Classification and Annotation of Open Internet of Things Datastreams

Abstract. The Internet of Things (IoT) is springboarding novel applications and has led to the generation of massive amounts of data that can offer valuable insights across multiple domains: Smart Cities, environmental monitoring, healthcare etc. In particular, the availability of open IoT data streaming from heterogeneous sources constitute a novel powerful knowledge base. However, due to the inherent distributed, heterogeneous and open nature of such data, metadata that describe the data is generally lacking. This happens especially in contexts where IoT data is contributed by users via cloud-based open data platforms, in which even the information about the type of data measured is often missing. Since metadata is of paramount importance for data reuse, there is a need to develop intelligent techniques that can perform automatic annotation of heterogeneous IoT datastreams. In this paper, we propose two novel IoT datastream classification algorithms: CBOS and TKSE for the task of metadata annotation. We validate our proposed techniques through extensive experiments using public IoT datasets and comparing the outcomes with state-of-the-art classification methods. Results show that our techniques bring significant improvements to classification accuracy.

Keywords: Internet of Things · Classification · Open Data · Metadata · Sensors

1 Introduction

The connected future is set to be dominated by a significant growth of the heterogeneous Internet of Things (IoT) devices that are estimated to surpass the total number of mobile phones by 2022 [1]. The IoT (underpinned by the principles of the Internet) has led to a phenomenal increase in the generation of IoT data that are contributed by users across the globe and can be publicly accessed via the Internet. In fact, a lot of data for such domains is available in open access forms from public platforms [14] either official, such as the Environmental Protection Agency (EPA) (`https://www.epa.gov/`), or user-produced, such as ThingSpeak (`https://thingspeak.com/`) and, until few years ago, Xively (`https://xively.com`) and SparkFun (`https://www.sparkfun.com/`). Such "open data" is a powerful source of information for developing novel IoT applications in domains such as smart cities, defense, healthcare and environmental monitoring, to name a few. A fundamental requirement in successfully re-purposing such open IoT data, in order to enable interoperability as envisioned by Semantic Web 3.0, is to be able to automatically characterize its metadata i.e. information such as observation type (e.g. temperature, humidity), unit of observation (e.g. Celsius,

Fahrenheit), location etc. However, as validated by a recent study in the literature [20], most publicly available IoT data lack availability of such accurate metadata and, in most cases, even the observation type is unclear (i.e. what is actually being measured).

Open IoT datastream[1] classification is a novel problem and has not been addressed well in the literature. Publicly available open IoT data is very diverse with varying degree of availability and accuracy of metadata (partial to none). In order to support the vision of the IoT-driven Web 3.0 – i.e. moving from streaming data to smart data (data that is well described, allowing interoperability and re-purposing across several domains) – in this paper, we propose algorithms to tackle the challenge of annotating open IoT datastreams produced by *heterogeneous* IoT environments. Such heterogeneity, attributed to the nature of the Internet that allows everyone to contribute, is due to diversities in the way data is captured (e.g. location, the accuracy and range of the sensor producing the IoT datastream, non-alignment in time, etc.) and imposes additional challenges in solving the IoT datastream classification and annotation problem. In particular, our focus is on classifying the observation type of an IoT datastream under the following two cases 1) lack of metadata and 2) partial, incomplete or inaccurate textual metadata e.g., an IoT datastream that produces temperature may be described by the user using non machine interpretable names such as "temp", or "t1", or "T (°C)". To this end, this paper makes the following contributions:

- A novel Class-wise Bag of Summaries (CBOS) approach, based solely on the numerical characteristics of the sensor readings.
- A novel Top-k Sequential Ensemble (TKSE) approach, which uses a combination of any available textual metadata describing the datastream and numerical summaries.
- Extensive experimental evaluations to validate the accuracy and the performances of the proposed approaches against the current state-of-the-art approaches in time series classification.

Our proposed sequential ensemble approach is a pioneering effort in the classification of IoT datastream that considers a combination of the numerical characteristics and partially available metadata of the IoT datastream. The rest of the paper is organized as follows: Section 2 provides the state-of-the-art in similar data classification and annotation tasks, Section 3 introduces our IoT data classification problem and describes the proposed CBOS and TKSE approaches, Section 4 describes our experimental design and the IoT datasets used in our evaluations, Section 5 presents the results of our experimental evaluations and, finally, Section 6 concludes the paper with recommendations for future work.

[1] We use the term "datastream" to refer to an individual stream of data, together with its metadata, produced by a sensor on an IoT device. In contrast, we refer to "IoT data" as the collection of all the data produced by several IoT devices.

