Data-Scanner-4C: Format Inconsistency Identification by Means of Crowdsourcing

Shaochen Yu[1], **Lei Han**[1], **Marta Indulska**[2], **Shazia Sadiq**[1], **Gianluca Demartini**[1]

[1]School of ITEE; [2]School of Business
University of Queensland, Brisbane, Queensland, Australia
shaochen.yu@uq.net.au, tomhanlei@hotmail.com, m.indulska@business.uq.edu.au, shazia@itee.uq.edu.au, demartini@acm.org

Abstract

Format consistency is one of the most frequently appeared data quality issue that need to be addressed in data cleaning tasks. The existing (semi-) automated approaches either involving human input or not to deal with format inconsistencies are limited by means of lacking applicability and generalisability, because these approaches typically require writing regular expressions which would be difficult for non-experts. This paper proposes a novel human-machine hybrid system that we call "Data-Scanner-4C" to leverage crowdsourcing to address syntactic format inconsistencies in an effective and cost-efficient way. The proposed method incorporates rule-based learning approaches, and thus having experts writing regular expressions is no longer needed. We first ask crowd workers to select training examples through data selection and result validation. Then, we make use of a learning algorithm to infer the regular expression that works for the format consistency issues in a given structured dataset. In this way, we are able to apply the created regular expression to the entire dataset to find more consistency issues. Our Data-Scanner-4C system integrates crowdsourcing and format extraction techniques in a single workflow.

Introduction

Data quality issues may appear due to various reasons, such as outliers, duplicate data, and rule violations (Ilyas and Chu 2019). Data cleaning tasks are tedious and time-consuming, taking data workers up to 80% of overall time in a data analytics activity (Zhang, Zhang, and Yang 2003). In this paper, we focus on one of the most frequently appeared data quality issue – namely format inconsistency (Dasu and Johnson 2003). In practice, data may come from heterogeneous sources, and records in a dataset may be represented in various formats (Bleiholder and Naumann 2009). A well-formatted record generally follows structured formatting rules. For example, a correct date format can be "YYYY/MM/DD" and the format of a contact number can be "+(1)(XXX)XXX-XXX". This gives the opportunity to use regular expressions to match the types of data formats in a given dataset. However, to ensure high format identification effectiveness, we require to construct regular expressions and this is a hard and tedious job that requires a significant amount of time from data experts. Although there are some automated approaches to produce regular expressions (Bartoli et al. 2016; Brauer et al. 2011; Bartoli et al. 2014), these methods put a limited sight on the selection of training examples, which may affect the quality of generated regular expressions significantly. Cochran et al. (2015) proposed an approach called "Program Boosting" to program synthesis, which proves that crowd workers can improve the quality of regular expressions effectively in a human-machine hybrid system. However, "Program Boosting" requires domain experts to write some initial regular expressions and provide examples for golden sets.

In this paper, we propose a novel human-machine hybrid system, which we call "Data-Scanner-4C", to carry out data cleaning activities for tabular datasets. Our system leverages the scalability of crowdsourcing and asks crowd workers to select training samples for learning algorithms, which is much cheaper than recruiting domain experts. These samples are then validated by another group of crowd workers and those passing the validation are given to our learning algorithm as input. The learning algorithm can then infer regular expressions following a rule-based approach to identify more errors in the entire dataset. The main contributions of this paper are as follows:

- We introduce a human-machine hybrid workflow involving crowdsourcing and design the hybrid system named "Data-Scanner-4C" to identify format inconsistencies in a large structured datasets. Our system can automatically generate regular expressions to describe data each format. The created regular expressions are then used to identify more format inconsistency issues in the entire dataset.
- We design two crowdsourcing tasks, namely "data selection" and "result validation". By asking crowd workers to select records in the same format, we classify formats that are then validated by another group of crowd workers.

Crowdsourcing Tasks

"Data-Scanner-4C" consists of two main parts: (i) crowdsourcing tasks: selecting records by crowd workers as the training examples for an inference algorithm. (ii) format extraction: learning rules from examples and constructing regular expressions. It starts with inputting a dataset with format inconsistency issues into the "Data-Scanner-4C". The system will first post a crowdsourcing task named "data selection" to 20 crowd workers, asking each worker to select a

set of records sharing the same format. Then, the selected 20 sets of records will be randomly assigned to another group of 6 crowd workers. Each set need to be validated 3 times by different workers. Thus, there are 60 sets (20 sets $*$ 3 times) of records that need to be validated, and each worker is assigned 10 sets. These 6 crowd workers are asked to select records in the most dominant format for each set, discarding records in other formats. This crowdsourcing task is called "result validation", which aims to remove record sets that are incorrectly selected in the "data selection" task.

