and TrueSkill), was first evaluated using simulation. The purpose of this evaluation was to see how the algorithms performed in a noisy setting. Before making any user tests, we wanted to see which algorithms showed the most promise, so as to not waste annotation resources evaluating an algorithm that required an infeasible amount of comparisons. To this end, the algorithms were used to order a list, with known ground truth ordering, using simulated noisy pairwise comparisons.

4.1 Experimental Setup

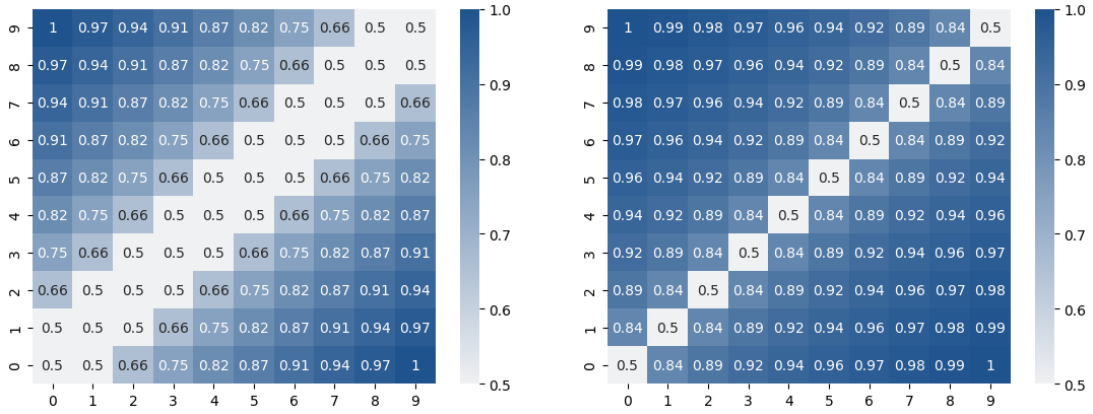
A list of integers was to be ordered. The sorting algorithms decided which elements to compare, and the simulated annotator would subsequently answer which number was the largest. The probability that the correct number was chosen to be the largest was

$$\frac{1}{2} + \gamma,$$

where γ depends on the similarity of the values. The main idea is that γ is small for values that are close to each other, and large for values that are far apart. It was assumed that this would best reflect the noise encountered in subjective assessments. For this reason, we defined the function γ as

$$\gamma(x_1, x_2) = \frac{log(c \cdot |x_1 - x_2|)}{2 \cdot log(c \cdot m)},$$

where x_1 and x_2 are two different integers from the list. Here, m denotes the largest achievable difference in the set (e.g. if the list consists of numbers between 1 and 100, then $m = 100 - 1 = 99$), and c is a parameter representing the clearness of the comparison. The impact of different levels of comparison clearness on the probability of a correct assessment can be visualized in Figure 4.1. Figure 4.1a shows the probability of a correct comparison in the case of $c = 1$. In contrast, Figure 4.1b shows the case where $c = 100$, which ensures that almost all rulings are accurate. This is demonstrated by the fact that neighboring items in Figure 4.1b are almost certain to be correctly assessed, whereas adjacent elements in Figure 4.1a still have a 50 % chance of being assessed incorrectly. In situations where $x_1 = x_2$ (i.e.



(a) Comparison clearness equal to 1 (b) Comparison clearness equal to 100

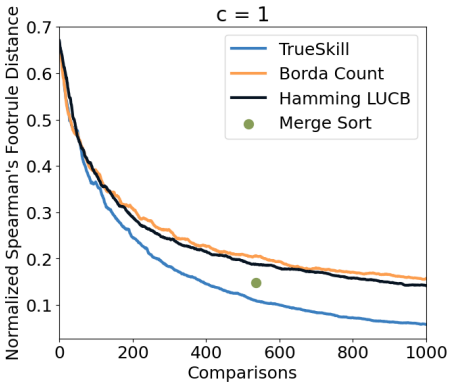
Figure 4.1: A visualization of how the probability of a correct comparison is affected by different comparison clearness. This assumes that the list only contains integers between 0 and 9 (i.e. $m = 9$).

the gamma function is undefined), the comparison is arbitrary, and the probability of both possible outcomes were 50 %.

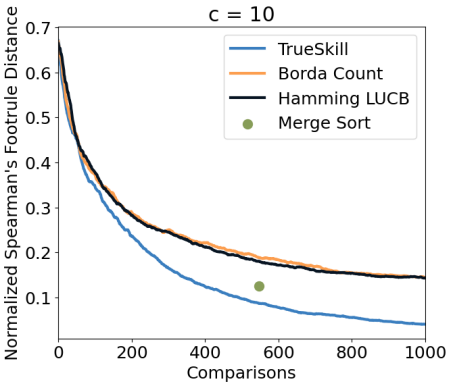
This way, the probability of a correct comparison ranges from $\frac{1}{2}$ when the values are next to each other, to 1 when the values are as far apart as possible. The parameter c can be set to any value larger than 1 in order to control how quickly the function converges. Larger values for c decrease the noise of the comparisons. The only difference level used is *Normal* as we reason that inserting a probability of the other levels appearing would not offer any more information as the simulation is simply a very rough estimation of a real annotation scenario. While this simple model will not be able to capture what the noise of the comparisons in the final application will look like, it does work to give us an indication of how the different algorithms perform compared to each other as the noise varies.

A list of 100 numbers between 1 and 200 was generated through uniform sampling. Borda Count, Hamming LUCB, and TrueSkill were run for 1000 pairwise comparisons (i.e. simulated annotations). After each comparison, the normalized Spearman's footrule distance was measured between the current estimation of the algorithm and the correct order of the list. Merge Sort was run for the number of comparisons it required, and the footrule distance was only calculated once using the final sorting.

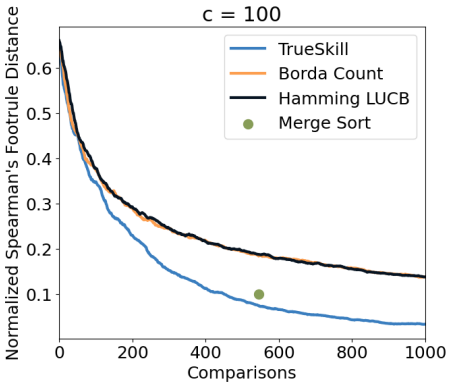
In the case of Hamming LUCB, a few parameters had to be set. Firstly, it was decided to let the algorithm attempt to split the list into 5 different subsets of equal size. Further, a margin of 4 elements between each set was used. Lastly, the tolerance parameter d was set to 0.99. For the TrueSkill algorithm, each element was initiated with the default parameters, that is $\mu = 25$ and $\sigma^2 = 8.33^2$.



(a) Graph of the normalized Spearman's footrule distance with $c = 1$.



(b) Graph of the normalized Spearman's footrule distance with $c = 10$.



(c) Graph of the normalized Spearman's footrule distance with $c = 100$.

Figure 4.2: The normalized Spearman's footrule distance, with different comparison clearness', between the current estimation of the algorithm over 1000 comparisons, averaged over 10 runs.

4.2 Results

After running these simulations ten times, each with a newly generated list, the average distance per comparison was calculated. The result of this with the comparison clearness parameter c set to 1 can be seen in Figure 4.2a. Likewise, the results of running the same experiments with $c = 10$ and $c = 100$ can be seen in Figures 4.2b and 4.2c respectively. In the graphs, the amount of comparisons of Merge Sort is the average of the 10 runs.

4.3 Conclusion

There are several takeaways from these results. Most interesting is the fact that TrueSkill appears to be converging significantly faster than any of the other algorithms. Not only does it achieve the lowest distance after 1000 comparisons, but it also outperforms Merge Sort given the same amount of comparisons. It should

however also be noted that, as the clearness parameter is increased, Merge Sort shows the greatest increase in performance. This is to be expected as the algorithm is constructed to find the optimal ordering, with as few comparisons as possible, in a noiseless setting.

Furthermore, it would seem that Borda Count and Hamming LUCB are both performing poorly in comparison. Perhaps the largest reason for this similar performance is the fact that the algorithms always compare the element of interest to another randomly selected one. This non-dynamic selection technique will lead to difficulties distinguishing between elements that are of similar quality. Although there are parameters for the Hamming LUCB algorithm that can be tuned, testing during implementation indicated that the tweaking of these will not have a noticeable effect compared to the aforementioned issue. These instead have a greater impact on the termination criteria, which was not reached during any of the 10 runs. While robust and noise resistant, these algorithms do not utilize information gained from previous comparisons, most likely making them too resource-demanding for the target application.

5

Evaluation of Arbitrary Subjective Annotation

With the knowledge obtained from running the simulations, the next step was to test the most promising sorting methods on users performing an actual annotation task. It was decided that the subjective annotation task in question would be to order images of people based on how old they appeared to be. We reasoned that this task would be simple enough for the average person to perform, while also being difficult and subjective enough to cause disagreements (i.e. noise).

Although there is no ground truth ordering, we can still measure how consistent the results produced by the algorithms are. We speculate that if multiple groups were to order the same list, a good-performing algorithm that is noise resistant would be able to produce more similar outputs compared to one that is not. Moreover, the potential added efficiency of letting the user order a shorter list, as opposed to making pairwise comparisons, can also be measured, with the caveat that the results are only representative of this specific annotation task. This, as the cognitive workload could increase if the annotation task became more difficult. As such, the goals of this experiment were to see which system produced higher quality orderings in terms of reliability, as well as to evaluate the efficiency of different comparison sizes. With these goals in mind, the two main questions posed for these initial tests were:

1. How consistent are the sorted lists produced by the different algorithms?

2. In the case of this annotation task, is there a gain in letting the user sort a list of four images instead of making pairwise comparisons?

To answer these questions, two tests were performed. The first test discussed further in Section 5.1.1, had annotators order lists as groups via pairwise comparisons using different algorithms. The orderings produced by the groups could subsequently be compared to see if any algorithm seemed to produce more consistent results. The second test that was performed, which is covered in Section 5.1.2, instead had participants annotate individually, using both pairwise comparisons, as well as letting them order a shorter list. The purpose of this test was instead to measure the time required per annotation, to see if any of the options proved more efficient.

