Using LDA to Uncover the Underlying Structures and Relations in Smart City Data Streams

Daniel Puschmann, Payam Barnaghi, *Senior Member, IEEE*, Rahim Tafazolli, *Senior Member, IEEE*
Institute for Communication Systems
University of Surrey
Guildford, United Kingdom GU2 7XH
Email: {d.puschmann, p.barnaghi, r.tafazolli}@surrey.ac.uk

Abstract—Recent advancements in sensing, networking technologies and collecting real-world data on a large scale and from various environments have created an opportunity for new forms of real-world services and applications. This is known under the umbrella term of the Internet of Things (IoT). Physical sensor devices constantly produce very large amounts of data. Methods are needed which give the raw sensor measurements a meaningful interpretation for building automated decision support systems. To extract actionable information from real-world data, we propose a method that uncovers hidden structures and relations between multiple IoT data streams. Our novel solution uses Latent Dirichlet Allocation (LDA), a topic extraction method that is generally used in text analysis. We apply LDA on meaningful abstractions that describe the numerical data in human understandable terms. We use Symbolic Aggregate approXimation (SAX) to convert the raw data into string-based patterns and create higher level abstractions based on rules.

We finally investigate how heterogeneous sensory data from multiple sources can be processed and analysed to create near real-time intelligence and how our proposed method provides an efficient way to interpret patterns in the data streams. The proposed method uncovers the correlations and associations between different pattern in IoT data streams. The evaluation results show that the proposed solution is able to identify the correlation with high efficiency with an F-measure up to 90%.

Index Terms—Data Streams, LDA, Data interpretation, Internet of Things, Data analytics

1 Introduction

With the Internet of Things (IoT) as one of the most rapidly growing technologies, more and more data is produced and has become readily available. According to the annual report 2015 by IBM [1], it is estimated that 2.4 quintillion bytes are published each day by more than 9 billion connected devices. To put this number into perspective, this amounts to more than one hundred million movies in high definition. While this abundance of data is certainly impressive, without processing methods that can deal with the huge volume and velocity, the vast majority of information gets lost in the deluge of data. If we can extract this information from the data, we can build intelligent applications, for example smart city applications that improve the life of citizens in big cities. On average, a citizen of London spends over 100 hours stuck in traffic. This comes with a huge environmental and temporal cost and is a cause for lots of frustration and aggravation. By analysing the data from traffic sensors, we can build solutions that regulate the traffic in a more intelligent way.

Additional to spatio-temporal context dependency, patterns found in the streaming data can also have a significance depending on the output of other data streams at the same time in the vicinity. When correlations are known, these can be used to increase the performance and efficiency applications such as smart grid frameworks for homes or cities [2]

Data from multiple sources poses a key challenge to create actionable information from raw data for developers and researchers alike. The huge variety of features and their characteristics make comparing and combining data from different sources very demanding and complex. This challenge is alleviated in the case of numerical input data where different values or patterns within the data may not translate straightforward into a meaning.

While there are several approaches that can extract information from a single data source for classification and prediction purposes with varying degrees of success, the most existing solutions are limited by not taking into account information from other sources which can enrich the information extraction task by providing complimentary or essential contextual information. When analysing the user behaviour of urban bicycle sharing, Froehlich *et al.* [3] have argued that incorporating contextual information such as weather data into their methodology could greatly improve the prediction rates. Similarly, the prediction rates of tidal events from coastal data in Ganz *et al.* [4] could be improved.

1.1 Motivation

Given various heterogeneous data streams that are produced in the same environment (i.e. same location and at the same time) we are interested in:

- Understanding the meaning of the current output produced by the individual data sources.
- Understand how different data sources are correlated and how they create a meaningful interpretation of the the situation as a whole in this environment.

