Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Special Track on AI for Good

Creating Dynamic Checklists via Bayesian Case-Based Reasoning: Towards Decent Working Conditions for All

Eirik Lund Flogard[1,2], **Ole Jakob Mengshoel**[2] and **Kerstin Bach**[2]
[1]Norwegian Labour Inspection Authority
[2]Norwegian University of Science and Technology
eirik.flogard@arbeidstilsynet.no, {ole.j.mengshoel, kerstin.bach}@ntnu.no

Abstract

Every year there are 1.9 million deaths worldwide attributed to occupational health and safety risk factors. To address poor working conditions and fulfill UN's SDG 8, "protect labour rights and promote safe working environments for all workers", governmental agencies conduct labour inspections, using checklists to survey individual organisations for working environment violations. Recent research highlights the benefits of using machine learning for creating checklists. However, the current methods only create static checklists and do not adapt them to new information that surfaces during use. In contrast, we propose a new method called Context-aware Bayesian Case-Based Reasoning (CBCBR) that creates dynamic checklists. These checklists are continuously adapted as the inspections progress, based on how they are answered. Our evaluations show that CBCBR's dynamic checklists outperform static checklists created via the current state-of-the-art methods, increasing the expected number of working environment violations found in the labour inspections.

1 Introduction

Checklists are extensively used in high-stakes decision making, such as in surgery or food inspections [Jelacic *et al.*, 2020; Ho *et al.*, 2018]. They are also used by government agencies in labour inspections, to survey individual organisations for non-compliance to working environment regulations [Dahl and Søberg, 2013; Karanikas and Hasan, 2022]. Such inspections are important to globally achieve UN's SDG 8, "protect labour rights and promote safe working environments for all workers", specifically SDG 8.8. Despite the inspection efforts, there are still 1.9 million registered deaths world-wide each year that are attributed to occupational health and safety risk factors [World Health Organization, 2021]. Our goal in this paper is to attack this problem by introducing dynamic checklists created via machine learning.

A conceptual view of a labour inspection checklist is shown in Figure 1. Here, a checklist consists of a subset of K of N items where each item corresponds to a specific regulation. Each item has a yes or no answer which indicates the inspected organisation's compliance to the corresponding regulation. Any non-compliant item found at an inspection is cited individually in a report which is sent to the inspected organisation, with an order to rectify the violations.

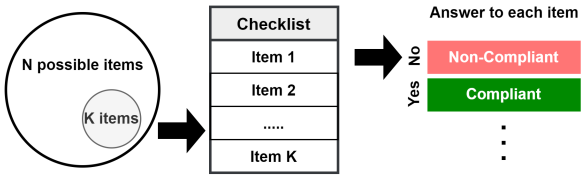

Figure 1: A conceptual view of a checklist for labour inspections. For a given inspection, a checklist ideally contains a subset of the K items most likely to be non-compliant, out of N possible items.

Each inspected organisation may be subjected to several hundred different regulations, which vary according to its size, location and industry. Prioritizing the correct regulations for inspections, in terms of risks to workers' health and safety, is very difficult. Assessing compliance to regulations is also time-consuming, so a checklist should ideally contain a small subset K of N possible items that are most likely to be found non-compliant at the inspection. Creating or using such checklists is difficult, partly because they are situation dependent [Ho *et al.*, 2018; Catchpole and Russ, 2015].

In order to create optimized checklists efficiently, Flogard et al. [2021] propose to use machine learning (ML) to construct checklists and a new ML problem called the Checklist construction problem (CCP). They introduce a method called BCBR, which constructs checklists for labour inspections. BCBR uses Bayesian inference (BI) to construct features for cases used in case-based reasoning (CBR) [Aamodt and Plaza, 1994]. The input to BCBR is an organisation targeted for a labour inspection. BCBR then selects cases that contain a set of the K out of N possible unique items that are most likely to be found non-compliant at the given organisation. The output is an optimized checklist that consists of the selected K items.

Motivation. A shortcoming with BCBR is that it only creates static checklists, which can be inaccurate in many real-world situations where the context changes over time [Grigg, 2015]. For instance, new information obtained during an inspection could suggest that inquiries should be made into regulations that are not represented by the K items currently on the checklist on Figure 1. A possible solution for this

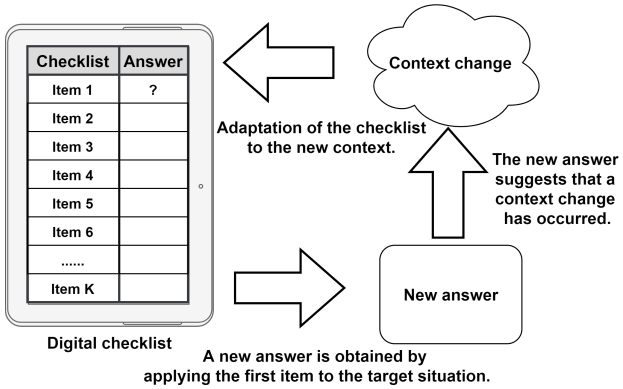

Figure 2: A dynamic checklist is a digital checklist that can adapt to changes in environment or context during use. Such adaptations to the checklists can increase their relevance and efficiency in aiding users for their intended tasks.

is to dynamically adapt the checklists during use. Figure 2 shows how a dynamic checklist can be adapted to a situation, inferred from how a user answers the checklist. This idea has been proposed and studied in clinical surgery trials [Kulp *et al.*, 2021], where the current approaches for adapting checklists are based on process [Christov *et al.*, 2016] or rule-based models [De Bie *et al.*, 2017]. However, these models need to be built and maintained manually by domain experts, which is infeasible for creating the ideal high-detailed models needed for many complex tasks. Kulp et al. [2021] also discuss this limitation and highlight the need for a technical solution that can adapt checklists without relying on manually created models.