2 Related Work

IoT datastreams are ordered sequences of sensor readings which can naturally be seen as attributes that form what in the literature is called a "time series". Time Series Classification (TSC) problems, indeed, differ from ordinary classification problems in that features are ordered (not necessarily in the dimension of time). Within the last years, several TSC approaches have been proposed [4] as an alternative to the one-nearest-neighbor (1NN) approach using the simple pointwise Euclidean Distance as a similarity measure between series. The common agreement accepted among researchers as a "hard to beat" standard distance measure between series has been Dynamic Time Warping (DTW) [5], for which several alternatives have been proposed in order to contrast its high time complexity [10]. The above mentioned methods consider the whole series, since they extract series similarities by pointwise comparison. Other recent TSC approaches aim to find a subsequence, called "shapelet", yielding the highest information gain that can discriminate among classes and using a tree-based classification algorithm [22]. Such *shapelet-based approaches* have been improved over time, especially due to their high time complexity [16]. Finally, a third type of well-performing methods, namely *dictionary-based approaches*, split the time series in time windows and extract patterns out of each window as new features. Such methods tend to be faster than the aforementioned ones due to feature numerosity reduction. Examples of such methods are Bag-of-Patterns (BOP) [13], which uses piecewise aggregate approximation (PAA) through Symbolic Aggregation approXimation (SAX) words [12] and *Bag-of-SFA-Symbols* (BOSS) [19], which encodes subsequences through Discrete Fourier Transform (DFT).

The classification of IoT data stemming from heterogeneous IoT devices has been already considered in literature. In [7], the authors imply a PAA-based approach which treats the sensor data classification as a dictionary-based TSC problem and uses interval slopes as features. A different approach has been taken in [6], which aims to assess open IoT data validity by comparing it with a certified ground truth. In [14] The data streams are classified only on top of the metadata provided, i.e. the user-assigned stream name. However, no approach currently employs ensemble methodologies in order to consider varying degree of metadata quality and availability for classification; i.e. if no metadata is available it relies only on the numerical characteristics of data, whereas if limited and inaccurate metadata in the form of text is available, it uses a combination of numerical data and textual metadata.

3 CBOS and TKSE: Approaches for Classification and Annotation of IoT Datastreams

In this section we first formulate the problem of IoT datastream classification and annotation and then present our proposed algorithms: Classwise Bag of Summaries (CBOS) and Top-k Sequential Ensemble (TKSE). Our algorithms are based on generic data mining and machine learning approaches that are extensible to other problem domains.

3.1 Problem Formulation

Formally, we are given n IoT sensor datastreams [2] $\mathbf{NS} = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n}\}$, each of which can be represented as an ordered tuple that resembles a time series with metadata. That is, $\forall i \in [n] : \mathbf{S_i} = \langle \mathbf{D}_i, r_{i,1}, r_{i,2}, \ldots, r_{i,m} \rangle$ where $[n]$ represents the interval from 1 to n, $\mathbf{D}_i$ denotes a dictionary of metadata on sensor stream i with or without annotations, and each $r_{i,j}$ is a numerical sensor reading (typically a double value) from i-th stream at time τ_j. For instance, consider a temperature stream $\mathbf{S_i}$ with annotated metadata 'name' and 'description' and the annotation for metadata 'type' missing. Then $\mathbf{D}_i = \{(name : \text{"}outdoorTemp\text{"}), (description : \text{"ESP8266 with DHT11"}), (type : \text{""}), \ldots\}$. Without loss of generality and for simplicity, we assume datastreams being of the same length m (thus, **NS** can be viewed as a column-ordered matrix of size $n \times m$) and time intervals $\{\tau_{j+1} - \tau_j\}$ between two consecutive readings in each datastream being near-uniform. As, in our scenario, the textual metadata 'type' values that indicate the classes of datastreams are missing, our *goal* is to recover the datastream class (the 'type' value in $\{\mathbf{D}_i\}$) from the information of sensor readings and/or other aiding metadata. To achieve this, types or classes are mapped to numerical labels $\{y_i\}$, i.e. datastreams become of the form $\{\mathbf{S_i}, y_i\}$. Specifically, streams with known classes in **NS** form the training set of size t, otherwise the testing set of size $n - t$. From the training set, existing classes are mapped to c distinct numerical labels $\mathbf{L} = \{l_1, l_2, \ldots, l_c\}$[3] and, normally, $c \ll n$. In training phase classifiers are built for the later testing phase to inference, from **L**, which class each missing y_i in the test set belongs to. Throughout the paper, we use bold symbols to denote multi-dimensional data structures such as vectors, matrices and dictionaries.

For the above formulated classification and annotation problem, we first propose CBOS, a preliminary data mining solution based on statistical summaries of numerical readings $\{r_{i,j}\}$ – what we call as bag of summaries (BOS). Then, we introduce the best performer TKSE that sequentially and incrementally ensembles multiple classifiers trained from the textual metadata and numerical summaries respectively.

3.2 Classwise bag of summaries (CBOS)

Our first approach took place in investigating TSC algorithms for inferencing the type of data measured by each datastream. Given the noisy fashion of each datastream, we attempted to tackle the problem using Bag-of-Patterns-Features (BOPF) [11], a recent phase-invariant dictionary-based approach based on SAX words to encode local patterns. In essence, such approach splits the z-normalized time series through a sliding window in time-ordered chunks and extracts a SAX word by aggregating symbolic mappings of the chunks' mean values. Features are then represented by the distribution of SAX words across sliding windows. In the training set, features are first ranked by their global ANOVA-F value (the

[2] we use the term IoT datastream and IoT sensor datastream interchangeably

[3] For convenience, we can treat 'class' and 'label' equivalently.

mean variance of a feature value across the whole dataset divided by the sum of within-class variances of that feature) and then by means of a leave-one-out cross validation together with the computed ANOVA-F ranking, a subset of features yielding the highest cross-validation accuracy is selected.

