Feature-Based and Adaptive Rule Adaptation in Dynamic Environments

Alireza Tabebordbar¹ · Amin Beheshti¹ · Boualem Benatallah² · Moshe Chai Barukh²

Received: 10 March 2020 / Revised: 4 May 2020 / Accepted: 8 June 2020 / Published online: 25 June 2020
© The Author(s) 2020

Abstract

Rule-based systems have been used increasingly to augment learning algorithms for annotating data. Rules alleviate many of the shortcomings inherent in pure algorithmic approaches, in cases algorithms are not working well or lack from enough training data. However, in dynamic curation environments where data are constantly changing, there is a need to craft and adapt rules to keep them applicable and precise. Rule adaptation has been proven to be painstakingly difficult and error-prone, as an analyst is needed for examining the precision of rules and applying different modifications to adapt the imprecise ones. In this paper, we present an autonomic and conceptual approach to adapt data annotation rules. Our approach offloads analysts from adapting rules; it boosts rules to annotate a larger number of items using a set of high-level conceptual features, e.g. topic. We utilize a Bayesian multi-armed-bandit algorithm, an online learning algorithm that adapts rules based on the feedback collects from the curation environment over time. We propose a summarization technique, which offers a set of high-level conceptual features for annotating items by identifying the semantical relationships among them. We conduct experiments on different curation domains and compare the performance of our approach with systems relying on analysts for adapting rules. The experimental results show that our approach has a comparative performance to analysts in adapting rules.

Keywords Rule adaptation · Data annotation rules · Rule-based systems · Data curation systems

1 Introduction

Data curation indicates processes and activities related to the integration, annotation, publication, and presentation of data throughout its lifecycle [8]. One category of data curation is data annotation, which aims at labelling the raw data to generate value and increase productivity. Data annotation has been used extensively in various computational machine learning algorithms for information extraction, item classification, record-linkage [6, 38, 39]. However, in dynamic environments, e.g. Twitter (twitter.com/) and Facebook (facebook.com/), where data are continuously changing, relying on pure algorithmic approaches do not scale to the need of businesses that need to annotate data over an

extended period of time. Because algorithms make predictions based on the historical data only. While, in dynamic environments, the distribution of data is changing, and algorithms need to be updated to capture the changes, which is expensive and time-consuming.

In recent years, several pioneering solutions (e.g. [5, 23, 30, 33, 44, 51]) have been proposed to augment algorithms with rule-based techniques. Rules can alleviate many of the shortcomings inherent in pure algorithmic approaches. Rules can be written by non-technician analysts, which is less expensive than training algorithms [23]. Updating rules is faster than training algorithms and can supplement algorithms in cases they are not working well [44].

To keep a rule applicable and precise [5, 23, 33, 44, 51], there is a need for an analyst to adapt¹ the rule based on changes in the curation environment. Rule adaptation has been proven to be painstakingly difficult as the analyst needs to understand the context of data and the impact of modifications she applies to the rule [32]. In many cases, the analyst needs to apply different changes to identify the optimal one.

✉ Alireza Tabebordbar
alireza.tabebordbar@mq.edu.au

Amin Beheshti
amin.beheshti@mq.edu.au

¹ Macquarie University, Sydney, Australia

² University of New South Wales (UNSW), Sydney, Australia

¹ The process of modifying a rule to become better suited to the curation environment.

This problem exacerbated in dynamic curation environments as adaptation is not a single rule modification task, and the rule needs to be updated over its lifetime.

In this paper, we take the first step toward creating an adaptive rule adaptation model in dynamic and constantly changing environments. While previous approaches rely on analysts to identify the optimal modifications for rules, we propose a different learning task. We focus on incrementally adapting a rule based on the changes in the curation environment. Hence, we focus on offloading analysts and updating a rule autonomously. We do this by utilizing a Bayesian multi-armed-bandit algorithm, which learns the optimal modification by observing rules performance over time. Besides, previous systems adapt rules at the syntactic level, e.g. keyword and regular expression. Syntactic-level adaptation limits rules ability in annotating data as rules skip a large number of semantically related items. Instead, we focus on coupling syntactic-level features with conceptual features to boost rules to annotate a larger number of items.

Overall, our solution is made up of the following stages: (1) Each time a rule annotates a set of items, we extract a set of candidate features (e.g. syntactic and conceptual features) as the potential modifications. (2) Then, a Bayesian multi-armed-bandit algorithm determines the optimal modification for the rule by estimating a probability distribution for candidate features. (3) Over time, by annotating more items, the algorithm learns the performance of candidate features better and modifies the rule to keep the rule applicable and precise.

Example: rule adaptation through analyst Consider a government that intends to analyze citizens' opinions regarding the quality of social services, e.g. health care, domestic violence, and aged care services. Social media is one of the sources that decision-makers may rely on to understand public satisfaction levels. Social media allows users to express their opinions regarding their communities through their Posts, Tweets, or comments. The government may rely on learning algorithms to analyse and understand the people opinions. However, learning algorithms need training data, which is not available in many cases. One solution is to craft rules to curate and extract the data to train or augment the learning algorithms. For example, an analyst can write the rule R_1 to curate items relevant to 'Health Care'.

$Rule_1 = \text{IF tweet contains } (\text{'health'}) \text{ keyword THEN tag with } \text{'Health Care'}.$ (1)

However, rule $Rule_1$ will tag a large number of irrelevant items as every item, e.g. Tweets, Posts, comment, that contain the 'Health' keyword is not expressing an issue relevant to 'Health Care'. Thus, the analyst may adapt the rule $Rule_1$ by adding new keywords to make the rule more precise. For example:

$Rule'_1 = \text{IF tweet contains } (\text{'Health'}) \text{ keyword AND tweet contains } (\text{'Care'}) \text{ keyword THEN tag with } \text{'Health Care'}.$

After adaptation rule, $Rule'_1$ curate items that contain both 'Health', and 'Care' keywords. But the topic 'Health Care' contains a large number topical sub-spaces, which rule $Rule'_1$ will ignore as they may not contain both 'Health' and 'Care' keywords. Adapting a rule through an analyst is both time-consuming and challenging. Either due to the fact the analyst does not really know how or what is the perfect rule, e.g. is anything missing, or simply a certain type of analysis contains too many different conditions that would require many complex rules. This problem exacerbated as the social media data are 'ever-changing and never-ending' [23], and social media users trickling in a million pieces of different information every day. Thus, rule adaptation is not a one-time rule modification task and the analyst needs to update a rule over time based on changes in the curation environment. Figure 1 shows a typical workflow of adapting rules through the analyst.

Contribution By considering the work proposed by Tabebordbar et al. [21], in this paper we propose a feature-based technique for enhancing rules to annotate a larger number of items. Our solution couples both conceptual and syntactical features for annotating data. It leverages a summarization technique to identify the semantical relationship among items and extract the conceptual-level features.

The rest of this paper is organized as follows: we discuss the related works in Sect. 2. Then, in Sect. 3, we explain the problem. We discuss our solution in Sect. 4. Next, we present the performance of our approach on three different curation domains: mental health, domestic violence, and budget. The experiment results showed our approach could significantly improve the precision of rules in annotating data (by as much as 29% precision compared to the initial results). Finally, we discuss the future works and conclude the paper in Sect. 7.

2 Related Works

In this section, we discuss prior works related to rule adaptation (Sect. 2.1), and online learning algorithms (Sect. 2.2). In particular, we discuss the usage of a Bayesian multi-armed-bandit algorithm in unstructured and constantly changing environments. Besides, we consider it appropriate to discuss

approaches on feature extraction to position our proposed feature summarization technique (Sect. 2.3).