If we can address the above issues, this allows us to monitor, analyse and predict the situation of various environments (e.g. smart cities) in order to provide more intelligent solutions for smart applications. To interpret the individual data streams, we can extract and process patterns. With these abstract representations of the streaming data we can further process the data streams and analyse their co-occurrences and correlations. Since the data comes from various heterogeneous sources, it has to be first transformed into a common, homogeneous semantic space previous to the correlation analysis. Afterwards we can apply latent topic models to extract the underlying relations between different data sources. These models can then be used to give an abstract representation of the data streams. As the models are generative, we can also use them to fold in incoming data from one or multiple sources to predict the expected patterns within other co-related data sources.
Our approach deals with one specific sub-problem in the IoT domain, namely, the analysis of correlations and dependencies of different data streams to each other. We are using data from a Smart City use case that is measuring the traffic flow in the city and analyse if we can find correlations between the traffic flow and the weather circumstances.

1.2 Overview of Method and contributions

We create a solution which looks at different sensor data streams in an unsupervised fashion and extracts the patterns, hidden factors and their relationships and therefore the underlying structure of the input streams. The solution is applicable for any data stream processing task within the IoT domain.
Initially the data is converted into a more abstract form, representing the patterns found in the data streams. These patterns are then translated into human understandable higher level abstractions by a rule engine. In the IoT domain, data comes mainly in the form of numerical data streams. We apply an abstraction method Symbolic Aggregate Aproximation (SAX), which transforms the numerical time series data into a textual representation. We extend SAX for the analysis of heterogeneous data sources by using multiple alphabets, one for each different kind of data stream. We group the extracted labels from the different features of the data stream together, essentially creating virtual *documents*, which opens up the possibility to use a new range of data analytic methodologies; methods developed for extracting information from text *documents*. These methods are able to extract the underlying relations within textual *documents* through the use of probability distributions and latent variable models. The analysis methods have been successfully developed to extract representative topics from Wikipedia articles and have also been adapted for the use in ontology learning [5].
In the past we have successfully applied SAX for pattern recognition in numerical sensor data as a part of the abstraction chain of transforming raw data into actionable information [6]. With this technique we can pre-process the data to be used by the topic extraction methods and exploit the capabilities of the text analysis methods to uncover hidden relations and structures.
Hereby we face several challenges that have to be overcome. The first challenge is finding the correct format to create the patterns from sensor data that are produced at different points in time and from different types of sensors. If we succeed with the latter, we then have a textual representation of the continuous numerical data streams and are one step closer to applying the latent semantic analysis methods such as Latent Dirichlet Allocation (LDA) [7]). The smallest unit of analysis for this method are text *documents.* The question is how to represent the continuous stream of data as one or multiple *documents*. Therefore the next challenge is deciding which parts of the stream we can interpret virtually as a *document.*
The other challenge is to apply the latent semantic analysis methods on the virtualised *documents* and interpret the result of this analysis in a meaningful way. We also need to design effective methods that can identify the relationships between the heterogeneous data sources.
The structure of the paper is as follows. Section 2 presents the methods used and discusses related work. In Section 3 we provide mathematical background information about the techniques used by the proposed solution. Section 4 explains in detail how the solution and how the individual components are implemented. It also discusses the impact of different parametrisations. Section 5 describes the set-up of the experiments that are carried out and shows the feasibility of the proposed solution. The outputs of the individual components are presented and different parameter selections are compared. Finally, Section 6 summarises the presented work and discusses the research impact and future work.

2 Methods Used and Related Work

In order to provide a common understanding throughout the paper, this section is split into three parts. First we present the methods used for the data discretisation. We then present the latent semantic models and give examples where they have been successfully applied to the existing topic extraction methods to other domains such as audio or image retrieval. We discuss related works which are concerned with finding correlations within different data sources.

2.1 Data Discretisation

One of the common data discretisation methods introduced in 2001 by Keogh *et al.* [8] is Piecewise Aggregate Approximation (PAA) which is used to reduce the dimensionality of streaming data. It splits the input data into window segments which are represented by their mean. For batch input it means that the data will be reduced from an original dimension n into N chunks of size n/N.
SAX is a way to transform continuous, numeric data into discretised values while retaining sequential information. Intuitively SAX transforms the data into patterns, represented by words. The words are generated using a given alphabet and a predefined word size. Initially the data is normalised with a mean of zero and a standard deviation of one [9]. A Gaussian distribution is assumed for the values, which is split into areas of the same likelihood. These are called equi-probable areas and each area is assigned to a

SAX literal from the given alphabet. Using pre-computed breakpoints the PAA coefficients are mapped to the SAX literals. Put together the literals form a string based representation of the pattern, called a SAX word. Section 3 provides a more detailed description of the mathematical background of PAA and SAX.