Although TSC approaches are widely known to work well on homogeneous datasets, such algorithm, as well as other TSC approaches we attempted, did not perform the way we expected on IoT datasets (as shown later in Section 5) due to heterogeneity in IoT data. To cope with this phenomenon, after some experimentation, we observed that a set of global statistical summaries of each non-normalized datastream can be highly discriminative for certain classes. For example, pressure values tend to have a mean around 10,000 hPa, thus the mean has a higher information gain since is far from any other; likewise, values like RSSI (the received signal strength by a wireless appliance) are often negative, thus the minimum value of the datastream is likely to be indicative. Hence, we adopt a set of meaningful statistical summary features $\{F_1, F_2, \dots F_f\}$ – defined as *bag of summaries* (BOS) – i.e. mean, median, minimum value, maximum value, root mean squared error (RMS) and standard deviation. Moreover, we propose the algorithmic approach CBOS, built on BOS features, that differentiates classwise features, i.e. each class of instances is trained to have its own/different discriminative subset of BOS features. This is in contrast with the aforementioned BOPF approach, where identical global features are shared by all classes. The proposed CBOS algorithm is presented in Algorithm 1.

Algorithm 1 CBOS Algorithm

Require: Training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \dots, (\mathbf{S_t}, y_t)\}$, test example $(\mathbf{T}, y)$
Ensure: $y \in \{1, 2, \dots, c\}$
{TRAINING PHASE}
$\{(\mathbf{F_1}, y_1), \dots, (\mathbf{F_t}, y_t)\} := ExtractSummaryFeatures(\mathbf{TRAIN})$
for $i := 1$ **to** c **do**
    $\mathbf{AF_i} := NormalizedCANOVA(i, \{(\mathbf{F_1}, y_1), \dots, (\mathbf{F_t}, y_t)\})$
    $\mathbf{Cen_i} := CalculateCentroids(\{(\mathbf{F_1}, y_1), \dots, (\mathbf{F_t}, y_t)\}_i)$
end for
$h^* := LeaveOneOutCV(\{(\mathbf{F_1}, y_1), \dots, (\mathbf{F_t}, y_t)\}, \mathbf{AF}, \mathbf{Cen})$

{TESTING PHASE - 1NN}
for $i := 1$ **to** c **do**
    $d_i := 0$
    for $j := 1$ **to** h^* **do**
        $d_i := d_i + \|(\mathbf{F}[j] - \mathbf{Cen_i}[j]) \cdot \mathbf{AF_i}[j]\|$ # $\mathbf{F}$ is the feature value vector of $\mathbf{T}$
    end for
end for
return $y := \arg\min_i d_i$

In Algorithm 1, instead of ranking global features by ANOVA-F values, we compute *classwise* ANOVA (CANOVA) distributions $\mathbf{AF}$ on the training set

(line 4), where $AF_{i,j}$ is the global variance of feature F_j divided by the local variance of F_j among instances of class i ($i \in [c]$ and $j \in [f]$). Moreover, values in $\mathbf{AF_i}$ are normalized into *weights* (for our later weighted cross validation and testing) by forcing them to sum up to 1. Similar to BOPF, both our cross validation and testing phases rely on a simple 1NN feature distance classification against the training class centroids $\mathbf{Cen_i}$ (line 5), obtained by averaging out the BOS feature values within the same class. The main difference is our leave-one-out cross validation (line 7), which performs classwise discriminative feature selection on all classes, i.e. it incrementally tries the $h \in \{1, 2, \ldots, f\}$ highest CANOVA-ranked features of each class and eventually finds the best h^* that yields the maximum cross-validation accuracy. For each class with its respective centroids and CANOVA values, the distance between a new or cross-validated example and the centroids of such class is calculated as their sum of feature distances *weighted by* the respective feature CANOVA values (lines 10 to 16).

3.3 Top-k sequential ensemble (TKSE)

Within the scope of classification on data streams and time series, ensemble algorithms have been shown to be effective and able to capture different facets of the type of data [18]. However, most of the existing ensemble algorithms are designed in a parallel fashion, in that a number of classifiers are built and trained on the original data and the class of an unseen example is typically guessed via majority vote. In our case, as stated in Section 3.1, IoT datastreams may come with partial and inaccurate metadata (e.g. the ThingSpeak dataset in Section 4.1), which can provide a powerful source of information from a different dimension, for instance the dimension of the natural language. For such reason, we propose a novel *sequential ensemble* of classifiers that sequentially combines the text-based Natural Language Processing (NLP) and numerical value-based classification techniques (on the metadata and sensor readings respectively), so that they both contribute in classifying a datastream enriched with annotated metadata. In particular, our proposed *Top-k Sequential Ensemble* (TKSE) algorithm aims to independently train two or more classifiers of different nature and then classify a new example in a pipeline, rather than doing it in parallel.

