Towards Semantic Search in Building Metadata

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Master of Science

by

Akshat Pandey

May 2019

APPROVAL SHEET

This Thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

Author Signature: 

This Thesis has been read and approved by the examining committee:

Advisor: Hongning Wang

Committee Member: Kevin Sullivan

Committee Member: Yanfeng Ji

Committee Member: _____

Committee Member: _____

Committee Member: _____

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

May 2019

Abstract

This report presents the design of a search engine that can be used to search across internet-of-things sensor data and metadata in the context of building managment. Search engines are most commonly used to allow humans to retrieve relevant web-pages from the internet given a natural language query. The system presented here also takes natural language queries as input but instead of returning web-pages, its primary function is to return visualizations of time-series data in the form of line charts. The system also allows users the ability to name visually-defined phenomena in the returned line charts and search across sensor data for similar data shapes.

Acknowledgements

I would like to thank my advisor Professor Hongning Wang. He has been an excellent mentor throughout this process and has taught me a lot about information retrieval and performing research in general. I would also like to thank the University of Virginia Facilities Management team for making this project possible. In addition, I would like to thank my colleagues at the University of Virginia, Elijah Lewis, Dezhi Hong, Karthik Chinnathambi for their assistance in this effort. Finally, I would like to thank my family for all their love and support.

Contents

Chapter 1

Introduction

1.1 Background

Arguably one of the largest increases in data in the past decade has been the data generated by social media platforms. Likes, comments, posts, videos, and shared links are all part of this data landscape, each with their own variant set of usable and constraining characteristics. Understanding, interpreting, and deriving benefit from this feature-diverse web of data has become a key focus of modern information technology, and will remain so for many years to come.

The coming decade will see a similar massive increase in data. However this new expanse of data will be generated from sensors as part of the internet of things and a vast number of new cyber physical systems rather than humans alone. Unlike the diversity of data types seen in social media, data generated from these sensors is relatively uniform. In most cases sensors deliver only a steady stream of time stamps with corresponding integer or floating point readings.

The uniformity of the data generated by sensors makes the problem of searching and interpreting this data very unique. Humans cannot draw insight from the raw data alone, as they might be able to with a text document or an image. The data needs to be visualized for human interpretation. Determining what portion of the data is significant, why it is significant, and how it should be displayed will become increasingly important as the amount of data quickly exceeds any amount that could reasonably be investigated by humans in its raw format. This report will explore methods of traversing this data using natural language queries, and allowing users to retrieve sensor data visualizations as they see fit on-the-fly by means of a search engine.

The particular context for search discussed in thie report is building management. Modern buildings have a wide array of sensors available constantly collecting time-series data.

Building management groups use output from these sensors to both maintain regular upkeep of buildings, as well as to make decisions as to future enhancements to improve building energy efficiency. Currently building managment teams rely on hardware-specific or external dashboarding tools to provide them with visualizations in order to gain insight on the day-to-day functioning of buildings under their purview. The aim of the proposed system is to allow such teams to search dynamically across their data to retrieve relevant information using natural language, rather than having to monitor a large array of dashboards in real time.

1.2 Intelligent Buildings

The internet-of-things (IOT) has ushered in a new wave of "intelligent" buildings designed to make intelligent and informed decisions about aspects of building management such as power management and increasing energy efficiency using data provided by sensors located throughout a building.

According to Flax[11], an intelligent building is not only one that attempts to maximize its own potential in regards to cost or efficiency, but also one that allows for increased productivity in regards to the management of its facilities. In Wong, Li, and Wang[10] the notion of what makes a building "intelligent" is examined from a historical perspective. While older iterations of the term "intelligent building" referred exclusively to the peices of technology that may be used to enhance a building's efficiency, researchers in the field began to acknowledge the importance of including human interaction in the definition of an "intelligent building". Human interaction ultimately has now become an important part of the definition, both context of those that manage building facilities as well as those in the general public that simply occupy and use the building on a regular basis.

While a great portion of the research in the realm of intelligent buildings have been focused on the integration of facilities systems such as heating, ventilation, and air conditioning (HVAC) and alarm systems, there has not been as much work produced to enable more efficient managment of these systems. Currently in order for facilities management teams to parse through the data produced by these intelligent building sensors, team members use dashboards containing metrics from multiple sensors to manually determine the productivity of an intelligent building. This report aims to increase their productivity by introducing a natural-language query based search engine to traverse the data produced by building sensors.

1.3 Natural Language Queries and Sensor Data

Two popular methods of searching across time series data sets in the field currently are query-by-sketch, and query-by-example. Query-by-sketch systems are relatively new in the area, and allow users to search across time-series shapes by drawing shapes and producing similar windows of time-series data. Mannino and Abouzied [12] describe *Quetch*, one such query-by-sketch tool which allows users to define a search using scale-less hand-drawn patterns to return similar results.

In contrast to query-by-sketch methods, which rely on user abilities to produce significant patterns for search, query-by-example time-series search methods allow users to select significant patterns from existing time-series data. Selected patterns are then defined as the ground truth in the search for similar patterns across other time-series sources. Hochheiser and Shneiderman[13] describe such a system that allows users to visually select rectangular portions of time-series data, which are then used to return similar shaped subsequences of time-series data in their data set.

While query-by-sketch and query-by-example methodologies of searching across time-series have been studied and can currently be applied to time-series data, these two methods of search alone are not sufficient to productively traverse across sensor time-series data in the domain of intelligent buildings. This is because data shape alone does not convey all that is significant about a time-series. In addition, different line shapes may have vastly different meanings given a sensor type.

For example, suppose a member of a facilities management team were interested in the current status of temperatures in rooms across campus. Based on the location of a temperature sensor, temperature values may fluctuate greatly with a wide range, or they may remain fairly stable at a particular level. In this scenario, specifying a data shape may not then return all the relevant sensors. Specifying a high variance shape will return data from sensors with high variance, regardless of whether they are temperature sensors or not. Conversely, specifying a flatter data shape might return any sensor data with low variance, regardless of the sensor type.

Where query-by-sketch and query-by-example methods of time-series search fail is in their inability to take categorical data into account. The information provided by a time-series shape can be of significant importance, but does not reflect all that is important about a sensor data stream. Natural language querying can provide users with the ability to specify categorical requirements for sensor search, mitigating some of the issues that arise from a pure time-series based search method such as query-by-sketch or query-by-example.

However, natural language based querying cannot be accomplished through the use of

time-series data alone, especially if users are required to search across categorical metadata for building sensors. The system proposed in this report will use IOT sensor metadata as well as pure analysis of the time-series data itself to allow users the best of both data-based search as well as natural language based search. Providing both methods of search gives the user maximum flexibility in determining what kind of data they need, whether it be data from particular sensor types, or data represntative of a particular time-series phenomenon across all sensors.

1.4 Problem

The specific problem this report aims to address is the retrieval of relevant time-series sensor data given a natural language query. Given such a query, this system must be able to search across all the sensors part of the system and return data from those sensors determined to contain data most relevant to the query. Due to the nature of time-series data, returned data must come in the form of a visualization that can be used to provide insight.

Due to the limited and often techinical vocabulary used to name and describe sensors in the context of building managment, a system allowing natural language search over such data must be able to effectively derive semantic meaning from query terms relevant in this particular setting. The acceptable query vocabulary must be pertinent to the domain, but also allow for flexibility in terms of natural language to allow for a general user. In addition, in order to remain effective as the number and types of sensors increase over time, the query vocabulary must also be extensible.

1.5 Proposed Solution

The solution proposed in this work is a search engine similar in usage to Google or Bing, but instead of providing relevant webpages to the user as a list of results, this engine provides a set of time-series line charts from sensors determined to be relevant to the user query.

This system allows users to query across time-series data using a semi-structured language that maintains the simplicity of natural language while still providing users with the ability to enforce logical constraints on the returned results. It also will allow users to extend the search vocabulary to include terms and determine their semantic significance as they see fit.

Finally, this system will also leverage the largely visual-reliant nature of time-series interpretation to allow users to label and search over what they determine to be significant visual events in sensor data. Users will be able to label shapes in time series charts and search

for similar shapes in the data, further enhancing their ability to traverse large amounts of sensor data effectively.

The remainder of this text is organized as follows: Chapter 2 discusses prior works in this area and how they relate to the system proposed in this work. Chapter 3 provides a system overview, including further detail on the sensor data used in the system, an overview of how search is performed, and a high-level map of the search functionality. Chapter 4 discusses in detail the rules of structured querying language used in this system, Chapter 5 details how users may extend the searchable vocabulary, and Chapter 6 discusses the process of labelling and retrieving visually defined events in the sensor data. Chapter 7 describes how the queries is evaluated in conjunction with user-defined vocabulary and events. Finally, Chapters 8 and 9 discuss the evaluation of the system, what conclusions can be drawn from this work, and where it may lead in the future.

