Towards Semantic Search in Building Sensor Data

Andrew Villca-Rocha[1], Max Zheng[1], Chengzhu Duan[2], Hongning Wang[1]
[1]University of Virginia, Charlottesville, VA 22901
[2]University of California San Diego, San Diego, CA 92093
{av3dj,zz5ra,hw5x}@virginia.edu,c3duan@ucsd.edu

ABSTRACT

This paper presents a search engine system for sensor time series data and metadata in the context of building management. It takes natural language queries as input and retrieves sensor time series data, ranks them with respect to their relevance to a given query, and visualizes the results as graphs. In addition, the system allows users to interact with the search results: they can define events of interest in the visualized results and search across sensor data for time series with similar shape, i.e. the search by example scheme. We leverage both a feature based cosine similarity model and DTW to find similar time series and rank them by relevance. Our quantitative evaluations and user studies demonstrate the value of this system for managing building sensor data.

CCS CONCEPTS

• **Information systems** → Database query processing; • **Computer systems organization** → *Sensor networks.*

KEYWORDS

Building search engine, natural language query, query by example

ACM Reference Format:
Andrew Villca-Rocha[1], Max Zheng[1], Chengzhu Duan[2], Hongning Wang[1]. 2021. Towards Semantic Search in Building Sensor Data. In *The 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '21), November 17–18, 2021, Coimbra, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3486611.3486647

1 INTRODUCTION

The internet-of-things has ushered in a new wave of "intelligent" buildings. They use sensor data to make decisions on aspects of building management such as power consumption and energy efficiency. While a multitude of research in the realm has been focused on the algorithmic advancement for monitoring, controlling, and diagnosing building systems, little work has been dedicated to effective and efficient handling of the data generated by those systems. Currently, team members have to use dashboards with a fixed set of predefined functionalities and data accessing scheme (e.g., SQL-like queries). This inevitably limits their productivity and cannot fully unleash the potential of intelligent building management.

In this work, we aim to increase the productivity of facilities management by enabling building managers, who are usually not equipped with programming skills, to directly interact with the sensor data via natural language and time series queries. Due to the domain-specific and often esoteric vocabulary used to name sensors in the context of building management, a system allowing natural language searches must be able to effectively derive correct semantic meanings from the input query terms. For example, to retrieve results for the query "overheated rooms on the second floor", the system needs to map "overheated room" to the room temperature sensor in each room and "second floor" to the location attribute of a room. Furthermore, it also needs to correctly understand the required condition on each attribute. For instance, in the aforementioned example query, returned room(s)' temperature should be well above their setpoints, and must also be located on the second floor of a target building. Only at this level of query understanding can the system match the input query with the stored sensor data and evaluate their relevance for result ranking and display.

Leveraging the advancement in modern information retrieval techniques, we develop a simple set of query syntax for query parsing, and use both lexicon-based [11] and pre-trained word embedding [3] techniques for input query expansion. We map parsed queries to a structured SQL format to search against the metadata of sensor points for retrieval and ranking purposes. We shall note that we assume in this work the sensor metadata is provided, and thus the extraction as well as the representation of metadata are out of the scope of this work. Even when such information is not available, there are already studies for automated mapping of sensor metadata [2, 6, 7], which can be employed as a pre-processing component of our system.

In addition, our system features a "drag&search" function, which allows users to select a subsequence of a presented time series as *events* of their interest and then search against the other time series data for similar events, i.e., query by example. A rich set of statistical features, including slope, skewness, entropy, and etc. [10], are used to depict the shape patterns of user-selected events and locate similar subsequences in the entire building sensor database. As the length of user-selected subsequence is indeterminate beforehand, query-time feature construction and similarity matching will be extremely expensive and slow. In order to improve the search efficiency, we pre-compute and store the feature vectors offline, and only match and compute similarity against subsequences of the same length of the original subsequence. We also use Dynamic Time Warping (DTW) [12] as another way to quickly find subsequences that match the user's selected time series segment.