5.1 Experimental Setup

In order to perform the tests, a graphical user interface was developed. This interface supported both pairwise comparisons as can be seen in Figure 5.1, as well as ordering of lists of size three or four, as can be seen in Figure 5.2. More images and further descriptions of the final version of the interface can be seen in Appendix A. For the pairwise comparison *Of Similar Age*, *Younger/Older*, and *Clearly Younger/Older* would directly correlate to the difference levels *None*, *Normal*, and *Large*. Similarly, for the ordering of lists, these are instead displayed above the image with *No Difference* and *Large Difference* naturally correlating to *None* and *Large* whilst no text correlates to *Normal*. Apart from saving the current state of the sorting algorithm after every comparison, the result of each comparison was also stored as a row in a CSV file, along with auxiliary information such as the time it took to make the annotation as well as which user performed it. This CSV file was vital as it would allow for later recreation of the annotations, step by step, helping us investigate convergence.

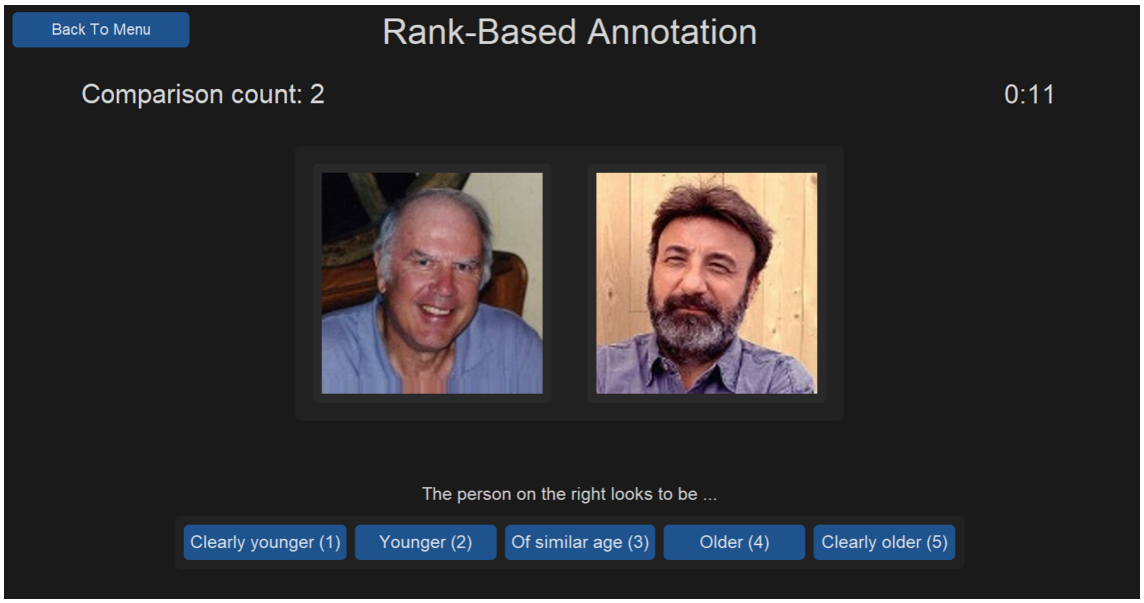


Figure 5.1: An example screenshot of the user interface when sorting by pairwise comparisons.

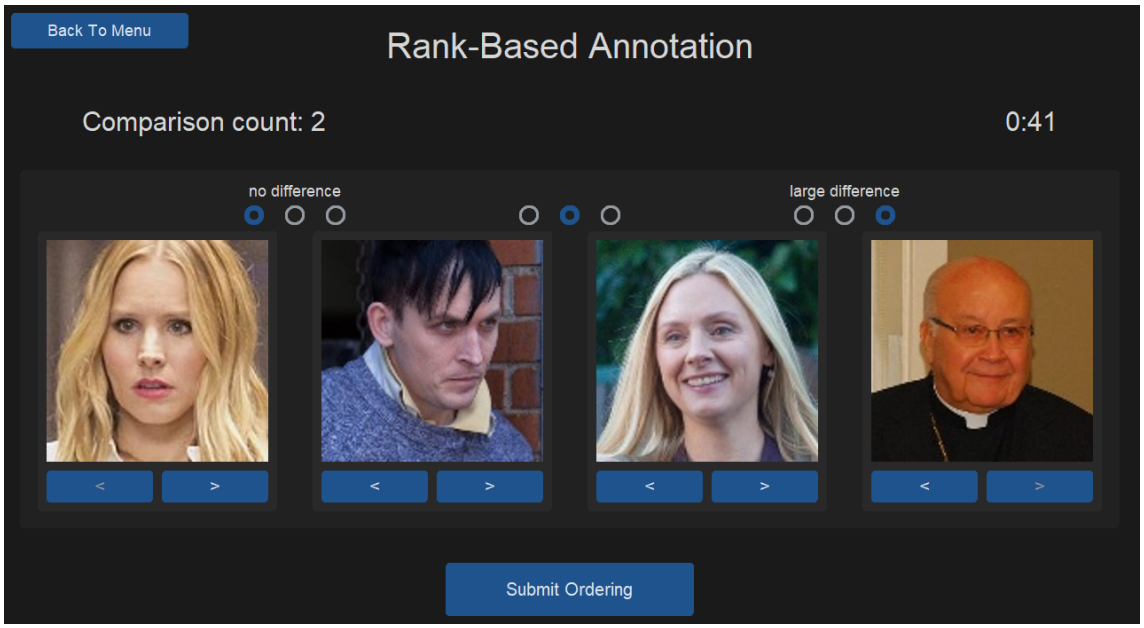


Figure 5.2: An example screenshot of the user interface when ordering a shorter list.

The images used for the experiments were collected from Kaggle [41]. An initial cleaning of the dataset was made to remove images of low quality. As this is a publicly available dataset, most of the images are of celebrities. Participants of the study were however asked to only take the appearances in the images into account, disregarding any eventual further knowledge of the person. The dataset also included labels in the form of ages, but these were ignored as the objective ground truth was not of interest for this specific study.

5.1.1 Group Testing Setup

As previously discussed, the main goal of the group annotation test was to compare how consistent the outputs of the algorithms were. The results from the simulation indicated that TrueSkill showed the most promise, so we subsequently wanted to test it against a benchmark algorithm. It was decided that this benchmark would be Merge Sort as it, apart from TrueSkill, seemed to be the best performing one given a reasonable amount of comparisons.

Nine annotators were recruited and subsequently split into three groups. Each group was to order the same list of 100 images which was noted as data set A (with different initial shuffled orderings) according to how old the person in the image appeared to be. This was done using the TrueSkill algorithm, and having each member of the group make 220 pairwise comparisons, one after another, totaling 660 comparisons. This total amount of comparisons was selected as previous simulations indicated that the algorithm had converged to a reasonable extent at this point (see Figure 4.2). Similar to Section 4, the default parameters were used when initializing TrueSkill. The groups were then randomly shuffled, and the same procedure was done for another list of 100 images which was noted as data set B, this time using the Merge Sort algorithm. Each member except the last was asked to make 180 comparisons,

with the final member making comparisons until the algorithm was finished. Table 5.1 shows the group assignment for the two steps in more detail.

User	Algorithm	Data	Group
1	TrueSkill	A	1
2	TrueSkill	A	1
3	TrueSkill	A	1
4	TrueSkill	A	2
5	TrueSkill	A	2
6	TrueSkill	A	2
7	TrueSkill	A	3
8	TrueSkill	A	3
9	TrueSkill	A	3

User	Algorithm	Data	Group
1	MergeSort	B	3
2	MergeSort	B	3
3	MergeSort	B	2
4	MergeSort	B	1
5	MergeSort	B	2
6	MergeSort	B	1
7	MergeSort	B	2
8	MergeSort	B	1
9	MergeSort	B	3

Table 5.1: The distribution of users across groups. Two different datasets of 100 images were used for the different steps.

5.1.2 Individual Testing Setup

When performing the individual tests, four users were recruited and allowed to order two lists by themselves. Half of the users started by ordering a list using pairwise comparisons, while the other half annotated by ordering lists of four images. The lists consisted of 50 images, and the participants making pairwise comparisons were asked to make 220 annotations, while the ones ordering shorter lists were asked to make 60. Similarly to the group tests, these numbers were based on convergence observed during simulation, as well as on how much total time we estimated that the user would have to spend to complete the task. After having finished this first round of annotations, each user annotated a second list of images using the method that was not initially utilized. The algorithm used was TrueSkill for all scenarios. The exact distribution of comparison sizes and corresponding image sets can be seen in Table 5.2.

User	Algorithm	Size	Data
10	TrueSkill	2	C
11	TrueSkill	2	D
12	TrueSkill	2	C
13	TrueSkill	2	D

User	Algorithm	Size	Data
10	TrueSkill	4	D
11	TrueSkill	4	C
12	TrueSkill	4	D
13	TrueSkill	4	C

Table 5.2: The distribution of images and how they were sorted by participants.

5.2 Results

This section will cover the results of the two initial user tests performed. First, the consistency of the outputs of the large group tests will be discussed. Thereafter, the efficiency tests of ordering a shorter list will be covered.

5.2.1 Group Testing Results

After the trials had concluded, the annotation process for each group was reconstructed using the collected data. When doing this, the distances between the three different orderings could be calculated after each comparison. This distance was more specifically achieved by using normalized Spearman's footrule to measure the distance between groups 1 and 2, groups 1 and 3, and groups 2 and 3. Figure 5.3 shows the average distance between these group orderings estimated by TrueSkill at each time step. It also shows the average distance between the final orderings produced by Merge Sort (the exact amount of comparisons required has been averaged over the three runs). The lowest average distance achieved by TrueSkill is approximately 0.151, while Merge Sort reaches a distance of about 0.233. Apart from this significant difference, it would also appear that TrueSkill converges more quickly, reaching the average distance acquired by Merge Sort in 368 comparisons as opposed to 550.