Chapter 2

Related Works

While there is no previous all-inclusive work detailing all the different facets of the system presented here, many of the components of this system have been studied significantly both in academia and in industry. There are also multiple products that are widely used in industry today that are designed to solve some of the same problems as the proposed system.

2.1 Query Segmentation

Query segmentation is the process of dividing queries into significant semantic subsections, and is often used in the process of understanding what a user's intention is when performing a query on a search engine. It has been studied significantly in recent years within the context of information retrieval and is an important component of the system proposed in this work.

Prior works can be partitioned into those that recommend continued use of an external data source for segmentation, and those that do not. While both groups do not necessarily require an external data source for continued use, they both groups may still require an initial data set for machine learning.

Bergsma and Wang [1] propose a machine learning-based approach to query segmentation using AOL search query data to train a support vector machine model to determine where segments should be placed in a user's query. The learning in this method involves looking at a set of lexical, statistical features to determine whether position x of a given query represents a segment or not.

The lexical features are hand-crafted binary features that are triggered given the presence of particular things, like the word "the" or a part of speech tag. Some of the statistical features they use are counts of the pair of words w_{x-1} and w_{x+1} surrounding x appearing together in the training data, or how often they appear in the genitive format w_{x-1}'s w_{x+1} in

the training data. Similarly, they also have features that are meant to examine the context provided by prior and later words near position x in the query.

The system proposed in this work also examines position x of a query given particular features, but simply relies on the features alone to determine whether the position should be considered a segment or not. However, given significant usage of the system, user query data could be collected and used to train a model to segment similarly. This could allow greater flexibility in future iterations of querying in the system.

Tan and Peng [3] suggest an unsupervised machine-learning based approach with the addition of assistance from Wikipedia in order to determine the segment points. Their method relies on the assumptions that queries are generated from single or multi-word concepts, and that these concepts are independent and identically distributed. In this method, a query is a sequence of words $w_1, w_2, ...w_n$, and also a sequence of concepts $s_1, s_2, ...s_m$. Assuming a unigram model of concepts, the probability of a sequence of concepts $P(S^Q)$ can be calculated as

$$\begin{aligned} P(S^Q) &= \Pi_{s_i \in S^Q} P_C(s_i) \\ &= \Pi_{s_i \in S^Q} P(s_i | s_1, s_2, ...s_{i-1}) \end{aligned}$$

The segmentation that maximizes $P(S^Q)$ is then determined to be the proper segmentation of query x. They incorporate the use of query-relevant subcorpuses in order to make calculating this feasible. In addition, Wikipedia is then incorporated to ensure the soundness of concepts by introducing a term in their calculation of $P_C(s_i)$ that takes the frequencies of sequences in Wikipedia article titles into account.

This approach is an effective and interesting way to segment queries in an unsupervised manner, but requires quite a large data set in order to work effectively. However the use of Wikipedia article titles to improve segmentations could be incorporated into this system in the future given that an appropriate alternate data source could be found. Wikipedia does not have enough, or in come cases any at all, articles relating to the different kind of sensors evaluated in this system.

It should be noted that semantic concepts in the context of IOT sensors are significantly different than the semantic concepts needed in web-page retrieval. For example, in a web search engine, "computer science courses" correctly segmented as "computer science", and "courses". This segmentation could help retrieve better results for a user. However, in an IOT sensor search, segmentation can have an entirely different meaning. For example "Room temperatures > 72" can be split into "room temperature", and "> 72". In this scenario, "room temperature" defines a search over sensor metadata to determine the type of sensor to

return. "> 72" defines a search over data, rather than metadata. In this way segmentation in sensor search can help define the appropriate search space, similar to an ontological search.

In Fernandez, Lopez, Sabou, Uren, Vallet, Motta, and Castellis[19] a search engine is described that takes a user query and performs search in a two-tiered approach. First the appropriate ontological subject of search is determined, and then the appropriate data is retrieved. This two-tiered approach can be used in sensor search to identify the appropriate sensor via metadata and then constraints on the data can be applied on the data itself to return appropriate results.

2.2 Time-Series Feature Selection and Classification

Feature selection is the process of choosing significant variables from data in order to avoid using all available variables and reduce complexity. It has been widely studied for a significant period of time, and there is also a considerable amount of work available considering feature selection specifically in the context of time-series data. The prior work in this field was taken into great consideration in the event labelling feature of the proposed system.

Classification is a heavily studied aspect of machine learning relating to assigning labels to data given prior data. It has been studied significantly in the context of time-series data. Similar to feature selection, the prior work in this field had a large influence on the decisions made in the development of the event labelling strategy in the proposed system.

In Baydogan, Runger, and Tuv[4] time series classification is done using a bag-of-features representation of time-series data. To generate a bag-of-features for a series, data is divided in to subsequences in order to account for local features. For each subsequence, the slope of the fitted regression line, mean, variance are calculated. In addition, the overall mean and variance of the series as well as the starting and ending points are also collected and added as features for the subsequence. Class probability estimates are calculated for each subsequence feature vector and each possible class, and are designated as feature vectors for each subsequence. The vector formed from the concatenation of all the subsequence feature vectors is designated as the feature vector for the entire sequence and used to train a classifier.

General search over time series data does not involve class labelling in the sense that a machine learning practitioner might consider class labelling. But the features and methods of feature selection in Baydogan et al.[4] can be used to create summarizing feature vectors for time series data in the context of event detection.

Schafer and Leser[7] use a bag-of-patterns in order to classify time-series data. In this work, a bag-of-patterns is generated by first taking windows for a time-series and approxi-

mating it using a Fourier transform. The real and imaginary Fourier values that provide the best separation amongst classes are used to define discretized bins. Bins are translated into words, and finally a bag of patterns is generated by looking at the unigrams and bigrams taken from all the windows of a series.

Transforming time-series into a discretized sequence of unigrams and bigrams is an interesting and effective way to summarize a time-series for the context of time-series classification. However the complexity of the creation of the bag-of-patterns can only be warranted in the scenario that simpler methods provide significantly worse results. This method of time-series summarization may be ideal for the purposes of training a classifer once, but is not necessarily feasible for the continuous transformation of data. In the case of a live building management system of many sensors continously producing data, the complexity of creating such a bag-of-patterns for the purposes of searching over is not ideal.

2.3 Dasboarding Tools

Dashboarding tools are suites of software tools designed to be placed on top of a data set in order to provide a visual display of metrics and various kinds of information derived from the data set. There are a variety of such tools in the market that could be used on time-series data in the context of building management. These existing systems are a general solution that could provide some of the same features as the system proposed in this work.

One of the most popular of these software suites is Tableau, which is used widely in business and and analytics settings in industry to generate live visualizations of datasets without the need for the user to have any programming ability.

Tableau allows users to create dashboards, or groups of charts and visualizations by dragging, dropping, and sizing components onto a canvas. Once a set of visualizations is selected, a required data source is specified, and the visualizations become active.

Visualizations provided in this dashboarding tool as well as many of the similar competitors can include line charts, scatter plots, maps, heatmaps, bar graphs, and others. In addition some of these tools also provide the ability for users to specify the display of particular mathematical operations on their visualizations, like the addition of a regression line on a scatter plot.

Dashboarding tools are very general, easy to use and provide a certain level of customizability and interactivity to data analysis. In the context of building management, charts and visualizations created using one of these tools could be connected to sensors in order to display live information about a particular sensor.

However, in the context of managing a large set of sensors, where dashboards suffer

is in their inflexibility in regards to defining visualizations. Determining insight from a created visualization is not difficult, and visualizations are easy to create, but if one wishes to examine date from sensors other than those in already created visualizations, new ones must be defined. For the purpose of examining and identifying issues in particular sensors, this workflow is not ideal.

In addition to this, supposing that one had the screen space to display data from many many sensors simultaneously, the use of such visualizations becomes increasingly questionable. A worker can only effectively evaluate output from so many visualizations at one time. Too many may be just as ineffective as not enough.

The system presented here aims to mitigate some of these issues by enabling users to use natural language to search for and annotate sensor data as they see fit on-the-fly. The system aims to provide some of the interactivity of existing dashboarding technology with the added functionality of language-based search for added flexibility.

Chapter 3

System Overview

Generally speaking, a search engine returns relevant documents given a user query. Most commonly in terms of public usage, documents consist of text and image-based web pages. In the most naive sense, the relevance of said web pages is determined by performing some function of comparison between the natural language query provided by the user and the text in the web pages being searched upon. Documents are then returned to the user in the order of their successive relevance the the query provided.

In the context of searching across sensor data, documents are fundamentally different, as they do not contain any words, but are rather just a stream of floating point values. In order to ensure functionality of a search engine across time-series data while leveraging the current work on natural language based search engines, a system would need to be able to effectively evaluate time-series data as a document using any available text as well as text derived from time-series phenomena.

