Using Text-Based and Example-Based Querying to Improve Usage of IoT Sensor Data in Smart Buildings

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Andrew Villca-Rocha

Spring, 2020.

Technical Project Team Members

Andrew Villca-Rocha

Max Zheng

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Hongning Wang, Department of Computer Science

Using Text-Based and Example-Based Querying to Improve Usage of IoT Sensor Data in Smart Buildings

Andrew Villca-Rocha
Department of Computer Science
University of Virginia
Charlottesville, Virginia, United States

ABSTRACT

As IoT sensor data use becomes more prevalent in smart buildings, building managers are left to search for data themselves by using dashboard metrics. This paper proposes an improvement to current building management systems by using text-based and example-based querying. The system accomplishes this by utilizing Information Retrieval techniques and Dynamic Time Warping algorithms. Through these two types of querying the user is able to better express their intentions in their searches which leads to an overall productivity improvement to their workflow.

ACM Reference Format:
Andrew Villca-Rocha. 2021. Using Text-Based and Example-Based Querying to Improve Usage of IoT Sensor Data in Smart Buildings. In . ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

With the increasing use of IoT devices in smart buildings, comes a need for software that is able to retrieve this information efficiently for building managers. This is especially important because building management such as power management and energy efficiency need effective methods of using their vast IoT sensor data.

Smart buildings are buildings that incorporate IoT devices into their infrastructure to achieve some efficiency. Improvements to infrastructure could also include increasing productivity to the management of its facilities[2]. Most research has been limited to integrating various facilities systems such as heating, ventilation, and air conditioning. Currently, building managers are limited to using dashboards with metrics for various sensors to determine the productivity of the smart building. This process is limited because it relies on the users to manually search rather than letting the system conduct the search. This technical report aims at improving the retrieval of IoT sensor data by providing new methods. These new methods for building management systems take the form of text-based and example-based querying.

2 RELATED WORK

As described earlier, current solutions fail to provide efficient methods for retrieving information from IoT sensors. These systems do little to innovate on the methods for retrieving data. Current solutions involve a dashboard where users have to manually inspect sensors to retrieve the information they are looking for. This is why I propose adding text-based and example-based querying.

Text-based querying is common in search engines. Sites like Google and DuckDuckGo dedicate their business to retrieving relevant documents based on a text query. Current practice involves tokenizing a query into segments. Each segment represents a semantic subsection of the overall query. This process is conducted to understand the user's intention when performing a query. Most research around query tokenization is limited to retrieving webpages. In this case the internet query deviates from queries on IoT sensors. Query tokenization in the context of the internet treats each segmentation as a term that needs to be matched in a document's metadata. In IoT sensors, each segmentation isn't bound to a document's metadata but also to the time-series data that is associated with the document.

There has also been previous work towards example-based querying. Query-by-sketch allows users to draw time-series shapes and the system will produce similar windows of time-series data [3]. This method relies on the user's ability to produce rich patterns for search. Instead of relying on the user to retrieve relevant data, I propose improving this by having the user select pre-existing examples from the time-series data. This way the examples the user produces are realistic and can better match existing time-series data.

In order to accomplish example-based querying, we need methods for extracting characteristics from time-series data. Using these characteristics, we can then compare time-series data for relevance determination. This process is known as feature selection. Current methods involve dividing time-series into subsequences and extracting basic statistics like slope, mean, and variance [1] . This method suffers from the simplicity of its characteristics. Since slope can only distinguish between different linear trends, other trends like exponential or quadratic are not properly described.

Other methods include a bag-of-patterns approach. In this approach, a bag-of-patterns is generated by dividing time-series data into windows and then using Fourier transforms to approximate the shape of each window. The Fourier values are then used to create classes amongst the windows and the bag-of-patterns is generated by looking at the unigrams and bigrams from each window in the time-series-data [5].