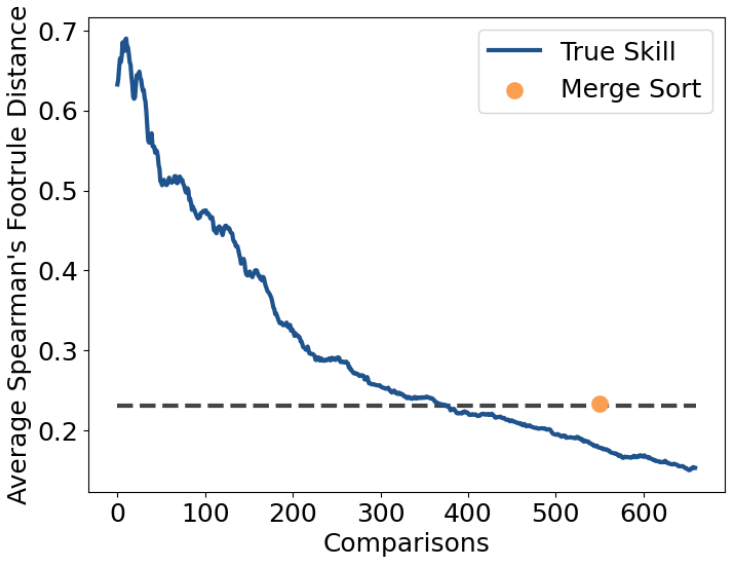


Figure 5.3: The average normalized spearman's footrule distance between the three orderings produced by each algorithm.

Apart from looking at how the orderings produced by different groups converge toward each other, the convergence of the individual orderings was also measured. Figure 5.4 shows the moving average of the root mean squared error of the estimated scores of the current iteration, compared to the one before. The vertical lines highlight changes of annotator. The interesting thing to note is that for two of the three groups, the change between annotators had a great effect on the current rate of convergence.

Lastly, when groups annotated with the TrueSkill algorithm, they had the option to state that one person was *Of Similar Age*, *Younger/Older*, or *Clearly Younger/Older*. To better understand the perceived difficulty of the task, the frequency of the use of these different options was calculated. The distribution of these difference levels can be seen in Table 5.3. This will be further discussed in Section 6.3

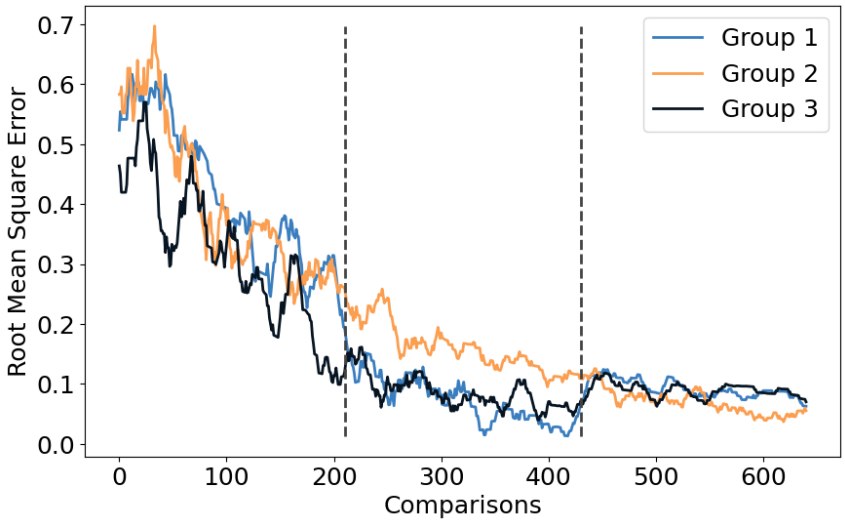


Figure 5.4: Moving average (with a window size of 20) of the RMSE of the estimated TrueSkill scores of the current iteration, compared to the one before. The dashed lines indicate a change in user annotating.

Group	Difference Level					
	Of Similar Age		Younger/Older		Clearly Younger/Older	
1	183	27.7 %	362	54.8 %	115	17.4 %
2	230	34.8 %	381	57.7 %	49	7.4 %
3	241	36.5 %	377	57.1 %	42	6.4 %
	33 %		56.6 %		10.4 %	

Table 5.3: The distribution of differences supplied by annotators when ranking the apparent age of people in images using the TrueSkill system.

5.2.2 Individual Testing Results

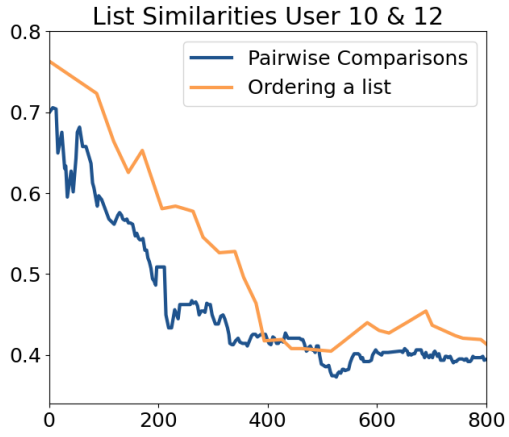
The time per annotation was recorded for each user. Table 5.4 shows the average time on a user and method level. Exactly how to compare these numbers is not obvious. Each annotation made by ordering a list of four images leads to six pairwise comparisons being made in the background. Using this measurement, each of the users made more comparisons per second using the list ordering method. However, the selection strategies are fundamentally different, meaning that the six comparisons made would potentially not have been the same six as the pairwise algorithm would have selected. Furthermore, as the annotation task itself becomes different, the quality of each comparison can vary. If one annotation task would be more difficult, then noise could increase.

To exemplify the problem of comparing the annotation speeds, we can compare the orderings of the different users at different times. As can be seen in Table 5.2, users

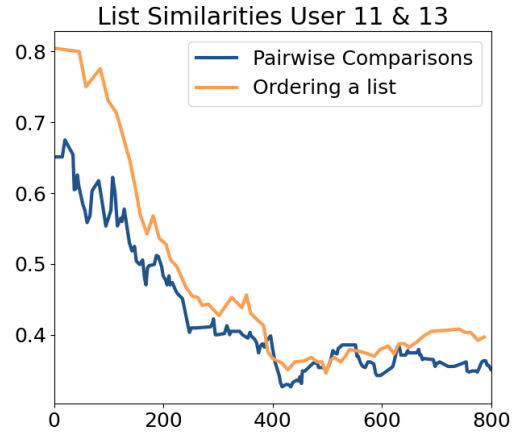
User	Average Time Per Pairwise Comparison	Average Time Per List Ordering
10	3.9 s	17.8 s
11	2.9 s	13.9 s
12	5.6 s	27.7 s
13	4.4 s	11.2 s

Table 5.4: The average time per annotation of users when making pairwise comparisons, as well as when ordering shorter lists.

10 and 12 ordered the same collections of images using the same techniques. The same is true for users 11 and 13. Consequently, the rate at which the orderings converge toward each other can be compared. Figure 5.5a shows the similarity of the orderings for users 10 and 12 when they utilized pairwise comparisons, as well as ordering shorter lists. Similarly, Figure 5.5b shows the distance between the orderings produced by users 11 and 13. The time of each annotation has been calculated by averaging the times of the two annotators. While these examples are too few to draw any conclusions (as the user's subjective opinions could differ to varying degrees), it can be seen that the cases where the annotators ordered shorter lists did not converge faster.



(a) Similarity of orderings produced by users 10 and 12 over time.



(b) Similarity of orderings produced by users 11 and 13 over time.

Figure 5.5: Similarity of orderings produced by different users on separate sets of images, where the y-axis represents the normalized Spearman's footrule distance

5.3 Conclusion

The results from the group tests indicate that TrueSkill is better at producing consistent orderings. We reason that this is due to the fact that the algorithm is able to manage disagreements and inconsistencies among annotators better, allowing it to better capture the general consensus of a group. Moreover, TrueSkill is able to

reach this better ordering in fewer comparisons than what would be required of Merge Sort.

When looking at the convergence of the scores of TrueSkill, as seen in Figure 5.4, it becomes clear that annotator preference is important. As different annotators perceive the annotation task differently, some will be quicker to state that there is no difference between two images, while others may be more inclined to try to decide upon a winner. Furthermore, as the annotation task is subjective, an annotator that is in agreement with the current ordering will make fewer updates than one that is not. For these reasons, it can be difficult to use the current rate of convergence as a well-defined finish condition for the TrueSkill algorithm. Instead, convergence on a user basis can be one factor to consider when deciding if more comparisons are required.

Lastly, it is difficult to draw any conclusions from the individual tests. While the time per comparison did decrease when ordering a shorter list, there is still a lack of understanding of the quality of these comparisons compared to the standard pairwise version of the algorithm. The consequence of this is that even though the annotator is technically able to make more comparisons in a specific set of time, the ordering may still not converge faster than the pairwise case. For this reason, we believe that, while ordering shorter lists shows some promise, further user testing is required before drawing any conclusions.

6

Evaluation of Bronchial Wall Thickening Annotation

The final step of the project was to evaluate the annotation system on the application area of interest. That is, to rank CT images of the lungs according to the severity of the bronchial wall thickening present. The goal was to take the most promising rank-based algorithm and compare it to a rating-based system. This evaluation step aimed to evaluate and address two key features, the quality of the annotation and the resource requirements. In order to evaluate these metrics, the following questions were considered:

1. Do the annotations produced by the radiologists align better when using the rank-based system compared to a rating-based one?
2. For the rank-based systems, how is the similarity of the produced labels affected as the amount of comparisons increases?

Through the previous evaluations, it has become evident that TrueSkill is the algorithm displaying the greatest potential. Not only is it noise resistant, but it also displays properties of quick convergence. For these reasons, TrueSkill will be used as the rank-based algorithm for this experiment. Moreover, as the results from Section 5.2.2 were inconclusive, it was decided that only pairwise comparisons would be used for the test. As a lack of resources does not permit us to test both solutions, it was reasoned that the standard pairwise alternative would be the safer option, as the annotation task may potentially be more difficult.