SAX has significant advantages over other symbolic representation techniques: Applied to the output of PAA, it can capitalise on its dimensionality reduction power. A distance measure can be defined in the symbolic space with a high correlation to the distance measures defined on the raw data. There is no need to have access to all the data before transforming the data into the symbolic representations which makes SAX applicable to streaming data. Because the representation space provides a lower bound for the distance function, the SAX output can be used for improved performance in clustering, classification and indexing tasks. Furthermore, it can be used for motif discovery and to visualise activity from different kinds of data, for example to visualise physical activity from accelerometer data [10].

Ganz *et al.* [4] developed an abstraction scheme which starts from raw sensory data and extracts discretised SAX patterns. The SAX patterns are then associated to events (wherever applicable) by applying Parsimonious Covering Theory (PCT) [11]. Using hidden markov models (HMM) and pre-defined events, temporal relations of the events are then defined. However, the method focuses only on one type of sensor data, producing 23% false positives and 7% false negatives. Ganz *et al.'* experiments indicates that these reported results might be lower boundaries, however the use of other data streams as contextual information might be able to pierce these boundaries.

2.2 Latent Semantic Models

Latent Semantic Analysis (LSA) was developed in 1990 [12] as a way for automatic indexing and retrieval of text documents. Initially the documents are represented by a *term-document matrix* storing the information which terms are present in the *document* and which are not. Using singular-value decomposition, a semantic space is created in which closely associated terms and documents are near each other even if the term did not appear in the *document.* This solves the problem that the same information can be described using a completely different vocabulary and can therefore deliver more meaningful results for *document* retrieval.

One of the main drawbacks of LSA, however, is that it assumes that related terms and documents will end up being close within the created semantic space without strong theoretical foundation.

An adaptation of the method called Probabilistic Latent Semantics Analysis (pLSA) was introduced in 1999 by Hofmann *et al.* [13]. LSA provides a strong statistical foundation for the model. Using an aspect model as a starting point, the joint probability of terms and words is calculated by maximising the likelihood (i.e. Expectation Maximisation) [13].

pLSA has been shown to have a substantial performance increase over LSA and even works in cases where the original method failed such as the CISI dataset [1]. However pLSA still has two major drawbacks:

(i) the number of parameters increases with the number of documents analysed
(ii) documents outside the training set cannot easily be assigned with a probability

The above problems are addressed by Latent Dirichlet Allocation (LDA), a generative probabilistic model that extracts topic models from textual documents [7]. Each *document* is considered to be a distribution over topics and A Dirichlet distribution is chosen as the prior distribution and the posterior distribution for the latent variables is inferred. This way LDA is able to represent a topic by its most probable words. At the same time the underlying hidden structure of the documents is uncovered.

Hu *et al.* [14] successfully applied a variation of LDA called Gaussian-LDA to audio retrieval. The audio files are represented as histograms which can be interpreted as *bag-of-word* representations by regarding the audio feature clusters as words in the dictionary. Having this interpretation gives them the ability to utilise the topic and structure detection powers of LDA. Since the histograms have a shortcoming of treating similar items as unrelated in border regions of the histogram clusters, Hu *et al.* instead treat topics as a Gaussian distribution which allows them to step over the vector quantification step. To evaluate these experiments Hu *et al.* use labelled data and compute precision and recall of their audio retrieval [14].

Both pLSA and LDA have also found successful applications in image classification [15], [16]. There are extensions to LDA to improve the performance of the algorithm for the image classification. For example Rasiwasia *et al.* [17] developed two extensions of LDA, (topic-supervised LDA and class-specific-simplex LDA) for image classification. These methods work by representing images as *bag-of-visual-words*. These representations are then used as input to train the latent models that are employed in LDA and pLSA. LDA has also been applied in the context of IoT. Bian *et al.* [18] have applied the topic model together with sentiment analysis on Twitter data to analyse the public perception of the IoT. . Xin *et al.* [19] have created hierarchical topic model taxonomies on IoT data using LDA. However, both approaches can only be applied to textual information within the and not to numerical data streams within the IoT domain.

