Towards Finding Animal Replacement Methods

Nadine Dulisch and Brigitte Mathiak(✉)

GESIS - Leibniz Institute for the Social Sciences,
Unter Sachsenhausen 6-8, Cologne, Germany
{nadine.dulisch,brigitte.mathiak}@gesis.org

Abstract. Protecting animal rights and reducing animal suffering in experimentation is a globally recognized goal in science. Yet numbers have been rising, especially in basic research. While most scientists agree that they would prefer to use less invasive methods, studies have shown that current information systems are not equipped to support the search for alternative methods. In this paper, we outline our investigations into the problem. We look into supervised and semi-supervised methods and outline ways to remedy the problem. We learned that machine assisted methods can identify the documents in question, but they are not perfect yet and in particular the question about gathering sufficient training data is unsolved.

Keywords: Classification · Animal welfare

1 Introduction

Researchers from Life Sciences that contemplate to use animal testing are motivated by ethical, financial and often legal incentives to try and find alternatives that enable them to answer the same research questions, but with less animal involvement.

Unfortunately, current strategies for literature search do not support search for animal test alternatives very well. Dutch animal welfare officers have been asked for their strategies in finding alternative methods [2]. None found this task easy and reported that the most successful way of finding good alternatives was word of mouth.

When interviewing experts in finding such documents, it becomes clear that there are two criteria. *Similarity* is how close the document is to the experiment we want to replace. *Relevance* measures how likely it is that the document describes a method that causes less animal suffering. In order to give a comprehensive list of candidate documents, we need to take both criteria into account.

2 Related Work

Our attempt at solving the animal test replacement problem is not the first one. Go3R[1] is a semantic search engine based in PubMed and ToxNet[2] prioritizing

[1] http://www.gopubmed.org/web/go3r/.
[2] http://toxnet.nlm.nih.gov.

J. Kamps et al. (Eds.): TPDL 2017, LNCS 10450, pp. 595–598, 2017.
DOI: 10.1007/978-3-319-67008-9_51

3R and toxicology. While the toxicology use case is interesting and useful, the focus on recall makes it hard to handle basic research questions, which typically do not fit semantic categories as neatly. AltBib[3] is not an independent search engine, but rather suggests query term expansions that, among other factors, utilize the MeSH classification for animal testing alternatives. This classification is not systematically used to tag all methods that are developed as alternatives, but seems to focus on documents about animal testing alternatives on a meta level. In 2015 there were less than 3000 documents with that MeSH term.

Table 1. Overview over # of positive, negative and not classified instances for the different use cases.

Use case	Reference document (PMID)	Relevance			Similarity			Animal test		
		+	-	/	+	-	/	+	-	/
1	16192371	13	85	2	15	77	8	13	70	17
2	11932745	21	70	9	13	78	9	47	39	14
3	11489449	13	80	7	4	75	21	37	55	8

3 Corpus

To our knowledge, there is no corpus available to use as training data, a problem that has been plaguing Information Retrieval from the very beginning [1]. We structured our corpus around individual use cases, mimicking the application process for getting a permission to conduct a specific animal experiment. The starting point was a document describing an animal test (reference document), which was chosen by the domain expert. To search for possible alternative methods, we used the PubMed functionality to find similar documents based on substring similarity[4].

The first 100 hits of documents similar to the reference document were downloaded and then assessed by the domain expert according to criteria the expert set down beforehand. Additionally to the aforementioned dimensions of result similarity and replacement relevance, we also asked the expert to give us information on whether the document described animal experiments or not.

Non-classification occurred when there was not enough information to make a sensible decision, e.g. missing abstract, missing relevant information, non-available full text or, in rare cases, when the expert felt not knowledgeable enough about the domain to make a judgement call. In the following experiments, we only used the classified documents.

[3] http://toxnet.nlm.nih.gov/altbib.html.
[4] http://www.ncbi.nlm.nih.gov/books/NBK3827/#pubmedhelp.Computation_of_Similar_Articl.

For each document we collected the following metadata: title, abstract, PubMed Central ID, URLs, journal name, availability status of the full text and MeSH term information.

Table 1 gives an overview over our created use cases. The table shows the number of positive (+) and negative (−) instances included in the datasets. The datasets include only few positive instances, leading to strong bias in learning.