6.1 Experimental Setup

Slight adjustments were made to the interface in order to better support comparing images of CT scans. Figure 6.1 shows an example screenshot of pairwise comparison. The annotators had five options. Stating that *A & B Are Equal* corresponded to the *None* difference level, while the *A/B More Severe* and *A/B Much More Severe* options corresponded to the *Normal* and *Large* levels respectively. In order to conduct the experiment, the annotators also had to be able to provide ratings for single images. A simple image rating system was as such developed, an example image of which can be seen in Figure 6.2. For this system, the annotators were allowed to assign one of four labels to the image, stating that the bronchial wall

thickening appeared to be: *None*, *Mild*, *Moderate*, or *Severe*.

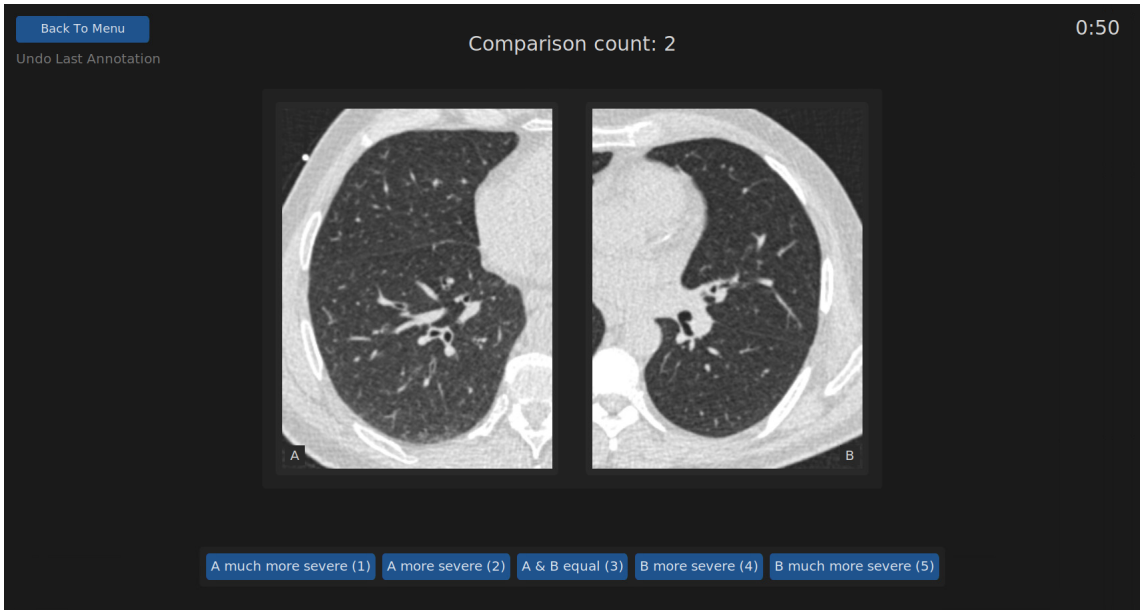


Figure 6.1: An example screenshot of the user interface when making pairwise comparisons of CT images.

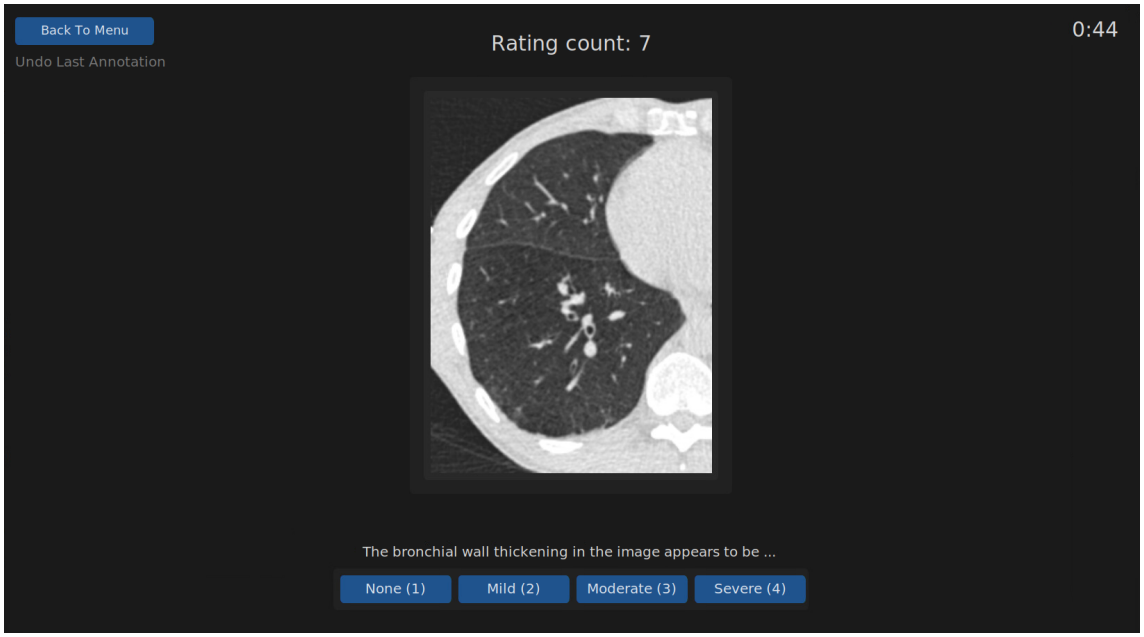


Figure 6.2: An example screenshot of the user interface when rating the degree of bronchial wall thickening in a single image.

The CT scans used for this experiment are a small subset from the SCAPIS study [14]. An initial sample of 77 subjects was selected for the experiment. For each patient, a series of slices covering the entire lungs were available. The slices were spaced at an interval of 0.6 mm, providing comprehensive coverage of the lungs.

Additionally, two previously generated segmentations were provided, offering the regions corresponding to the left and right lung respectively. These segmentations could be used to only display one lung at a time, making sure the image presented to the annotator only contains the relevant information.

Additional preprocessing of the data was necessary in collaboration with an experienced radiologist in the field. In order to obtain more diverse data, it was first necessary to delete slices that were adjacent to one another since they frequently showed similarities and overlapped bronchial wall thickening. To mitigate this, a decision was made to consider slices at regular intervals of 1 cm. This interval was approximated by selecting every 17th slice, as depicted in Figure 6.3 with black lines. Moreover, it was observed that certain regions at the top and bottom of the lungs rarely displayed visible bronchial wall thickening. Consequently, slices in these specific regions were excluded from further analysis. To ensure a diverse range of slices for evaluation, every fourth slice was retained while discarding the first and last extracted slices for each individual lung. Figure 6.3 illustrates an example of this selection process, with the remaining slices of interest indicated by the green lines for each individual.

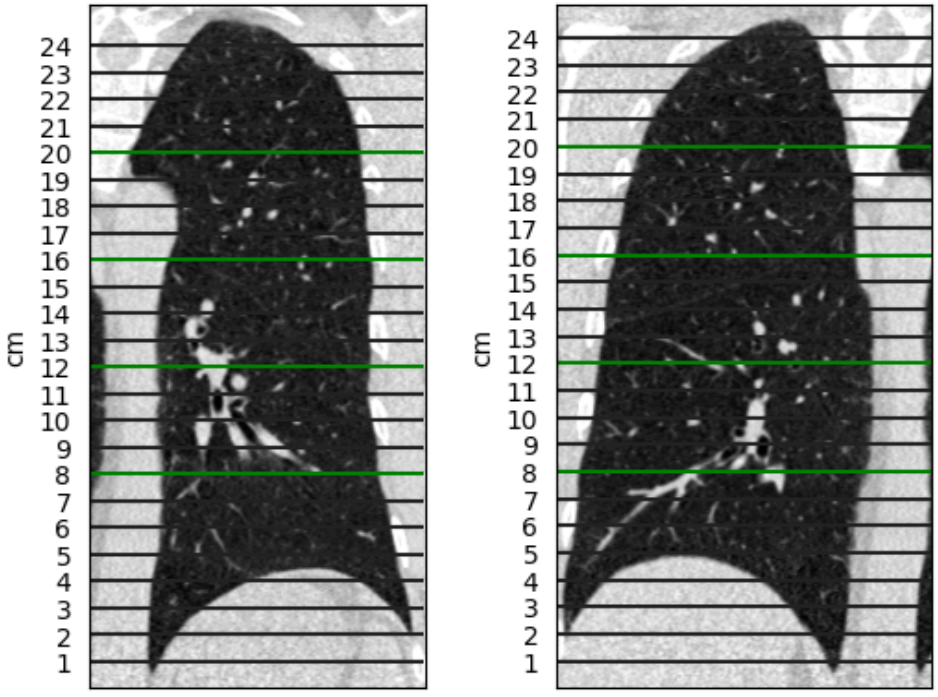


Figure 6.3: Slice selection of a CT scan. Slices highlighted in green were extracted to be annotated.

However, in order to get an interesting collection of slices for the experiment, further selection had to be made. The are a few reasons for this. Firstly, the distribution of degrees of bronchial wall thickening is very unbalanced, with most slices showing no signs of bronchial wall thickening, and very few showing signs of severe. Secondly, some slices did not have any visible bronchial walls at all, making them uninteresting

for the target experiment. To make this selection, a radiologist, independent from the annotation evaluation that was to be done, organized a sample from the previously extracted slices into the categories *Non-Representative*, *None*, *Mild*, *Moderate*, or *Severe*. We could then use this rough categorization to select the final 75 images such that all images had visible bronchial walls, and the degree of wall thickening was more reasonably distributed.

Thereafter, four experts, one radiologist and three radiology residents (in training), were recruited for the test. Each annotator was to both rate all 75 slices, as well as create an ordering of the elements by making 500 pairwise comparisons. Two participants began by rating, while the other two started by ordering the images.

6.2 Results

For the final experiment, a group of trained experts was involved in annotating CT scans of the lungs to assess the extent of visible bronchial wall thickening. The objective was to measure the level of agreement among the annotators regarding both the direct labels they assigned and the labels and rankings provided by the ranking system for the 75 ordered CT images.