Scientific Contributions. There are two main contributions in this work. The first contribution is to establish ML as a means to create dynamic checklists. As far as we know, this is a difficult problem where ML has not been used before. A challenge here is to develop a method that is accurate and fast enough to make real-time dynamic adaptations of checklists.

The second contribution is a new method called Context-aware Bayesian Case-Based Reasoning (CBCBR), which is an extension of BCBR. The novelty, compared to BCBR, is a context-aware naive Bayesian inference model that enables dynamic adaptations to the constructed checklists during use. This means that CBCBR both creates new checklists and adapts them. The adaptations are done as recommendations of new items for the checklists. The recommendations are based on answers of the checklists, which can be considered a part of the temporal context of the situations where the checklists are used [Haruna *et al.*, 2017]. CBCBR is also a fully transparent, online model that enables real-time creation and adaptation of checklists, which is crucial for their usability in real-world situations. Online learning models are known for their effectiveness in dynamic or non-stationary situations [Huleihel *et al.*, 2021; Idrees *et al.*, 2020], such as labour inspections.

Social Impact. Our work focuses on improving checklists used for labour inspections, which will have a direct impact on the promotion of decent work (see SDG 8). We show that CBCBR's dynamic checklists can improve task completion and increase the number of violations (non-compliance) found during inspections, which in turn will increase levels of compliance with working environment and labour rights and also reduce injuries (SDG indicator 8.8.1 and 8.8.2). More effective checklists could also reduce social dumping (SDG 8.5), human trafficking and forced labour (SDG 8.7).

2 Related Work

Checklist Construction. Until recently, checklists have been created manually by domain experts via approaches like the Delphi method [Morgan *et al.*, 2007; Amaya *et al.*, 2017]. However, creating checklists manually is difficult and time consuming [Hasson *et al.*, 2000]. Instead of relying on human experts, Flogard et al. [2021] propose BCBR for constructing static checklists. In experiments, they show that labour inspection checklists constructed by BCBR outperform human domain experts and other ML methods. Zhang et al. [2021] propose an ML method for constructing checklists for medical diagnosis, assuming that a checklist is a binary M-of-N decision problem. They use an integer program to find an optimal checklist of N items that most accurately predict a diagnose, given that at least M items are checked. However, their method only creates static checklists and is also not usable for creating labour inspection checklists, as they are not binary M-of-N decision problems. An approach for generating checklist items for construction site inspections is proposed by Cai et al. [2020]. The approach is based on Natural Language Processing and generates new items from regulation document texts. Checklists are generated via SQL queries, by matching generated items with keywords specified by a user. The checklists are described as dynamic, but only in the sense that they are presented digitally, enabling users to manually customize them.

Context-Aware Recommender Systems. Research shows that dynamically elevating items on checklists can be effective for adapting them to context changes [De Bie *et al.*, 2017]. CBCBR's recommendations can also be seen as a way to elevate context-relevant items and is inspired by context-aware recommender systems (CARS) and contextual modelling [Adomavicius and Tuzhilin, 2011].

Case-Based Reasoning. In recommender systems, CBR is used to address explainability and the long tail problem, which are known issues in collaborative filtering [Alshammari *et al.*, 2017; Jorro-Aragoneses *et al.*, 2020]. Reasoning in CBR (and CBCBR) is based on retrieving the best past cases to solve a given problem [Aamodt and Plaza, 1994]. To improve reasoning, Bayesian methods are used to infer missing case features or information in CBR systems [Nikpour and Aamodt, 2021; Kim *et al.*, 2014]. For the same reason, CBCBR's cases are augmented with probability estimates to improve case retrieval. CBCBR also learns from answered checklists items, by using them for checklist adaptations or retaining them to create future checklists.

Summary. To our knowledge, there is no previous work where ML is used to dynamically adapt checklists. In our work, we show that checklists can be significantly improved by doing so, using answers to past items in a checklist to

adapt future items in the same checklist. The approach is also generic and can likely be used in many other domains, since most checklists have some kind of answers recorded in them. Thus, our work can potentially be seen as a starting point for future ML research into dynamic context-aware checklists.

3 Definitions

Data Set and Cases. A data set $\mathcal{D}$ for variables $\mathbf{Z} = \{E, X, L\}$ is a tuple $(\mathbf{d}_1, ..., \mathbf{d}_N)$ where a case $\mathbf{d}_j \in \mathcal{D}$ is an instantiation of $\mathbf{Z}$ [Darwiche, 2009]. A case can be defined as a tuple $\mathbf{d} = (e, x, l)$ where e denotes a single item of a checklist, x is the target organisation for the item and $l \in \{0, 1\}$ denotes the binary answer to the item. A case in the data set can be viewed as a past experience where an item e has been applied to x to obtain the answer l. Any e, x and l are instantiations of the variables E, X and L, respectively.

Organisation. Every case in the data set contains a target organisation description x, that consists of multiple sub-features. These sub-features are all categorical and describe the organisation's location and industry [Flogard *et al.*, 2021]. For brevity we treat x and X as categorical in our work.

Item. Each case in the data set contains an answerable item e, used to survey the organisation x. There are N unique items in the data set, so an item e is a categorical value that may appear in multiple cases. Thus, E is also categorical.

Checklist. The item in each case of the data set belongs to a checklist $\mathbf{y}$, which is a solution applied to the organisation x. A checklist consists of a set of items such that $\mathbf{y} = (e_1 \in \mathbf{d}_1, e_2 \in \mathbf{d}_2, ..., e_n \in \mathbf{d}_n)$. An item can only occur once per checklist, such that $e_i \neq e_j$ for every $e_i \wedge e_j \in \mathbf{y}$. Thus, $\mathbf{y}$ consists of items from multiple cases.