Fig. 1 The overview of adapting rules through analysts and crowd workers

2.1 Rule Adaptation

Rule adaptation is a continuous process focused on modifying a rule to fit the rule to the curation environment better. However, rule adaptation is a challenging and error-prone task. Thus, many solutions [23, 24, 30, 33, 48, 51] have been proposed to assist analysts in adapting rules. Several solutions [24, 30, 32, 33, 48] focused on interactively adapting rules. In these solutions, a system proposes possible adaptations for a rule, and an analyst adapts the rule through interacting with the system. For example, Milo et al. [33] proposed a cost-benefit approach for generalizing or specializing fraud detection rules. The approach developed a heuristic algorithm to interactively adapt rules with domain experts until the desired set of rules is obtained. Volks et al. [48] proposed a cost function to adapt the integrity constraint (IC) rules. The approach relies on the analyst feedback to update the cost function and resolve the inconsistencies in IC rules. Liu et al. [30] proposed an interactive approach for refining a rule using a set of positive and negative results. The method uses a provenance graph to identify candidate changes that can eliminate negative results. However, these solutions focus on adapting rules that operate on structured data, where a rule may adapted with a limited number of features. Besides, many of these solutions assume the analyst can access to a ground truth, e.g. a dataset of items tagged

with the correct label, to verify the effectiveness of an adaptation.

Alternatively, to adapt rules in both unstructured and dynamic environments, some solutions [23, 44, 51] focused on augmenting interactive rule adaptation systems by coupling crowds with analysts. These solutions rely on crowd workers to determine the precision of rules. For example, Xie et al. [51] proposed an approach for validating rules for information extraction purposes. The approach relies on a voting technique to identify whether an adaptation of a rule produces a positive impact in extracting information or not. GC et al. [23] designed an interactive system by coupling analysts and crowds for adapting rules. The system verifies items annotated with a rule using crowd workers and assists analysts in identifying the optimal modification using a relevance feedback algorithm (Rocchio). Sun et al. [44] proposed a rule-based technique (Chimera) for large-scale data classification systems. First, the approach identifies the misclassified items in cooperation with crowd workers. Then, forwards the items to analysts to write rules and address the errors. Bak et al. [5] relies on visualization by showing the result of applying a rule on a set of data records. The system requires crowd workers to verify the outcome of applying the rule on the data record, indicating the optimal adjustment for the rule.

Although coupling crowd workers with interactive systems provide more flexibility in adapting rules in dynamic

Table 1 A sample list of different rule adaptation techniques and their domains

		Crowd	Analyst	Visualization	Machine learning
Information extraction	[23, 44, 47, 51]	✓	✓		✓
	[5]	✓	✓	✓	
	[4, 14, 15, 30, 34, 45]		✓		✓
Databases	[16, 17, 24, 27, 48]		✓		✓
Fraud detection	[26, 32, 33]		✓		✓

environments, these systems still rely on analysts for identifying the optimal modification of rules. In contrast, our approach not only offloads analysts but also it autonomically modifies a rule regarding changes in the curation environment. Table 1 shows a sample list of techniques used for adapting rules and their domains.

2.2 Multi-armed Bandit Algorithm

In this section, we discuss how a Bayesian multi-armed-bandit algorithm has been used in dynamic and constantly changing environments. This algorithm increasingly used in large-scale randomized A/B experimentation by technology companies [28]. One area of work that used a Bayesian multi-armed-bandit algorithm is educational learning to facilitate the learners' learning rate. For example, Williams et al. [50] proposed a system (AXIS) to improve explanation generation for online learning materials by employing a combination of crowds and a Bayesian multi-armed-bandit algorithm. Clement et al. [18] used a multi-armed bandit algorithm in intelligent tutoring systems to choose activities that provide better learning for students. Other areas that relied on a Bayesian multi-armed-bandit algorithm are feature engineering [3], gaming [31], and online marketing [11]. In this context, we follow a similar trend by employing a Bayesian multi-armed-bandit algorithm with the crowd workers. Over time, the algorithm based on the collected feedback determines an adaptation for the rule to keep the rule applicable and precise.

2.3 Feature Extraction

In addition to interactive systems for helping analysts in adapting rules, we consider it appropriate to include approaches in feature extraction to position our proposed summarization technique. Feature extraction is the process of identifying a set of variables that best describe the data [19]. Feature extraction is an ongoing task and requires to iteratively explore the curation environment to identify features that capture the salient aspect of data. Several approaches have been proposed to aid analysts in feature extraction (e.g. [7, 9, 10, 10, 12, 29, 46, 49]). For example, Anderson et al. [2] proposed BrainWash, a system that

provides a pipeline to ease the process of feature extraction in large datasets. The system focused on helping a user to explore, extract, and evaluate features faster. Cheng et al. [13] relied on crowd workers for feature extraction. The approach refines the performance of machine learning algorithms based on the feedback received from crowds. Veeramachaneni et al. [46] proposed an approach to engage crowd workers in extracting features and predicting students' stop-out on massive open online courses (MOOC) systems. The approach provides a pipeline for evaluating and examining the relevancy features through crowd workers.

Another type of works focused on facilitating feature extraction through visualization techniques. For example, Patel et al. [35] relied on visualizing the confused region of machine learning classifiers to help analysts in extracting features. Brooks et al. [10] provide a visual summary of the data to aid a user to create a dictionary of features. Stoffel et al. [43] relied on visualization for examining machine learning features error. The system iteratively interacts with a user to remove ineffective features.

In contrast, we propose a summarization technique that identifies the semantical relationship among keywords and extracts features at the conceptual level. Each conceptual feature represents a group of semantically related keywords, which boosts rules to annotate a larger number of items.

3 Preliminaries and Problem Statement

We first introduce the component of rules used in this paper (Sect. 3.1). We then describe the problem in Sect. 3.2. Finally, we provide an overview of our solution in Sect. 3.3.

3.1 Preliminaries

Feature We express a rule R in forms of features, where each feature $f \in R$ corresponds to a function in forms of

$$\langle \text{Dataset}, \text{Function}, \text{Operator} \rangle \rightarrow \text{Value}$$

where *Dataset* is the data source such as Twitter and Facebook, *Function* performs the curation task (e.g. feature extraction), *Operator* represents the condition for a feature

Fig. 2 The overview of the proposed approach for adapting rules

to curate the data, and *Value* is the output of a feature. An example of a feature is extraction functions, e.g. named-entities, or similarity extraction. Expressing a feature as a function allowing us to leverage the standard data-types as the feature's operator. For example, if a feature operates over textual data, the operator for the feature will include string operators, such as *contains* and *exact*. Similarly, if a feature curates integer data the feature will include integer operators, such as *equals* and *less-than*. As an example, consider the feature $f_1 = \langle \text{Tweet.Keyword.Contains}(\text{'Mental'}) \rangle$, which curates Tweets that contain 'Mental' keyword. In this example, *Tweet* represents the dataset the feature operates for curating the data, *Keyword* represents the function of the feature, and *Contains('Mental')* is the operator and represents the condition for curating a Tweet.

Rule We represent a rule R as a tree of features, where each feature $f \in R$ can have K children. We denote a path p in the tree as a sequence of features $f_1, \dots, f_m$, where f_1 represents the root feature and f_m represents the last feature in the path. More precisely, a path p is a conjunction of features in the form $f_1 \wedge \dots \wedge f_m$. To curate an item with a rule, the item should be annotated with all features within a path. Notice that, we do not require inventing our rule language. Rather, the benefit of rules being expressed as features, we can adopt any suitable functional or rule-expression language for our purpose.

Tag A Tag is the label, e.g. 'Mental Health', a rule assigns to a curated item, e.g. Tweet, to describe the item. In this paper, we use the term *tag* and *annotate* interchangeably. As an example, consider the rule presented in Figure 2. This rule is made up of three features $\{f_1, f_2, f_3\}$, and tags a Tweet with 'Mental Health', if the Tweet curated with features $f_1 \wedge f_2$, or $f_1 \wedge f_3$. More clearly, Rule_1 tags a Tweet with 'Mental Health', if the Tweet contains 'Mental' and 'Health' keywords or the Tweet contains 'Mental' and a keyword related to 'Medical' topic.