To evaluate the effectiveness of our system, we deploy the system over a large collection of sensor data collected from a campus office building, with more than 3,300 sensor points in a period of 498 days. There are more than 21 million data points of sensor readings indexed on this data set. Both quantitative evaluations and user studies demonstrate the system's utility in managing building sensor data, especially its flexibility and generality in satisfying users' information needs in building management.

2 RELATED WORK

Supporting natural language queries in structured datasets has been studied in the database and information retrieval communities for decades [13]. The key research challenges lie in query parsing and mapping to predefined data schema. Early-stage solutions depend on hand-crafted rules tailored to each individual database, which are hard to generalize. Recent developments focus on learning-base solutions to automate the process. For example, Tan and Peng [16] assume queries are generated from single or multi-word semantic concepts and these concepts are independent and identically distributed in queries and external text corpus. Once a query is segmented, it will be mapped to a SQL query with respect to the underlying database schema [8]. To address vocabulary mismatch, e.g., keywords in a user query do not match any entry in the database, query expansion is often needed [14].

However, natural language queries cannot fully describe the curvature of a desired time series sequence. Query-by-example allows users to select significant subsequences from existing time series data and search for similar patterns across other time series sources. Hochheiser and Shneiderman [5] describe such a system that allows users to visually select rectangular portions of a time series, which are then used to return similar shaped subsequences of other time series in their data set. The key in query-by-example is the representation of time series data. Baydogan et al. [1] develop a bag-of-features representation. They divide time series data into subsequences to account for local patterns, including the slope of the fitted regression line, mean and variance. We adopt this solution in our system for time series similarities.

3 THE SEARCH SYSTEM

A search engine is expected to return relevant results given a user query. In the context of searching across sensor data, results are temporal streams of real values generated by sensors monitoring a physical environment. Similar to processing of textual data, modeling and indexing time series data are essential for supporting any useful search functionalities. In this work, we take advantage of existing sensor metadata – a short text string comprised of several abbreviations that encode the sensor type, location, relationships with other points – to support natural language based queries. We also index sensor readings with a set of statistical features to achieve the goal of search by example.

Figure 1 provides an overview of our system architecture. User queries are sent to the backend, segmented and translated into SQL statements to retrieve relevant time series data. User queries are automatically classified into one of three types: 1) keyword queries, e.g., "room temperature", which are executed by matching against the name entries in database schema; 2) conditional queries, e.g., "room temperature above 78", which are parsed with respect to a set of query syntax to realize user-specified filtering conditions; and 3) event queries, e.g., "fluctuating room temperature", which specify a shape-matching template for retrieval in time series data. The retrieved results are first ranked and then sent back to the frontend and visualized as line charts for display.

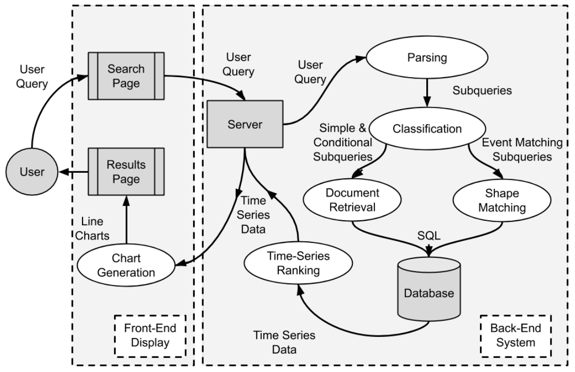

Figure 1: System diagram for the proposed building sensor data search engine.

Our system indexes the sensor points using a relational database. All attributes defined in the schema, e.g., site, room id, and equipment type, are thus directly searchable. To enable flexible natural language search, we also create a bag-of-words representation for sensor points. The text content of a sensor's attribute names is tokenized and encoded into n-grams [9]. To support keyword search, all attribute values are removed, and the terms in attribute names are expanded using WordNet [11] to improve the coverage of indexed terms. For example, for a sensor with the attribute texts "*site:SDH, room-id:282, supply fan-id:1*", we create its bag-of-words representation as ["*site*", "*building*", "*room*", "*office*", "*supply-fan*"]. In addition to the text information, each sensor point is also associated with its time series readings, which are represented and indexed by a set of statistical features.