4 Experiments

4.1 Classification

What we are most interested in, is if trained algorithms are able to distinguish between positive and negative instances. In this experiment, we compared the prediction performance of standard data mining algorithms, which included J48 (C4.5 decision tree), JRIP (Propositional rule learner), SMO (Sequential minimal optimization), Naïve Bayes, Bayes Net and LWL(Locally weighted learning). We conducted the experiments applying the data mining software Weka[5] (version 3.6) and used Weka's implementation of the aforementioned algorithms. For classification we used Weka's "FilteredClassifier", applying the "StringToWordVector" filter to handle string attributes. This filter transforms string attributes into an attribute set that represents word occurrence information[6]. We used leave-one-out evaluation for all experiments, based on the original dataset. For the results see Table 2.

Table 2. Average F-Score over all three use cases, differentiated after algorithm and metric. Note that the F-Score is calculated from the point of view of the positive instances, therefore the expected value for random choice is very low due to the bias.

Target attribute	Unbalanced dataset					
	J48	JRIP	Naïve Bayes	Bayes Net	SMO	LWL
Relevance	0.51	0.36	0.44	0.66	0.52	0.42
Similarity	0.14	0.08	0.36	0.28	0.18	0.07
Animal test	0.79	0.87	0.91	0.87	0.94	0.83

We immediately discovered that the unbalancedness of the datasets, with as little as 4 positive examples were creating serious problems as especially relevant and similar documents were only rarely correctly classified (cf. Table 2). This is particularly devastating for similarity, where most results are worse than or close to random. Results for relevance are not ideal, but clearly better than random. Animal tests can be detected quite reliably, but positive and negative instances are much better distributed, as you can see in Table 1.

[5] http://www.cs.waikato.ac.nz/ml/weka/.

[6] http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html.

Table 3. F-Score for semi-supervised learning. Averaged over all use cases. Number in parentheses is the original value.

Target attribute	Naïve Bayes	SMO
Relevance	0.56 (0.44)	0.53 (0.52)
Similarity	0.38 (0.36)	0.43 (0.18)

Semi-supervised Learning. As discussed before, we had only comparably few labeled documents available, and in real life, we might have even less. What we have not leveraged so far is that we had a high number of unlabeled documents available to us that fit the general topic. Following the self-training methodology laid out by [3] we used a semi-supervised learning approach in which unlabeled documents were used to counteract the scarcity of training data.

The semi-supervised approach improves results for relevance compared to the original values for the unbalanced dataset. As Table 3 shows, the F-Score value for relevance increases for both top algorithms, but only moderately. The SMO F-Score for similarity, however, raises more significantly, which seems to indicate that the lack of training data impacted the classifiers ability to successfully predict similarity.

5 Conclusions and Future Work

While we do not have a workable prototype for up-ranking animal replacement methods yet, we believe we have made important inroads and identified some roadblocks. On the bright side, we have shown that relevant documents can be found with machine learning given enough training data. Methods to reduce the need for training data have been tested and were found to be successful.

A more direct approach would be to use un-supervised methods of finding similar documents. Bibliometric methods seem hopeful, but positive and negative effects overlap and cancel each other out. We tried using classifiers across use cases, but without any improvements. On all fronts, it becomes clear that more training data is needed.

References

1. Jones, K.S., van Rijsbergen, C.J.: Information retrieval test collections. J. Documentation **32**(1), 59–75 (1976)
2. van Luijk, J., Cuijpers, Y., van der Vaart, L., de Roo, T.C., Leenaars, M., Ritskes-Hoitinga, M.: Assessing the application of the 3rs: a survey among animal welfare officers in The Netherlands. Lab. Anim. **47**(3), 210–219 (2013)
3. Zhu, X.: Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)

Environmental Monitoring of Libraries with MonTreAL

Marcel Großmann[1(✉)], Steffen Illig[2], and Cornelius Matějka[1]

[1] Computer Networks Group, University of Bamberg, 96047 Bamberg, Germany
marcel.grossmann@uni-bamberg.de,
cornelius-lucian.matejka@stud.uni-bamberg.de
[2] University Library of Bamberg, 96047 Bamberg, Germany
steffen.illig@uni-bamberg.de

Abstract. An ever-increasing amount of devices connected over the Internet pave the road towards the realization of the 'Internet of Things' (IoT) idea. With IoT, endangered infrastructures can easily be enriched with low-cost, energy-efficient monitoring solutions, thus alerting is possible before severe damage occurs. We developed a library wide humidity and temperature monitoring framework MonTreAL, which runs on commodity single board computers. In addition, our primary objectives are to enable flexible data collection among a computing cluster by migrating virtualization approaches of data centers to IoT infrastructures.