3.2 Problem Statement

In the followings, we discuss two major problems in rule-based systems.

Adaptation through analyst Typically, to adapt a rule, an analyst examines the annotated items to identify the potential modifications that make the rule precise [23, 30, 32, 33]. However, rule adaptation is challenging and error-prone as the analyst needs to evaluate the impact of each modification she applies to the rule. Such a problem is categorized under the category of *online learning* problem, where an analyst does not have access to the entire knowledge to craft the adequate type of rule. Instead, over time she learns to better adapt a rule through examining the items.

To offload analysts from adapting rules, we formulated the problem as a Bayesian multi-armed-bandit algorithm. The algorithm is suitable when the required information for making a decision is provided piece-by-piece in a serial fashion. Each time a rule annotates a set of items, the algorithm collects feedback over the number of items the rule correctly/incorrectly annotated. Then, by receiving more feedback the algorithm learns a better adaptation for the rule over time.

For example, consider a rule R that operates over a dataset and must annotate data with a threshold j . Assume that at time τ_i rule R annotated a set of items $I_{\tau_i} = \{i_1, i_2, \dots, i_n\}$. We denote by $P[R_{\tau_i}]$ as the precision of the rule observed at time τ_i . Our algorithm adapts rule R at time τ_{i+1} , where $P[R_{\tau_{i+1}}] > j$.

Syntactic-level data annotation Typically, an analyst adapts a rule at the syntactic level, e.g. keywords and regular expressions. Using syntactic-level features allowing the analyst to more conveniently modify a rule by replacing irrelevant keywords or phrases with new ones. However, relying on syntactic-level features limits the capacity of a rule

in annotating data as these features skip a large number of semantically related items. For example, consider the rule:

$Rule_{11} = Tweet.Keyword.Contains(\{Mental'\}) \wedge Tweet.Keyword.Contains(\{Health'\}) : \{Mental\}Health'$

This rule tags a Tweet if the Tweet contains 'Mental' and 'Health' keywords. However, there exists a large number of Tweets relevant to 'Mental Health', which could not be tagged with the $Rule_{11}$ as those Tweets may not contain both 'Mental' and 'Health' keywords.

3.3 Solution Overview

The overview of our proposed solution is shown in Figure 2. The approach consists of four steps, feature extraction, observation, estimation, and adaptation.

Feature extraction The initial step in the workflow is feature extraction, which extracts a set of candidate features $T = \{t_1, t_2, \dots, t_n\}$ from annotated items. The approach extracts candidate features both at the syntactic and conceptual levels. Each syntactic-level feature represents a keyword extracted from items annotated with a rule, while a conceptual feature represents a group of semantically related keywords. In Sect. 4.1, we accentuate how our approach extracts candidate features for adapting a rule.

Observation The second step in the workflow is observation, which gathers feedback to update a Bayesian multi-armed-bandit algorithm about changes in a curation environment. For gathering feedback, we rely on the crowd workers². Each time a rule annotates a set of items $I = \{i_1, i_2, i_3, \dots, i_n\}$, the algorithm receives feedback over a sample of annotated items $S = \{i'_1, i'_2, i'_3, \dots, i'_n\}$, where $S \subset I$ to identify the latest rule performance in annotating the data. Crowds verify whether a rule correctly tagged an item or not. In Sects. 4.2 and 5, we review how crowd workers contribute in verifying items.

Estimation The third step in the workflow is *Estimation*, where a Bayesian multi-armed-bandit algorithm determines the performance of candidate features by estimating a probability distribution θ . The algorithm calculates the performance of features using workers collected feedback

To formulate workers feedback as a Bayesian multi-armed-bandit problem, we propose a reward/demote schema. Each time the rule annotates a set of items, the schema calculates a reward/demote for candidate features to update the algorithm about changes in the curation environment. In Sect. 4.3, we review how the approach estimates the probability distribution for candidate features.

Adaptation Given a set of candidate features T along with their probability distribution θ , we identify potential modifications that keeps a rule applicable and precise. We do this by removing or restricting features that deteriorate the rule performance. In Sect. 4.4, we review how our approach modifies a rule.

4 Adaptive Rule Adaptation

In this section, we explain components (feature extraction, observation, estimation, and adaptation) of our proposed solution.

4.1 Feature Extraction

The first step in our workflow is feature extraction, where we extract a set of candidate features as the potential modifications for a rule. Each time a rule annotates items, we extract a set of candidate features to calculate their performance in adapting the rule. We extract two types of candidate features, syntactic and conceptual. A syntactic-level feature represents a keyword within an annotated item, while the latter represents a group of semantically related keywords. Followings explain how we extract features.

Syntactic candidate feature

For extracting syntactic candidate features, we conduct a preprocessing task on annotated items I . The preprocessing performs tokenization, normalization, and noise removal. In tokenization, we split each item $i \in I$ into smaller tokens. Normalization removes stop words and conduct stemming, and noise removal skips certain characters, e.g. emoji, URLs, that occur in items. We consider the remaining tokens as the candidate feature type of *keyword*.

Conceptual candidate feature

Conceptual candidate features are proposed to alleviate shortcomings exist in annotating data using syntactic features. Although syntactic features allow an analyst to modify a rule more conveniently, relying on these features cannot capture the salient aspect of data and limits rules capacity in annotating items. Thus, there is a need for more productive features to boost rules to annotate a larger number of items. We propose a *summarization* technique, which extracts and groups semantically related keywords and forms a concept. Summarization consists of two steps: (1) mapping, and (2) grouping. In the mapping step [37, 42], we map each syntactic feature using a knowledge base to an abstract concept and associate a descriptor to it. In the grouping step, we group features with an identical descriptor and consider each group as a conceptual candidate feature. Following explains how our proposed technique extracts two new conceptual features using two readily available knowledge bases: WordNet [20]

² <https://www.figure-eight.com/>.

and Empath [22]. Algorithm 1 shows the pseudo-code of summarization technique.

Algorithm 1: Feature Summarization

```

1 Function Feature_Summarization():
   Input:  $T$ 
   Output:  $T'$ 
    $set\_map \leftarrow \emptyset$ ,  $T' \leftarrow \emptyset$ ;
2   foreach  $t \in T$  do
3      $set\_map.Add(Abstract[t])$ ;
4   foreach  $t_{map} \in set\_map$  do
5     foreach  $t \in T$  do
6       if  $Abstract[t] == t_{map}$  then
7          $T'[t_{map}].Add(t)$ ;
8   return  $T'$ ;

```

1. **WordNet** WordNet³ is a semantic lexicon, which grouped English words into sets of synonyms called synsets. We use WordNet to identify semantic relations between keywords using their hypernyms relation. A hypernym is a relationship between a generalized term and a specific instance of it. For example, based on the hypernym relationship in the wordNet, we can describe the keyword ‘doctor’ as a ‘*medical_practitioner*’. Thus, as the mapping step, we mapped each keyword using its hypernym relation to a more generalized form, where the hypernym acts as the descriptor for the keywords. Next, in the grouping step, we group features with the same descriptor and consider each group as a conceptual candidate feature. For example, consider $Rule'_{11}$, which tags Tweets with mental health.

$Rule'_{11} = Tweet.Keyword.Contains(\{Mental'\}) \wedge Tweet.Topic.Contains(\{Medical_Practitioner'\} : \{Mental\ Health'})$

This rule tags a Tweet, if the Tweet contains ‘Mental’ keyword, and a keyword relevant to ‘*Medical_Practitioner*’. The topic ‘*Medical_Practitioner*’ represents a large number of semantically related keywords, including doctors, physician, dentist.