Based on the aforementioned data processing and indexing procedures, a sensor point in our system is represented in three different forms: 1) a structured form based on the database schema, which directly supports our keyword and conditional queries; 2) a semi-structured form based on the parsed attribute names, which supports our keyword queries; and 3) an unstructured form based on the statistical features extracted from sensor time series readings, which supports our event queries. Next, we will discuss our detailed solutions for handling natural language queries and shape-based event queries in detail.

3.1 Query by Natural Language

In our system, a user query may consist of natural language search terms, relational operators, as well as predefined keywords. The system supports searching across sensor points by attributes or point names, where users can place constraints on their search results by using conditional operators to define numeric constraints for retrieval, and/or grouping search results by attributes of interest. Supported natural language queries can be divided into three categories: keyword queries, conditional queries, and event queries. Users can also use the "and"/"or" keywords to chain combinations of these three query types to express their complex search intents.

Keyword Queries. Keyword queries are the least complex, and consist exclusively of search terms, such as "temperature" or "supply fan". Such queries are evaluated by comparing their tokenized query terms against each indexed sensor point's attribute and n-gram

Table 1: Supported query syntax for conditional queries.

Keyword	Usage	Example
<, ≤, >, ≥, =	Specifies data constraint on relevant sensors	"temp > 75"
and/or	Returns sensor data from multiple queries	"temp and occupancy"
by [attr]	Groups results based on specified attribute	"temp by room"
by [attr] = [val]	Groups results based on specified attribute and value	"temp by room = 123"

text representation. Due to the domain-specific nature of most attribute terms, query expansion is needed to increase the coverage of searchable terms. To give user's original query higher priority, query expansion is only executed when no result can be retrieved for the user's original query. The BM25 ranking function [9], which is the most effective keyword-based ranking method, is used to measure the relevance of matched sensor points for result ranking. In addition, we also used a pretrained sentence encoder [3] to embed the historical queries. If the expanded query still cannot match any result in the database, we will search against those historical queries in this embedding space and return their associated search results (i.e., this query expansion method has the lowest priority).

Conditional Queries. Conditional queries are recognized and parsed against a set of predefined syntax in our system. This type of queries extend users' search scope by allowing them to specify search conditions jointly over sensor metadata and time series data. Table 1 summarizes the supported query syntax in our system.

We identify the conditional queries by matching against the keywords defined in our query syntax, as shown in Table 1. Once recognized, a conditional query is parsed into the corresponding SQL statement to match against the sensor metadata and time series. Our simple query syntax helps mitigate the difficulty of query segmentation in the sense of distinguishing search terms from constraints, while imposing very little cost to the users. Since there are only three built-in keywords (as well as 5 relational operators) with predefined meanings, a user can employ natural language with a wide range of flexibility to express their search intents.

All queries are split into a group of simple and/or conditional queries and mapped to the appropriate SQL statements. For example, evaluating the "and" keyword and relational operator ">" in the query "temp > 75 and occupied" leads to two sub-queries "temperature > 75" and "occupancy > 0". Each is mapped to corresponding SQL statements to retrieve relevant time series data. Once a relevant sensor is retrieved, filtering conditions are evaluated by examining the time series data table joined by the sensor id. The resulting SQL query is similar to "SELECT timestamp, sensor_value FROM timeseries_data WHERE sensor = [relevant sensor]". This query is evaluated for all sensors deemed relevant in the first query.

Although our query syntax is simple, it empowers users to specify expressive conditional filtering statements without requiring deep domain knowledge or experience with SQL. In addition, the lower level organization of sensor data is also completely shielded away from the users.