Answer. The label $l \in \{0, 1\}$ of each case $\mathbf{d}_j \in \mathcal{D}$ is the recorded answer from applying the item e to the organisation x. A positive answer (non-compliance) is observed if $l = 1$.

The Checklist Construction Problem. Given a candidate target organisation x^{cnd}, a model $\mathcal{M}$ needs to select K out of N unique items $(e_1, e_2, ..., e_K)$ for an initial candidate checklist $\mathbf{y}^{cnd}$. Each item $e_i \in \mathbf{y}^{cnd}$ needs to be selected so that it maximizes the probability for observing the answer $l_i = 1$ when applied to the targeted x^{cnd}.

Dynamic Adaptation of Checklists. Let $(l_1, ..., l_i)$ be the observed answers to the items $(e_1, ..., e_i) \in \mathbf{y}^{cnd}$ where $i \leq K$. Given the target organisation x^{cnd} and the answered items, adapt the checklist so that the posterior probability for observing a positive answer to any unanswered items on the changed checklist is maximized. The adaptation could be done by adding or removing items to the checklist in many ways. In this work adaptation is done by recommending any additional M of N items $\hat{e} \notin y^{cnd}$ that have higher posterior probability for observing $\hat{l} = 1$ (given $(l_1, ..., l_i)$), compared to any existing items on y^{cnd}. The M recommendations are appended to the existing K items of $\mathbf{y}^{cnd}$.

4 CBCBR Framework

The purpose of CBCBR is to create a dynamic checklist for any given x^{cnd}, by retrieving past cases with items used in similar organisations, and with high estimated probabilities for non-compliance. Answering the checklist may change the estimates, triggering dynamic updates to the checklist.

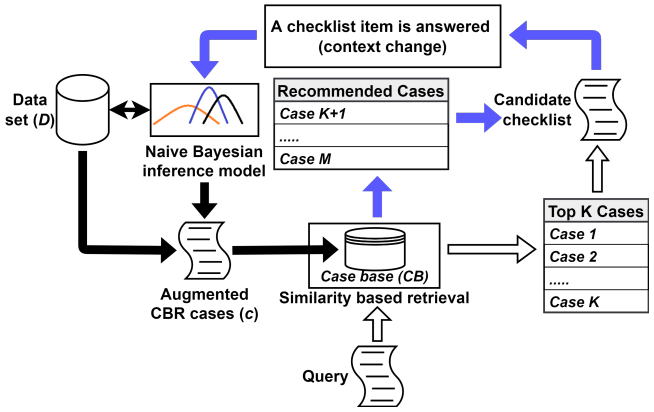

Figure 3: An overview of CBCBR. The black arrows show how the case base is created or updated. The white arrows show the creation of a candidate checklist. The checklist is dynamically updated via the blue (and black) arrows, starting from the candidate checklist.

An overview of CBCBR is shown in Figure 3 and consists of the following steps: (1) a naive Bayesian inference (NBI) model generates probability estimates (θ^{be}) for answers based on empirical distributions of the data set $\mathcal{D}$. (2) A case base $\mathcal{CB}$ of augmented CBR cases $\mathbf{c}_j$ is created by adding θ^{be} as feature to the instances $\mathbf{d}_j \in \mathcal{D}$. (3) Similarity based retrieval is used to retrieve K cases from $\mathcal{CB}$, given a query $\mathbf{q}$ that contains the target organisation x^{cnd} and a fixed target value for θ^{be} in the cases. The retrieved cases contain the items for the initial candidate checklist $\mathbf{y}^{cnd}$, as illustrated by the white arrows on Figure 3. (4) The novel recommendation part for adapting $\mathbf{y}^{cnd}$ is shown by the blue arrows in Figure 3. First, an item on $\mathbf{y}^{cnd}$ is answered, which prompts CBCBR to refresh the $\mathcal{CB}$ with updated posterior θ^{be} estimates (via NBI). CBCBR then retrieves any new cases from $\mathcal{CB}$ with sufficiently increased θ^{be}, containing M recommended items which are appended to $\mathbf{y}^{cnd}$.

Further details for how θ^{be} is calculated by the NBI model in steps 1 and 4 are covered in Section 4.1 and 4.2, respectively. The other details for step 2-4 are found in Section 4.3.

4.1 Naive Bayesian Inference for Checklist Construction

NBI is based on using empirical distributions of the data set $\mathcal{D}$ to estimate the probability for the event $L = 1|x, e$, expressed as the mean of a Beta distribution [Darwiche, 2009]:

$$\theta^{be}(L = 1|x, e) = \frac{\mathcal{D}\#(L = 1 \wedge X = x \wedge E = e) + \psi_{L=1|x,e}}{\sum_{l=0}^{1} \mathcal{D}\#(L = l \wedge X = x \wedge E = e) + \psi_{L=l|x,e}} \quad (1)$$

where $L = l$, $X = x$ and $E = e$ denote the event where the outcomes l, x and e are observed. Both $(\mathcal{D}\#(L = 1 \wedge X = x \wedge E = e)$ and $(\mathcal{D}\#(L = 0 \wedge X = x \wedge E = e)$ are parameters for the Beta distribution, and $\psi_{L=l|x,e}$ denotes the prior parameters. Equation 1 is used to calculate the proba-

bility estimates needed for selecting the optimal items when constructing the initial checklist $\mathbf{y}^{cnd}$.