6.2.1 Rating-Based Annotation Quality

Once the radiologists completed the task of annotating the CT scans and assigning ratings to indicate the amount of visible bronchial wall thickening, the collected ratings were subjected to analysis to assess the inter-annotator agreement. To measure this agreement, Krippendorff's alpha was calculated, utilizing the ordinal difference function as it was deemed to be the most appropriate metric given the ordinal nature of the labels. The resulting agreement score, approximately 0.38, was observed to be relatively low. A score in this range suggests that there are significant inter-annotator variations in the assessment of bronchial wall thickenings.

6.2.2 Rank-Based Annotation Quality

There are multiple ways that the annotator agreement can be compared using the ordered lists of images. One quite apparent method is to simply compare the ranks of the elements. When comparing the rankings of the 75 CT scans by the annotators, Krippendorff's alpha was once again employed using the ordinal difference function. Each of the 75 ranks was treated as a distinct label. This approach yielded an agreement score of approximately 0.73, which is considered acceptable for alpha.

While Krippendorff's alpha is designed to function on any span of numbers, the case can be made that it is difficult to compare the value when using 75 labels instead of four. To further evaluate the rank-based method, the agreement was also calculated after distributing the four labels used in the rating-based system across the ordered lists. The distribution used for each of the four orderings was the average distribution observed from the rating-based system. When using this system, 35 % of the cases were labeled as *None*, 43 % as *Mild*, 20 % as *Moderate*, and 2 % as *Severe*. Assigning

labels to the ordered lists based on this average distribution resulted in an alpha value of approximately 0.65.

Lastly, to gain additional understanding of the effect of the rank-based system, the four labels were once again distributed on the orderings. However, this time the distribution used was the same as the individual distribution obtained by each annotator during the rating experiment (meaning that each of the four orderings used a different distribution). Using these labels, a much-decreased agreement score of approximately 0.40 was observed. This highlights the benefits of being able to apply the same distribution, and is something which will be discussed further in Section 6.3.

When ranking, the radiologists were to classify one of the images as *Equal*, *More Severe*, or *Much More Severe*. Table 6.1 visualizes the distribution of differences provided by each annotator when making the comparisons. A classification of *Equal* was assigned to roughly 65,7 % of the comparisons. Even though *Equal* was the classification for the vast majority of comparisons, *More Severe* variances were regularly seen, found in about 32.7 % of the comparisons. Notably, the *Much More Severe* classification was found in only 1.6 % of the instances.

User	Difference Level					
	Equal		More Severe		Much More Severe	
1	327	65.4 %	165	33 %	8	1.6 %
2	409	81.8 %	88	17.6 %	3	0.6 %
3	284	56.8 %	199	39.8 %	17	3.4 %
4	294	58.8 %	202	40.4 %	4	0.8 %
	65.7 %		32.7 %		1.6 %	

Table 6.1: The distribution of differences supplied by annotators when annotating bronchial wall thickening using the rank-based system.

6.2.3 Resource Requirements

Using the saved annotations, the ordering of the images could be recreated after every comparison. Indirect labels could subsequently be distributed, which allows us to investigate how the inter-annotator agreement converges. For this analysis, it was decided to use four labels and distribute them according to the average distribution observed during rating as it makes the agreement score the most clearly comparable to the rating-based system. Figure 6.4 shows Krippendorff's alpha after every comparison. Furthermore, as the time of each annotation was recorded, the amount of resources that were spent on the two different methods can be quantified. Figure 6.5 shows how alpha varies over time, where the time has been calculated by averaging the time it took the four annotators to make a specific comparison (e.g.

if it took on average 10 minutes for the annotators to make 100 comparisons, then the alpha obtained after 100 comparisons is placed at 10 minutes in the graph).

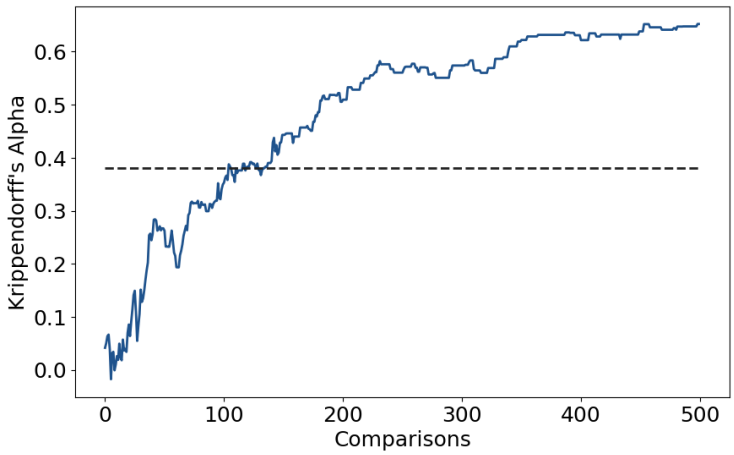


Figure 6.4: The inter-annotator agreement when distributing the four labels provided during rating according to the average observed distribution after every comparison. The dashed line represents the agreement achieved by the rating-based approach.

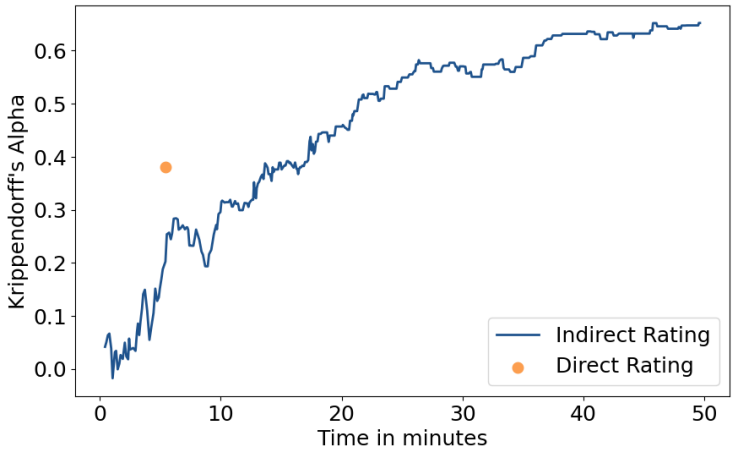


Figure 6.5: The inter-annotator agreement over the averaged time.

6.3 Conclusion

When letting radiologists annotate using both methods, it has become clear that the rank-based method can produce better-aligned labels. The primary reason for this is the ability to use the same distribution of labels on all orderings. While it can be argued that this is an unfair comparison, we reason that it is instead one of the advantages of rank-based systems. In this case, one of the difficult and subjective assessments radiologists had to make was where to draw the line between two different degrees of bronchial wall thickening (e.g. decide where the images go

from mild to moderate). In some sense, the indirect rating avoids this problem by instead focusing on creating an internal ordering of the data points. The implication is that if one was to label a set of images, the choice of annotator will not have as large of an impact on the end result if they instead use the rank-based version and the distribution is predetermined.

Another interesting thing to note is the apparent difficulty of the task. As can be seen in Table 6.1, the majority of presented comparisons were deemed to be equal. This is in clear contrast to the distribution of difference levels observed during the initial user tests, as seen in Table 5.3. There are multiple possible explanations for this. Firstly, an uneven distribution could mean that there are in fact a lot of images displaying a similar degree of bronchial wall thickening, implying that there is no large difference between elements. Secondly, the subjective assessment could also simply be very difficult, making it hard to state which of the two images has more visible bronchial wall thickening. The interesting thing to note is that regardless of this, an agreement score of 0.73 was achieved when comparing the ranks produced. This implies that while the task may have been difficult, there are clear similarities in the orderings produced by the different annotators.

Finally, it is clear that the rank-based method is more resource-demanding. As the trade-off between annotation quality and resources is of interest, it is important to note that it took on average 50 minutes to make 500 comparisons, as opposed to the meager 5 minutes spent on average directly rating the images. It should, however, be noted that while 500 comparisons were made to achieve the alpha values which are discussed, Figure 6.4 shows that it only took about 100 comparisons before the rank-based system started to outperform the rating-based one. This is important as 100 pairwise comparisons is a small sample compared to the 2775 possible, implying that the TrueSkill algorithm is able to make effective use of the resources.

7

Conclusion

This thesis has produced and justified a novel rank-based annotation method through gradual testing and evaluation. The following sections will discuss the implications of the results of this project. The advantages of this system, as opposed to existing systems, will be highlighted, while the current limitations and areas which required further testing will also be brought up.

7.1 Discussion

The efficiency of any rank-based system will rely on the sorting algorithm used. Throughout all stages of evaluation, TrueSkill has shown to be a high-performing algorithm in multiple regards. Not only is it able to produce more consistent outputs compared to Merge Sort, as shown in Section 5.2, but it also converges toward these more consistent orderings in fewer comparisons than Merge Sort requires. This combination of quick convergence and noise resistance makes it a good candidate for any rank-based system. While Borda Count and Hamming LUCB would most likely be able to produce equally consistent orderings given enough annotations, the simulation covered in Section 4 indicated that the number of comparisons this would require would in all likelihood be infeasible for the target application.

In the case of annotation of bronchial wall thickening, this thesis has shown that a rank-based system can palpably increase the quality of labels through higher reliability. However, exactly which labels should be generated using the ordered lists is a question that remains. While the thesis has shown that the rank-based system increased annotator reliability when using the four labels provided by the rating-based system, this number was decided upon somewhat arbitrarily. The reason four labels are used in the rating-based system is to make the task simple enough for the annotator to perform. In reality, the degree of bronchial wall thickening visible in images is most likely better represented as a continuous value. This highlights another advantage of the rank-based system in that, not only does it have the potential to increase the quality of labels through higher reliability, but also through offering more information by increasing the granularity of the annotations. Apart from being able to choose any amount of labels up to the size of the list, one can also imagine for example using the underlying representations of the TrueSkill algorithm and letting the label of each image be its estimated score.