Event Queries. Event queries will be identified by matching against the user-registered event names. WordNet-based query expansion is also applied here to relax exact keyword matching in event names. When a registered event is matched, it will be executed to retrieve the related sensor time series. If multiple events are matched, we will keep the one with the highest BM25 score [9], i.e., the most relevant event, to search for the time series data.

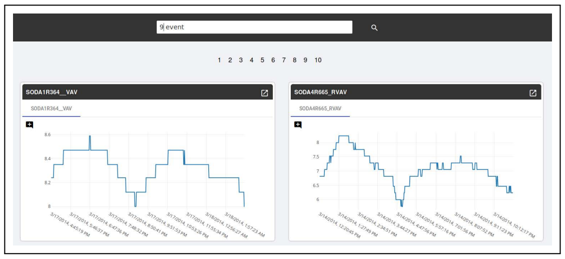

Figure 2: A sample search result from our query by example scheme. The left panel is the user-created event query, and the right panel is the top ranked search result.

3.2 Query by Example

Building managers usually have a good sense of what they are looking for visually in the data, e.g., sudden shifts in consumption, or stagnation. Yet, even provided the support of natural language queries, describing such patterns is still non-trivial. To this end, we aim to equip users with the capability to *query by example*, which allows them to visually define what they consider to be a significant event and search for similar shapes using the specified event name.

Specifically, users create an example event by highlighting a portion of a time series for a single sensor and registering a name for the event. Once an event has been created, users can search for the event by name. The search results consist of the originally created event as well as any subsequences of other sensor time series data with a similar shape. To provide fast and accurate result retrieval, we implemented two solutions to measure and rank time series by sequence-to-sequence similarity.

Our first solution involves significantly reducing and standardizing the dimensionality of the search problem then searching for similarities in parallel. We begin by splitting each indexed time series into fixed-length overlapping windows, and then transform each window into a vector of 14 statistical features [10]. We first compute 7 features in a given window, including mean value, standard deviation, skewness, Kurtosis, absolute energy, sum of absolute changes, and percentage of values above mean. We then extract the same set of features from the differential vector of this input subsequence. The differential vector is computed by the difference between the input subsequence and this subsequence with a predefined time lap, e.g., shift the input sequence backwards by two points. This feature generation happens offline and the feature vectors are stored in our database and cached. When a user creates an event, the system finds and returns a list of all the feature vectors within its start and end timestamp. Afterwards, we compare that list of feature vectors with every other list of identical length in our database using a sliding window. Between each pair of lists, we compute the cosine similarity and store the top n lists with the highest cosine similarity.

Our second solution repurposes Dynamic Time Warping (DTW) [12], which determines similarity between time series by warping the time axis so they are aligned. We modify its original implementation [15] to return the top results instead of only the best result, and our DTW implementation can compute the ranked results across millions of datapoints in seconds. After computing the ranked top n results from cosine similarity and DTW algorithms, we use Borda rank aggregation method [4] to merge the two rankings.

4 EVALUATION

Dataset and Experiment Settings. We use a dataset consisting of 3,322 unique sensors from an office building with 21 distinct sensor types. The sensor timestamps range over 498 days with varying sampling rates depending on the sensor type. In addition, the set of sensors have 151 unique metadata attributes, such as room id, sensor type, and equipment id; and each sensor has 1 to 8 attributes associated with it. We index the entire dataset based on the approaches described in Section 3.

Effectiveness of Natural Language Queries. We asked 12 participants familiar with building facilities management terms to attempt to use our system to solve real-world problems. Each participant was asked to solve two predefined tasks using our system with detailed problem statements. The first task is designed for them to look for specific type of sensors; and the second is to verify if the sensor reading supports a given assertion. For example, Task 1: "You are looking for sensors that detect the temperature of rooms. Can you provide a list of sensor names that are associated with the temperature of rooms?"; Task 2: "Someone just reported Room 300 in building SODA was too cold, i.e., its room temperature was below 65 degree. Please determine whether this event occurred." From their interactions (e.g., their issued queries and clicked results), we identified high quality natural language queries from their attempts at solving these problems. Each participant evaluated the relevancy of the results returned by their queries. From these evaluations, we determined the search effectiveness with mean averaged-precision (MAP), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR). Table 2 summarizes the evaluation results, which support the value of our support in natural language based queries for building management scenarios.