2. **Empath** The second knowledge base we rely on for extracting conceptual features is EMPATH [22]. EMPATH is a deep learning skip-gram network, which categorizes text over 200 built-in categories. It represents a token as a vector using a vector space model (VSM) [36] and assigns tokens to categories based on their vector similarity. To extract conceptual candidate features, we query the EMPATH vector space

model to map each keyword to a category. We use categories to represent keywords in the abstract concept. Then, we group keywords with the same categories and consider each group as a conceptual candidate feature. For example, consider the following keywords $T = \{t_1 : fund, t_2 : illness, t_3 : budget, t_4 : disease\}$. To generate conceptual features, we query the EMPATH vector space model to map each keyword to a category. Assume, the following categories are identified $T = \{t_1 : Economy, t_2 : Health, t_3 : Economy, t_4 : Health\}$. Then, we group keywords with the identical categories and represent $\{fund, budget\}$ keywords as the *Economy* topic, and $\{disease, illness\}$ keywords as the *Health* topic.

4.2 Observation

The second step in our proposed approach is observation, which gathers feedback to update a Bayesian multi-armed-bandit algorithm about changes in the curation environment. For gathering feedback, we rely on crowd workers. Each time a rule annotates a set of items, we take a sample of the items $S = \{i'_1, i'_2, i'_3, \dots, i'_n\}$, where $S \subset I$ to send to the crowd. The crowd workers verify whether an item correctly tagged with the rule or not, e.g. if a rule tags an item with ‘Mental Health’. The task was to confirm whether the item is relevant to ‘Mental Health’ or not. For taking samples, we divided annotated items into subgroups [25] and represented each subgroup by a candidate feature – the population of subgroups determined by the frequency of candidate features in annotated items.

More clearly, consider the following candidate features $T = \{t_1 : fund, t_2 : illness, t_3 : budget, t_4 : economy\}$ that extracted from items annotated with a rule. The approach divides annotated items into four subgroups, where feature t_1 represents items contain *fund*, feature t_2 represents items contain *illness*, and so forth.

Our sampling strategy boosts a Bayesian multi-armed-bandit algorithm (see Sect. 4.3) to better learn the performance of features in adapting a rule. For example, if we used more obvious techniques, such as random sampling, then the algorithm considers all items equally likely. Thus, it takes a longer time to learn the performance of candidate features.

4.3 Estimation

This step computes a probability distribution θ for candidate features to determine their performance in adapting the rule. This step consists of two components: (i) reward/demote schema, which calculates a reward/demote for candidate features using workers feedback, and (ii) a Bayesian multi-armed-bandit algorithm, which estimates the performance of candidate features based on their collected rewards/demotes.

³ <https://wordnet.princeton.edu/>.

Reward/demote schema To adapt rules, we formulated rule adaptation as a Bayesian multi-armed-bandit algorithm. This algorithm is suitable when a system needs additional improvement to their decisions over time. The algorithm based on the feedback collects from the curation environment learns more consistent patterns of changes and takes a decision that maximizes its performance. A Bayesian multi-armed-bandit algorithm is a good fit for our problem because each time a rule annotates a set of items, it gets updated by the workers' feedback.

To frame the rule adaptation as a Bayesian multi-armed-bandit problem, we propose a reward and demote schema using the feedback collected from workers. The schema assigns a reward/demote to candidate features $t \in T$ appear in annotated items. The schema rewards r , a candidate feature, if it appears in an item that verified as relevant. Similarly, it demotes d , a candidate feature, if a feature appears in an irrelevant item. Over time, as a rule, annotates more items the schema updates candidate features reward/demote, allowing a Bayesian multi-armed-bandit algorithm to update its estimation regarding the performance of features in adapting the rule.

As each conceptual candidate feature represents a group of keywords, we calculate the reward/demote for these features based on the rewards/demotes collected by their associated keywords. More precisely, consider feature t as a conceptual candidate feature. Suppose $t = \{t'_1, t'_2, \dots, t'_n\}$, where t' represents a keyword associated with t . We calculate reward as $r_t = \sum_{t'=1}^n r_{t'}$ and demote as $d_t = \sum_{t'=1}^n d_{t'}$. Clearly, consider the candidate feature 'Medical_Practitioner', that was introduced in the previous section. Suppose the following features are associated with it: doctor, dentist, and physician. We calculate reward/demote for 'Medical_Practitioner' by summing up the rewards/demotes collected by doctor, dentist, and physician.

Algorithm 2: Estimating expected performance of candidate features

```

1 Function Est_Probability_Dist():
   Input:  $T$ 
   Output:  $\theta$ 
2    $r \leftarrow \emptyset, d \leftarrow \emptyset, \theta \leftarrow \{\}$ ;
3   foreach  $t \in T$  do
4     foreach  $I' \in S$  do
5       if  $t \in I'$  AND  $I'$  is verified as Irrelevant then
6          $d_t \leftarrow d_t + 1$ ;
7       else if  $t \in I'$  AND  $I'$  is verified as Relevant then
8          $r_t \leftarrow r_t + 1$ ;
9        $\theta_t \leftarrow \text{Beta}(r_t, d_t)$ 
10  return  $\theta$ ;

```

Bayesian multi-armed-bandit algorithm This section explains how a Bayesian multi-armed-bandit algorithm estimates the performance of candidate features. First, we explain the algorithm, and then we discuss how it operates in our settings.

We utilized Thompson sampling [41] for Bernoulli bandit problem, e.g. when the rewards are either 0 or 1. Thompson sampling is a Bayesian multi-armed-bandit algorithm that provides a dynamic policy for choosing which feature should be selected for adapting a rule, and an algorithm for incorporating new information to update this policy based on the candidate features rewards/demotes. Thompson sampling stores an estimated probability distribution θ for each candidate feature to indicate their performance in adapting the rule. Each time, the algorithm receives a set of candidate features $T = \{t_1, t_2, \dots, t_n\}$ along with their reward/demote it updates candidate features probability distributions $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$, where $0 < \theta < 1$ using the Bayesian formula:

$$P(\theta | t) = \frac{P(t | \theta) \times P(\theta)}{P(t)} \propto P(t | \theta) \times P(\theta)$$

$P(t | \theta)$ represents the likelihood and $P(\theta)$ is the prior. The likelihood is a Bernoulli distribution, and the prior is a Beta distribution.

It has been proven that Beta distribution is a good choice of priors for Bernoulli rewards [1]. Beta distributions form a family of continuous probability distributions on the interval (0, 1). The pdf of $\text{Beta}(\alpha, \beta)$, the Beta distribution with parameters $\alpha > 0, \beta > 0$, is given by

$$f(X; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} X^{\alpha-1} (1 - X)^{\beta-1}$$

Beta distribution is useful for Bernoulli rewards because if the prior is a $\text{Beta}(\alpha, \beta)$ distribution, then after observing a Bernoulli trial, the posterior distribution is simply $\text{Beta}(\alpha + 1, \beta)$ or $\text{Beta}(\alpha, \beta + 1)$, depending on whether the trial resulted in a success or failure, respectively.

The Thompson sampling algorithm initially assumes feature t to have prior $\text{Beta}(1, 1)$ on θ_i , which represents $\text{Beta}(1, 1)$ has a uniform distribution on (0,1). Each time, a feature appears in feedback, receives a reward r_t , or failures d_t , the algorithm updates the distribution on θ_i as $\text{Beta}(r_t + 1, d_t + 1)$. The algorithm then samples from these posterior distributions of the θ_i 's and selects feature according to the probability of its mean being the largest.