One of the most prominent limitations of the developed system is the lack of ter-

mination criteria. As opposed to Merge Sort or Hamming LUCB, TrueSkill does not have an indicator of when sorting is completed. While the system has been empirically shown to have good convergence properties, there is a lack of theoretical guarantees. For this reason, our current suggestion is to use the heuristic provided in [40] and use an initial comparison amount of about $n \log(n)$, where n is the size of the data that is to be sorted. We propose that while making the comparisons, the current rate of convergence can be monitored, as discussed in Section 5.3, and the remaining number of comparisons can either be decreased or increased accordingly.

While this project has made clear the benefits of a rank-based system, it has also highlighted the increase in resource demands. Although the improvement in quality is large, one should take into account the vast increase in time required to perform these comparisons, as shown in Figure 6.5. The silver lining is that these tests indicate that, for this specific task, the amount of time required to achieve higher reliability than that of the rating-based system was only about three times longer, and not ten times as the entire rank-based annotation process. Still, the usefulness of a rank-based system such as this will depend on the target application. Before gathering labels, users will have to decide if the increase in quality is worth the added resource requirements. If the subjective assessment is easy enough, perhaps a rating-based system would be preferred.

Furthermore, the comparison that has been made between a rank-based and rating-based system has been on an individual level, meaning that radiologists have either rated or ranked entire lists by themselves. In reality, it would most likely be common to have groups of radiologists either rate or order the data together. While TrueSkill was chosen due to its ability to accommodate the opinions of multiple annotators, it has yet to be compared to a system that aggregates multiple ratings, such as the simple majority vote. As the aggregation of multiple ratings is performed in order to account for differing assessments, it would be interesting to compare the systems in a group setting, but this was unfortunately outside the scope of the thesis given the limited amount of radiologist annotations.

Lastly, in Section 5 the efficiency of ranking by ordering shorter lists was evaluated. Unfortunately, while there was an increase in underlying comparisons made per second, there was no clear increase in convergence speed. Although a larger study would be required to measure the actual difference in efficiency between this method and the classical pairwise approach, it should also be considered that the results could vary depending on the application. That is, changes in the difficulty of the annotation task may make one option favorable. This makes it difficult to draw any general conclusions.

7.2 Future Work

This thesis has had its limitations, some as a result of time constraints and others due to a lack of resources. However, the results of this thesis offer a number of prospective paths for further study and advancement.

One significant area for future work, which was not explored in this thesis, is the

application of the results on a supervised learning algorithm. It would be intriguing to investigate the potential application of neural networks to analyze the outputs generated by the developed annotation system. This could involve utilizing the neural network to predict labels, ranks, or even the underlying scores associated with the annotated data. As a result, using this rank-based annotation approach may make it possible to gather enough labeled data to train a supervised learning algorithm and eventually remove the dependence on manual labeling.

Currently, the version of TrueSkill where the annotator is presented with a smaller list shows no advantage compared to pairwise comparisons. While it is intuitive to pair the two items with the most overlap, this approach may not be the most representative when multiple data points are involved. In such cases, the pair with the highest overlap may not provide additional valuable information, whereas there could be other pairs that would offer a greater advantage if compared. Therefore, it would be beneficial to enhance the selection strategy aiming for a more informed and effective approach.

An additional challenge would be to investigate criteria for determining when the annotation process should be considered complete. One problem with the technique, as previously mentioned, is that it has no clear termination criteria, i.e. it is unclear how to tell when the data is sufficiently sorted. If the label distribution is known or can be calculated, it could potentially be utilized for the finishing condition, indicating that there have been enough comparisons to accurately depict the desired label distribution, similar to Hamming LUCB.

Another area of interest involves implementing a hybrid method that initially uses a rating-based system and later switches to a rank-based algorithm. This strategy has various potential advantages. Firstly, upon initialization of the TrueSkill algorithm, the estimated score of each element could be set according to the ratings that element has received, creating a rough initial ordering and bootstrapping the ranking process. Secondly, initial ratings could offer insight into the distribution of elements. This information could potentially be used to decide how labels should be distributed across the final ordering or to create some form of finishing condition for the algorithm.

Finally, it would be interesting to see how the system would perform in a setting where a group of annotators is to rank a larger dataset together. As has been mentioned, while the system is developed to accommodate multiple annotators, this has not been compared to the rating-based approach. Furthermore, it would be interesting to explore if the most efficient way would be to allow users to annotate using the same list and algorithm instance, or if it would be beneficial to have them sort the lists individually and later aggregate them.

Bibliography

[1] D. Bhatt, C. Patel, H. Talsania, *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10202470. [Online]. Available: https://www.mdpi.com/2079-9292/10/20/2470.

[2] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of subjective and objective quality assessment of video," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, 2010. DOI: 10.1109/TIP.2010.2042111. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7344627.

[3] G. N. Yannakakis and H. P. Martínez, "Grounding truth via ordinal annotation," in *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2015, pp. 574–580. DOI: 10.1109/ACII.2015.7344627. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7344627.

[4] G. N. Yannakakis and H. P. Martnez, "Ratings are overrated!" *Frontiers in ICT*, vol. 2, Jul. 2015. DOI: 10.3389/fict.2015.00013. [Online]. Available: https://doi.org/10.3389/fict.2015.00013.

[5] M. Yang, G. Yin, Y. Du, and Z. Wei, "Pair comparison based progressive subjective quality ranking for underwater images," *Signal Processing: Image Communication*, vol. 99, p. 116 444, Nov. 2021, ISSN: 09235965. DOI: 10.1016/j.image.2021.116444. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596521002125.

[6] I. Jang, G. Danley, K. Chang, and J. Kalpathy-Cramer, "Decreasing annotation burden of pairwise comparisons with human-in-the-loop sorting: Application in medical image artifact rating," Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.04823.

[7] C. Raherison and P.-O. Girodet, "Epidemiology of copd," *European Respiratory Review*, vol. 18, no. 114, pp. 213–221, 2009. DOI: 10.1183/09059180.00003609. eprint: https://err.ersjournals.com/content/18/114/213.full.pdf. [Online]. Available: https://err.ersjournals.com/content/18/114/213.

[8] W. H. Organization, *Chronic obstructive pulmonary disease (copd)*, Mar. 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/chronic-obstructive-pulmonary-disease-(copd)#:~:text=Chronic%5C%20obstructive%5C%20pulmonary%5C%20disease%5C%20(COPD)%5C%20is%5C%20the%5C%20third%5C%20leading%5C%20cause,%5C%

2Dincome%5C%20countries%5C%20(LMIC).&text=Tobacco%5C%20smoking%5C%20accounts%5C%20for%5C%20over,cases%5C%20in%5C%20high%5C%2Dincome%5C%20countries. (visited on 05/16/2023).

[9] V. Koblizek, B. Novotna, Z. Zbozinkova, and K. Hejduk, "Diagnosing COPD: Advances in training and practice - a systematic review," *Advances in Medical Education and Practice*, p. 219, Apr. 2016. DOI: 10.2147/amep.s76976. [Online]. Available: https://doi.org/10.2147/amep.s76976.

[10] J. Vestbo, S. S. Hurd, A. G. Agust, *et al.*, "Global strategy for the diagnosis, management, and prevention of chronic obstructive pulmonary disease," *American Journal of Respiratory and Critical Care Medicine*, vol. 187, no. 4, pp. 347–365, 2013, PMID: 22878278. DOI: 10.1164/rccm.201204-0596PP. eprint: https://doi.org/10.1164/rccm.201204-0596PP. [Online]. Available: https://doi.org/10.1164/rccm.201204-0596PP.

[11] D. M. Hansell, A. A. Bankier, H. MacMahon, T. C. McLoud, N. L. Müller, and J. Remy, "Fleischner society: Glossary of terms for thoracic imaging," *Radiology*, vol. 246, no. 3, pp. 697–722, 2008, PMID: 18195376. DOI: 10.1148/radiol.2462070712. eprint: https://doi.org/10.1148/radiol.2462070712. [Online]. Available: https://doi.org/10.1148/radiol.2462070712.

[12] M. K. Han, E. A. Kazerooni, D. A. Lynch, *et al.*, "Chronic obstructive pulmonary disease exacerbations in the COPDGene study: Associated radiologic phenotypes," en, *Radiology*, vol. 261, no. 1, pp. 274–282, Oct. 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3184233/ (visited on 05/18/2023).

[13] D. A. Lynch, C. M. Moore, C. Wilson, *et al.*, "CT-based visual classification of emphysema: Association with mortality in the COPDGene study," *Radiology*, vol. 288, no. 3, pp. 859–866, Sep. 2018. DOI: 10.1148/radiol.2018172294. [Online]. Available: https://doi.org/10.1148/radiol.2018172294.

[14] Hjärt-Lungfonden, *Scapis*, Dec. 2022. [Online]. Available: https://www.hjart-lungfonden.se/forskning/scapis/.

[15] G. Bergström, G. Berglund, A. Blomberg, *et al.*, "The swedish CArdioPulmonary BioImage study: Objectives and design," en, *J. Intern. Med.*, vol. 278, no. 6, pp. 645–659, Dec. 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4744991/ (visited on 05/18/2023).

[16] X. Xie, A. E. Dijkstra, J. M. Vonk, M. Oudkerk, R. Vliegenthart, and H. J. M. Groen, "Chronic respiratory symptoms associated with airway wall thickening measured by thin-slice low-dose CT," en, *AJR Am. J. Roentgenol.*, vol. 203, no. 4, W383–90, Oct. 2014. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/25247967/ (visited on 05/18/2023).

[17] H. O. Coxson, "Quantitative computed tomography assessment of airway wall dimensions: Current status and potential applications for phenotyping chronic obstructive pulmonary disease," *Proceedings of the American Thoracic Society*, vol. 5, no. 9, pp. 940–945, Dec. 2008. DOI: 10.1513/pats.200806-057qc. [Online]. Available: https://doi.org/10.1513/pats.200806-057qc.

[18] H. O. Coxson, B. Quiney, D. D. Sin, *et al.*, "Airway wall thickness assessed using computed tomography and optical coherence tomography," *American Journal of Respiratory and Critical Care Medicine*, vol. 177, no. 11, pp. 1201–

1206, Jun. 2008. DOI: 10.1164/rccm.200712-1776oc. [Online]. Available: https://doi.org/10.1164/rccm.200712-1776oc.