Data Selection

We designed a task for crowd workers to select records within a data format from a given dataset. We build an interface based on HTML5, CSS, and JavaScript that allows crowd workers to go through as many records as they want. We recruited 20 crowd workers for each dataset for the data selection task. They are first required to complete a questionnaire covering their basic information (e.g., age, occupation, qualifications, etc.). Next, they are asked to read a material that describes their task and how to use the interface we designed. It takes a crowd worker at least 30 seconds to read the material, and then this worker is allowed to go to the next page. A test is on the next page, where we give workers a dataset for testing, and only those workers who choose the correct results are allowed to enter the formal experiment. In the formal experiment, the crowd workers are asked to pick one data format based on their preferences and then select at least 5 records that are in the chosen format and have correct values. We collect 20 sets of records submitted by 20 crowd workers respectively as the result of the data selection task.

Result Validation

To reduce the number of errors incorrectly selected by workers in the data selection task, we designed a follow-up task called "result validation", which requires another group of crowd workers to validate the results of data selection. The input of the result validation task is 20 sets of records generated in the data selection. We recruited 6 workers for the result validation task. Like data selection, result validation starts from collecting basic information via a questionnaire. Then, we asked crowd workers to spend at least 30 seconds reading material about their task. In this task, we require crowd workers to tick all records in the most dominant data format in a set of records.

Experiments

"Data-Scanner-4C" aims to classify data formats and infer regular expressions for each data format in a given dataset. We set up an empirical study collecting training examples by means of crowdsourcing. Then, a rule-based algorithm infers regular expressions from these training examples. We use the standard F-measure to evaluate the quality of generated regular expressions comparing identified format issues with ground truth information on existing issues.

Dataset We use "Contact Number" datasets in our experiments. This is a synthetic dataset with synthetically injected errors, which is created by Parallel Data Generation Framework (PDGF). There are 8 types of data formats and 13000 records in total, while 6295 records have incorrect values (e.g., missing value, imprecise value).

Results

In table 1, we can observe a statistically significant improvement in accuracy after the validation of the "Contact Number" dataset (Mann-Whitney U test, $p < 0.05$). "Result validation" can effectively improve the quality of selected examples from "data selection" for this "Contact Number" dataset.

Experiment		Accuracy of Examples		
Task	Dataset	Mean	Median	Std
Data Selection	Contact Number	0.74	0.8	0.229
Result Validation	Contact Number	0.86	1.0	0.218

Table 1: Performance of Crowdsourcing

Table 2 illustrates the quality of generated regular expressions based on the selected examples, and the table shows that 8 data formats are inferred after the aggregation under the "Contact Number" dataset, 7 of them are correct formats, and 1 inferred format is imprecise. The regular expressions of "type 1", "type 2", "type 4", "type 5" and "type 7" can describe the corresponding format correctly. For "type 3" and "type 6", the generated regular expressions have a recall of 1.0, but also match some records with imprecise values. This is due to the fact that even though the selected examples have been validated, a few incorrect records remain. These incorrect records lead to incorrect learning of the rules.

Format	Example	Precision	Recall	F1-Score
type 1	+61-08-5104-3016	1.0	1.0	1.0
type 2	61-(07)-6413-8565	1.0	1.0	1.0
type 3	+61(08)40246347	0.837	1.0	0.911
type 4	61-02-3566-7449	1.0	1.0	1.0
type 5	+61-(03)-5484-6862	1.0	1.0	1.0
type 6	610413246019	0.814	1.0	0.898
type 7	61(02)15972549	1.0	1.0	1.0
imprecise	+61-06-23-1	0.0	0.0	0.0

Table 2: Contact Number (8 correct formats in the dataset)

Conclusions

The system proposed in this paper can effectively identify format inconsistencies in a structured dataset. The system can identify the presence of different data formats in a dataset and produce regular expressions to detect each of the formats. Therefore, format inconsistency issues can be identified when we apply the generated regular expressions to match records in the given dataset. "Data-Scanner-4C" integrates a hybrid human-machine design, acquires examples by means of crowdsourcing, and infers regular expressions based on selected examples. In our empirical study, we show that crowd workers are able to select useful examples in the data selection task. At the same time, the result validation tasks can effectively eliminate errors and improve the quality of selected examples.

References

Bartoli, A.; Davanzo, G.; De Lorenzo, A.; Medvet, E.; and Sorio, E. 2014. Automatic synthesis of regular expressions from examples. *Computer*, 47(12): 72–80.

Bartoli, A.; De Lorenzo, A.; Medvet, E.; and Tarlao, F. 2016. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5): 1217–1230.

Bleiholder, J.; and Naumann, F. 2009. Data Fusion. *ACM Comput. Surv.*, 41(1).

Brauer, F.; Rieger, R.; Mocan, A.; and Barczynski, W. M. 2011. Enabling information extraction by inference of regular expressions from sample entities. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, 1285–1294.

Cochran, R. A.; D'Antoni, L.; Livshits, B.; Molnar, D.; and Veanes, M. 2015. Program boosting: Program synthesis via crowd-sourcing. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 677–688.

Dasu, T.; and Johnson, T. 2003. *Exploratory data mining and data cleaning*. John Wiley & Sons.

Ilyas, I. F.; and Chu, X. 2019. *Data cleaning*. Morgan & Claypool.

Zhang, S.; Zhang, C.; and Yang, Q. 2003. Data preparation for data mining. *Applied artificial intelligence*, 17(5-6): 375–381.