We evaluate our prototype of the system MonTreAL at the University Library of Bamberg by collecting temperature and humidity data.

Keywords: IoT · Single board computer · Container · Virtualization · Monitoring · ARM · Sensor

1 Introduction

The environment in archives and libraries should be maintained within specified tolerances in order to guarantee cultural heritage, e.g. books, magazines, electronic media to be conserved and prevented from serious damage [1]. Our university library maintains several depots to store stocks and equipment, which are partly hosted in buildings under monumental protection. Occurring problems range from simple things, such as broken lights, to more problematic ones, like water damages which could lead to mold formation. In that case a regular supervision of a depot is absolutely essential. To overcome severe issues and to avoid health risks while doing manual data collection, the endangered depot was equipped with temperature and humidity sensors attached to a single board computer (SBC). The relation of both, humidity and temperature, is absolutely necessary for an appropriate climate control in depots [1].

These requirements lead to the idea of creating a distributed sensor environment that covers more areas within a single depot and can be distributed over multiple depots. The data captured by those sensors should then be aggregated

J. Kamps et al. (Eds.): TPDL 2017, LNCS 10450, pp. 599–602, 2017.
DOI: 10.1007/978-3-319-67008-9_52

to a single point, where it can be processed and used, e.g., for creating graphs and analyzing trends. Also the captured data should be made accessible via a web interface once the sensor network is fully operational.

2 IoT Monitoring Solutions in Other Areas of Application

Already existing monitoring solutions are provided in several other areas. For instance, Lewis *et al.* [2] propose an environmental monitoring in a quality-controlled calibration laboratory. In the e-health sector, Jassas *et al.* [3] implemented a prototype with e-health sensors attached to the RPi. Medical sensors measure patients' physical parameters and the RPi collects and transfers them to the cloud environment, as real-time data. However, the former prototypes miss an architectural concept and are not considering the privacy of sensor data. Instead, they send, process, and store the data on remote cloud servers. Only Hentschel *et al.* [4] introduce a concept with local supernodes, which are sensor enhanced RPis. In their model the nodes behave autonomously and carry out simple tasks like processing data or communication with other devices. Unfortunately, each supernode needs to be managed and thus, attaching new devices increases maintenance complexity.

In contrast to those approaches, we focus on transferring data center technology to energy-efficient devices, to enable an easily updatable, scalable, and manageable framework. Moreover, obtained sensor data are processed locally and are saved on the RPis without the need to send them to cloud services.

3 Realisation of MonTreAL

We built MonTreAL (Monitoring Treasures of all libraries) to measure temperature and humidity in our library. Due to our initial goal to perform the measurements with SBCs, MonTreAL takes advantage of Docker[1] and Docker Swarm as underlying technologies.

The system consists of two main components as depicted in Fig. 1. Devices that act as *Workers* are connected to sensors and are distributed within a building. Their tasks is a regular interaction with their environment to gather and process information. Therefore, a *sensor* container running an application to interact with the sensors is responsible to correctly process and send the gathered data to a messaging *queue* container. *Sensor* containers are able to address several different sensor types, which are connected via *GPIO* or *USB* interfaces to a SBC and will port the diversity of manufacturers' dependent data formats into a more uniform format for further processing. Every *sensor* container is configured to add a unique ID to every package it sends to the queue to distinguish and match the data with their origin later. The sensors we are using are simple temperature and humidity sensors like the *DHT-22*[2], which is directly

[1] https://www.docker.com/.
[2] https://www.adafruit.com/product/385.

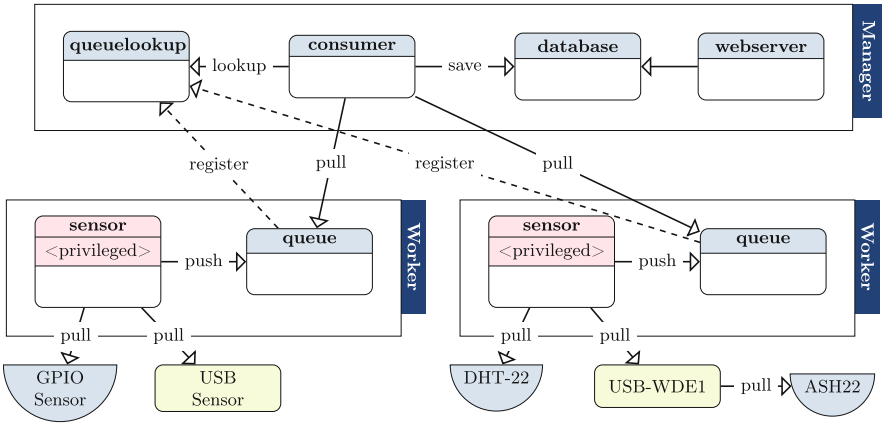

Fig. 1. Architectural overview of MonTreAL

connected to the GPIO interface of the RPi. Or, a more complex sensor system, which consists of a receiver (*USB-WDE1*[3]) that is connected via USB and several autonomous sensor devices (*ASH22*[4]), which regularly send their data to the receiver by radio.