With time-series data we cannot immediately apply traditional information retrieval methods and instead must consider other possible ways to retrieve data. Basic statistical methods of previous works in information retrieval and natural language processing are not easily afforded due to the basic differences in document structure. Given stream data and data of the attributes associated with these streams, the system must be able to determine a sensor data's relevance and return sensor data to users in order of relevance to their natural language query.

3.1 Data

The data used in this project was all obtained from the University of Virginia Building Management Group. The data contains time series data from 2858 different sensors in buildings on the University of Virginia campus. The overall group of sensors contain 34

distinct sensor types. The number of time-series data points for each stream range from 1 data point to 127,662 data points, and there are a total of 21,909,993 time-series data points in the entire system. The timestamps on the streams range from 6/1/2013 03:00:19 to 10/12/2014 17:38:05. However it should be noted that the data is not completely continuous within those date ranges.

1942 of the 2858 sensors also have additional descriptive text attributes associated with them. There are a total of 151 distinct such attributes for all the sensors. Attributes can be single or multiple terms and may have corresponding values. Some example attributes are "occupancy sensor" and "room-id:386". "occupancy sensor" is a two word attribute with no corresponding value, while "room-id:386" is a single word attribute ("room-id") with an associated value ("386"). Sensors in the data set range from having no attributes at all, to having a maximum of 8.

3.2 Document Generation

Given the list of attributes for a sensor, the system defines "documents" using a bag-of-words model incorporating all the terms in each attribute name. Multiple term attributes are split into a list of their distint terms and added to a bag-of-of words document for their corresponding sensor. An example sensor with its corresponding attributes and document transformation is shown in Figure 3.1.

Sensor Name	Attribute Text
Siemens_229387	site:SDH, room-id:282, supply fan, supply fan-id:1

Sensor Name	Document
Siemens_229387	fan, fan-id, room-id, site, supply

Figure 3.1: Document creation

For the purposes of basic document search, all attribute values are removed, and the document associated with the sensor is simply the set of all terms in the attribute names associated with the sensor.

3.2.1 Attribute Expansion with WordNet

One issue that arises when using the given attribute names is the relatively small number of attributes associated with each sensor. Exact search or even a fuzzy search method across

the available attribute names will not lead to many matches unless the user is already very knowledgeable of the data set.

WordNet is a lexical database developed at Princeton containing various different relationships between words rooted in their categorization in to "cognitive synonyms", or "synsets". Relationships between words in WordNet can be representative of various different kinds of established linguistic relationships, such as hypernomy, hyponomy, meronymy, antonymy and others. For the purposes of this system, no differentiation was taken into account with regard to the specific kind of linguistic relationship between words when expanding documents.

In order to help mitigate the problem of small document sizes, synsets extracted from WordNet were used to expand the available list of attribute names to help provide a larger and more general set of document terms to search across when determining relevance to a user query.

3.2.2 Attribute Expansion with User Input

Due to the fact that the vocabulary used in attribute names was so domain-specific, often times even expansion using WordNet is not sufficient to provide sufficient expansion of documents. For example, the word "site" is an attribute for many of the sensors in the data set. WordNet identifies "locale" in the group of synsets for "site". "site" in the context of the data is closely associated with the concept of a "building site". "locale" is somewhat related to this notion, but not entirely.

While "locale" is still somewhat relevant, another term in the group of synsets for "site" is "web-site", which is wholly irrelevant to the context of this system. Including "web-site" will not really help provide better search results here because if one were looking for sensor streams at a particular place on the University of Virginia campus, they would not use the word "web-site" in their search query.

Because there is no data-set of terms that are exclusively significant to the context of building management or sensor data, the only other way to mitigate this problem of low attribute counts is to allow users to tag sensors with attributes that they determine to be relevant.

The system allows users to enter their own attributes along with corresponding values if they determine it to be necessary and associate these newly-created attributes to sensors. Once an attribute is entered into the system it is then added to the document corresponding to the appropriate sensor, and can be used as a search term to return data from the relevant sensor.

3.3 System Functionality

The system's high-level approach to basic querying is to treat the list of attributes and values associated with a sensor as a document. This document is to be associated strictly with the sensor from which the attributes are taken. A basic search can then be simplified to mimic the standard document search retrieval process - sensors are deemed relevant to user queries depending on how relevant the documents associated to the sensors are to the terms in the user query. Figure 3.2 and Figure 3.3 display how a simple query is evaluated in the system.

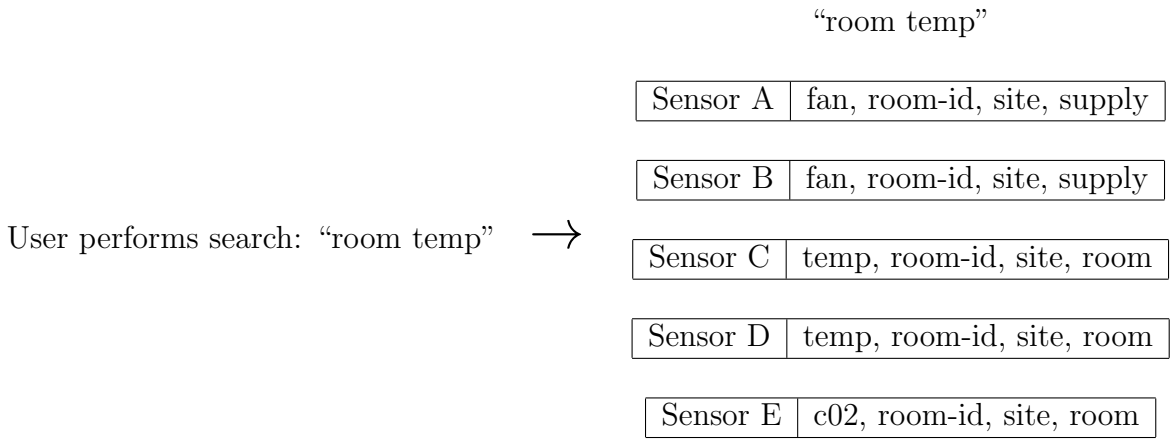


Figure 3.2: Query evaluation 1

The user performs search with the query "room temp". The query terms "room" and "temp" are evaluated against the documents for each sensor in the system generated using the attributes provided in the data set.

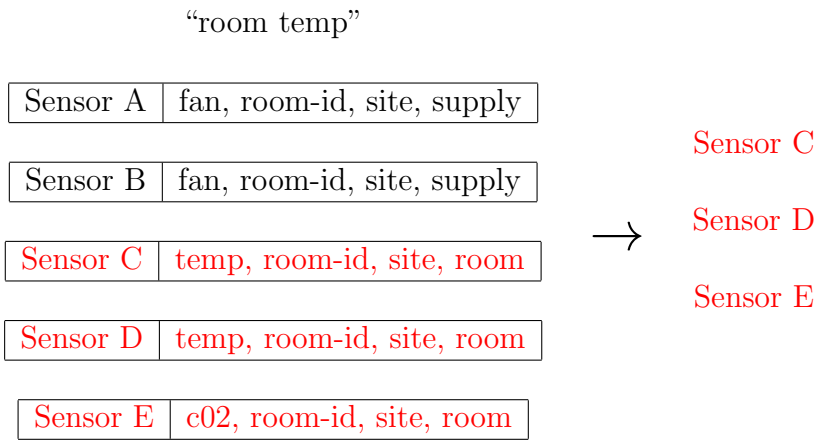


Figure 3.3: Query evaluation 2

"Sensor C", "Sensor D", and "Sensor E" are deemed as relevant sensors to the query "room temp" due to the co-occurrences of the words "room" or "temp" in the documents corresponding to each respective sensor. It should be noted that "Sensor C" and "Sensor D" would be regarded as being more relevant than "Sensor E" because their documents contain both "room" as well as "temp", unlike "Sensor E", whose corresponding document only contains "room". Finally, the search results page will display line charts for time-series data retrieved for "Sensor C", "Sensor D", and "Sensor E". This displays the basic functionality of the system - how a natural language query is interpreted to return relevant time series data.

3.4 Document Term Co-occurrence

In this system's implementation of relevant attribute term search, the system uses a simple co-occurrence count of user search terms against attribute document terms. In modern information retrieval systems, a wide array of techniques have been developed to determine a document's relevance based upon a comparison between user search terms and search document terms. One very popular metric used in such instances is term frequency inverse document frequency, or the tf-idf metric.

tf-idf has had wide use in information retrieval systems for many years as a way to determine the importance of a word by taking into account the frequency of word occurrence in searched documents as well as the frequency of a word across all documents. The term frequency is used to determine whether or not a document is relevant to user search terms, and the inverse document frequency is used to determine how much importance should be given to a particular user search term in finding relevant documents in a corpus.