4.2 Naive Bayesian Inference for Item Recommendations

After answering an item e_i on the checklist $\mathbf{y}^{cnd}$, the obtained answer l_i holds evidence that can be used to update the posterior belief in any unobserved answer $\hat{l}$ for a candidate item $\hat{e}$.[1] This assumption can be used to update the probability estimates from Equation 1, by estimating new additional Beta distribution parameters with the following equation:

$$p(\hat{l}, x, \hat{e}, e_i, l_i) = \mathcal{D}\#(L = \hat{l} \wedge X = x \wedge E \in \{\hat{e} : (\hat{e}, e_i) \in \mathbf{y} \wedge (x, e_i, l_i) \in \mathcal{D}\}). \quad (2)$$

The parameters are made by counting cases in $\mathcal{D}$. Each case in $\mathcal{D}$ is counted if it contains the given x, $\hat{l} \in \{0, 1\}$, $\hat{e}$ and if there exist at least one other case $\mathbf{d} = (x, e_i, l_i)$ and a past checklist $\mathbf{y}$ in $\mathcal{D}$ such that both e_i and $\hat{e}$ exists in that checklist. The posterior estimates can now be updated by inserting the parameters above into the Beta distribution from Equation 1:

$$\theta^{be}(L = 1|x, \hat{e}, \mathbf{y}^{cnd}) = \frac{\beta_{L=1|x,\hat{e}} + \sum_{(l_i, e_i \in \mathbf{y}^{cnd})} p(1, x, \hat{e}, e_i, l_i)}{\sum_{\hat{l}=0}^{1} \beta_{L=\hat{l}|x,\hat{e}} + \sum_{(l_i, e_i \in \mathbf{y}^{cnd})} p(\hat{l}, x, \hat{e}, e_i, l_i)} \quad (3)$$

where $\beta_{L=\hat{l}|x,\hat{e}} = \mathcal{D}\#(L = \hat{l} \wedge X = x \wedge E = \hat{e}) + \psi_{L=\hat{l}|x,\hat{e}}$. The θ^{be} estimate in Equation 3 is calculated by summing the parameters for every $e_i \in \mathbf{y}^{cnd}$ that has an observed answer l_i. It should be noted that the equation above "naively" assumes that all applied items e_i are mutually independent of each other given x. This is done to decrease the amount of θ^{be} estimates based on low or zero case counts [Darwiche, 2009]. Equation 3 is an important technical contribution of this paper, since it enables online adaptations of checklists.

4.3 Case Base and Retrieval of Checklist Items

This section defines the details for the augmented CBR cases, case base and similarity based retrieval from Figure 3.

Creating Augmented CBR Cases and Case Base. Algorithm 1 shows the creation of a case base $\mathcal{CB}$ with augmented cases $\mathbf{c}$. The κ feature is included to adjust for the case counts of the probability estimates when retrieving cases [Flogard *et al.*, 2021]. The algorithm iterates through each case $\mathbf{d}_j \in \mathcal{D}$ and creates κ and θ^{be} estimates for the case via Equation 1. Both θ^{be} and κ are conditioned on the case features x and e_j and added as features to each $\mathbf{d}_j$, to create cases $\mathbf{c}$ for $\mathcal{CB}$.

Case Retrieval for the Candidate Checklist. To retrieve items e_i for the candidate checklist $\mathbf{y}^{cnd}$, a query case $\mathbf{q}$ and a similarity function are used. The query consists of the target organisation x^{cnd} and the desired values for both the probability estimates and the case count features. The similarity function assigns a score $Sim(\cdot, \cdot) \in [0, 1]$ to every pair $(\mathbf{q}, \mathbf{c}_j \in \mathcal{CB})$. The function is the same linear weight similarity function used by Flogard et al. [2021]. The function is applied to every $\mathbf{c}_j \in \mathcal{CB}$ and a set of unique items $(e_1, ..., e_K)$ is then retrieved from the K cases with the highest similarity score, to create an initial checklist $\mathbf{y}^{cnd}$ with K items.

[1] In this setting, $\hat{e}$ is a potential candidate for recommendation.

Algorithm 1 Creating or updating $\mathcal{CB}$ with cases $\mathbf{c}$ containing new θ^{be} estimates.

Input: $\mathcal{D}$, $\mathbf{y}^{cnd}$; //$\mathbf{y}^{cnd}$ is only included when updating $\mathcal{CB}$ with new posterior θ^{be} estimates after answering items.
Output: $\mathcal{CB} \leftarrow ()$; //Initialize $\mathcal{CB}$.
for each $\mathbf{d}_j \in \mathcal{D}$ **do**
  //Both x_j and e_j are found in the current case $\mathbf{d}_j$.
  if $\mathbf{y}^{cnd}$ is empty **then**
    $\theta \leftarrow \theta^{be}(L = 1|x_j, e_j)$; //Eq. 1
  else
    $\theta \leftarrow \theta^{be}(L = 1|x_j, e_j, \mathbf{y}^{cnd})$; //Eq. 3
  end if
  $\kappa \leftarrow \mathcal{D}\#(L = 1 \wedge X = x_j \wedge E = e_j)$;
  $\mathbf{c} \leftarrow Join(\mathbf{d}_j, \theta, \kappa)$; //merge $\mathbf{d}_j$, θ, κ to a single case.
  $\mathcal{CB} \leftarrow Insert(\mathbf{c})$; //Add the new case $\mathbf{c}$ to $\mathcal{CB}$.
end for
return $\mathcal{CB}$; //The $\mathcal{CB}$ is now ready for any case retrievals.