Another more efficient method is proposed by using Dynamic Time Warping (DTW). Through an exhaustive literature search, researchers found that DTW outperforms other distance measures by a statistically significant amount. DTW is an algorithm that measures similarity between time-series data that may vary in

, ,

speed. This is important for IoT data as different sensors may take measurements at different rates. Additionally, time-warping allows us to conduct one-to-many and many-to-one matches which can be useful in comparing data points that occur at different indexes in the time-series data. This is demonstrated in Figure 1. Finally, efforts have been made to optimize this method for the case of time-series data. This provides a fast implementation that can be easily utilized within our system [4].

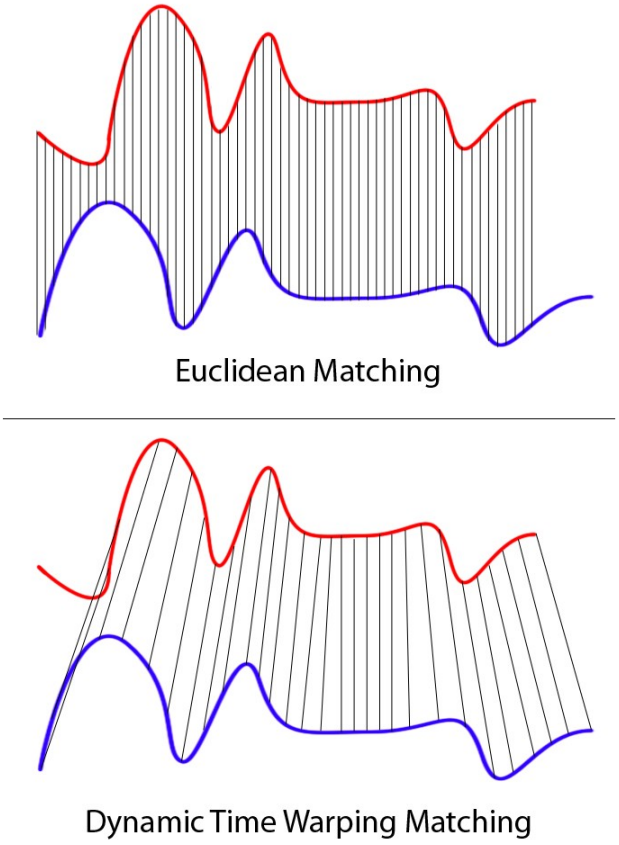

Figure 1: Visual Representation of Euclidean (one-to-one) Matching vs. DTW (one-to-many/many-to-one) Matching

3 SYSTEM DESIGN

The system provides two ways of querying for information: text-based and example-based querying. The solution is then provided in a web-app with a distributed system design. This solution also assumes data is static.

3.1 Text-based Querying

Text-based querying is a common method for retrieving information. Our system allows for metadata querying which is common in search engines like Google. This type of querying can also be used to search over the time-series data. Queries like "temperature > 65" can be used to search through temperature data that exceeds the value of 65. This is accomplished by translating text queries into queries that can be issued to the database system. See Figure 2 for an example of this translation. Even though this provides searching for time-series data, it does not provide the user the flexibility needed in their searching.

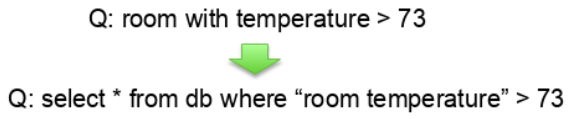

Figure 2: Example of text-based query translation to database query

3.2 Example-based Querying

Text-based querying is limited in its lack of quality representation of the user's intent. In order to complement text-based querying another method must be proposed to better retrieve information related to the time-series data. One way of accomplishing this is with example-based querying.

Example-based querying involves using an example of a graphical trend in time series data to retrieve other time series data from other sensors that match the trend provided. For example, a user can select a portion of continuous time-series data from a sensor that shows the temperature linearly decreasing. The system then uses this selection and retrieves time-series data from other sensors that show a linear decrease similar to the one provided by the user. This method is not limited to only linear trends but allows for a comparison with a variety of graphical trends. This improves on the text-based querying because the system is no longer bounded by categorical methods. Instead it gives the user more flexibility in determining what data they want to retrieve from the database.