In its most naive formation, the tf-idf metric can be calculated using the forumula in Figure 3.4. In Figure 3.4, t represents a term, d is a document, D is all the documents in the corpus, n_t is the number of documents containing term t, and N is the total number of documents.

$$\begin{aligned} tf - idf &= tf(t,d) \times idf(t,D) \\ &= f_{t,d} \times \log\left(\frac{N}{1+n_t}\right) \end{aligned}$$

Figure 3.4: tf-idf calculation

As shown in Figure 3.4, the tf portion is simply the frequency of the word in the document, which is what this system uses to determine document relevance. However the *idf*

portion of the metric uses the total number of documents containing term t to specify how important t is in calculating document d's relevance.

In a practical setting, the *idf* term helps systems ignore terms that are so frequent in every document that they are not relevant in document search. For example, if a user were to query Wikipedia for "The New York Times", the *idf* metric may help the Wikipedia search engine determine that articles only containing the word "the" may not be strictly relevant to the user query, because "the" is generally such a frequently used word. In this way, the *idf* metric can help increase a search engine's precision by assisting in the identification of irrelevant documents.

The use of the *idf* metric in this context makes the assumption that the more frequent a term is in a corpus of documents, the less meaningful it is, and therefore it is less important in regards to document relevancey determinations. This assumption can be summarized using Zipf's law, which states that word frequency is inversely proportional to its rank.

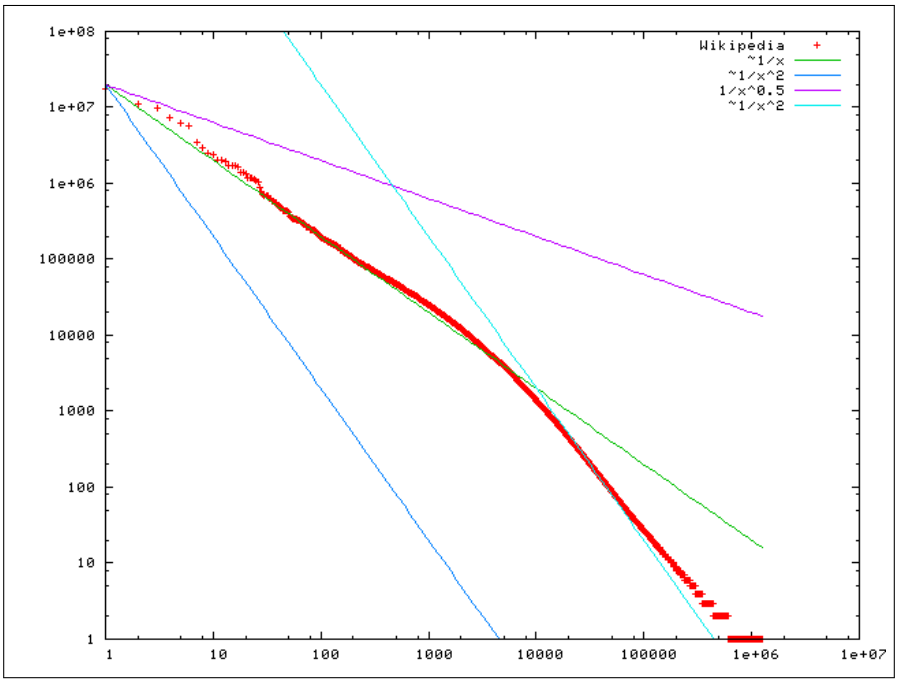


Figure 3.5: Zipf's Law in Wikipedia Text[14]

Figure 3.5 displays Zipf's law in the context of Wikipedia text as of the year 2006. This figure is in the log-log scale, and the x axis represents word rank, and the y axis total word frequency across Wikipedia. In this figure Zipf's law corresponds to the dark green line, while the red is calculated using actual Wikipedia term frequencies. It is clear from this figure that Zipf's law has empirical precedent in the case of Wikipedia terms, and even across natural language in general.

However this assumption does not necessarily apply in the case of the attribute-based documents used in the proposed system. Because they are bags of words and not documents written in natural text, the assumption that a frequent word is not important is not one that can be immediately taken without further examination. In this domain, document sizes do not reach the lengths that one would expect in a Wikipedia article for example. And because they are bags-of-attributes, there are no direct equivalencies in this limited vocabulary to words like "the" in the general English language.

Furthermore the inapplicability of what is often considered a very basic assumption in language-based information retrieval (Zipf's law) to this domain states that many of the classical assumptions must be closesly examined before they can be deemed relevant in this space. In light of this, document co-occurrence, or simply term frequency was chosen as the exclusive metric for determining document relevance, as it does not make any assumption on the inherent importance of attribute terms.

Chapter 4

Querying

In this system a query may consist of search terms, mathematical operators, as well as predefined keywords. Users may search across documents by attribute tags or stream names. In addition they may place constraints on their search results by using conditional operators to define the numeric bounds for retrieval and they may group results by attribute tags to return results.

Querying is performed in a manner similar to most web-page search engines on the market. Users enter natural language queries, and relevant results are returned to them in the form of relevant line graphs. Unlike most search engines, however, there are a few constraints on how language may be used in the system. The system is queried using a structured language consisting of the keywords "and", "by", as well as the mathematical operators $>, <, >=, <=$, and $=$. The "and" keyword is used in chained querying, which will be discussed in section **4.1.4**. The "or" keyword is used in grouped querying, which is discussed in **4.1.3**, and the matematical operators listed are used in conditional querying, which is detailed in **4.1.2**.

4.1 Simple, Conditional, and Grouped Querying

Acceptable user queries can be divided into three different categories: simple queries, conditional queries, and grouped queries. In addition to these three categories users may use the "and" keyword to chain combinations of these three query types together in order to return more specific search results.

4.1.1 Simple Querying

Simple queries are the least complex and consist exclusively of search terms. Examples of such queries may be "temperature" or "supply fan". Such queries will be evaluated by comparing the search terms against a WordNet expanded list of terms stored for each stream in the system. Results will be returned based simply on the co-occurrence count of search terms in the expanded list of terms for each stream. Figure 4.1 displays the functionality of the simple query "temp" in the system. After searching for "temp" in the search bar, the top two relevant results are shown in the figure - scrolling down will display further results. They are ranked according to co-occurrences counts between search terms and the attribute documents associated with each vector.

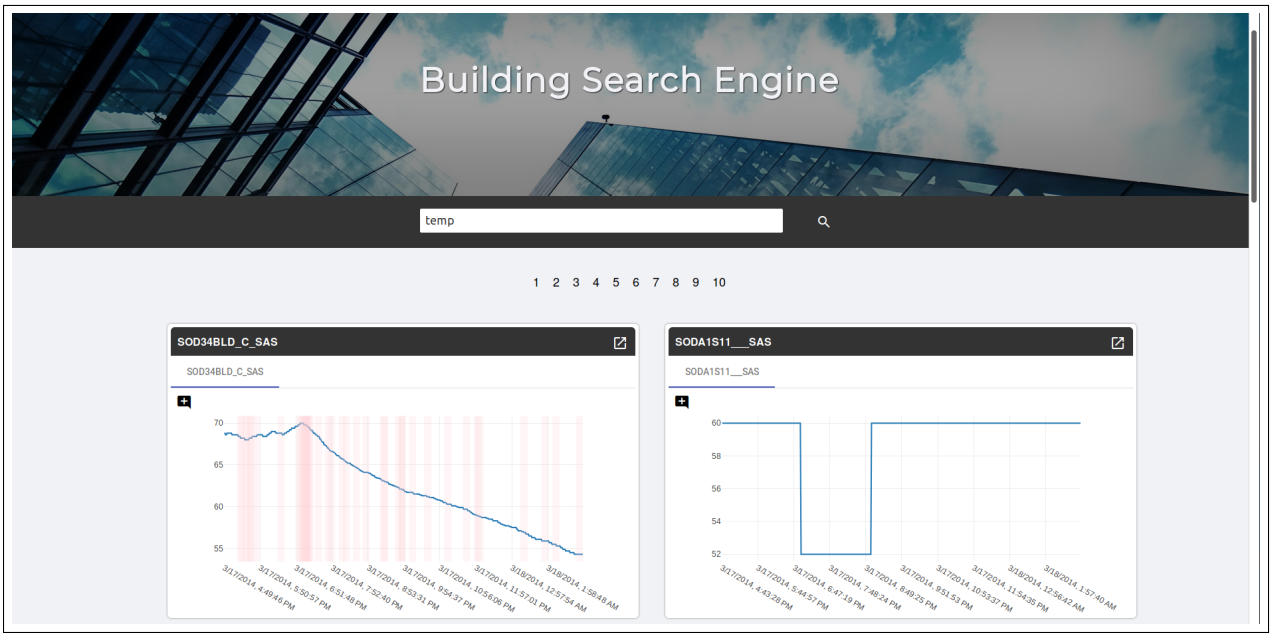


Figure 4.1: Simple query search results

4.1.2 Conditional Querying

Conditional queries are simple queries that include a numerical constraint on the values of the time-series data retrieved by the system. Examples of such queries may be "temperature > 75" or "occupancy $<= 2$". Similar to the evaluation of simple queries, search terms will be compared against a WordNet expanded list of terms stored for each stream in the system. However these results will then be filtered to only retrieve those streams whose values do not break the constraint specified by the operator.