In order to formulate features feedback for Bernoulli bandit problem, we treat the total observation of features as a Bernoulli variable. The number of successes is the number of positive feedback, and the number of failures is the number of feedback minus successes. The update to $\text{Beta}(\alpha, \beta)$ is simply $\text{Beta}(\alpha + \text{number of new successes}, \beta + \text{number of new failures})$.

Fig. 3 Adapting a rule by replacing/restricting its features

Algorithm 2 shows how the approach estimates the value of θ for candidate features. As an example consider the following rule.

$Rule_1 = Tweet.Keyword.Contains(\}Mental') : \}MentalHealth'$

Which tags Tweets with 'Mental Health'. Assume, the following candidate features $T = \{t_1 : medical, t_2 : health, t_3 : wellbeing, t_4 : care, t_5 : qanda\}$ are extracted from annotated items as the potential modifications for the rule. First, to identify the performance of candidate features, the algorithm calculates their reward/demote using workers feedback. Then, a Bayesian multi-armed-bandit algorithm estimates a probability distribution θ for candidate features. Each time the rule annotates a set of items, the algorithm updates the value of θ based on the feedback gathers from workers to better understand features performance in adapting rules.

4.3.1 Discussion on Thompson Sampling

We have selected Thompson sampling as it has shown the near-optimal regret bound compared to other alternatives. Regret is the difference between the best performance from an option and the performance from the option chosen by the algorithm at time τ_i [1]. Although other variants of multi-armed bandit algorithm, such as ϵ -Greedy and Upper Bound Confidence (UBC), could be used for adapting rules. The ϵ -Greedy may stuck in local optima by exploiting the sub-optimal option. This algorithm greedy explores the environment for a period and then, exploits the best-identified option ϵ percent of the time and $1 - \epsilon$ for alternatives. For example, if we set the value of $\epsilon = 0.05$, the algorithm will exploit the best option for 95% of the time and explore for random alternatives for 5% of the time. UCB has been

proposed to reduce the randomness in ϵ -Greedy. The algorithm aims at reducing the parameter ϵ over time and is optimistic about options with high uncertainty. For example, it first chooses an option and observes its reward making us less uncertain about the option performance. The algorithm continues this behaviour until the uncertainty about the option reduces below a threshold.

4.4 Adaptation

In this section, we explain how our approach modifies a rule. Recall from Sect. 3.1, that we introduced a rule R as a tree of features, where each feature $f \in R$ can have K children. We also defined a path p in a rule as a conjunction of a set of features in forms of $p = f_1 \wedge f_2 \wedge \dots \wedge f_n$. First, to adapt a rule, we identify imprecise paths that annotate data with a precision below a threshold j . The threshold represents the minimum precision a path should have to be considered as precise. We determine the precision of paths by calculating the number of relevant/irrelevant items their features annotated. After identifying imprecise paths, we determine whether to replace or further restrict their features.

We would replace a feature in an imprecise path if the number of annotated items was below the average number of items annotated with its siblings, indicating the feature is imprecise and incapable of adequately annotating data. Conversely, we restrict a feature if the number of annotated items was greater than or equal to average, indicating the feature is applicable but should be restricted to be precise. For replacing or restricting features, we select candidate features that yielded the highest probability distribution θ estimated by a Bayesian multi-armed-bandit algorithm.

2 people die in rural areas each day due to mental health due to lack of mental health services Good luck!@amelieapp #winning #mentalhealth @StartupVic

Specify Tweets Expressing an Issue Relevant to 'Mental Health'

☐ Yes ☐ No ☐ I don't know

@FoxyFoxylyn growing a boy need mental stimulation!

Specify Tweets Expressing an Issue Relevant to 'Mental Health'

☐ Yes ☐ No ☐ I don't know

Fig. 4 Sample of questions to workers to verify the tag of items

Example Suppose, after annotating a set of items at time τ_i the algorithm identifies that the rule is imprecise⁴. Thus, it examines the number of annotated items and adapts the rule by appending K candidate features⁵ that yielded the highest probability distribution (restriction) (Figure 3b). After adaptation, $Rule_1$ annotates an item if the item curated with features in paths $p_1 = f_1 \wedge f_2$, or $p_2 = f_1 \wedge f_3$, or $p_3 = f_1 \wedge f_4$. Alternatively, the algorithm may replace a feature if it identifies the feature annotates data below the average number of items annotated with its siblings⁶. For example, suppose at time τ_{i+n} feature f_2 is identified as imprecise and incapable of annotating data adequately⁷. Thus, the algorithm removes feature f_2 and replaces the feature with a candidate feature that yielded the highest probability distribution value (Figure 3c). On the other hand, to select a candidate feature, the algorithm performs a feature extraction task and estimates candidate features probability distribution θ based on the reward/demote features accumulated from time τ_1 to τ_{i+n} .

The proposed adaptation strategy allows to adapt rules according to changes in the curation environment. For example, by replacing an imprecise feature with a content bearing feature that obtained a high value of θ over an extended period of time, we keep the rule applicable as the new feature better captures the salient aspect of data. Similarly, by restricting an imprecise feature that annotates a large number of items, we make the rule precise by filtering out the irrelevant items.

5 Gathering Workers Feedback

This section explains how we contribute workers to verify items annotated with rules. We created a task on Figure Eight⁸ micro-tasking market. The workers' task was to confirm whether an item is relevant to the tag assigned by a rule or not. Workers could choose 'Yes' if they identify the item is related to the tag, and 'No', if they determine the item is irrelevant. In cases workers could not verify an item, they could choose 'I don't know'. For example, we present a Tweet to workers, which a rule tagged as relevant to 'Mental Health'. Then, workers task was to verify whether the Tweet is related to 'Mental Health' or not. Besides, we provide workers with a textual instruction to explain to them how to confirm items.⁹ We explained steps workers need to follow and provided them with three positive¹⁰ and three negative¹¹ examples. For verifying each item, we paid 1 cent, and each worker varied ten items per page. At each round of the annotation task, we sent 3% of annotated items to workers. Figure 4 shows a sample question to workers.

5.1 Stopping Condition

In the previous section, we explained how workers verify annotated items. However, continuously sending items to crowds increases the cost of the adaptation task. Thus, there is a need to identify when a rule is stabilized to stop verifying more items. To address this problem, we developed a

⁴ Annotates data with a precision below j .

⁵ As feature f_1 is the root feature it annotates data above the average, thus satisfies the restriction condition.

⁶ features $\{f_3, f_4\}$ are siblings for feature f_2 .

⁷ Annotates data below the average number of items annotated with its siblings

⁸ <https://www.figure-eight.com/>.

⁹ For example, in this job, we will have you to identify whether a Tweet is expressing an issue relevant to mental health or not. An issue can be a shortcoming that exists in services provided for mental health, or a threat that lack of mental health services may cause to the society, or a suggestion that helps to improve the quality of mental health services.

¹⁰ @lucianaberger: Mental health services facing serious shortages of mental health nurses decrease of 12% since 2010 psychiatrists.

¹¹ if I have to hire a car and drive home from Belgium i am going to go mental stupid french air traffic control wanks on strike.

solution using the probabilistic policy defined in Thompson sampling algorithm to determine whether a path in a rule is stabilized or not. For each path, we estimate a probability distribution θ based on the number of relevant/irrelevant items annotated. Then, we define a smoothing window Q to record the value of θ . We set the size of smoothing window $Q = 3$ and average as the smoothing function. We consider a path as stabilized, if the value of Q increases or remains stable within 3ϵ , where $\epsilon = 0.01$ ¹². More clearly, consider path $p_3 = f_1 \wedge f_4$ presented in Figure 3. Each time the rule annotates a set of items, the algorithm records the value of θ for the path. Then, the approach computes the value of Q , where $Q_1 = \text{AVG}(\theta_1, \theta_2, \theta_3)$, and $Q_2 = \text{AVG}(\theta_2, \theta_3, \theta_4)$ and so forth. The algorithm stops sending items to workers, when the value of $Q_{i+1} + 3\epsilon \geq Q_i$ indicating the path is stabilized.