[19] S. S. Skiena, "The algorithm design manual," en, in 2nd ed. Guildford, England: Springer, Apr. 2009.

[20] B. Kumar, *Time complexity*, Apr. 7, 2007. [Online]. Available: https://dev.to/bks1242/time-complexity-5c3 (visited on 05/03/2023).

[21] M. Braverman and E. Mossel, "Sorting from noisy information," Oct. 2009. arXiv: 0910.1191 [cs.DS]. [Online]. Available: https://arxiv.org/abs/0910.1191 (visited on 05/03/2023).

[22] R. Heckel, M. Simchowitz, K. Ramchandran, and M. Wainwright, "Approximate ranking from pairwise comparisons," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, A. Storkey and F. Perez-Cruz, Eds., ser. Proceedings of Machine Learning Research, vol. 84, PMLR, Sep. 2018, pp. 1057–1066. [Online]. Available: https://proceedings.mlr.press/v84/heckel18a.html.

[23] L. Busse, M. H. Chehreghani, and J. M. Buhmann, *Approximate Sorting*. 2013, pp. 142–152. DOI: 10.1007/978-3-642-40602-7_15. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-40602-7%5C_15.

[24] C. M., "Pattern recognition and machine learning," en, in (Information Science and Statistics), 1st ed., Information Science and Statistics. New York, NY: Springer, Aug. 2006, pp. 21–24, 68. [Online]. Available: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf (visited on 05/04/2023).

[25] A. A. Johnson, M. Q. Ott, and Mine Dogucu, *Bayes rules!* (Chapman & Hall/CRC Texts in Statistical Science). London, England: CRC Press, Mar. 2022. [Online]. Available: https://www.bayesrulesbook.com/.

[26] I. E. of the Social Sciences, *Distribution, normal*, May 2023. [Online]. Available: https://www.encyclopedia.com/science-and-technology/mathematics/mathematics/normal-distribution.

[27] R. Herbrich, T. Minka, and T. Graepel, "Trueskill™: A bayesian skill rating system," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19, MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2006/file/f44ee263952e65b3610b8ba51229d1f9-Paper.pdf.

[28] A. E. Elo, "The proposed uscf rating system, its development, theory, and applications," *Chess Life*, vol. 22, no. 8, pp. 242–247, Aug. 1967. [Online]. Available: http://uscf1-nyc1.aodhosting.com/CL-AND-CR-ALL/CL-ALL/1967/1967_08.pdf#page=26 (visited on 05/24/2023).

[29] M. J. W. Nihar B. Shah, "Simple, robust and optimal ranking from pairwise comparisons," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7246–7283, 2017.

[30] A. Slivkins, "Introduction to multi-armed bandits," *CoRR*, vol. abs/1904.07272, 2019. arXiv: 1904.07272. [Online]. Available: http://arxiv.org/abs/1904.07272.

[31] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaa, and L. E. Meester, *A modern introduction to probability and statistics* (Springer Texts in Statistics), en, 1st ed. London, England: Springer, May 2005. [Online]. Available: https://link.springer.com/book/10.1007/1-84628-168-7 (visited on 05/25/2023).

[32] L. d. Bruijn, *Inter-annotator agreement (iaa)*, Jul. 2020. [Online]. Available: https://towardsdatascience.com/inter-annotator-agreement-2f46c6d37bf3.

[33] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology* (Content Analysis: An Introduction to Its Methodology). Sage, 2004, ISBN: 9780761915454. [Online]. Available: https://books.google.se/books?id=q657o3M3C8cC.

[34] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[35] M. J. Warrens, "Five ways to look at cohen's kappa," *Journal of Psychology & Psychotherapy*, vol. 5, no. 4, p. 1, 2015.

[36] J. Vikgren, M. Khalil, K. Cederlund, *et al.*, "Visual and quantitative evaluation of emphysema: A case-control study of 1111 participants in the pilot swedish cardiopulmonary bioimage study (scapis)," *Academic Radiology*, vol. 27, no. 5, pp. 636–643, 2020, ISSN: 1076-6332. DOI: https://doi.org/10.1016/j.acra.2019.06.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1076633219303253.

[37] M. Lidén, O. Hjelmgren, J. Vikgren, and P. Thunberg, "Multi-readermulti-split annotation of emphysema in computed tomography," *Journal of Digital Imaging*, vol. 33, no. 5, pp. 1185–1193, Aug. 2020. DOI: 10.1007/s10278-020-00378-2. [Online]. Available: https://doi.org/10.1007/s10278-020-00378-2.

[38] K. Krippendorff, "Computing krippendorff's alpha-reliability," 2011. [Online]. Available: https://repository.upenn.edu/asc_papers/43/ (visited on 05/08/2023).

[39] P. Diaconis and R. Graham, "Spearman's footrule as a measure of disarray," *Journal of the royal statistical society series b-methodological*, vol. 39, pp. 262–268, 1977.

[40] J. Hees, B. Adrian, R. Biedert, T. Roth-Berghofer, and A. Dengel, "Tssort: Probabilistic noise resistant sorting," Jun. 2016.

[41] M. Frenescu, *Age prediction*, May 5, 2023. [Online]. Available: https://www.kaggle.com/datasets/mariafrenti/age-prediction.

A

Appendix 1 - Graphical User Inteface of Application

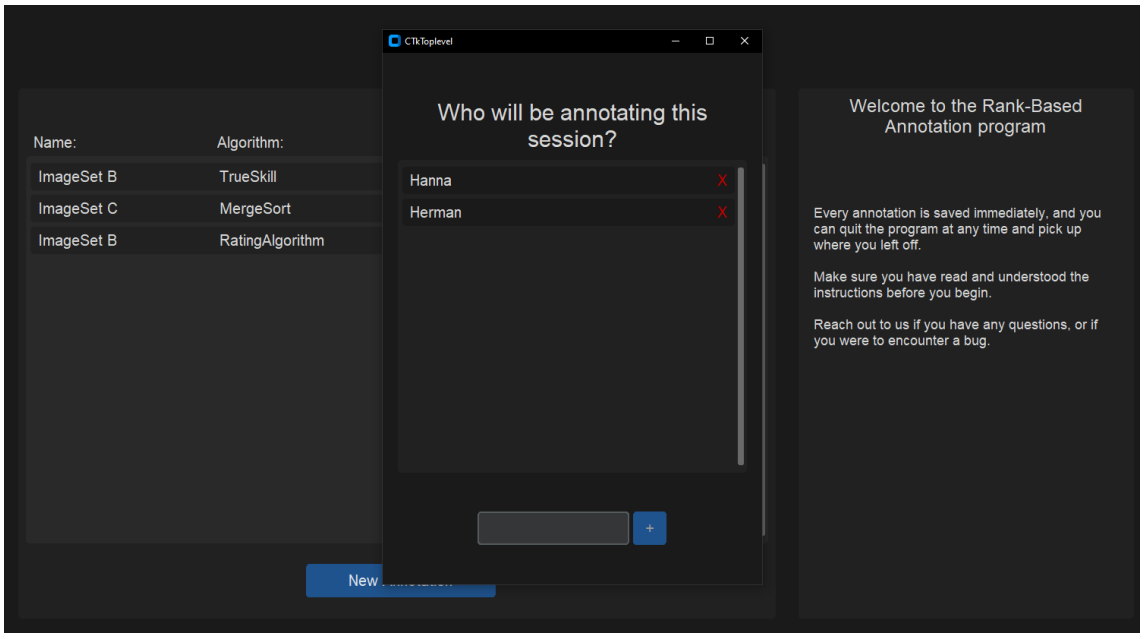


Figure A.1: The initial view when introduced to the application. A mandatory pop-up urging the user to provide who will be annotating the current session.

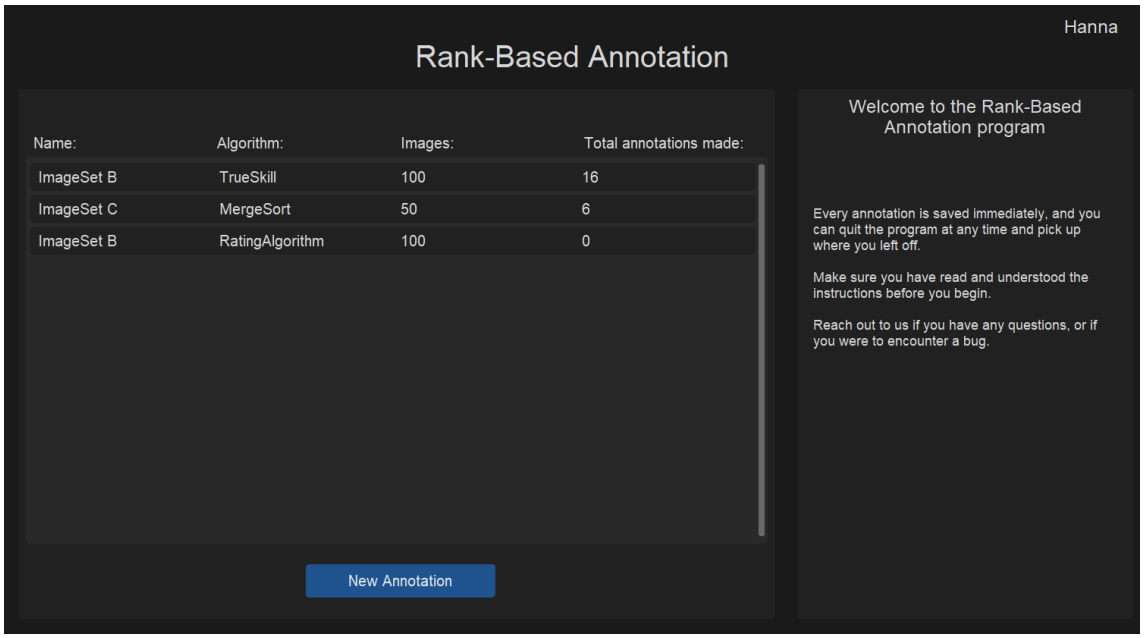


Figure A.2: The main page view essentially provides some overview of the application. On the left side of the screen all available algorithms as well as the active algorithm, the number of images in the set, and a total number of annotations made. In the top-right corner the currently annotating user is presented and below are some instructions for the user.