It should be noted that currently the system will only evaluate the operator values at

the most recent timestamp available for each stream in the system. For example, the query "temperature > 75" will only return sensor data with the appropriate attribute matches for "temperature" but also where the latest timestamp available contains a value greater than 75. Previous values are not evaluated in the current implementation of conditional querying.

Conditional queries will provide results similar to those displayed in Figure 4.1 - however, the last value in each chart will be compliant with the value specified by the conditional statement of the query.

4.1.3 Grouped Querying

Grouped querying allows users to return results in groups with respect to their specified search terms. A grouped query may consist of a simple or conditional query with the addition of the keyword "by" followed by an attribute search term with which to group their returned results. Examples of such queries may be "occupancy by room-id", or "temperature <= 68 by building-id". The simple or conditional query to the left of the grouping statement will be evaluated as listed above, and then additional search will be done upon the resulting streams to identify those whose expanded attribute terms contain the terms in the specified grouping statement. The secondary evaluation of grouping terms is evaluated similar to the simple queries - grouping terms are evaluated based on co-occurrences in WordNet expanded terms stored for each stream.

In addition, users may also specify equality constraints on the grouping statements to further constrain the returned results. An example of such a query may be "occupancy by room-id = 504". This query would first evaluate the statement "occupancy by room-id", and then take the further step of removing all those stream matches where "room-id" is not equal to "504". The conditional matching on grouping statements in the current implementation only accounts for exact matches.

Figure 4.2 displays the results obtained from the grouped query "temp by site". Results here look similar to the results displayed in Figure 4.1, however, here each result box has a list of tabs on the top. Once a tab is clicked, the line graph for the corresponding sensor is displayed rather than the current chart being displayed. The chart tabs in each card correspond to the grouping value listed on the title. In Figure 4.2, the card on the left displays all sensors with "SDH" listed as a "site" value, and the card on the right displays all sensors with "SOD" listed as a "site" value.

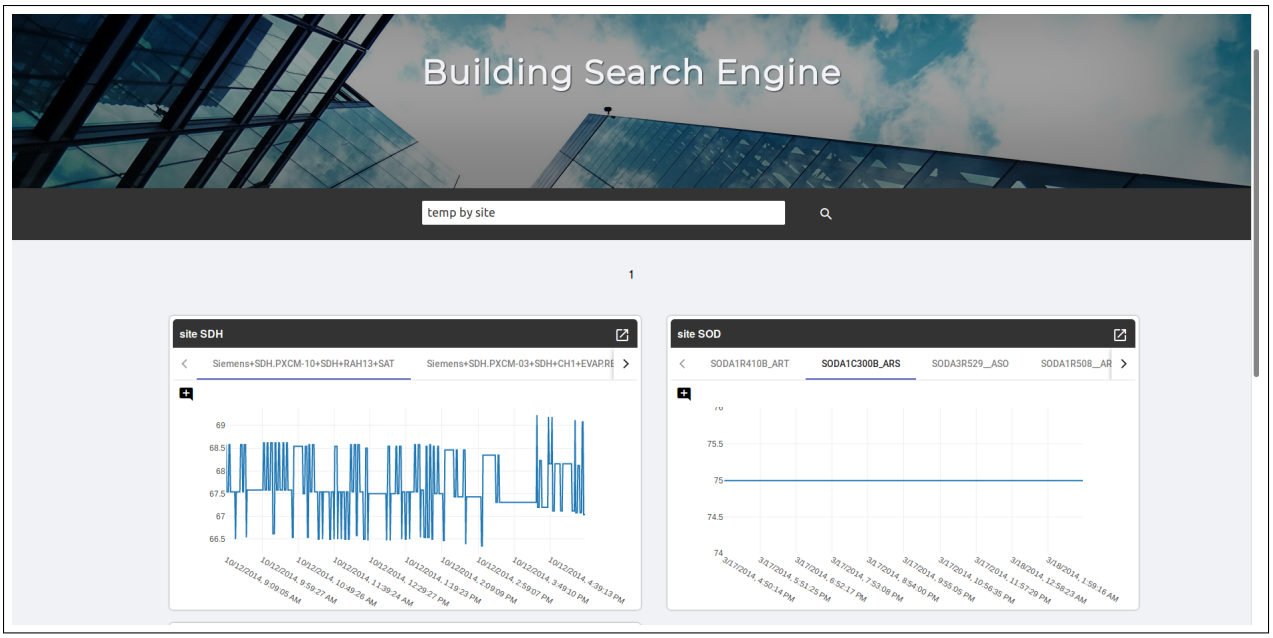


Figure 4.2: Grouped query search results

4.1.4 Chaining

The system also allows for the chaining of simple and conditional queries using the "and" keyword. An example of such a query may be "occupancy and temperature >= 75". This query will act as an "or" operation on the separate queries "occupancy" and "temperature >= 75". Chained queries can contain as many "and" statements as the user deems necessary. In addition grouping statements can also be appended to any chained query. For example "occupancy <= 0 and temperature >= 72 by room-id" will display room and occupancy measures for each available room-id with the specified constraints applied.

Chained queries will return results similar to those shown in Figure 4.1 if there is no grouping statement attached to the query. However if a grouping statment is at the end of the chained query, results will look similar to Figure 4.2, where result cards are tabbed with each sensor containing the same grouping value as the rest of the sensors on the card.

4.1.5 Imposed Query Structure

Bergsma et al.[1] as well as Tan et al.[3] both suggest methods of automatically determining the segmentation of queries into their significant semantic subunits with the use of machine learning methods. Bergsma et al.[1] use AOL search log data to train their segmentation model, and Tan et al.[3] use AOL search log data as well as Wikipedia article titles in their implementation of a query segmentation model. Both wish to split user subqueries into

semantic concepts rather than simply terms, for improved search results.

In the context of search in this system semantic concepts can be similar to those suggested in prior works on query segmentation. However segmentation in this system must also be able to distinguish between what could be defined as commands and simple search terms. For example, in the search query "room temperature > 75", "room temperature" could be regarded as a semantic concept whose meaning is much more relevant than the meanings of "room" and "temperature" as seperate search terms. The latter part of the query "> 75" however, does not consist of search terms at all. Rather it specifies a contraining command to be placed upon the results of a search for "room temperature" sensor data. This requirement distinguishes search in the sensor domain from search across text-based documents.

Taking this additional query segmentation constraint into regard, the models suggested Bergsma et al.[1] and Tan et al.[3], or any similar model cannot be used in this space, due to the fact that there is no similar training data to train such a segmentation model with. Due to the lack of data available in this space, it is not clear how to apply machine learning methods to train any model to learn the ability to segment IOT sensor-based user queries into semantic concepts.

The enforcment of a particular query structure was chosen for this system because it helps mitigate the difficulty of query segmentation in the sense of distinguishing search terms from commands while coming at very little cost to the user. Since there are only three keywords (as well as 5 mathematical operators) that are not allowed in search phrases, the user can still employ natural language with a wide range of flexibility while still being able to express commands specific to search in this domain effectively.

Chapter 5

Attributes and Labelling

Given only the time series data without any extension or modification, search would be limited to the names of streams, which are often unintuitive and impossible to extend on their own. An example of such a stream name is "Siemens+SDH.PXCM-09+SDH+WIN+7FS5_TEMP.2". This is not a search term that one could reasonably expect a user to enter to find window temperatures (which is the sensor the stream name is attached to). Without prior information about the stream name, it is also not immediately intuitive how to extend this stream name to correspond to searches for "window" or "temperature".

Even assuming that basic stream names are available, such as "temperature" or "occupancy sensor", it is not easy to then associate these names with relevant descriptive search terms that a user may want to use, such as "warm rooms" or "empty rooms". The "attribute" and "label" features provide the system with the flexibility to accomodate more natural and expressive queries, and allow users to extend the searchable vocabulary as they see fit.

5.1 Attributes

In this system "attributes" are defined as key value pairs that are attached to individual sensors. An attribute has a name and can have a value if specified. Since stream names are often not in natural language, all search queries are matched against the attributes associated with each sensor. An example sensor and their corresponding attributes are displayed in Figure 5.1.