The second main component of MonTreAL is one (or possibly more) device(s) representing the *Manager*. Its task is to accumulate and store the gathered data and provide interfaces to allow user interaction. To fetch the gathered sensor data from the queues, a *consumer* container running a dedicated consumer implementation periodically requests the addresses of all running queues from a *queuelookup* container and fetches all available data. The consumer then stores the data in a *database* container for long-term storage. MonTreAL also provides the user with the possibility to view the gathered data in a suitable way. It runs a simple *webserver* container, which allows the user to view gathered data of every single sensor in a graph and to filter it by date. Furthermore, the front end indicates, which sensors are currently online or offline, or when problems with devices occur, e.g., inoperable devices, broken sensors, empty batteries, etc.

To summarize, MonTreAL allows to set up a distributed system of IoT devices equipped with specific sensors and to gather, accumulate and store the collected data without the need of a powerful backend. It also features a simple front end to view the data. MonTreAL relies on the existence of a network infrastructure (LAN, WLAN) and operates with only a few needed configuration options. We constantly evaluate our prototype at the University Library of Bamberg by collecting temperature and humidity data with the capacity of three RPis.

[3] https://www.elv.de/output/controller.aspx?cid=74&detail=10&detail2=44549.
[4] https://www.elv.de/elv-funk-aussensensor-ash-2200-fuer-z-b-usb-wde-1-ipwe-1.html.

4 Conclusion and Future Work

Our IoT prototype MonTreAL implements a distributed temperature and humidity monitoring framework, which is delivered by Docker images for several Docker supported SBC architectures like ARM, AARCH64 and x86_64. It is made publicly available at the Github[5] repository *unibaktr/MonTreAL* containing the source code. By defining those images, virtualization is achieved at IoT level and by these means MonTreAL offers an easily manageable solution. Furthermore, deploying the system to a container enabled cluster achieves reliability and resilience through the underlying Docker Swarm paradigm. Even, if services are unavailable or are scaling up or down, collected data remain in the distributed message queue and is processed as soon as the consumer recognizes the queue again. Data collected from message queues are processed and stored in a database for long term evaluations. However, if sensors, services, the underlying network, or in the worst case SBCs fail, the consumer notifies the user by alerts.

By now, the RPis, more precisely the plugged in SD cards, provide a bottleneck for MonTreAL. Their limited lifetime is evoked by a relatively small number of write cycles and stands in contradiction to a persistent storage solution. This might be a less relevant problem for a stateless container that only produces runtime data. However, for the database we plan to evaluate the possibility to persist data on a reliable storage solution, which is attached to the database container. Moreover, MonTreAL currently supports only one depot with several sensors attached to multiple RPis. In the future, more containers should enhance the framework to remotely manage multiple depots within one web service.

Acknowledgement. The authors would like to thank the Hypriot team (https://blog.hypriot.com/crew/) for their great effort to port Docker to ARM platforms and for providing Hypriot OS.

References

1. Glauert, M.: Klimaregulierung in Bibliotheksmagazinen. In: Hauke, P., Werner, K.U. (eds.) Bibliotheken bauen und ausstatten. Institut fü Bibliotheks- und Informationswissenschaft (2009). http://edoc.hu-berlin.de/docviews/abstract.php?id=30204
2. Lewis, A., Campbell, M., Stavroulakis, P.: Performance evaluation of a cheap, open source, digital environmental monitor based on the Raspberry pi. Measurement 87, 228–235 (2016). http://www.sciencedirect.com/science/article/pii/S0263224116001871
3. Jassas, M.S., Qasem, A.A., Mahmoud, Q.H.: A smart system connecting e-Health sensors and the cloud. In: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 712–716, May 2015
4. Hentschel, K., Jacob, D., Singer, J., Chalmers, M.: Supersensors: Raspberry pi devices for smart campus infrastructure. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 58–62, August 2016

[5] https://github.com.