6 Experiments

First, we discuss the dataset was used for examining the performance of our proposed approach in Sect. 6.1. Then, in Sect. 6.2, we explain three scenarios have been defined to show the applicability of our approach. Finally, we discuss the results in Sect. 6.3.

6.1 Experiment Settings and Dataset

The core component of techniques described in the previous sections is implemented in Python. Three months of Twitter data (Australian region) were used as the input dataset (from May 2017 to August 2017) with ≈ 15 million Tweets. MongoDB and Elasticsearch were used for storing and indexing the input dataset. We demonstrate the performance of our approach in three different curation domains (domestic violence, mental health, and budget). We show how our approach learns to adapt a rule to annotate data more precisely over time. As the initial rules for annotating the data, we used rules that contain only one feature. For example, the initial rule for annotating Tweets in the mental health domain was in the form of *Tweet.keyword.contains({Mental'}) : }Mental Health'*, which tags Tweets that contain 'Mental' keyword. Then, at each timestep rules annotate a set of items, our approach adapts rules to make them more precise. We demonstrate the performance of the approach within five rounds of rule adaptation.

6.2 Experiment Scenarios

To evaluate the performance of our solution and the applicability of the proposed algorithm, we have defined three different experiment scenarios:

1. *Evaluating the performance of a Bayesian multi-armed-bandit algorithm in adaptation* We explain an experiment scenario to represent the performance of a Bayesian multi-armed-bandit algorithm in adapting rules. We demonstrate how the algorithm keeps a rule precise and applicable by adding or removing features. We adapt rules with two different choices of features ($K = 10$, $K = 20$) (see Sect. 3.1). Adapting a rule with a higher number of features allows a rule to annotate a larger number of items, but with less precise ones.
2. *Evaluating the proposed feature-based adaptation* This scenario aims at demonstrating the performance of the proposed feature-based technique in augmenting rules to annotate a larger number of items. We demonstrate the improvement rules make in the number of annotated items while adapting rules using both syntactical- and conceptual-level feature. We also compare the obtained results with technique that adapts rules at the syntactic level only.
3. *Comparison with existing studies* The third scenario, we conducted a controlled experiment and compared the performance of our approach with a system proposed by GC et al. [23]. The proposed system is an interactive rule adaptation system, which relies on analysts for adapting rules. Each time a rule annotates a set of items, the system sends a sample of items to crowds and receives feedback over the number of items correctly/incorrectly tagged by the rule. Then, the system tokenizes items and weights every token using the TF-IDF weighting scheme. Subsequently, the system ranks tokens based on their TF-IDF weights and iteratively shows tokens to an analyst to adapt a rule. The system continues showing tokens until the analyst is satisfied with the resulting rule. To help the analyst to more effectively adapts the rule, the system incorporates the analyst feedback by adjusting the weight of tokens using a relevance feedback algorithm [40]. Whenever the analyst selects a token, the algorithm increases the weight of other candidate tokens that co-occurred with the selected token.

¹² we set the value of ϵ and Q , experimentally using simulated data.

Fig. 5 The performance of a Bayesian multi-armed-bandit algorithm in adapting rules. As presented the algorithm could improve rules precision in all domains

6.3 Result

6.3.1 Performance of a Bayesian Multi-armed-bandit Algorithm in Adapting Rules

In this section, we demonstrate the performance of a Bayesian multi-armed-bandit algorithm in adapting rules (see Sect. 4.3). We show the precision of the rules adapted with two different choices of candidate features ($k = 10, k = 20$) that yielded the highest probability distribution θ . As presented in Figure 5 by adapting rules with 10 candidate features, the algorithm could significantly improve rules precision in all curation domains. For example, in the budget domain, the algorithm could improve the precision for 36.65%, from 54.56% to 91.21%. Similarly, in the domestic violence and the mental health domains,

Fig. 6 Comparison between the number of items annotated using conceptual- and syntactic-level features (Budget Domain)

Fig. 7 Comparison between the number of items annotated using conceptual- and syntactic-level features (mental health domain)

the algorithm could improve the precision for 18.20% and 32.47%, respectively. Also, to demonstrate the applicability of the algorithm in adapting rules, we repeated the experiment with a higher number of features ($K = 20$). This boosts rules to annotate a larger number of items, but with less precise features. Figure 5 shows the obtained results for each domain. As presented, adapting rules with a higher number of features decreases the precision of rules; however, the algorithm could learn the performance of features and adapts the rule to improve its precision over time. For example, in mental health domain, the precision is improved by 30.81%, and in budget and domestic violence domains the precision is improved by 33.22% and 16.36%, respectively.

Fig. 8 Comparison between the number of items annotated using conceptual- and syntactic-level features (domestic violence domain)

In this experiment, we considered features that annotate data with a precision below 75% ($j < 75\%$) as imprecise.

Discussion on rules performance As presented in Figure 5, the initial rules added to the curation system were imprecise and annotated a large number of irrelevant items. For example, the initial precision of rules in Budget and Mental Health domains was below 55%. However, after collecting a set of feedback the algorithm identifies the need to restrict rules by adding a new set of features. Although the restricting rules could improve their precision, this limited rules to only annotate those items that contain the features selected by the algorithm during the adaptation. As presented in Figures 6, 7, and 8, after adaptation rules are annotating fewer items compared to their initial states. For example, in Budget domain the number of annotated items has reduced by 15240, after two rounds of adaptation. We can see similar trends for other curation domains as well. But, the promising fact is that a Bayesian multi-armed-bandit algorithm can learn a better adaptation for rules by incrementally collecting more feedback over time. This can be seen in Figure 5 that the algorithm could dramatically improve the rule precision. For example, in Budget domain the difference in precision between the adaptation that occurred at τ_2 and τ_3 is over 10%. This difference for the

Table 2 Precision of the approach in adapting rules using *summarization* approach $K = 20$

Curation domain	Round 1	Round 2	Round 3	Round 4	Round 5
Budget	54.56	73.12	78.72	81.11	84.21
Mental health	54.74	57.35	71.40	80.16	80.61
Domestic violence	74.32	83.59	84.60	85.75	84.43

Mental Health domain is over 20%. Based on the obtained results, we concluded that a Bayesian multi-armed-bandit algorithm by collecting more feedback learns a better adaptation for rules over time, and if we can adapt rules with more robust features, we can improve both precision and recall. This fact, can be approved by comparing the precision and the number of annotated items between Figures 5 and Figures 6, 7, and 8. As presented by adapting rules with a higher number of features ($K = 20$), the algorithm could annotate a larger number of items, and at the same time maintain rules precision. In the next section, we discuss how feature-based adaptation augments the performance of rules to annotate a larger number of items.

6.3.2 Feature-Based Adaptation.

As we discussed in the previous Sect. 6.3.1, adaptation limits the ability of rules in annotating items. To alleviate this problem, we discussed that adapting rules with higher number features could boost rules to annotate a larger number of items. However, increasing the number of features has a negative correlation with precision (by increasing the number of features in adaptation the precision of rules drops). Thus, to diminish the impact of an adaptation and maintaining the performance of a rule in annotating items, we proposed feature-based adaptation. In feature-based adaptation, we hypothesizes that adapting a rule with a group of semantically related features would have a similar impact on the rule precision when adapting with a single feature. Thus, in this section, we study the impact of feature-based adaptation on rules performance. The goal is to study whether adapting a rule with a group of related features can enhance the performance of rules to annotate a larger number of items, and at the same time maintain their precision. To test our hypotheses, we conducted two sets of experiments. First, we discuss the precision of rules adapted through our approach. Then, we compare the number of annotated items with rules adapted using syntactic-level features.