Sensor	Attribute Name	Attribute Value
Siemens+SDH.PXCM-09+SDH+_TEMP.2	Type	Temperature
	Room-Id	300
	Building-Id	Rice

Figure 5.1

5.2 Labels

Labels are defined as user defined terms that expand to conditional queries when evaluated at search time. An example of a label would be "warm", which, if so defined, would return all temperature sensors with a value greater than or equal to 75. Labels can be created by users, and once created are immediately usable as part of simple or conditional query chains.

5.2.1 Creation and Usage

Labels can be created by specifying a label term, an attribute, and a corresponding numeric operator and value. Once these have been specified, the label term can be used in searches. If the user defined the label term "warm" to the attribute "temperature" and the operator and value ">=" and 75. Upon searching "warm", the system would expand the label to the equivalent conditional query "temperature >= 75".

Once a label has been defined its usage is similar to that of a conditional query. It can be chained along with other simple queries or conitional queries. In addition, grouping statements can also be used to further constrain search results.

Attributes are created using a simple form as shown in Figure 5.2. In Figure 5.2, the user is defining the label "warm" to be equivalent to the conditional query "room temp > 75". Upon saving the defined label, searching "warm" will return results equivalent to searching for "room temp > 75" from then on.

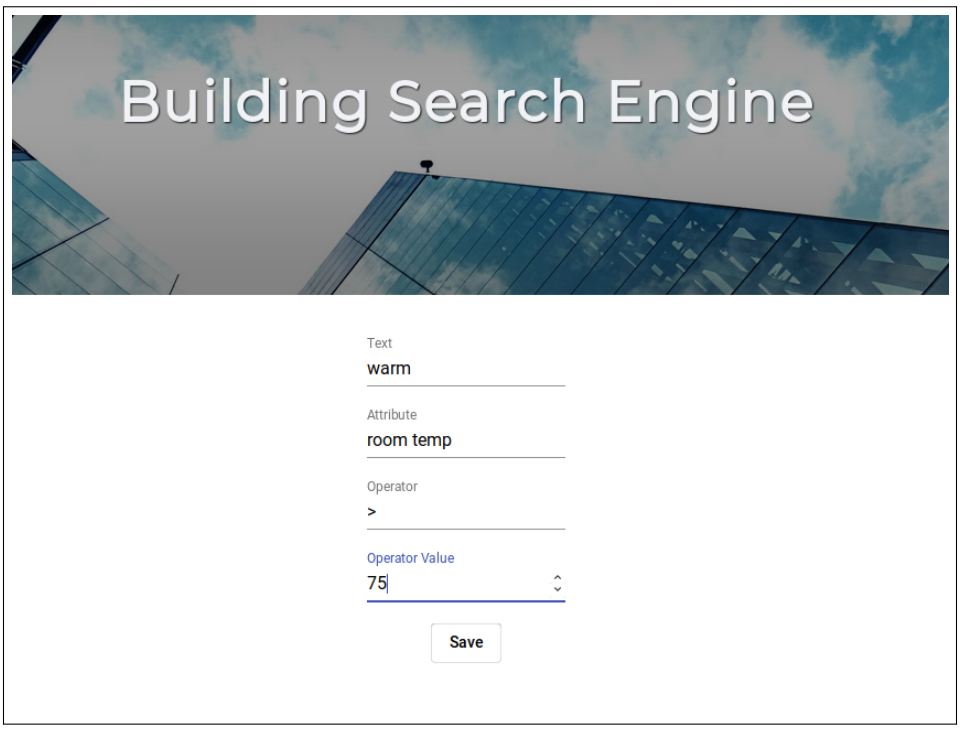


Figure 5.2: Label creation form

Chapter 6

Event Labelling

Event labelling allows users to visually define what they consider to be a significant event and search across the data for subsections of sensor data with similar shapes using the specified event title. Automatic detection anomaly detection can detect a limited amount of shapes within data, but does not allow users the flexibility to define what kind of event is relevant.

6.1 Event Creation

Events are created by highlighting a portion of time series data for a single sensor and specifying a name for the event. Once the event has been named, users can search for the event by name. The results will display the original created event as well as any subsequences of other sensor data with a similar data shape. Data is displayed in the order of relevance to the user-defined event. Events are created with the use of a single form, where the user simply specifies the name of the event.

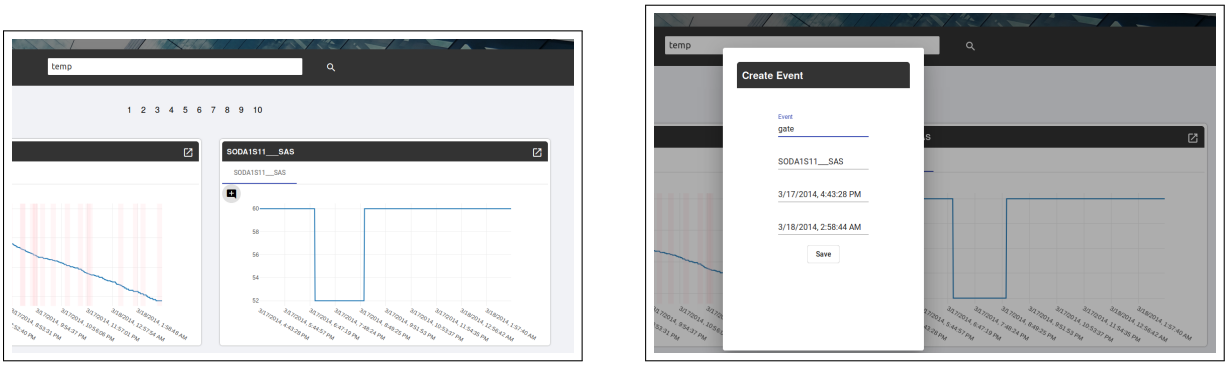


Figure 6.1: Event creation

Figure 6.1 displays the creation of an event “gate” created using one of the first results from the search “temp”. Searches for the “gate” event will now return a list of sensor charts

(as in Figure 4.1) where the graphs are visually similar to the chart labelled as a "gate" event.

6.2 Event Matching

In order to determine similarity between defined events and the rest of the data in the data set, the time series data specified by the start and end points of the event must be vectorized and compared to similar vectors of the rest of the data.

Computing feature vectors for all the available data at every event search would not be feasible, so feature vectors must be precomputed for the data set so they can be used for comparison immediately at search time.

6.2.1 Feature Vectors

Feature vectors were defined using data from each sensor examined in 500 data point windows. Windows were generated for the entireity of data available for each sensor with a step size of 100.

The following features were used in defining an event vector: minimum, maximum, median, mode, mean, standard deviation, skewness, kurtosis and entropy. Each of these values was calculated for a window, and then again on each fifth of the window. Overall, each of the nine features was calculated six times for each vector resulting in feature vectors of size 54 for each window.

After precomputing feature vector values for the entire data set, the feature vector values were all normalized to remain between zero and one. This was in order to prevent features with inherently large values from overshadowing those with inherently smaller values overall.

6.2.2 Time-Series Features

There is a large body of work available detailing the creation of appropriate features for time series data in the field of classification. Baydogan et al.[4] suggest the use of class probability estimates on a group of local features to define a feature vector, and Schafer et al.[7] suggest the use of patterns generated through approximated Fourier transform values.

The clearest difference between prior work in this field and the event search proposed in this work is the existence of prior class labels. In the proposed system users define events classes on-the-fly. Not only that, but users only identify one shape in a time series to define an event, so there is not enough data to train a model to identify the event being specified by a user using a machine learning approach.

In addition, in a live implementation of this system, feature vectors would need to be continuously calculated and stored for sensor data as it comes in, with as little delay as possible. So a solution for feature vector creation in this context must be both rapid and expressive. Given these constraints, only mathematically computable features that do not require further data outside of a given time-series were pursued.

Dividing time-series into sub-windows was used due to the successes previous works have described in this method's ability to determine local events in a larger time-series[4]. The selected features were chosen due to their ability to summarize sensor data shapes while being very efficient in terms of complexity.

6.2.3 Matching

Upon search of an event, a feature vector is created for the sensor and time period specified by the user and then compared to all the precomputed vectors. Each of the precomputed vectors is ranked based upon their cosine similarity to the event feature vector. The cosine similarity calculations are done on all the precomputed feature vectors using the map reduce paradigm in order to return results in a reasonable amount of time. The top one hundered precomputed vectors are then returned, and the corresponding sensor data is returned to the user.

In Figure 6.1, an event called "gate" was created by the user. In order to search the sensor data for similar shapes, the user must specify the name of the event followed by the keyword "event". In order to return similar shapes to the gate event, the user would then query "gate event".