The system designed is able to conduct example-based querying by gathering descriptive information about the user selected time-series data. The descriptive information is gathered by utilizing DTW. As proposed in the paper [4], DTW was optimized by utilizing EBSM. This meant significant offline preprocessing computations were conducted. This is fine as earlier we assumed that data associated to these sensors are static. To conduct the search for similar trends, a window is slid across time-series data and the data in that window is compared to the user-selected trend. The most similar results are then retrieved from the database.

3.3 Query Pipeline

In order to systematically parse a text-query provided by the user, a pipeline was created. This pipeline consists of operations conducted on the original query.

A query is first passed into the spell check. Each token in the query is checked for spelling mistakes. If a spelling mistake is found, then the token is corrected. This is accomplished by using the NLTK package. Now that the query is sanitized, a cache check is performed. If the query is in the cache, then the pipeline is halted and the results are returned to the user. Else, the query is issued to the database. If the database does not retrieve any relevant results then

the query is passed onto the next step. The final step is a last effort at retrieving results. The query is expanded by using a Wordnet lexical database. This lexicon contains semantic relations between words. For instance, the word "temperature" can get expanded to other semantically similar words like "heat" or "cool". By expanding the initial query we are able to conduct a more extensive search of our database. Any results acquired from this pipeline is then used to update the cache for future lookups. This process is represented graphically by Figure 3.

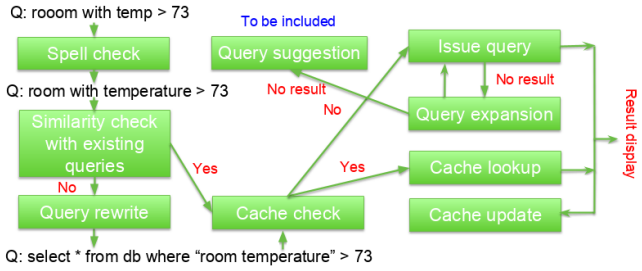

Figure 3: Overview of the query pipeline

3.4 System Architecture

The text-based querying and example-based querying solutions described earlier are housed in a search engine web app. This web app is designed using Docker containers. The web app is split into 6 containers: front-end, API, Database, Microservices, Event, and DTW.

The front-end does not deviate from common practice. Its sole purpose is to serve web pages to the user and to issue any network requests to the API for additional work to be conducted. These webpages will accept queries provided by the user and display time series data associated with retrieved sensors.

The API in my design acts as a mediator between the front-end and the other docker containers. Depending on the operation needed, the API will either conduct some light-weight work or delegate the work to other containers that are more equipped. Additionally, the API handles the search engine pipeline where it pre-processes the query provided by the user and applies pipeline operations to get more meaning from the user's query.

The database houses the dataset used for this project (described in Section 3.5). Additionally, it houses the cache used to speed up commonly issued queries. Since we assume the sensor metadata in our system is static, we are able to store the results of previous queries. When a user issues a query that is the same as one in our cache we are able to retrieve the results without having to execute the query pipeline.

The microservices container handles computationally intensive work. This container houses the WordNet expansion used in the query pipeline.

The Event and DTW both handle the example-based querying functionality. The DTW container conducts the Dynamic Time Warping computations (described in Section 3.2). Since the DTW implementation used is written in C, the Event container is created to write network communication between the DTW and the API in Python.

3.5 Datasets

In order to test the system's capabilities of retrieving information, a dataset was acquired from an existing building's sensor data. The data used in this project was obtained from the University of Virginia Building Management Group. This dataset contains time series data from 2858 different sensors in buildings on the University of Virginia's campus. The dataset also contains data from 34 different sensor types. There are a total of 21,909,993 time-series data points in the entire system. These data points range from 6/1/2013 03:00:10 to 10/12/2014 17:38:05. The data from each sensor is also labelled with metadata in the form of descriptive text attributes. Some example attributes include "occupancy sensor" and "room-id:386". These attributes describe an occupancy sensor that exists in room 386.