Suppose that two classifiers Γ_1 and Γ_2 can be independently trained on the same training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \ldots, (\mathbf{S_t}, y_t)\}$ with training/ground truth classes $|\mathbf{L}| = c$ and $\forall i \in [t] : \mathbf{S_i} = \langle \mathbf{D}_i, r_{i,1}, r_{i,2}, \ldots, r_{i,m} \rangle$. Then, for an unseen example $(\mathbf{T}, y)$, these classifiers' Top-1 [4] predicted classes are $y_1 = \Gamma_1(\mathbf{T}, 1, \mathbf{L})$ and $y_2 = \Gamma_2(\mathbf{T}, 1, \mathbf{L})$ respectively. During testing, TKSE instead first applies the classifier $\Gamma_1(\mathbf{T}, k, \mathbf{L})$ that outputs Top-k ranked classes $\subseteq \mathbf{L}$ (this can also be viewed as a filtering classifier) based on learning from annotated textual metadata. Then, these output classes from Γ_1 are fed as the input class labels of Top-1 Γ_2 trained from sensor readings. Therefore, the final ensembled prediction result becomes $y = \Gamma_2(\mathbf{T}, 1, \Gamma_1(\mathbf{T}, k, \mathbf{L}))$. Note that if $k = 1$ then TKSE reduces

[4] In many cases, classifiers produce a (probabilistic) rank of classes based on classification accuracy and the best ranked class is selected as the final predication.

to $\Gamma_1(\mathbf{T}, 1, \mathbf{L})$, and if $k = c$ it reduces to $\Gamma_2(\mathbf{T}, 1, \mathbf{L})$. If we have more useful annotated metadata, TKSE can then extend to a chain of Θ classifiers as below:

$$y = \Gamma_\Theta(\mathbf{T}, 1, \Gamma_{\Theta-1}(\mathbf{T}, k_{\Theta-1}, \ \ldots \ (\Gamma_1(\mathbf{T}, k_1, \mathbf{L})) \ \ldots \ ))$$

where $\forall\theta \in \{2, 3, \ldots, \Theta\} : 0 < k_\theta < k_{\theta-1}$ and $k_1 < c$.

In all our experiments we use as Γ_1 a simple supervised dictionary-based NLP classifier, which we will refer to as a Dictionary Damerau-Levenshtein NLP (DDL-NLP) classifier, first introduced in [14]. The algorithm focuses on the similarity of the metadata 'name' attributed to data streams as a classifying parameter. Algorithm 2 outlines the proposed TKSE algorithm: in training phase, a "dictionary" for each class L_j is constructed in the form of Bag-of-Words (BOW_j) including all stream names in metadata attributed to data streams within the same class (line 2); in testing phase, for each class L_j, the classwise minimum edit distance $d_j = min\{ed(w, s) \,|\, s \in BOW_j\}$ of a testing example w is computed (lines 4-5). For the distance function ed, we leverage the Damerau-Levenshtein edit distance [8] normalized by the maximum length between the two words. The algorithm then picks the closest classes through the TOP-k smallest distances among $\{d_1, \ldots, d_c\}$ (line 7), and subsequently inputs these to a second vanilla machine learning classifier Γ_2 (line 8) such as decision tree, random forest and SVM (as experimented in Section 5.1) trained on all the BOS features.

In order to properly combine two classifiers and achieve the best accuracy, the optimal value of k has to be determined, since, an inappropriate k would impact negatively on the performance by either missing many classes or introducing much noise. The straightforward approach would try all *discrete* values of $k \in [c]$ and choose the value k^* which yields the highest accuracy in k cross-validation rounds of Γ_2 on the training set, however, such method could be slow when c is large. This is also unlike applying stochastic gradient descent on approximating continuous non-convex functions. Instead, we perform a faster logarithmic search heuristic as a simple approximation: first let ACC_k denote the cross-validation accuracy for the chosen k, then from 1 to k we incrementally try values $\{2^0, 2^1, 2^2, ..., k\}$ and pick $k' = 2^p$ with the highest $ACC_{k'}$. As intervals between $[2^{p-1}, 2^p]$ and $[2^p, 2^{p+1}]$ might get larger, we can then recursively

Algorithm 2 TKSE Algorithm with Γ_1 as DDL-NLP

Require: Training set $\mathbf{TRAIN} = \{(\mathbf{S_1}, y_1), \ldots, (\mathbf{S_t}, y_t)\}$, test example $(\mathbf{T}, y)$, k
Ensure: $y \in \{1, 2, \ldots, c\}$
1: **for all** $(\mathbf{S}_i, y_i) \in \mathbf{TRAIN}$ **do**
2: $\quad BOW_{y_i} \leftarrow \mathbf{D_i}('name')$ $\quad \#\ \mathbf{S_i} = \langle \mathbf{D}_i, r_{i,1}, r_{i,2}, \ldots, r_{i,m} \rangle$
3: **end for**
4: **for** $j := 1$ **to** c **do**
5: $\quad d_j = min\{ed(\mathbf{D}('name'), s) \mid s \in BOW_j\}$ $\quad \#\ \mathbf{T} = \langle \mathbf{D}, r_{i,1}, r_{i,2}, \ldots, r_{i,m} \rangle$
6: **end for**
7: $\mathbf{C} = \{L_j \mid d_j \in kmins\{d_1, \ldots, d_c\}\}$ $\quad \#\ kmins$ picks the k lowest values
8: $y = \Gamma_2(\mathbf{T}, 1, \mathbf{C})$

perform the above logarithmic guesses in these intervals until they are small enough and find the overall best guess $k^* = \arg\max_{k'} ACC_{k'}$.