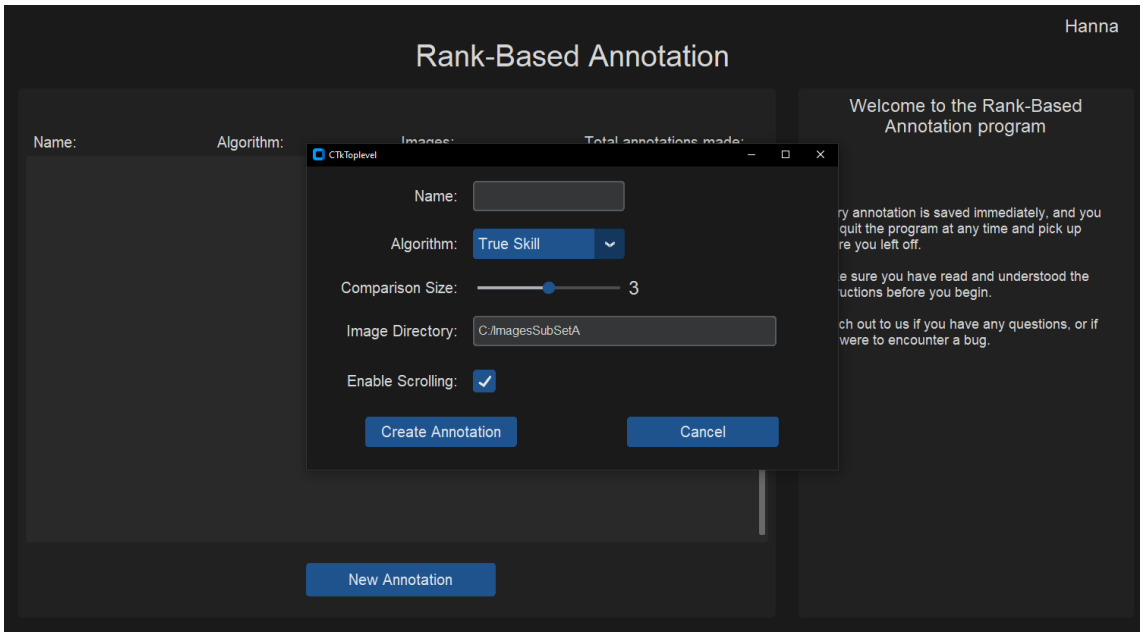


Figure A.3: The fields the user is offered when creating a new ranking instance. When Scrolling is enabled, the user can load nifti files containing multiple slices and scroll between them.

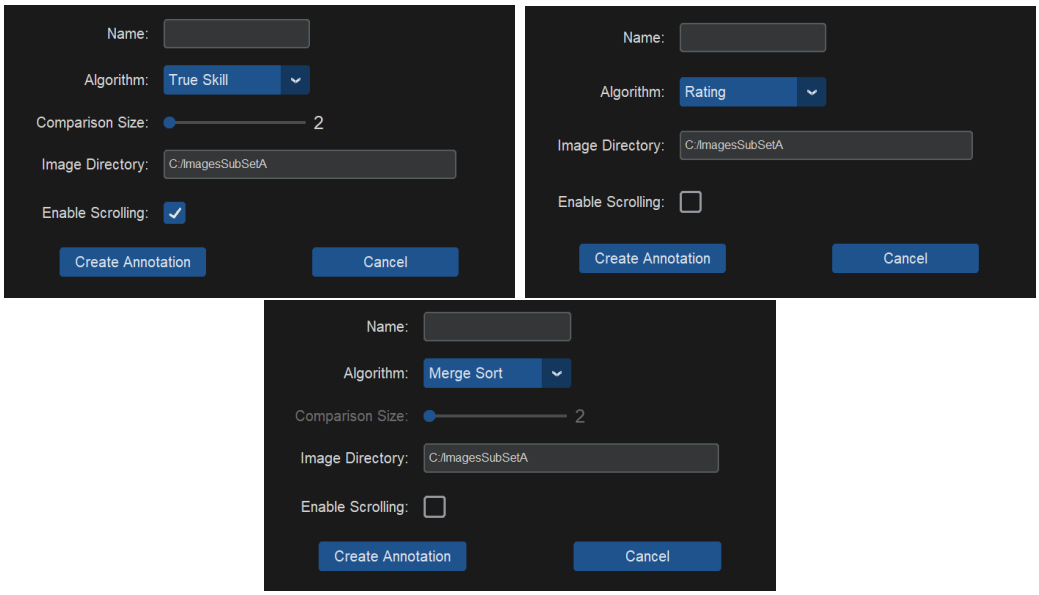


Figure A.4: There are three algorithms to choose from, the rank-based algorithms TrueSkill and Merge Sort, as well as the option to rate images. The comparison size field differs depending on the selected algorithm.

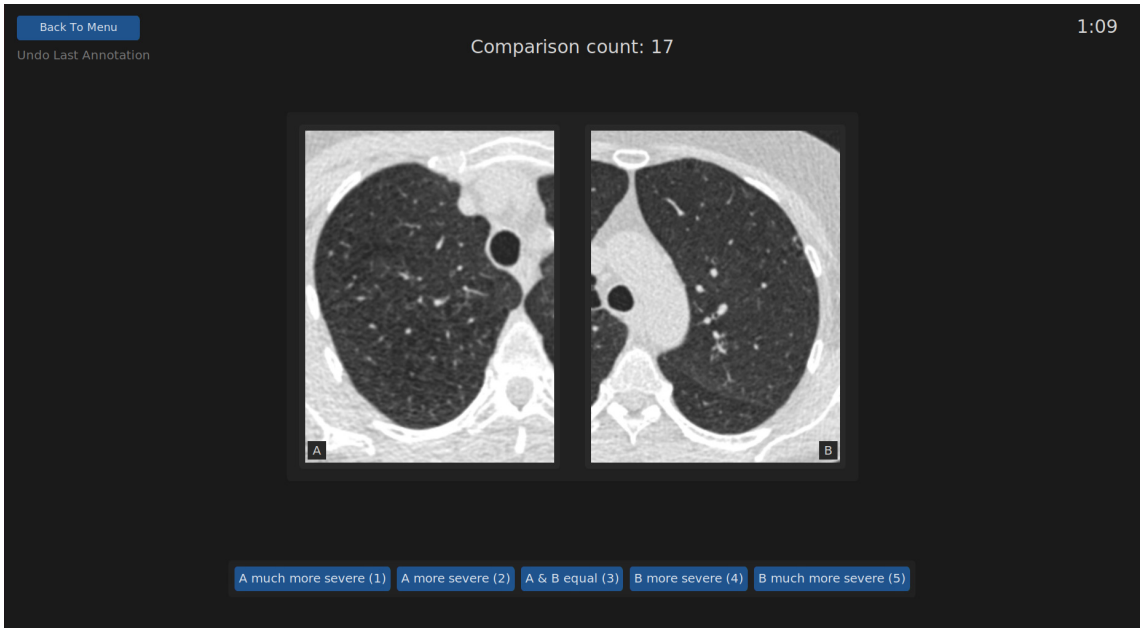


Figure A.5: The view when annotating pairwise comparisons. The user provides order and difference level by using one of the five buttons. Here it is also visible to the user that it is possible to undo the last annotation.

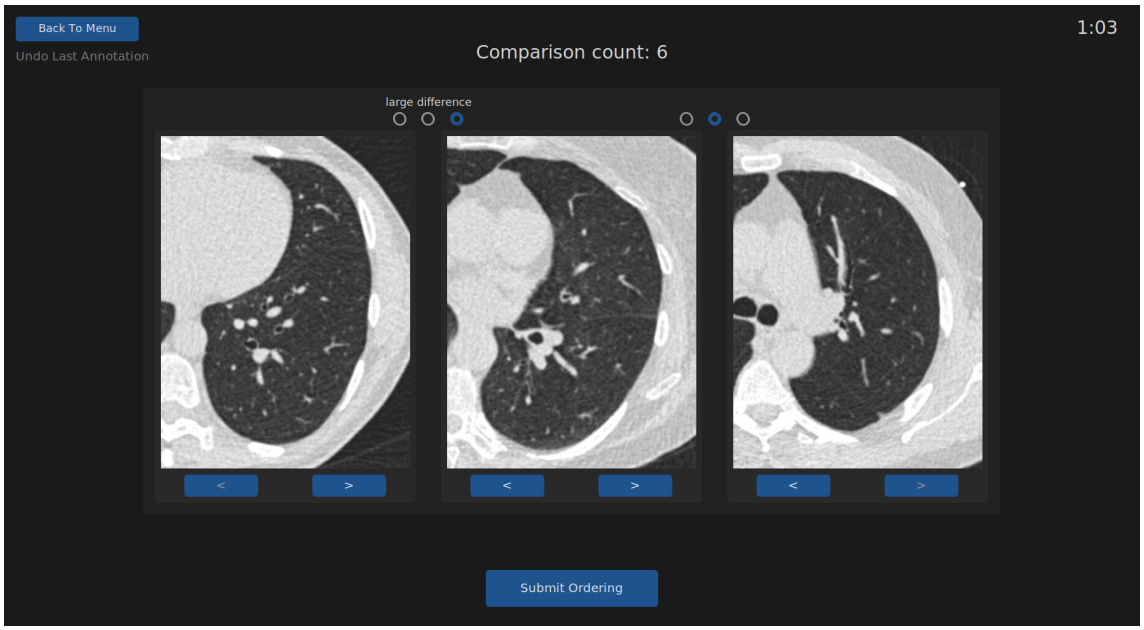


Figure A.6: The interface when ordering a list of three images. The annotator orders the images from the least to the most amount of visible bronchial wall thickening, left to right. Difference levels are assigned by using the radio buttons between two images.