2.3 Correlation Analysis in Smart City Data

Because it is a relatively new topic, there is still limited research in the area of finding correlations within urban environment data. Froehlich *et al.* [3] have explored the spatio-temporal patterns of bicycle share usage with relation to urban behaviour. Their prediction model using simple classifiers indicates an lower bound error rate of 12%. However, Froehlich *et al.* argue that taking data from additional sources into account such as weather data to enrich the prediction with contextual information can help break this boundary.

Lathia *et al.* [20] have found out that there are correlations

1. http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/

between the movement of citizens from different areas to the degree of social deprivation. To verify the latter, they use the Pearson Correlation to find a relation between public transport mobility data and community well-being census data. This can help identify linear correlations. However, if the patterns produced from the different sensors correlate in a non-linear fashion, the Pearson correlation is unable to detect these. The information about correlating patterns would therefore be lost.

Correlations between the temperature and traffic flow have been found and modelled with a Poisson distribution model by Jara *et al.* [21]. Their results suggest that both traffic flow and temperature follow a daily cycle. This follows the common perception that temperature rises and falls naturally with the amount of solar radiation. At the same time traffic flow follows a similar pattern as people are going to and from work at times close to sunrise and sundown. A more interesting analysis would be finding out how the traffic flow on the same days at the same time but in different weeks correlate with higher or lower temperature.

Semantic road and Traffic Analytics and Reasoning for City (STAR-CITY) [22] is an integrated traffic analysis system which takes both historical and real-time heterogeneous data into account. The system is able to analyse current traffic conditions and predict future traffic flows. By semantically enriching the data and using a predictive reasoning component, correlations between traffic flows, social media and weather data streams are automatically expressed by creating rules.

Originally deployed in Dublin, the STAR-CITY system has shown limitations in terms of flexibility and scalability when being applied to other cities. Therefore Lécué, *et al.* [23] have introduced the "any-city" architecture which addresses some of these issues to a degree. Since the individual components are strongly connected within the system, applying the solution to other Smart City applications is hindered.

The relationship between weather conditions (specifically solar radiation, temperature and humidity) and power demand has been analysed by Hernández *et al.* [2]. They have developed a smart grid framework which tackles power consumption prediction on small (Smart Grid, Smart Building) and large (Smart City, Smart Environment) scales. Hernández *et al.'s* work shows how known correlations between different data streams can be exploited to improve the efficiency of energy usage and cut down costs.

As discussed above most of the existing solutions for smart city data analysis work on applying different correlation analysis methods. However, using and interpreting multimodal data and integrating information from different sources is always a key limitation in making the existing solutions flexible and scalable. This suggests that more research should be dedicated to analysing multi-sourced data streams to reveal previously unknown correlations, as these can then in turn be exploited in the future.

3 Mathematical Background

In order to provide a common understanding throughout the descriptions and to keep the paper self-contained, we provide ashort mathematical explanation about the techniques used in our solution.

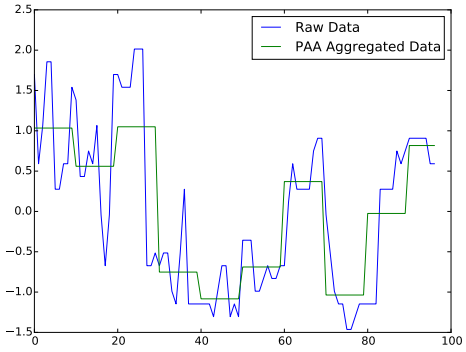


Fig. 1: Sample PAA coeefficient generation ($N = 10$)

3.1 Piecewise Approximate Aggregation (PAA)

PAA can significantly reduce the dimensionality of the data while maintaining lower bounds for distance measures in Euclidean space. PAA exploits that in time series data, the individual data points are usually strongly correlated to their neighbours. With this in mind, a time window containing multiple data points can be represented by its mean without losing too much information about the sequence. A time series subsequence $X = x_1, ..., x_n$ is reduced to a Vector $\bar{X}$ of size N (with $1 \leq N \leq n$), where each PAA coefficient x_i is calculated by equation 1 shown below.

$$x_i = N/n \sum_{j=n/N(i-1)+i}^{(n/N)i} x_j \tag{1}$$

Figure 1 shows the PAA representation of time series data.