Table 2: Natural language query evaluation results.

MAP	NDCG	MRR
0.569	0.633	0.597

Effectiveness of Event Queries. We performed our user study over 13 carefully selected event queries on Amazon Mechanical Turk to capture different shapes and curvatures in the sensor time series. We asked 42 participants to evaluate the relevance of the top 12 shapes returned by the system for each query. We interleaved results returned by cosine similarity, DTW, and a mixture of cosine similarity and DTW combined using weighted borda score (0.3 for cosine and 0.7 for DTW). The average inter-annotator agreement rate measured by kappa statistic is 0.3953, which reflects the difficulty in evaluating time series queries.

Table 3: Comparison across different ranking methods.

Cosine	DTW	0.3×Cosine+0.7×DTW
90	103	100

Table 3 reports the count of results annotators marked as relevant for each algorithm during our interleaved test. We see that all three algorithms performed similarly, with DTW being slightly preferred over cosine similarity. We then used majority vote to select relevance labels from the crowdsourcing labels and report reports the ranking performance of event queries under MAP, NDCG and MRR metrics in Table 4. The high ranking performance demonstrate demonstrated the effectiveness of our system in handling a wide variety of queries.

Table 4: Performance of Query-by-Examples.

MAP	NDCG	MRR
0.751	0.870	0.833

5 CONCLUSION

In this paper, we develop a search engine system for navigating through building sensor data. It allows users to directly interact with sensor time series via natural language queries and shape-based event queries. The system automatically parses user queries and maps them against the underlying database schema for relevance-driven result ranking. In addition, it uses bag-of-features and Dynamic Time Warping to represent time series data to support query by example. Our quantitative evaluation and user studies confirm the value of the developed system in assisting users to navigate through building sensor data.

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 (2013), 2796–2802.

[2] Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. 2015. Automated metadata construction to support portable building applications. In *BuildSys*. 3–12.

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 EMNLP: System Demonstrations*. 169–174.

[4] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th WWW*. 613–622.

[5] Harry Hochheiser and Ben Shneiderman. 2002. A dynamic query interface for finding patterns in time series data. In *CHI'02*. 522–523.

[6] Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. 2015. The building adapter: Towards quickly applying building analytics at scale. In *BuildSys*. 123–132.

[7] Jason Koh, Dezhi Hong, Rajesh Gupta, Kamin Whitehouse, Hongning Wang, and Yuvraj Agarwal. 2018. Plaster: An integration, benchmark, and development framework for metadata normalization methods. In *BuildSys*. 1–10.

[8] Fei Li and Hosagrahar V Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 SIGMOD*. ACM, 709–712.

[9] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering* 16, 1 (2010), 100–103.

[10] Yannis Manolopoulos and RJ Alcock. 1999. Time-Series Similarity Queries Employing a Feature-Based Approach. In *7th Hellenic Conference on Informatics*. 27–29.

[11] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography* 3, 4 (1990), 235–244.

[12] Meinard Müller. 2007. Dynamic Time Warping. In *Information Retrieval for Music and Motion*. 69–84.

[13] Neelu Nihalani, Sanjay Silakari, and Mahesh Motwani. 2011. Natural language interface for database: a brief review. *International Journal of Computer Science Issues (IJCSI)* 8, 2 (2011), 600.

[14] Yonggang Qiu and Hans-Peter Frei. 1993. Concept based query expansion. In *ACM SIGIR*. 160–169.

[15] 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*. 262–270.

[16] Bin Tan and Fuchun Peng. 2008. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th WWW*. 347–356.