Case Base Update and Retrieval for Recommendations. Algorithm 1 also shows how the case base is refreshed after answering items on the checklist. This procedure is done after an item $e_i \in \mathbf{y}^{cnd}$ has been applied to x^{cnd}. First the existing $\mathcal{CB}$ is cleared for cases. Then updated probability estimates are calculated for each case $\mathbf{d}_j \in \mathcal{D}$ via Eq. 3, conditioned on x_j, e_j and every $e_i \in \mathbf{y}^{cnd}$ with an obtained l_i. The newly calculated θ and κ features are concatenated to each $\mathbf{d}_j$ to create the updated cases $\mathbf{c}$ for $\mathcal{CB}$.

The updated cases are then retrieved for recommendation. First, the function $Sim(\mathbf{q}, \mathbf{c}_j)$ is again applied to $\mathbf{q}$ and every $\mathbf{c}_j \in \mathcal{CB}$. A filter is then applied to remove all cases with similarity scores less than any of the K cases that were selected for the initial $\mathbf{y}^{cnd}$. Duplicated items are also removed so that M cases with unique items remain. The size of M is usually very small (between 0 and 4) and is not known or fixed, but simply depends on how many eligible cases that remain after applying the filter. If there are $M > 0$ remaining cases, the items from these cases are recommended for the checklist.

5 Demonstration and Assessment of CBCBR

Data Set. We use the Checklist data set introduced by Flogard et al. [2021], which has 1 111 502 entries $\mathbf{d}_j$. The entries constitute 63 634 inspections conducted with 369 different unique checklists, which in turn contain $N = 1\,947$ unique items in total. The checklists cover a wide range of industries and the items on each checklist typically have an inspection topic that specifically relates to a few intended target industries. Mengshoel et al. [2021] also use a similar data set.

Query. We use the configuration from Section 6.1. A recommendation is done after answering each item. The query is $\mathbf{q} = (x^{cnd}, \theta, \kappa)$ and contains the desired values of the retrieved cases, where θ is set to 100% and κ to 70. The target organisation for the inspection x^{cnd} is a hotel in Oslo.

Constructed Checklist and Item Recommendations. Table 1 shows the constructed checklist $\mathbf{y}^{cnd}$ for the target hotel x^{cnd} where two of the items have been answered. The answers were randomly selected. The initial size of the checklist is $K = 5$ items, however two items have been added as

Checklist	
Does the employer ensure that the employees work within the framework in Chapter 10 of the Working Environment Act?	**Yes / No**
Has the employer ensured that the employment agreements are in line with the minimum requirements set out in the Working Environment Act §14-6?	**Yes / No**
Does the company have routines for how violence, threats and other unfortunate burdens as a result of contact with others are to be prevented, reported, handled and followed up?	**Yes / No**
Has the employer implemented the necessary measures and/or prepared a plan describing measures to remove or reduce hazards and problems at work?	**Yes / No**
Has the employer mapped the dangers and problems the employees may be exposed to in the company and on this basis assessed the risk of injury to or danger to the employees?	**Yes / No**
Added Items	
Does the company pay at least a 40% supplement to the salary for overtime work?	**Yes / No**
Does the employer have control over the working hours of the employees within the framework of the Working Environment Act, Chapter 10?	**Yes / No**

Table 1: The state of a dynamic checklist $\mathbf{y}^{cnd}$ with $K = 5$ initial items, generated for a hotel inspection in Oslo. Two items on the checklist have been answered (as indicated with blue circles). Two items have also been dynamically added to the checklist (at the bottom), based on the answered items.

recommendations based on the answered items. Further items may be added after more answers are given. The added items increases the checklist length, but allow the inspector to focus more on highly relevant risks in the inspected organisation.

Qualitative Assessment. The items in Table 1 cover a variety of topics such as working hours, overtime payment and violence. These are all common risks for the hotel industry, especially in a large city such as Oslo. The added items shown in the table are also relevant and closely related to the two answered items on the top of the checklist. Thus, the recommendations align well with the findings at the inspection. CBCBR also performed well when we, with domain experts, assessed a variety of other inputs (x^{cnd}) and lengths (K).

6 Experiment

In this section we conduct an experiment to measure the performance of CBCBR against BCBR and other baselines.

6.1 Experimental Setup

CBCBR and BCBR Configuration. Both CBCBR and BCBR generate the exact same initial checklists, but BCBR does not perform any additional recommendations or dynamic adaptations. The two methods use the same configuration as Flogard et al. [2021]. The NBI models for CBCBR and BCBR use fixed priors $\psi_{L=1|x,e} = 1$ and $\psi_{L=0|x,e} = 5$. For every query $\mathbf{q}$, the target matching value for θ^{be} is set to 100%. The target value for κ is set to 70. The NBI models are created and updated via MSSQL17 queries and stored as Python data frames. The similarity based retrieval is implemented via MyCBR [Bach *et al.*, 2019] for both methods. For performance reasons, CBCBR recommendations are made each time 5 items on the checklist have been answered.

Generating Checklist Answers for Predictive CBCBR Recommendations. To generate predictive recommendations for CBCBR in the experiment, we introduce a simulator. The approach is similar to user-behaviour simulations in recommender systems [Zhang and Balog, 2020], as the goal of the simulator is to perform a realistic walk-through of an initial checklist to obtain an answer l_i for each item $e_i \in \mathbf{y}^{cnd}$. These answers are then used to generate realistic item recommendations in the experiment below. This is done as follows:

1. For a given target organisation x^{cnd}, CBCBR (model $\mathcal{M}$) first constructs an initial checklist $\mathbf{y}^{cnd}$ with items e_i. Each e_i is retrieved from a case $\mathbf{c}_i \in \mathcal{CB}$ and has an unobserved answer l_i.
2. For each $e_i \in \mathbf{y}^{cnd}$, a value for l_i needs to be generated by drawing from the set $\{0, 1\}$.
3. To do so, the parameters $\mathbf{b}_i$ of a Bernoulli distribution $\mathcal{B}$ for l_i are estimated empirically from $\mathcal{CB}$ using the following expression: $\mathbf{b}_i = \frac{\mathcal{CB}\#(L=1 \wedge X=x^{cnd} \wedge E=e_i)}{\mathcal{CB}\#(X=x^{cnd} \wedge E=e_i)}$.
4. If e_i is an item that has been recommended earlier in the simulation, the recommendation context is taken into account when calculating $\mathbf{b}_i$. This is done by applying Eq. 3 to $\mathcal{CB}$ (instead of $\mathcal{D}$): $\mathbf{b}_i = \theta^{be}(L = 1|x^{cnd}, e_i, \mathbf{y}^{cnd})$.
5. The value $l_i \in \{0, 1\}$ is now drawn from a Bernoulli distribution, given by $\mathcal{B}(\mathbf{b}_i, 1 - \mathbf{b}_i)$. If 5 values have been generated since the last recommendation, then proceed to the next step. Otherwise, go to Step 2.
6. After 5 l_i-values are generated, CBCBR is updated with the new values in order to perform a recommendation. Any recommended items are then automatically appended to $\mathbf{y}^{cnd}$. After the recommendation, go to Step 2 if there are any unanswered items left in $\mathbf{y}^{cnd}$.

After the procedure above has been applied, the extended checklist is ready for evaluation.

Baselines for the Experiment. We use these following baselines to generate static checklists: Multi layer perceptron (NN), Random forest (RF), Naive Bayes inference (NBI), Decision tree (DT) and Logistic regression (LR).

Each baseline ($\mathcal{M}$) above generates a checklist of K items as follows: (1) $\mathcal{M}$ is first trained on a training set $\mathcal{D}_T$ of $\mathcal{D}$, using l from each instance $\mathbf{d} = (e, x, l)$ as the target label. Each $\mathcal{M}$ is trained with the goal of correctly classifying the value of l, given x and e. (2) Given a validation set $\mathcal{D}_{CB}$ of $\mathcal{D}$ and an input x^{cnd}, $\mathcal{M}$ generates a prediction score ($[0, 1]$) for every N possible items. Predictions are only generated for items (e) with at least one corresponding instance $(e, x, l) \in \mathcal{D}_{CB}$ where $x = x^{cnd}$. (3) The K items with the highest scores are selected for the candidate checklist $\mathbf{y}^{cnd}$.

We also use the original, domain expert created checklists (ECL) from the Checklist data set as baseline.

Baselines Configuration. The NBI-baseline is calculated using Equation 1 and is implemented via MSSQL17. The other baselines are implemented via Sklearn, using the default configurations for most of them. For the NN and RF baselines we used GridsearchCV (Sklearn) for hyperparameter tuning. The optimal configuration for NN was 20 layers, logistic activation function and L2 penalty of 0.0001. The optimal configuration for RF was 50 estimators, bootstrapping sample size of 50% and minimum sample size of 10 for node splits.

Environment. The experiment is conducted on a fully upgraded Dell Precision 5560 with Intel i9 11950h and 64GB RAM, in a Python environment using Scikit-learn (Sklearn).

Method	Acc	Prec (gt)	Prec	Rec	Avg	*Time (sec)*
ECL	0.365	0.177	0.176	0.592	0.328	-
DT	0.494	0.229	0.258	0.610	0.398	3 641
LR	0.513	0.247	0.272	0.664	0.424	205
RF	0.518	0.253	0.284	0.676	0.433	4 222
NBI	0.520	0.254	0.286	0.686	0.437	**7.2**
NN (MLP)	0.521	0.256	0.290	0.688	0.439	14 280
BCBR	**0.675**	0.313	0.479	0.764	0.558	8.7
CBCBR	**0.675**	**0.322**	**0.497**	**0.808**	**0.576**	8.6 (4.5)

Table 2: The mean accuracy *Acc*, ground truth precision *Prec (gt)*, precision *Prec* and recall *Rec* for the content of the checklists, created via the various methods in the table. The *Avg* column shows the average scores of the four preceding columns. The last column shows the training times in seconds for each method.

6.2 Evaluation of CBCBR's Dynamic Checklists

The goal of this experiment is to evaluate the performance of CBCBR against BCBR and other baselines.

Method and Data. The experiment is done on the data set $\mathcal{D}$ from Section 5. An 8-fold cross-validation is used where $\mathcal{D}$ is partitioned into training folds ($\mathcal{D}_T$) and validation folds ($\mathcal{D}_{\mathcal{CB}}$). $\mathcal{D}_T$ is used to calculate any probability estimates needed to select or recommend items for a checklist $\mathbf{y}^{cnd}$. $\mathcal{D}_{\mathcal{CB}}$ is used for the case base ($\mathcal{CB}$) and for performance evaluation. Each validation fold has 138 938 instances that constitutes 7 954 inspections. Each inspection contains a target organisation x that is used as input (x^{cnd}) to each checklist construction model $\mathcal{M}$. Each model $\mathcal{M}$ then generates a candidate checklist $\mathbf{y}^{cnd}$ for each given x as described in Section 4 and 6.1. The target length of $\mathbf{y}^{cnd}$ is set to $K = 15$.[2]