Table 2 shows the precision of rules adapted using the feature-based technique. The obtained results confirm that feature-based adaptation can dramatically increase the performance of a rule in annotating items. At the same time, a Bayesian multi-armed-bandit algorithm could learn the performance of features and improves rules precision over time. This improvement for the domestic violence domain is 10.11%, and for mental health and budget, domains are 25.87% and 28.47%, respectively. Although the learning rate of the algorithm using the feature-based approach is slower than syntactic-level features, still the algorithm could improve rules precision in all domains. In addition, Figures 6, 7, and 8 compare the number of items annotated with rules adapted using the syntactic and feature-based approach. As presented, adapting rules with different

Table 3 Precision of rules adapted through participants in *Budget*, *Mental Health*, and *Domestic Violence Domains*

	Round 1	Round 2	Round 3	Round 4	Round 5
Budget domain					
Participant 1	54.56	79.75	82.02	84.21	87.19
Participant 2	54.56	83.22	85.62	88.20	90.86
Participant 3	54.56	86.56	86.77	86.67	86.59
Mental health domain					
Participant 1	54.74	72.88	80.65	87.19	86.32
Participant 2	54.74	70.38	75.09	83.58	85.01
Participant 3	54.74	71.63	81.61	85.04	84.14
Domestic violence domain					
Participant 1	74.32	87.65	88.78	90.90	90.82
Participant 2	74.32	88.63	90.28	91.36	92.59
Participant 3	74.32	85.37	86.51	87.04	86.42

Fig. 9 Number of items annotated with rules adapted through participants after five rounds of annotations in three different curation domains: budget, mental health, and domestic violence

features could boost rules to annotate a larger number of items. For example, in the domestic violence domain, the rule could annotate over 12,000 items. In mental health and budget domains, rules could annotate 13,574 and 8304 items, respectively. These numbers are much higher than adapting rules using syntactic-level features. For example, in the budget domain, the rule ($k = 10$) could annotate 2137 items only. Annotating data using the syntactic-level features in mental health and domestic violence domains show a similar trend and rules could only annotate 5198, and 4127 items, respectively.

Discussion on feature-based adaptation An advantage of feature-based adaptation is that it allowing users to better investigate their information needs while seeking for topics that contain a large number of topical subspaces. Suppose a user intends to curate data relevant to ‘mental health’. There exists a large number of keywords, e.g. health, disorder, service, that are relevant to mental health but may not receive enough feedback to be considered for adapting the rule. Using, feature-based adaptation, we group all keywords that

are associated with a topic, thus the rule can easily curate a varied and comprehensive list of items relevant to the user information need.

6.3.3 Comparison with Existing Studies

In this section, we compare the performance of our approach with the state of the art technique on rule adaptation. We implemented the system proposed by GC et al [23] (see Sect. 6.2) and conducted a controlled experiment. We asked three Ph.D. students in a laboratory that were familiar with the concept of learning algorithms, e.g. true positive rate, false-positive rate, to participate in the experiment. We explained to them how the system works and how they can use the system to adapt rules. Also, we allowed them to work with the system to gain the required understanding for adapting rules. To better compare the performance of our approach with the interactive system, we have asked participants to adapt rules in all domains. Then, in each curation domain, we selected the rule with the highest

obtained precision and compared it with rules adapted by our approach. In this experiment, we asked participants to adapt rules with 20 features ($k = 20$). Table 3 shows the results. As presented, our approach has comparable performance to interactive systems. For example, in budget domain participants could adapt the rule with 90.86% precision, which is 3.08% higher than our proposed approach. In the domestic violence and mental health domains, participants could adapt rules with the precision of 92.59% and 86.32%, respectively. Besides, Figure 9 shows the number of items annotated with the rules adapted by participants. The figure shows the most precise rules in each domain. Although our approach and participants have shown a similar performance while using syntactic-level features for adapting rule, using the proposed feature-based technique our approach could significantly annotate a larger number of items. For example, the number of annotated items in budget domain is higher by 3233 items. The difference in mental health and domestic violence domains is 5320, and 4305, respectively. The overall cost that we paid for verifying items in the mental health domain is \$35.10, and in the budget domain is \$29.92, and in the domestic violence domain is \$21.22.

By comparing the precision and the number of items annotated with our approach and participants, we believe that our adaptive approach outperforms current rule adaptation techniques. In particular, by considering the prohibitive cost of analysts for adapting rules, our proposed approach can boost companies and data enthusiasts that need to annotate data in unstructured and constantly changing environments with a limited budget.

7 Conclusion and Future Works

In this paper, we proposed an approach for adapting data annotation rules in unstructured and changing environments. Our approach offloads analysts from adapting rules and automatically modifies rules based on changes in the curation environment. We utilize a Bayesian multi-armed-bandit algorithm, an online learning algorithm that learns the optimal modification for rules using the feedback gathers from the curation environment. In addition, our approach adapts rules at the conceptual level, which boosts rules to annotate a larger number of items compared to current methods that rely on syntactic similarity, e.g. keywords, regular expression, for adapting rules. We evaluated the performance of our approach on three months of Twitter data in three different curation domains: domestic violence, mental health, and budget. The evaluation results showed our approach has comparable performance to systems relying on analysts for adapting rules.

There are several exciting directions for future work. In this paper, we introduced a summarization, which boosts

rules to annotate data at the conceptual level. As a part of future works, we plan to identify more features for adapting rules. Specifically, we focused on adapting rules with three other types of features, including entities, word2vec, and relation. We believe adapting rules with different kinds of conceptual features not only enhance the performance of rules to annotate a more significant number of items but also allows rules to capture the salient aspect of data better.

Acknowledgements We Acknowledge the AI-enabled Processes (AIP) Research Centre for funding part of this research.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Agrawal S, Goyal N (2012) Analysis of Thompson sampling for the multi-armed bandit problem. In: Conference on learning theory, pp 39–1
2. Anderson MR, Antenucci D, Burgess VBM, Cafarella M, Kumar A, Niu F, Park Y, Ré C, Zhang C (2013) Brainwash: a data system for feature engineering. In: CIDR
3. Anderson MR, Cafarella M, Jiang Y, Wang G, Zhang B (2014) An integrated development environment for faster feature engineering. *Proc VLDB Endow* 7(13):1657–1660
4. Atzmueller M, Kluegl P, Puppe F (2008) Rule-based information extraction for structured data acquisition using textmarker. In: LWA, pp 1–7
5. Bak P, Dolev D, Yatzkar-Haham T Rule adjustment by visualization of physical location data, Sept. 11 2014. US Patent App. 14/483,158
6. Beheshti A, Benatallah B, Nouri R, Chhieng VM, Xiong H, Zhao X (2017) Coredb: a data lake service. In: Proceedings of the 2017 ACM on conference on information and knowledge management. ACM, pp 2451–2454
7. Beheshti A, Benatallah B, Nouri R, Tabebordbar A (2018) Corekg: a knowledge lake service. *Proc VLDB Endow* 11(12):1942–1945
8. Beheshti A, Benatallah B, Tabebordbar A, Motahari-Nezhad HR, Barukh MC, Nouri R (2018) Datasynapse: a social data curation foundry. *Distrib Parallel Databases* 37:1–34
9. Beheshti A, Moraveji-Hashemi V, Yakhchi S, Motahari-Nezhad HR, Ghafari SM, Yang J (2020) Personality2vec: enabling the analysis of behavioral disorders in social networks. In: Proceedings of the 13th international conference on web search and data mining, pp 825–828
10. Brooks M, Amershi S, Lee B, Drucker SM, Kapoor A, Simard P (2015) Featureinsight: visual support for error-driven feature ideation in text classification. In: 2015 IEEE Conference on VAST. IEEE, pp 105–112