4 Experimental Design

In this section we describe in detail the datasets we used to validate the proposed approaches, our experimental objectives and design consideration.

4.1 Public Open IoT Datasets

We chose a combination of unannotated and partially annotated open source IoT datasets, namely: *The Swiss Experiment* dataset from [2] and a dataset we extracted from the public channels on the *ThingSpeak* cloud platform [3] in order to validate the performance and accuracy of the proposed approaches.

The Swiss Experiment is a platform that allows publishing environmental sensor data located within the Swiss Alps mountain range on the web in real-time. Data is highly noisy, comes from different microscopic locations and it is taken within different time spans. The sampling rate is also different among sensors making the phase shift of data series very significant. Neither semantic annotation nor timestamps were originally provided, thus data is unusable as it is, since the only information provided is the numerical datastream. To the best of our knowledge, the Swiss Experiment dataset (for which a manually annotated version is available at `http://lsirpeople.epfl.ch/qvhnguye/benchmark/`) is one of the few heterogeneous datasets used in research for our type of problem [7]. The dataset consists in datastreams measuring 11 different environmental parameters: CO_2, humidity, lysimeter, moisture, pressure, radiation, snow height, temperature, voltage, wind speed and wind direction. Time series are of different length, however, some of the algorithms we have tested require the time series to have the same length, therefore we cut each time series to the length of the shortest stream in the dataset. The dataset is composed by 346 datastreams without metadata and 445 data points each.

ThingSpeak is an online cloud platform to which users can subscribe and push sensor data produced by their personal device onto their personal "channel" through dedicated APIs. Channels are optionally public and are composed by a set of time series (one for each sensor) and partially and mostly inaccurate user-annotated metadata: each channel has a name, a description, a name for each datastream and, optionally, a geolocation in GPS coordinates. Each name (as well as the other metadata) is user-assigned, thus it can be informative as well as useless. Each channel can be updated at any time rate above 10 seconds and the cloud keeps permanently in memory the last 8000 measurements. We built our dataset by scraping the first 500,000 channels through a dedicated HTTP call[5] returning a JSON object. With such call, the metadata and the last 8000

[5] `https://thingspeak.com/channels/\{$ch$\}/feed.json?results=8000\&start=2018-01-0100:00:00`

readings in year 2018 from all the datastreams belonging to the queried channel are retrieved and, subsequently, all the datastreams were made independent from their channel, thus the initial dataset is composed by more than 10,000 datastreams, each including a time series of sensor readings, with associated timestamps, and a set of metadata. We further filtered out private streams, non geolocated streams and streams with less than 5,000 readings in 2018; clustered the rest both by geo-location and time, selecting the most "populated" area and the 24-hour time period for which we have the highest number of measurements. We obtained 1,091 streams having a consistent number of data points on a defined day in an area in central Europe, which includes parts of Germany, Poland, Czech Republic, Slovakia, Hungary and Austria. We clustered, for each series, the data points in 15-minutes time chunks and interpolated the missing points by means of cubic splines. Datastreams have been manually annotated in 16 different classes: non-air temperature, humidity, pressure, wind speed, wind direction, UV, light, sound, air quality, electrical parameter, RSSI, indoor air temperature, outdoor air temperature, heath index, rain index, and dew point. In summary, the dataset is composed by 1,091 datastreams with metadata and 96 data points each. We made the dataset available for download at `https://github.com/stradivarius/TSopendatastreams`.

4.2 Experimental Objectives

In order to evaluate the effectiveness and efficiency of our proposed IoT datastream classification algorithms, we performed extensive experiments against other state-of-the-art approaches on the above open IoT datasets.

For the purpose of experimental evaluation, we use the following three metrics: *Accuracy, Macro averaged F1-Score* – defined in [21] as the harmonic mean of macro averaged precision and recall over all classes – and the *average runtime performance* over the number of folds in seconds. We chose to report the macro averaged F1-Score as the number of instances per class in each of the datasets is unbalanced and, while a high accuracy indicates the overall success, a high F1-Score implies that all classes have been equally considered. Through the usage of such metrics, we performed the following experiments:

Evaluation of Accuracy and F1-Score The objective of this experiment is to validate the proposed algorithms on both our IoT datasets against time-series and BOS-based algorithms from the literature. Each algorithm has been validated through a k-fold cross validation. On Swiss Experiment we used 5-fold cross validation (same as [7]), whereas on ThingSpeak, we used 10-fold, as the number of instances is much higher. The algorithms are tested both on the original and z-normalized time series, as the conventional homogeneous time series analysis often requires normalized data.