4 PROCEDURE & RESULTS

This section showcases how this system provides users with new methods for retrieving IoT sensor data. To do this I will walk through ways a user can use the system and afterwards I will discuss design criticisms provided by my peers from the user study.

4.1 Showcase

A user can directly type a query into the search box to search for sensors. The webpage serves the sensors that are relevant to the user's query. The time series data for each sensor retrieved is also displayed. If the user chooses to more closely inspect a retrieved sensor, he can expand each sensor to display the attributes associated with it and a more detailed visual of the time series data.

In the example provided by Figure 4, a user queries "temperature > 75". The system then retrieves temperature sensors whose values exceed 75.

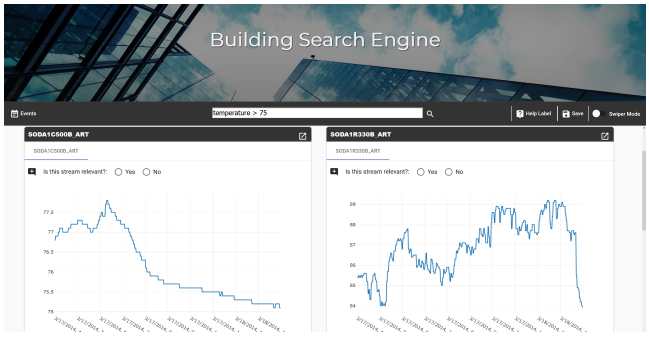

Figure 4: Screenshot of search engine returning results for "temperature > 75" query

If a user wishes to conduct an event based query, the user can select a window from the sensor's time series data and then the system will create an "event" for that window. The event is labelled for future querying. This event can then be queried by the search box or by using the left menu labelled "events". Once the event is queried, the user is provided time-series data that resembles the event they selected.

Figure 5 shows the results of an event query. The left time-series data is the event selected by the user and the right time-series data is a sensor who experiences a similar event.

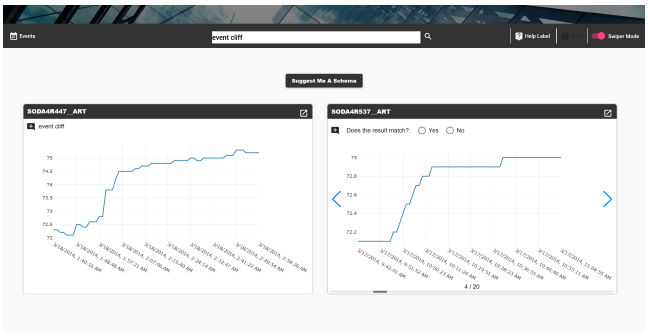

Figure 5: Screenshot of search engine returning results for event query

For both the text-based and example-based queries. The user is able to provide feedback about the relevancy of retrieved sensor data. This feedback can be used to further evaluate the system and to improve any hyper parameter tweaking.

4.2 Peer User Study

A small user study was conducted. Students of my technical advisor were contacted to test out our system and provide any comments and recommendations for improvement. The user study consisted of two parts: an exploration of text-based querying and event-based querying.

The text-based querying study involved users using their intuition to issue queries in order to solve building management problems. An example problem statement is the following: "I am interested in sensors that detect the temperature of rooms. Can you provide a list of sensor names that are associated with the temperature of rooms?". Users were tasked to determine what keywords to query for in order to find the names of temperature sensors for rooms. The second task given to users was to validate that an event occurred within the building. An example of this was "Room 500 was reported to be too cold. Can you confirm this?". Again like the previous tasks, users had to use their intuition to conduct queries into the database.

The event-based querying study was more simple. Users in this study were provided with events already queried into the system. Users simply had to evaluate whether or not the provided time series sub-sequences "matched" the even sub-sequence. This aimed at evaluating the event matching algorithm (DTW).