3.2 Symbolic Aggregate Approximation (SAX)

SAX [9] was introduced to provide a discretised, string based representation for time series data, where the stream is divided into windows. Each window is initially normalised. The PAA coefficients are then computed (see Section 3.1). SAX determines the corresponding symbols for the coefficient in a way that each symbol has the same probability of being assigned. Under the assumption that normalized time series data follows a Gaussian distribution the probability assumption is achieved by determining the breakpoints $\beta_1, ..., \beta_{n-1}$ which divide the Gaussian curve into n equi-probable areas. Each of these areas is assigned to a literal of the chosen alphabet. Depending on which area the individual PAA coefficients fall, they are assigned with the respective symbol. When put together for a set of windows they form a SAX word.

Figure 2 demonstrates this process. The blue line shows the data that has been aggregated with PAA into ten PAA coefficients. For each coefficient, we assign the literal associated to the area within the breakpoints. The first coefficient in this example is above the topmost breakpoint, so we assign an "a". The second coefficient is between the topmost and the second breakpoint from the top, therefore we assign a "b". Continuing this process, the SAX output for this example is the SAX word "abadedbecb".

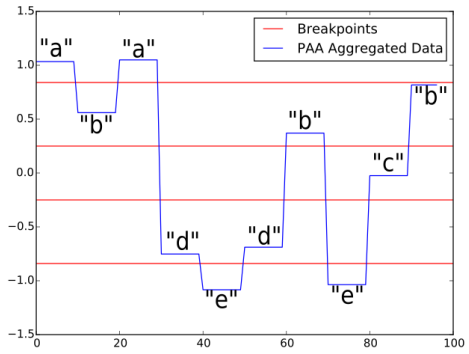


Fig. 2: Sample sax word generation

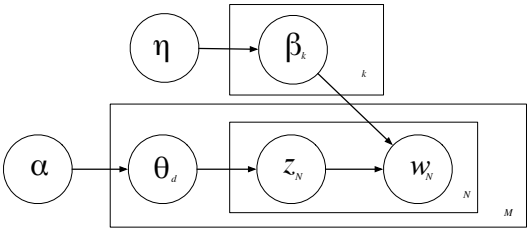


Fig. 3: Plate model representation of LDA [7]

3.3 Latent Dirichlet Allocation

LDA is a generative probabilistic model that is used to assign topics to documents from a given corpus. Figure 3 shows the plate model representation, which helps explaining the components of the LDA model. Here k is the number of topics (fixed on initialisation of the LDA model), M is the number of documents, N is the number of words in the document, which itself is represented by the vector w as a *bag-of-words*. β_k is the multinomial distribution words representing the topics and is drawn from the prior Dirichlet distribution with the parameter η. Similarly the topic distribution θ_d is drawn from a Dirichlet prior with parameter α. z_{ij} is the topic which is most likely to have generated w_{ij}, which is the jth word in the ith document.

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)} \quad (2)$$

Because a Dirichlet distribution is a conjugate prior for multinomial distribution, the posterior distribution is also distributed as a Dirichlet. The exact inference for the posterior is intractable; however, there are several methods which can be used to estimate the solution for equation 2. In the equation $\mathbf{z}$ and $\mathbf{w}$ are written in bold to make clear that they refer to the vector of topics and vector of words respectively. The original LDA [7] proposes the use of variational Bayesian approximation to solve the inference intractability problem. However, Gibbs sampling can also be applied for inference [24]. Gibbs sampling chooses initial topics for each word at random and updates the beliefs iteratively until convergence is achieved.

4 Methodology

This Section demonstrates how our proposed approach for integrating SAX symbols from various data streams and applying LDA to analyse them works. First a general overview over the workflow is given. Then each of the components are described in more detail in individual subsections. Each component is also configurable based on the parametrised initialisation, all of which have influence on the outcome. We carry out our experiments with different configurations and discuss how suitable parameters can be chosen based on the application domain and the input data streams. In general we make an assumption that the required data streams are readily available, meaning that all issues related to stream discovery and accessibility are not in the scope of this work. Furthermore in this paper we do not address any problems related to Quality of Service or Quality of Information in the data streams.