Impact of K on TKSE This experiment has the goal of validating the behavior of TKSE for different values of k. It is designed such that our BOS-based algorithms are tested as a second step in TKSE together with the DDL-NLP method (as

described in Section 3.3) in a 10-fold cross validation on the ThingSpeak dataset (with annotated names) on all values of k.

Evaluation of Runtime Performance This experiment has the purpose of validating the efficiency of the considered algorithm in time. They are tested on both datasets and the average runtime over the number of folds in seconds is reported.

4.3 Experimental Design

Our approaches CBOS and TKSE together with our adopted vanilla classifiers: decision trees (C4.5), random forest (RF)[6] (which have been shown to perform well on remote sensing scenarios [15]) and Support Vector Machines (SVM) are compared with the following time series classification (TSC) algorithms and the results obtained with the slope-based algorithm in [7] for sensor data:

The golden standard As sensor reading streams can be easily interpreted as time series, we took into account the most widely used TSC algorithm: *One Nearest Neighbor with Dynamic Time Warping* (1NN-DTW) [17] – considered as the golden standard for TSC [5].

Dictionary approaches Within the scope of TSC, we also included two recent dictionary-based approaches: *Bag-of-Pattern-Features* (BOPF) [11], which our CBOS is based on as outlined in Section 3.2 and it adopts SAX sequence encoding, and *Bag-of-SFA-Symbols* (BOSS) [19], which encodes subsequences through Discrete Fourier Transform.

All experiments are performed on a computer with an Intel Core i7-7700HQ CPU @ 2.80 GHz × 4 and 8GB RAM while running Linux Mint 18.2 64-bit. All tested algorithms are implemented in Python 3.5.2 except the implementations of BOPF [11] and BOSS [19], for which we used the original C++ code provided by the authors.

5 Experimental Results and Discussions

In this section we provide results and insights about the experiments outlined in Section 4.2.

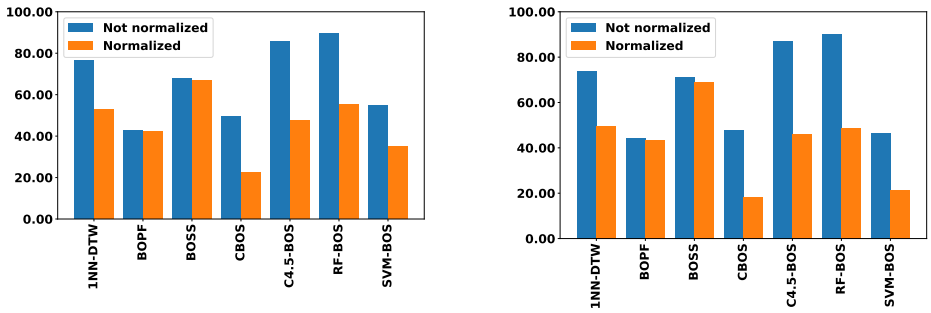
5.1 Evaluation of Accuracy and F1-Score

We evaluated ours proposed algorithms with the ones outlined in Section 4.2 using the Swiss Experiment and ThingSpeak datasets. Both accuracy and F1-Score are reported for the datasets (Swiss Experiment in Figure 1 and ThingSpeak in Figure 2) in their original and z-normalized form. Observing the outcomes of such analysis, it is immediately clear how data normalization causes the loss

[6] Although [9] has considered a more sophisticated RF, this is not the focus in dealing with heterogeneous IoT data.

of important features and hence impacts the accuracy of classification making this a much harder problem. In fact, only BOSS and BOPF achieve similar results for both normalized and unnormalized datasets, in that such algorithms were designed specifically to operate on z-normalized series and some parameters are hard coded to cope with the underlying data. Nevertheless, they still achieve a similar accuracy on the original series. Our first summary-based approach CBOS performed poorly on z-normalized data. This is expected, as z-normalization loses most of the information based on statistical summaries. But, on the non z-normalized data CBOS improves on BOPF, which it is based on, and is later shown to be faster than BOSS. In summary, the above mentioned TSC approaches still do not achieve the desirable IoT data classification accuracy. On the other hand, the golden standard DTW tends to perform better on non z-normalized data, sometimes achieving good results on ThingSpeak. However, this further validates our findings i.e. the trend of the time series is not as indicative as the absolute value ranges in heterogeneous IoT data. These absolute value ranges are better captured when the data is not z-normalized.