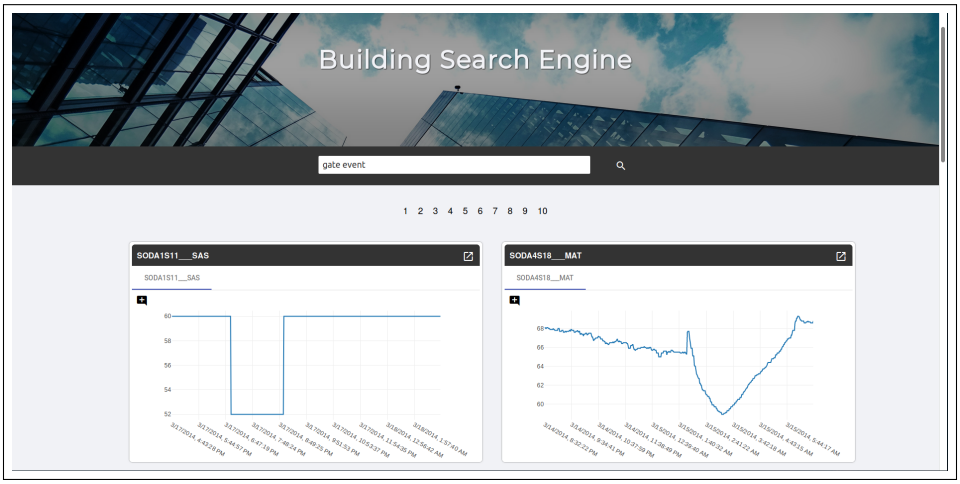


Figure 6.2: Event search

Figure 6.2 displays the results for the "gate event" query. Results are returned in order

of their cosine similarity to the feature vector created for the user defined event at creation time.

6.3 Event Label Testing

In order to test the effectiveness of event search, a user study was performed with three current students at the University of Virginia in the computer science department. Each student was given 20 ground-truth event shapes as well as the top 10 results returned by the system which were identified as similar. Participants were asked to look at the ground truth shape and identify whether each of the top 10 shapes (the first page of results) deemed similar by the system were relevant or not relevant to the original shape. Particpants were explained the use case that event search provides users, namely that users of the system may define data shapes they deem significant and that the system must return similar shapes. All participants were told that relevance judgements could be made at their own discretion, and were not provided any other instructions out side of the use case.

	Top 10 Results	**Top 5 Results**
User 1	.64	.68
User 2	.51	.55
User 3	.38	.43
Overall	.51	.55

Figure 6.3: Average Event Relevance

Figure 6.3 displays the results of the user study. Overall, when looking at the top 10 results returned by the system, users determined that 51% of the results were similar enough to the ground truth time-series shape to be considered relevant. When considering only the top 5 relevant results returned by the system, 55% of the results were deemed relevant.

Figure 6.4 displays the time-series shapes that test participants determined as having the most relevant and least relevant returned results. The 4 shapes on the top row are the shapes that performed best as determined by the test participants, and the 4 shapes on the bottom row are the shapes that performed the worst.

As displayed in figure 6.4, the system does well in retrieving relevant results for shapes that are consistent from start to finish, while the system fails to retrieve the similar results for shapes with sharp changes near the start and finish, or those with distinct hyper-local events.

Directionality of events also seem to have a detrimental effect in returning relevant results. For example the third shape from the left on the bottom row of figure 6.4 performed poorly

Figure 6.4: Best and Worst Performing Data Shapes

largely due to returned similar shapes appearing in the opposite direction to the ground truth shape rather than a far deviation from the original.

Distinct hyper-local features, such as the smaller increases and decreases midway through the fourth shape from the left on the bottom row of figure 6.4 also had a significant impact on how relevant results were determined to be by users. Very small sub-features makes it more difficult to find similar shapes because shapes must be able to match very distinct and small sub-features rather than just a general shape.

Chapter 7

Query Segmentation and Evaluation

There has been a significant amount of work done on query segmentation using machine learning techniques in the context of search engines. However, any machine learning technique requires training data, which is absent in this scenario. Without prior data available on how to segment queries for time series data, a machine learning approach to segmenting user queries was not a viable option.

This system instead uses a simple but structured query language that constrains user querying to a particular format. In order to maintain queries in a natural language format, the constraints are fairly small. Users should be able to effectively retrieve the information they want from the system without having to spend time learning how to use it.

The introduction of constraints on how queries may be formatted makes the process of segmenting user queries simpler. Search terms, conditional constraints, grouping statements, and event searches can all be identified by their position in the query. Each query, once segmented into its relevant subsections, can be evaluated by first returning results for the search terms, and then adding the constraints provided by conditional terms, and grouping statements afterwards.

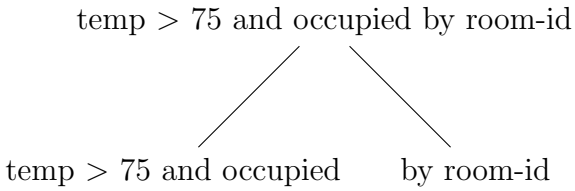
7.1 Segmentation

The three keywords the system takes into account are "and", "by", and "event". To demonstrate how segmentation of a query using the "and" and "by" keywords works in this system, take the following example query: "temp $>$ 75 and occupied by room-id".

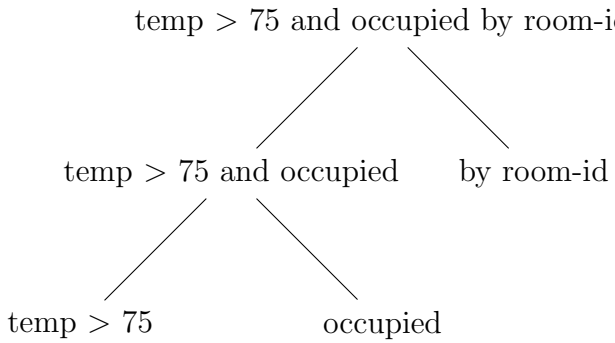
In this scenario, the assumption must be made that the user has created a label "occupied". For the purposes of examining the posed query, one can assume that the label "occupied" was defined as meaning "occupancy $>$ 0" by the user of the system.

"and" separates simple and conditional queries, while "by" adds a grouping constraint

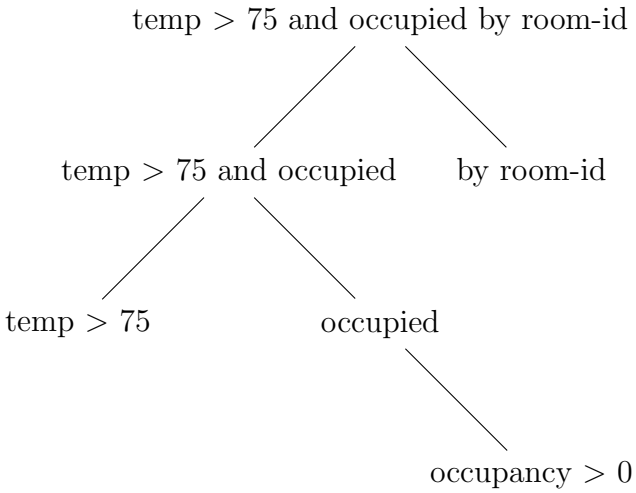
on the resulting returned values. Since the grouping is done on an already returned set of results, the grouping clause must be evaluated afterwards and is split from the rest of the query first:



The grouping clause "by room-id" cannot be broken down further. The clause "temp ¿ 75 and occupied" is then split on the keyword "and":



"temp > 75" is in itself a conditional query and cannot be broken down further, however "occupied" is neither a simple or conditional query nor is it a grouping clause, so it will be evaluated against the labels currently collected in the system and translated to its corresponding conditional query:



Once the original query has been segmented as described, the following query segments are extracted: "temp > 75", "occupancy > 0" and "by room-id", and can be evaluated.

7.2 Evaluation

Once a query has been segmented into its relevant subqueries, each subquery can then be evaluated. The order of evaluation is as follows: simple and conditional queries, and then finally grouping statements.

For a simple query, the search terms are compared against the WordNet extended list of attribute terms available for each sensor in the system. Sensors are ranked simply by the number of cooccurrences between the the search terms in the user query and the the attribute terms for each sensor. Assuming the simple query "temp", some sample sensors and their corresponding attributes are as follows:

Sensor Name	**Attributes**
Sensor A	site:SDH, supply air temp setpoint, room-id:302
Sensor B	site:SDH, supply air temp setpoint
Sensor C	site:Rice, occupancy sensor, room-id:504
Sensor D	site:Rice, occupancy sensor, room-id:404
Sensor E	site:Rice, occupancy sensor, room-id:304
Sensor F	site:Rice, occupancy sensor

By searching "temp" the sensors retrieved from the set would be "Sensor A" and "Sensor B", due to the inclusion of the word "temp" in their attribute list.