11. Burtini G, Loepky J, Lawrence R (2015) Improving online marketing experiments with drifting multi-armed bandits. *ICEIS* 1:630–636
12. Chen Y, Jiang H, Li C, Jia X, Ghamisi P (2016) Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans Geosci Remote Sens* 54(10):6232–6251
13. Cheng J, Bernstein MS (2015) Flock: Hybrid crowd-machine learning classifiers. In: *Proceedings of the 18th ACM conference on computer supported cooperative work and social computing*. ACM, pp 600–611
14. Chiticariu L, Krishnamurthy R, Li Y, Raghavan S, Reiss FR, Vaithyanathan S (2010) Systemt: an algebraic approach to declarative information extraction. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. In: Association for computational linguistics, pp 128–137
15. Chiticariu L, Krishnamurthy R, Li Y, Reiss F, Vaithyanathan S (2010) Domain adaptation of rule-based annotators for named-entity recognition tasks. In: *Proceedings of the 2010 conference on empirical methods in natural language processing*. In: Association for computational linguistics, pp 1002–1012
16. Chu X, Ilyas IF, Krishnan S, Wang J (2016) Data cleaning: Overview and emerging challenges. In: *Proceedings of the 2016 international conference on management of data*, pp 2201–2206
17. Chu X, Morcos J, Ilyas IF, Ouzzani M, Papotti P, Tang N, Ye Y (2015) Katara: a data cleaning system powered by knowledge bases and crowdsourcing. In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp 1247–1261
18. Clement B, Roy D, Oudeyer P-Y, Lopes M (2014) Online optimization of teaching sequences with multi-armed bandits. In: *7th international conference on educational data mining*
19. Domingos P (2012) A few useful things to know about machine learning. *Commun ACM* 55(10):78–87
20. Esuli A, Sebastiani F (2006) Sentiwordnet: a publicly available lexical resource for opinion mining. In: *Proceedings of LREC*, vol 6, pp. 417–422. Citeseer
21. Tabebordbar A, Beheshti A, Benatallah B, Barukh MC (2019) Adaptive rule adaptation in unstructured and dynamic environments. In: *International conference on web information systems engineering* (pp. 326–340). Springer, Cham
22. Fast E, Chen B, Bernstein MS (2016) Empath: understanding topic signals in large-scale text. In: *Proceedings of the 2016 CHI conference on human factors in computing systems*. ACM, pp 4647–4657
23. A.M. Ganan-Calvo, J.A. Hernandez Ramos, The fractal time growth of COVID-19 pandemic: an accurate self-similar model, and urgent conclusions. [arXiv:2003.14284](https://arxiv.org/abs/2003.14284) (2020)
24. He J, Veltri E, Santoro D, Li G, Mecca G, Papotti P, Tang N (2016) Interactive and deterministic data cleaning. In: *Proceedings of the 2016 international conference on management of data*. ACM, pp 893–907
25. Hunt N, Tyrrell S (2001) Stratified sampling. Retrieved November, 10:2012
26. Jarovsky A, Milo T, Novgorodov S, Tan W-C (2018) Goldrush: rule sharing system for fraud detection. *Proc VLDB Endow* 11(12):1998–2001
27. Khayyat Z, Ilyas IF, Jindal A, Madden S, Ouzzani M, Papotti P, Quiané-Ruiz J-A, Tang N, Yin S (2015) Bigdancing: a system for big data cleansing. In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp 1215–1230
28. Kohavi R, Longbotham R, Sommerfield D, Henne RM (2009) Controlled experiments on the web: survey and practical guide. *Data Min Knowl Disc* 18(1):140–181
29. Lai Z, Mo D, Wong WK, Xu Y, Miao D, Zhang D (2018) Robust discriminant regression for feature extraction. *IEEE Trans Cybern* 48(8):2472–2484
30. Liu B, Chiticariu L, Chu V, Jagadish H, Reiss F (2010) Refining information extraction rules using data provenance. *IEEE Data Eng Bull* 33(3):17–24
31. Liu Y-E, Mandel T, Brunskill E, Popovic Z (2014) Trading off scientific knowledge and user learning with multi-armed bandits. In: *EDM*, pp 161–168
32. Milo T, Novgorodov S, Tan W-C (2016) Rudolf: interactive rule refinement system for fraud detection. *Proc VLDB Endow* 9(13):1465–1468
33. Milo T, Novgorodov S, Tan W-C (2018) Interactive rule refinement for fraud detection. *EDBT*
34. Panahi F, Wu W, Doan A, Naughton JF (2017) Towards interactive debugging of rule-based entity matching. In: *EDBT*, pp. 354–365
35. Patel K, Drucker SM, Fogarty J, Kapoor A, Tan DS (2011) Using multiple models to understand data. In: *IJCAI proceedings-international joint conference on artificial intelligence*, vol 22, pp 1723
36. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp 1532–1543
37. Pham Q-K, Raschia G, Mouaddib N, Saint-Paul R, Benatallah B (2009) Time sequence summarization to scale up chronology-dependent applications. In: *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. ACM, 1137–1146
38. Ratner A, Bach SH, Ehrenberg H, Fries J, Wu S, Ré C (2017) Snorkel: Rapid training data creation with weak supervision. [arXiv preprint arXiv:1711.10160](https://arxiv.org/abs/1711.10160)
39. Ratner AJ, Bach SH, Ehrenberg HR, Ré C (2017) Snorkel: Fast training set generation for information extraction. In: *Proceedings of the 2017 ACM international conference on management of data*. ACM, pp 1683–1686
40. Rocchio JJ Relevance feedback in information retrieval. The SMART retrieval system: experiments in automatic document processing, pp 313–323
41. Russo D, Van Roy B, Kazerouni A, Osband I (2017) A tutorial on thompson sampling. [arXiv preprint arXiv:1707.02038](https://arxiv.org/abs/1707.02038)
42. Saint-Paul R, Raschia G, Mouaddib N (2005) General purpose database summarization. In: *Proceedings of the 31st international conference on Very large data bases*. VLDB endowment, pp 733–744
43. Stoffel F, Flekova L, Oelke D, Gurevych I, Keim DA (2015) Feature-based visual exploration of text classification. In: *Symposium on visualization in data science at IEEE VIS*
44. Sun C, Rampalli N, Yang F, Doan A (2014) Chimera: large-scale classification using machine learning, rules, and crowdsourcing. *VLDB Endow* 7(13):1529–1540
45. Tabebordbar A, Beheshti A, Benatallah B (2019) Conceptmap: a conceptual approach for formulating user preferences in large information spaces. In: *International conference on web information systems engineering*, pp 779–794. Springer
46. Veeramachaneni K, O'Reilly U-M, Taylor C (2014) Towards feature engineering at scale for data from massive open online courses. [arXiv preprint arXiv:1407.5238](https://arxiv.org/abs/1407.5238)
47. Vijayanarasimhan S, Grauman K (2014) Large-scale live active learning: training object detectors with crawled data and crowds. *Int J Comput Vision* 108(1–2):97–114
48. Volkovs M, Chiang F, Szlichta J, Miller RJ (2014) Continuous data cleaning. In: *2014 IEEE 30th international conference on data engineering (ICDE)*. IEEE, pp 244–255
49. Wiatowski T, Bölcskei H (2018) A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Trans Inf Theory* 64(3):1845–1866

50. Williams JJ, Kim J, Rafferty A, Maldonado S, Gajos KZ, Lasecki WS, Heffernan N (2016) Axis: Generating explanations at scale with learnersourcing and machine learning. In: ACM conference on learning@ Scale. ACM, pp 379–388
51. Xie J, Sun C, Yang F, Rampalli N (2014) Automatic rule coaching, Sept. 2. US Patent App. 14/475,470