Based on this inference, we further found through our experiments that the vanilla machine learning methods performed using our BOS features achieve significant accuracy and F1-Score gains. Our justification for such phenomenon lies behind the fact that, due to heterogeneity in IoT data, different classes exhibit different behavior (and, therefore, bias) in terms of such features. For such reason, tree-based classifiers (C4.5 and RF), built in a way in which the attribution of an example to a class is driven by a sequence of decisions based on the threshold of the feature itself, seem intuitively suitable for open sensor data. This is not the way in which SVM are designed, in fact, they do not seem to work as well. It is also interesting to notice that SVM tend to assign more likely the most populated classes, resulting in a fairly poor F1-Score. Furthermore, on the Swiss Experiment dataset, the authors in [7] have reported on normalized



(a) (b)

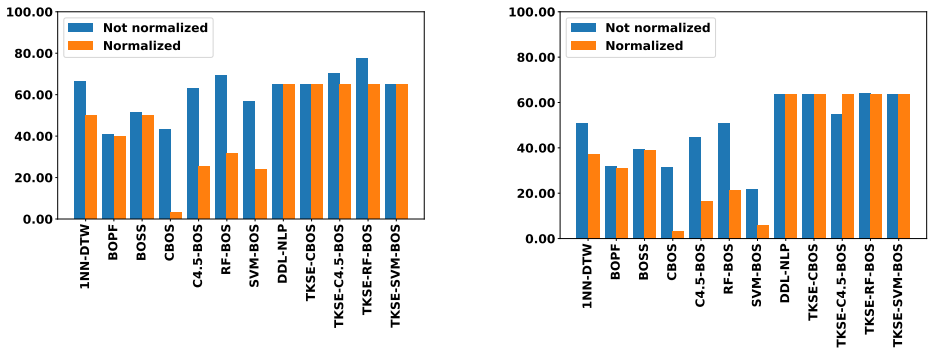
Fig. 1: (a) Accuracy and (b) F1-Score for Swiss Experiment Dataset

data in the form of a pattern-based time series approach, achieving an accuracy of 67.7%, whereas our RF approach on BOS easily achieves above 80%.

By looking at Figure 2 it is possible to see the performances of the DDL-NLP algorithm and the proposed TKSE (which includes DDL-NLP as a first step). DDL-NLP alone has been applied on open datasets previously (data extracted from ThingSpeak and SparkFun) [14] with an accuracy close to 88%, however, such datasets are purely composed by data streams annotated in English, whereas the ThingSpeak dataset presented in Section 4.1 is annotated in different languages and the performances of the DDL-NLP approach alone drop to 65% in both Accuracy and F1-Score. Looking at the bar charts, it is clear that TKSE with tree-based methods (RF and C4.5) bring significant improvements (at $k^* = 4$), whereas all others coupling with DDL-NLP diminish the performances as detailed in the next subsection, demonstrating the important influence of annotated metadata.

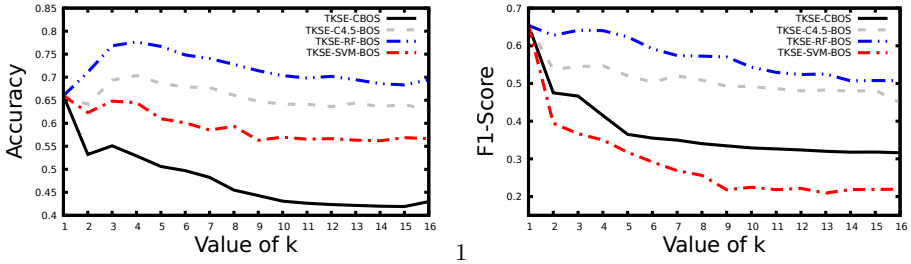
5.2 Impact of k on TKSE

The performance of the proposed TKSE approach has been evaluated via coupling the DDL-NLP classifier with other feature-based classifiers included in our previous test. For the purpose of this experiment we only used original data due to their preserved distinguishing power. For completeness in the illustration we tried each value of k rather than using a recursive logarithmic search. As stated before and shown in Figure 3(a), only tree-based approaches display a performance increase with TKSE, whereas the others tend to be pejorative. It is also interesting to notice how the curves are similar in their trend, in particular they all display a local maximum for $k \simeq 4$ while performance starts dropping for greater values (as more noises are introduced). On the other hand, F1-Score, as shown in Figure 3(b), is not positively affected by TKSE on ThingSpeak, since DDL-NLP is already the highest among the algorithms – this is mostly due to the



(a) (b)

Fig. 2: (a) Accuracy and (b) F1-Score for ThingSpeak Dataset



(a) (b)

Fig. 3: Accuracy (a) and F1-Score (b) of TKSE approaches evaluated on different algorithms. Tree-based algorithms display an evident increase in accuracy.

reason that the proposed TKSE searching/tuning process is designed with the objective of optimizing the classification accuracy instead of the F1-Score. This phenomenon would require our future investigation, but nevertheless TKSE-RF keeps the F1-Score to approximately the same as DDL-NLP alone. Therefore, along with its consistent superior accuracy, TKSE-RF on BOS is by far the best choice among all the algorithms.

5.3 Evaluation of Runtime Performance

We have tested extensively the runtime performances of all the evaluated methods. The results are reported in Table 1 and time is measured in seconds over the cross-validation process divided by the number of the folds (which gives the actual training and testing time over a training and a test set). In particular, the Swiss Experiments has a training set size of 273 and a test set size of 73, while the ThingSpeak dataset has a training set size of 974 and a test set size of 117. It is interesting to notice how algorithms perform differently on such datasets due to the relatively larger number of instances in ThingSpeak and longer series in the Swiss Experiment (e.g. BOPF performs better on the Swiss Experiment, whereas BOSS on ThingSpeak). The slow performance of DTW is expected due to its high worst-case time complexity $\mathcal{O}(n^2m^2)$, where n is the number of stream instances and m is the series length. Other time series based methods mostly have a linear complexity (w.r.t the input size of nm data points): BOPF and CBOS both have a complexity of $\mathcal{O}(nm)$, although, in practice, CBOS is significantly faster, and BOSS has a complexity of $\mathcal{O}(nm^{\frac{3}{2}})$. TKSE-based experiments were performed with logarithmic search that is relatively slower but more accurate. Overall, except for SVM and DTW all the other methods perform well in runtime, and, in particular standalone RF and TKSE-RF flexibly trade off some runtime for classification quality and exhibit better performances than others on both datasets.