Conditional queries are evaluated similarly to simple queries, but once the correct sensors are determined, their most recent values must be checked to identify if they match the conditional constraint. If the "temp" were made into conditional query "temp >= 75" then the two sensors "Sensor A" and "Sensor B" would again be returned, but then the further step of evaluating their most recent values would need to be completed. Their time-series data may look like the following:

	Sensor A	Sensor B
04/04/2018 10:00:01am	74	69
04/04/2018 10:00:02am	74	68
04/04/2018 10:00:03am	73	69
04/04/2018 10:00:04am	73	69
04/04/2018 10:00:05am	74	70
04/04/2018 10:00:06am	74	70
04/04/2018 10:00:07am	75	71
04/04/2018 10:00:08am	75	70
04/04/2018 10:00:09am	75	69
04/04/2018 10:00:10am	75	69

In this scenario, a query of "temp >= 75" at the time 04/04/2018 10:00:10am would then only return "Sensor A", because that is the only sensor that meets both the "temp" attribute requirement as well as the conditional requirement of having a value greater than or equal to 75 at query time.

Simple and conditional queries with the addition of a grouping statement or conditional grouping statement are evaluated in a similar way to conditional queries. Sensors with relevant attributes are identified first, and then are further reduced by examining grouping statement requirements. Assuming the same set of sensors, the query "occupancy by room-id" would only return "Sensor C", "Sensor D", and "Sensor D". While "Sensor F" also contains the term "occupancy" in its list of attributes, it would not be returned because it does not have an associated "room-id".

Specifying a particular grouping value works in the same way as well. The query "occupancy by room-id:504" would only return data from "Sensor C", as it is the only one that matches all the constraints specified by the query, namely that it contain "occupancy" in its attribute list, that it also contains "room-id" in its attribute list, and that the "room-id" attribute has a value of "504".

Overall, the query evaluation process depends heavily on the availability of attributes to function effectively. Given a large starting data set with very few attributes will limit the number of query terms that can be used to search the data. The more relevant attributes that are available for search, the more successful query evaluation will be in returning relevant results. Attributes that are more general and can be extended through the use of synset expansion are also beneficial to the overall relevance of results returned.

7.3 Query Evaluation Testing

In order to evaluate the effectiveness of the query evaluation of the system, sample queries were created that reflect the possible needs of a member of a building management team. There are 25 queries, in a variety of formats. The query types include simple queries, conditional queries, simple queries with grouping statements, conditional queries with grouping statements, as well as chained variations of all the query formats.

The ground truth in this evaluation is determined using manual database queries. Given prior knowledge of the data and knowledge of the intention of each of the queries, the exact sensors relevant to each query can be determined.

Evaluation of the queries was performed by comparing the ground truth sensor results for each of the queries with the results returned by the system using the same queries. Then the precision, recall, F-measure, and mean averaged-precision (MAP) were calculated for the two sets of query results. Figure 7.1 displays the results of the evaluation. Figure 7.2 displays the average precision of results for each query.

Precision	**Recall**	**F-Measure**	**MAP**
.181191	.660417	.284365	.227215

Figure 7.1: Query Evaluation Results

Figure 7.2 shows that while shorter queries often score average precision greater than zero, longer queries seem to have a higher probability of having high average precision scores. Longer, more specific queries seem to have a much greater variance in terms of average precision. Shorter more general queries do retrieve some relevant results at a higher rate, but average precision scores do not go as high as they do with larger queries.

This is due to the fact that that grouped conditions such as "by room-id = 180" reduce the search space very effectively. Adding more search terms prior to a grouping condition only serves to increase the search space, making the chances of returning relevant results inherently lower. Increased terms prior to the introduction of a grouping condition can also serve to include a larger set of irrelevant sensors. This can mean sensors that share one of the words in the query search terms, but not a sufficient amount to actually deem them relevant in the context of the search.

Query	**Average Precision**
"temp < 70"	0.242818
"temp by room-id = 180"	0.333333
"fan > 0 by vav"	0.000000
"temp set point by room-id = 181"	0.500000
"air pressure by room-id = 280"	1.000000
"alarm by room-id = 288"	1.000000
"fan speed by ahu-id = 1"	0.027778
"heating by floor = 1"	0.203571
"loop gain by room-id = 514"	1.000000
"operation status by floor"	0.000000
"fan speed by site = SOD"	0.018868
"air volume by site = SOD"	0.000000
"room temp by room-id = 535"	0.333333
"occupancy > 0 by room-id"	0.000000
"occupancy < 1 by room-id"	0.000000
"temp > 75 by site = SDH"	0.000000
"supply fan > 0 by site = SDH"	0.291667
"temp by room-id = 252"	1.000000
"exhaust fan > 0"	0.074554
"air pressure"	0.067625
"temp > 72 and occupancy sensor > 0 by site = SOD"	0.000000
"avg air pressure discharge"	0.012093
"window temp"	0.013559
"exhaust fan by site = SOD"	0.308817
"building wide sensor"	0.032258

Figure 7.2: Query Evaluation Results by Query

Chapter 8

Conclusion and Future Works

In this thesis a new system is proposed for the natural language querying of internet of things time series sensor data and metadata. The system allows users to use natural language to search across sensor data through expanded metadata attributes, as well as add constraints to search results based on numeric bounds on the sensor data. It also allows for users to define additional attributes and define relationships between text phrases and attribute and data value pairs.

Results of the query evaluation test show that the system is able to retrieve relevant sensor data through the use of natural language querying fairly well. Results vary somewhat depending on the number of search terms, and increase in relevance depending on how particular users are in their search specifications.

The system allows search by natural language query and also allows users to associate search terms with particular time-series data shapes. Given a named time-series shape, the system can return similar shapes in the time-series data set.

Results of the event matching feature of the system user study show that the system can successfully identify consistent or regular time-series shapes, but struggles to provide relevant matching time-series data shapes when ground-truth shapes contain a significant amount of distinct small hyper-local features or are significantly defined by their directionality.

Future works can improve on the querying of described system by increasing its ability to expand IOT sensor metadata terms in the context of building management. The event matching feature of the system could be improved by examining smaller windows of the events during the creation of feature vectors and including a wider array of features to match against relevant events.

Bibliography

[1] Shane Bergsma and Qin Iris Wang. *Learning Noun Phrase Query Segmentation*. In Proceedings of the Conferenceon Empirical Methods on Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL 07). 819-826.

[2] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. *Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 1321-1331.

[3] Bin Tan and Fuchun Peng. *Unsupervised Query Segmentation Using Generative Language Models and Wikipedia*. In Proceeding of the 17th international conference on World Wide Web (WWW '08). 347-356.

[4] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. *A Bag-of-Features Framework to Classify Time Series*. IEEE Trans Pattern Anal Mach Intell 25(11). 2796-2802.

[5] T. Warren Liao. *Clustering of time series data-a survey*. Pattern Recognition Volume 38, Issue 11, November 2005. 1857-1874.

[6] Abdullah Mueen, Eamonn Keogh, and Neal Young. *Logical-Shapelets: An Expressive Primitive for Time Series Classification*. KDD, 2011, 1154-1162.

[7] Patrick Schafer and Ulf Leser. *Fast and Accurate Time Series Classification with WEASEL*. [Online]. Available: https://arxiv.org/abs/1701.07681

[8] Lexiang Ye and Eamonn Keogh. *Time Series Shapelets: A New Primitive for Data Mining*. KDD, 947-956.

[9] Rohit J. Kate. *Using dynamic time warping distances as features for improved time series classification*. DataMining and Knowledge Discovery, online first, 2015.

[10] Wong, Li, and Wang. *Intelligent building research: a review.* Automat Constr2005;14(1):143-59.

[11] Flax. *Intelligent Buildings* IEEE CommunicationsMagazine, (1991 (April)) 24-27.

[12] Miro Mannino and Azza Abouzied. *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. 388:1-388:13. ACM, New York, NY, USA, 2018.

[13] Harry Hochheiser and Ben Shneiderman. *A Dynamic Query Interface for Finding Patterns in Time Series Data* In Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems (CHI03). 522-523, 2003.

[14] Wikipedia, the Free Encyclopedia. *File:Wikipedia-n-zipf.png.* Wikipedia, the Free Encyclopedia, 04/06/2019. Web.

[15] Melanie Swan. *Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0.* J. Sens. Actuator Netw.,vol. 1, no. 3, 217-253, 2012.

[16] Dave Evans. *The Internet of Things.* 2011. Cisco Blog. Available online: http://blogs.cisco.com/ news/the-internet-of-things-infographic/.

[17] Cisco. *Cisco Global Cloud Index: Forecast and Methodology, 20162021* Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html.

[18] Deborah Snoonian. *Smart buildings.* IEEE Spectr., vol.40, no.8, 18-23, Aug. 2003.

[19] Miriam Fernndez, Vanessa Lpez, Marta Sabou, Victoria Uren, David Vallet, Enrico Motta, Pablo Castells. *Semantic search meets the Web.* Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008). 253-260.