Statistics for all checklists created by each $\mathcal{M}$ are calculated as follows: For each generated $\mathbf{y}^{cnd}$, all items $e_i \in \mathbf{y}^{cnd}$ are considered as having predicted positive answers $l_i = 1$. The rest of the N possible items where $e \notin \mathbf{y}^{cnd}$ are considered as having predicted negative. The number of true positive (TP)/false positive (FP) answers and true negative (TN)/false negative (FN) answers of $\mathbf{y}^{cnd}$ are estimated from empirical distributions of $\mathcal{D}_{\mathcal{CB}}$, as described by Flogard et al. [2021]. This is done because every $\mathbf{y}^{cnd}$ is a new checklist, so most items $e_i \in \mathbf{y}^{cnd}$ do not have an observed ground truth answer l_i. The estimates are used to calculate accuracy, precision and recall statistics for each $\mathbf{y}^{cnd}$ via: $Acc_{\mathbf{y}^{cnd}} = \frac{TP+TN}{TP+FP+TN+FN}$, $Prec_{\mathbf{y}^{cnd}} = \frac{TP}{TP+FP}$ and $Rec_{\mathbf{y}^{cnd}} = \frac{TP}{TP+FN}$. The final mean *Acc*, *Prec* and *Rec* for each validation fold are found by averaging each statistic over all 7954 generated $\mathbf{y}^{cnd}$. We also included *Prec (gt)*, which is mean precision calculated by only using the subset of items on the checklists where ground truth answers are available. The statistics are comparable to mean average (MA), used to evaluate recommender systems [Natani and Watanabe, 2021].

Results and Discussion. The results are shown in Table 2. CBCBR has the highest *Avg* score and has higher scores than BCBR on most of the statistics. This is impressive as CBCBR's checklists are longer and should generally have lower *Acc*, *Prec* and *Prec (gt)*, since only 17.7% of the instances in the data set have positive labels ($l = 1$) and longer checklists means more predicted positives. NN has the third highest *Avg* score, only marginally higher than NBI. It may be possible to improve the results for NN with more advanced parameter tuning methods, but this will unlikely be enough to match BCBR or CBCBR's performance. The domain experts' checklists (ECL) have the lowest score.

On average, the checklists constructed by CBCBR contain 9.13 violations (true positives) against 7.19 for BCBR. CBCBR's checklists also contain 18.1 items against 15.0 for BCBR. This means that CBCBR recommends 3.1 items on average, because CBCBR creates the same initial checklists as BCBR. There are 1.94 violations found among the 3.1 recommended items, which corresponds to a precision score of 0.63. Thus, CBCBR's recommendations have much higher precision than static checklists created by BCBR or other baselines in Table 2. Overall, the results suggest that using CBCBR's dynamic checklists for labour inspections will increase efficiency and the amount of violations to working environment regulations found at the inspected organisations.

Time-wise, the slowest method is NN with an average training time of 14 280 seconds, excluding hyper parameter optimization. The fastest method is NBI with 7.2 seconds training time. CBCBR is nearly as fast, as the combined time for model training and generating an initial checklist is 8.6 seconds. This is much faster than LR and DT, which are known for their low training times. It also takes 4.5 seconds to update CBCBR and recommend new items after answering a checklist. However, the cross-validation took more than 20 days to complete because CBCBR must be trained and updated individually for each validation case. Still, the short individual training and update times for each checklist means that CBCBR can be used as an online model and we believe that further speed-ups can be achieved via parallelization.

7 Conclusion

In this work we show that dynamic answer-based adaptations to checklists can significantly increase the number of violations found in labour inspections. We introduce CBCBR for real-time generation and adaption of checklists, which could be employed by labour inspection agencies to increase the efficiency of their inspections. This will likely increase national and global levels of compliance with labour rights and reduce injuries (see SDG indicator 8.8.1 and 8.8.2). We are currently testing the real-world performance of CBCBR in a trial.

This paper only explores one of many possible approaches for dynamically adapting checklists. Future work could investigate other designs as well. An example is to dynamically build checklists from ground-up, starting with one item and adding K more, one-by-one as answers are obtained. Another direction for future work is to explore approaches for creating checklist items, where Cai et al. [2020] could be a starting point. It could also be interesting to look into other ML methods for dynamically adapting checklists, such as RNNs. CBCBR's dynamic checklists could also be tested in other relevant tasks, such as food inspections, aviation or surgeries.

[2]For CBCBR, each generated $\mathbf{y}^{cnd}$ also includes recommended items. The initial length of CBCBR's checklists are $K = 15$, before adding any extra items. The code for the experiment is published at https://github.com/ntnu-ai-lab/cbcbr.

References

[Aamodt and Plaza, 1994] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

[Adomavicius and Tuzhilin, 2011] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. 2011.

[Alshammari *et al.*, 2017] Gharbi Alshammari, Jose L Jorro-Aragoneses, Stelios Kapetanakis, Miltos Petridis, Juan A Recio-García, and Belén Díaz-Agudo. A hybrid cbr approach for the long tail problem in recommender systems. In *ICCBR*, pages 35–45. Springer, 2017.

[Amaya *et al.*, 2017] Marly Ryoko Amaya, Danieli Parreira da Silva Stalisz da Paixão, Leila Maria Mansano Sarquis, and Elaine Drehmer de Almeida Cruz. Construction and content validation of checklist for patient safety in emergency care. *Revista gaucha de enfermagem*, 37, 2017.

[Bach *et al.*, 2019] Kerstin Bach, Bjørn Magnus Mathisen, and Amar Jaiswal. Demonstrating the mycbr rest api. In *ICCBR Workshops*, pages 144–155, 2019.

[Cai *et al.*, 2020] Hubo Cai, JungHo Jeon, Xin Xu, Yuxi Zhang, Liu Yang, et al. Automating the generation of construction checklists. Technical report, Purdue University. Joint Transportation Research Program, 2020.

[Catchpole and Russ, 2015] Ken Catchpole and Stephanie Russ. The problem with checklists. *BMJ quality & safety*, 24(9):545–549, 2015.