4.1 Overview of the workflow

The overall workflow of our presented method consists of 4 steps which are described in more detail in the following Sections. They are:

- **Data Discretisation and Pattern Recognition** PAA is used to reduce the dimensionality of the data; SAX uses the output to discretise the continuous, numerical data streams and to recognise (lower-level) patterns.
- **Higher Level Abstractions** The information obtained from the pattern together with the statistical values are translated into higher level abstractions based on rules. The rules are explained in Section 4.3.
- **Virtual *Document* Generation** We create virtual documents to group the together the higher level abstractions from different sources within a certain time-frame
- **Latent Dirichlet Allocation (LDA)** We train and incrementally update an LDA model on the virtual documents that are generated from the data streams to identify and extract relations between the higher level abstractions.

The interaction of these components is shown in Figure 4. Initially the data coming from the streams is reduced by PAA. The patterns are generated by applying SAX on the output of PAA. A vertical split over a specific time frame along the different streams, groups the patterns within a time-frame into a virtual *document*. Together these *documents* form the corpus on which LDA is applied to extract the topics and find correlations between the higher level abstractions. As the method is designed for streaming data, the approach works in the following incremental online way. In an initial configurable training period, the virtual documents are stored together as the corpus, which is used to train the initial LDA model. From that point on we can use the model to analyse the correlations between the different features in the data streams. New incoming data is again processed in the same way, the SAX patterns are extracted and the virtual documents are created. Every time a new set of virtual documents is available, the LDA model is updated, so that the correlations that we can find with the LDA model stay current to the situation of the underlying data streams.

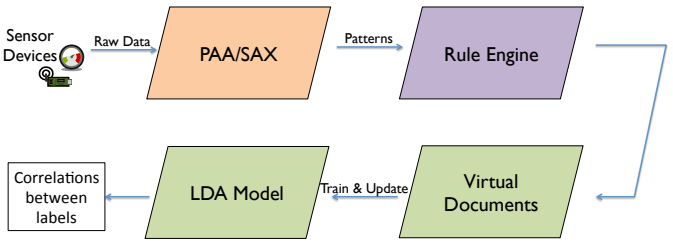


Fig. 4: Overall workflow visualisation

4.2 Data Discretisation and Pattern Recognition

As described in Section 2.1, PAA reduces the dimensionality of streaming data by splitting the input into data of a certain window size, represented by its mean. Defining the size of the slicing windows makes a huge difference in the overall outcome of the presented approach. Choosing an adequate size is heavily dependent on the input data. Here the most important factor is the throughput of the given stream. If the input data stream produces continuous output (e.g. new data values are produced every few seconds or less) the window size can be chosen big in relation with the included data points. If the aggregation is performed directly on the sensor node sinks, this has the added benefit of drastically reducing networking traffic and therefore improving the energy efficiency of the sink node. This has been analysed by Ganz *et al.* [4] where they applied a dynamic version of PAA as part of SensorSAX on the gateway level which aggregates more data together in times of low activity in the data stream and therefore reducing network load without losing information.

On the other hand, if the data stream has a very low updating rate, the data reduction factor (slicing window size) will approach 1 (equal or almost equal to no reduction in the data at all). If this is the case then the PAA step can be skipped altogether. The output of PAA is used to create the SAX patterns. In order to apply LDA on continuous data, we need a discretisation method. One way to discretise continuous, numerical data is to aggregate the values by clustering them into different categories through similarity. There are several way to do this, from hierarchical clustering to k-means. These methods have the common drawback that the information about the sequence of values is lost which is why we will not use them for this approach.

SAX is a way to transform continuous, numeric data into discretised values while retaining the sequential information. The method can capitalise on the dimensionality reduction powers of PAA by being applied to the output of PAA. In cases where the stream does not produce enough data and the dimension reduction would lead to a too low granularity, SAX can be applied directly to the streaming data instead of on the PAA coefficients. Intuitively SAX transforms the data into patterns, represented by words. The words are generated using a given alphabet and a word size. We use a different alphabet for each type of data in the streams in order to distinguish that the same shaped patterns in different features do not necessarily translate to the same meaning.