Here I will highlight a few findings from this user study. The text-based querying study exposed a flaw in the study itself. Users were tasked with taking the role of a building manager. This meant users had to be comfortable with the nomenclature of various sensors in the system to be able to issue queries. Thus, many users felt confused as to how to determine if a sensor was relevant to the query itself. Additionally, users struggled with validating the occurrence of events. 55.6% of users claimed the event did not occur and 44.4% claimed the event did occur. This result hints that users were guessing which further confirms the confusion of users. However, this confusion is stemmed from students taking on the role of a building manager. This confusion can be mitigated if the users were building managers. This way they would understand the naming convention.

The event-based querying study went more smoothly because graphical trends were intuitive to the students. Results showed that the event matching algorithm employed by the system was working very well. However when results from the algorithm began to deviate from the event, users struggled with determining when an event no longer matches what was provided by the system. An example of this ambiguous case is given by Figure 6. The event is an oscillating increasing trend. The retrieved sensor data exhibits oscillating trends but some would argue it deviates from the event.

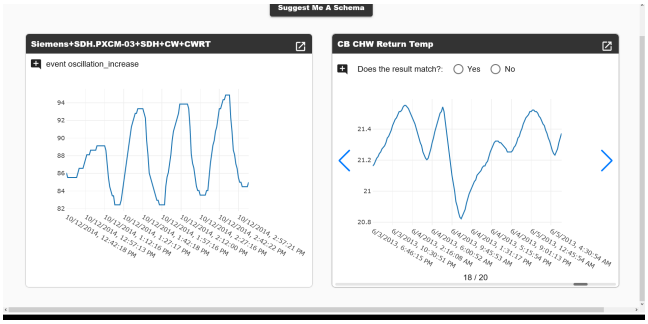

Figure 6: Screenshot of ambiguous event query match

5 CONCLUSION

In this technical project, we designed a system which is able to retrieve IoT time-series data by utilizing both text-based and event-based querying. This was accomplished by utilizing information retrieval techniques in the form of a query pipeline and by using a DTW-based algorithm for event-based querying. The system was showcased to demonstrate its ability to retrieve information from IoT sensor data. Additionally, a small user study was conducted. Findings showed that users without building management expertise struggled using the system. However, our stakeholders are building managers who presumably know the nomenclature of the system. The user study also showed that the event matching algorithm retrieved high quality matches.

6 FUTURE WORK

In this section I will suggest possible improvements on my solution. First, we assume the metadata associated with every sensor is static. This assumption does not hold in the real world as sensors are re-purposed or modified to fit a different use, thus making the metadata dynamic. Additionally, the time-series data in building management systems is handled in real-time. Future solutions should then assume the metadata to be dynamic. This will affect the cache system in our database. This will also affect the DTW implementation because it assumes the data is offline and static in order to compute preprocess computations to speed up the process.

Second, the user study showed that there was a learning curve in understanding the nomenclature of the system. This should be

addressed by introducing tools to allow the user to explore the sensors and their naming schema.

REFERENCES

[1] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. 2013. A Bag-of-Features Framework to Classify Time Series. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 11 (Nov. 2013), 2796–2802. https://doi.org/10.1109/TPAMI.2013.72

[2] B. M. Flax. 1991. Intelligent Buildings. *Comm. Mag.* 29, 4 (April 1991), 24–27. https://doi.org/10.1109/35.76555

[3] Miro Mannino and Azza Abouzied. 2018. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 388, 13 pages. https://doi.org/10.1145/3173574.3173962

[4] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. Association for Computing Machinery, New York, NY, USA, 262–270. https://doi.org/10.1145/2339530.2339576

[5] Patrick Schäfer and Ulf Leser. 2017. Fast and Accurate Time Series Classification with WEASEL. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. ACM, New York, NY, USA, 637–646. https://doi.org/10.1145/3132847.3132980