[Christov *et al.*, 2016] Stefan C Christov, Heather M Conboy, Nancy Famigletti, George S Avrunin, Lori A Clarke, and Leon J Osterweil. Smart checklists to improve healthcare outcomes. In *International Workshop on SEHS*, pages 54–57, 2016.

[Dahl and Søberg, 2013] Øyvind Dahl and Marius Søberg. Labour inspection and its impact on enterprises' compliance with safety regulations. *Safety Science Monitor*, 17(2):1–12, 2013.

[Darwiche, 2009] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

[De Bie *et al.*, 2017] AJR De Bie, S Nan, LRE Vermeulen, PME Van Gorp, RA Bouwman, AJGH Bindels, and HHM Korsten. Intelligent dynamic clinical checklists improved checklist compliance in the intensive care unit. *BJA: British Journal of Anaesthesia*, 119(2):231–238, 2017.

[Flogard *et al.*, 2021] Eirik Lund Flogard, Ole Jakob Mengshoel, and Kerstin Bach. Bayesian feature construction for case-based reasoning: Generating good checklists. In *ICCBR*, pages 94–109. Springer, 2021.

[Grigg, 2015] Eliot Grigg. Smarter clinical checklists: how to minimize checklist fatigue and maximize clinician performance. *Anesthesia & Analgesia*, 121(2):570–573, 2015.

[Haruna *et al.*, 2017] Khalid Haruna, Maizatul Akmar Ismail, Suhendroyono Suhendroyono, Damiasih Damiasih, Adi Cilik Pierewan, Haruna Chiroma, and Tutut Herawan. Context-aware recommender system: A review of recent developmental process and future research direction. *Applied Sciences*, 7(12):1211, 2017.

[Hasson *et al.*, 2000] Felicity Hasson, Sinead Keeney, and Hugh McKenna. Research guidelines for the delphi survey technique. *Journal of advanced nursing*, 32(4):1008–1015, 2000.

[Ho *et al.*, 2018] Daniel E Ho, Sam Sherman, and Phil Wyman. Do checklists make a difference? a natural experiment from food safety enforcement. *Journal of Empirical Legal Studies*, 15(2):242–277, 2018.

[Huleihel *et al.*, 2021] Wasim Huleihel, Soumyabrata Pal, and Ofer Shayevitz. Learning user preferences in non-stationary environments. In *AISTATS*, pages 1432–1440. PMLR, 2021.

[Idrees *et al.*, 2020] Mobin M Idrees, Leandro L Minku, Frederic Stahl, and Atta Badii. A heterogeneous online learning ensemble for non-stationary environments. *Knowledge-Based Systems*, 188:104983, 2020.

[Jelacic *et al.*, 2020] Srdjan Jelacic, Andrew Bowdle, Bala G Nair, Kei Togashi, Daniel J Boorman, Kevin C Cain, John D Lang, and E Patchen Dellinger. Aviation-style computerized surgical safety checklist displayed on a large screen and operated by the anesthesia provider improves checklist performance. *Anesthesia & Analgesia*, 130(2):382–390, 2020.

[Jorro-Aragoneses *et al.*, 2020] Jose Luis Jorro-Aragoneses, Marta Caro-Martínez, Belén Díaz-Agudo, and Juan A Recio-García. A user-centric evaluation to generate case-based explanations using formal concept analysis. In *ICCBR*, pages 195–210. Springer, 2020.

[Karanikas and Hasan, 2022] Nektarios Karanikas and Sikder Mohammad Tawhidul Hasan. Occupational health & safety and other worker wellbeing areas: Results from labour inspections in the bangladesh textile industry. *Safety Science*, 146:105533, 2022.

[Kim *et al.*, 2014] Been Kim, Cynthia Rudin, and Julie A Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *NeurIPS*, pages 1952–1960, 2014.

[Kulp *et al.*, 2021] Leah Kulp, Aleksandra Sarcevic, Megan Cheng, and Randall S Burd. Towards dynamic checklists: Understanding contexts of use and deriving requirements for context-driven adaptation. *ACM TOCHI*, 28(2):1–33, 2021.

[Mengshoel *et al.*, 2021] Ole Jakob Mengshoel, Eirik Flogard, Jon Riege, and Tong Yu. Stochastic local search heuristics for efficient feature selection: An experimental study. In *Norsk IKT-konferanse for forskning og utdanning*, pages 58–71, 2021.

[Morgan *et al.*, 2007] Pamela J Morgan, Jenny Lam-McCulloch, Jodi Herold-McIlroy, and Jordan Tarshis. Simulation performance checklist generation using the delphi technique. *Canadian journal of anaesthesia*, 54(12):992–997, 2007.

[Natani and Watanabe, 2021] Garima Natani and Satoru Watanabe. Knowledge graph-based data transformation recommendation engine. In *IEEE BigData*, pages 4617–4623. IEEE, 2021.

[Nikpour and Aamodt, 2021] Hoda Nikpour and Agnar Aamodt. Fault diagnosis under uncertain situations within a bayesian knowledge-intensive cbr system. *Progress in Artificial Intelligence*, pages 1–14, 2021.

[World Health Organization, 2021] W̃orld Health Organization. Joint estimates of the work-related burden of disease and injury, 2000-2016: Global monitoring report. Technical report, 2021.

[Zhang and Balog, 2020] Shuo Zhang and Krisztian Balog. Evaluating conversational recommender systems via user simulation. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1512–1520, 2020.

[Zhang *et al.*, 2021] Haoran Zhang, Quaid Morris, Berk Ustun, and Marzyeh Ghassemi. Learning optimal predictive checklists. *NeurIPS*, 34, 2021.