[7] This algorithm is written in C++ since the original code provided by the authors has been used – implying stronger runtime baselines to compare with.

Table 1: Runtime performances in training and testing of all the algorithms on both Swiss Experiment and ThingSpeak datasets

	Time in seconds (Swiss Experiment)	Time in seconds (ThingSpeak)
1NN-DTW [17]	5989.694	1528.244
BOPF[7] [11]	5.87	36.589
BOSS[7] [19]	79.028	15.628
CBOS	0.068	0.265
C4.5-BOS	0.045	0.138
RF-BOS	0.589	1.178
SVM-BOS	24.194	845.437
DDL-NLP	-	1.286
TKSE-CBOS	-	11.670
TKSE-C4.5-BOS	-	7.195
TKSE-RF-BOS	-	21.510
TKSE-SVM-BOS	-	5082.260

6 Conclusion

In this paper, we proposed novel algorithms to tackle the challenge of annotation and classification of open IoT datastreams produced from *heterogeneous* IoT environments. In particular, we first proposed CBOS, a bag of summary-based approach to classify IoT datastreams based on numerical characteristic of the underlying IoT datastream. Through experimental evaluations and validation we conclude, although IoT datastreams are reminiscent of time series datasets, due to the heterogeneity in the sensor readings produced by IoT devices, classic TSC approaches perform poorly while vanilla classifiers such as decision trees and random forest based on bag-of-summaries (BOS) perform significantly better when only considering the numerical characteristics of the IoT datastream. Our second proposed algorithm namely TKSE uses a novel sequential ensemble approach to take advantage of both 1) partially available textual metadata that describes the IoT datastream and 2) the numerical characteristics of the IoT datastream. Through extensive experimental evaluations and comparisons with state-of-the-art approaches in the literature, we validated the significant gain in accuracy of the proposed TKSE algorithm while imposing minimal impact on runtime performance. Future work can be devoted into further improving annotation quality with more sophisticated features and ensembles that leverage all useful metadata to some extent.

References

1. Internet of Things Forecast. https://www.ericsson.com/en/mobility-report/internet-of-things-forecast, accessed: 2018-04-30
2. The Swiss Experiment. http://www.swiss-experiment.ch/, accessed: 2018-04-30
3. ThingSpeak Channels. https://thingspeak.com/channels/public, accessed: 2018-04-30

4. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)
5. Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: Proceedings of the 2011 SIAM international conference on data mining. pp. 699–710. SIAM (2011)
6. Borges Neto, J.B., Silva, T.H., Assunção, R.M., Mini, R.A., Loureiro, A.A.: Sensing in the collaborative Internet of Things. Sensors **15**(3), 6607–6632 (2015)
7. Calbimonte, J.P., Corcho, O., Yan, Z., Jeung, H., Aberer, K.: Deriving semantic sensor metadata from raw measurements (2012)
8. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7**(3), 171–176 (1964)
9. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)
10. Jeong, Y.S., Jeong, M.K., Omitaomu, O.A.: Weighted dynamic time warping for time series classification. Pattern Recognition **44**(9), 2231–2240 (2011)
11. Li, X., Lin, J.: Linear Time Complexity Time Series Classification with Bag-of-Pattern-Features. In: Data Mining (ICDM), 2017 IEEE International Conference on. pp. 277–286. IEEE (2017)
12. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and knowledge discovery **15**(2), 107–144 (2007)
13. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**(2), 287–315 (2012)
14. Montori, F., Bedogni, L., Bononi, L.: A collaborative Internet of Things architecture for smart cities and environmental monitoring. IEEE Internet of Things Journal **5**(2), 592–605 (2018)
15. Pal, M.: Random forest classifier for remote sensing classification. International Journal of Remote Sensing **26**(1), 217–222 (2005)
16. Rakthanmanon, T., Keogh, E.: Fast shapelets: A scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining. pp. 668–676. SIAM (2013)
17. Ratanamahatana, C.A., Keogh, E.: Three myths about dynamic time warping data mining. In: Proceedings of the 2005 SIAM International Conference on Data Mining. pp. 506–510. SIAM (2005)
18. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review **33**(1-2), 1–39 (2010)
19. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6), 1505–1530 (2015)
20. Siow, E., Tiropanis, T., Wang, X., Hall, W.: TritanDB: Time-series Rapid Internet of Things Analytics. arXiv preprint arXiv:1801.07947 (2018)
21. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **45**(4), 427–437 (2009)
22. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)