In the original SAX, each time window is normalised using z-normalisation before applying PAA [9] and aggregating each window by representing it with its mean value. N of these means together are transformed into one SAX pattern, where N is the word length. Assuming a Gaussian distribution of the normalised data each of the areas is assigned to one literal from the alphabet as follows: parallel to the y-axis, breakpoints are used to determine in which of the K equi-probable areas each time window falls, where K is the alphabet size used. The whole data window is now represented by a SAX pattern.

The assumption of having a Gaussian data distribution is well founded on observing time-series data on a long term scale [25]. However, in the case of dealing with IoT data we may also look at the data in a more short term context. The ad-hoc correlations and meaning that we find might no longer be true in long term. The very nature of the IoT means that we are dealing with multivariate data in dynamic environments. We need to study the data outside the equilibrium (i.e. when in long term it follows a Gaussian distribution).

Therefore we propose determining the current probability distribution of the underlying data stream and use this distribution to create the equi-probable areas. This enables to represent the patterns found in the data more accurately in a short-term context. The time-frame we consider for the distribution is dependent on and is affected in two opposite directions. On the one hand, shorter-time-frames for the distribution computation lead to a faster reaction time in contextual changes of the environment. We call this effect the *delay impact*. On the other hand, too small windows can lead to flat data distributions as we have less values to estimate the distribution function. This will lead to less efficient results in the creation of the SAX patterns. Therefore an informed choice has to be taken based on the granularity of the input streams and the domain of the task. For coarsely granular input streams another option to deal with the *delay impact* would be to use interpolation to fill in the gaps between updates in the stream. However, we argue that this will introduce a bias in the input data leading to skewed results. We instead increase the time-frame in which the data distribution is computed. We consider different values for this in our experimental set-up (see Section 5).

Keogh *et al.* [25] state that in order for pattern analysis tasks through similarity measures (such as indexing or classification) to be successful, z-normalisation has to be applied before comparing the patterns. This is true in the case where the meaning of the patterns and therefore the similarity is only dependent on the shape of the time series signal without concern for magnitude or amplitude of the data in this time-frame. We argue that this is not the case with the data that we are dealing with. To give a simple example, consider the temperature measurements of $m1 = [-5, -5, -5, -3, -3, -5]$, $m2 = [-5, -4, -3, -2, -3, -1]$ and $m3 = [25, 25, 25, 23, 23, 25]$. If we normalise each time window by itself before comparing and classifying, the pattern created from the measurements $m1$ and $m3$ would be considered as similar (or even equal) and be classified into the same category. It should be clear that in this case the magnitude and amplitude do carry a meaning and should be taken into account for the classification; Therefore we need to classify $m1$ and $m2$ together and $m3$ in a different

category.
For this reason, we are omitting the z-normalisation step of SAX for short-term window analysis. However, in combination with using the current distribution of the data stream for the pattern creation, we are still able to extract meaningful patterns. To help us assign these meaning we apply statistical analysis on the data streams, using the mean, minimum and maximum values with simple rules for naming the class higher level abstractions. This statistical analysis gets updated each and every time the data distribution of one or more of the input streams changes. This way we can adapt to changes in the environment. Section 4.3 describes the rule-based engine which takes the SAX patterns and assigns higher level abstractions with the help of the statistics in the time period in which the patterns were produced. The next Section presents the steps needed to use the SAX patterns for the topic extraction model.

4.3 Higher Level Abstractions

We have implemented a rule-based engine which uses the SAX patterns and the statistical information of the time frame to produce human understandable higher level abstractions. The general form of the rules is given in Equation 3 where $< p >$ is the SAX pattern, $< l >$ is the level, $< f >$ is the feature, $< mod >$ a modifier for the pattern movement and $< pm >$ is the pattern movement. Equation 4 shows the possible values that the individual output variables can take. As indicated by the square brackets around $< mod >$ in Equation 3, the modifier is optional. Furthermore, there are restrictions on which modifiers can be used, depending on the pattern movement that was extracted.

$$\{< p >< mean >\} = \{< l >< f > [< mod >] < pm >\} \quad (3)$$

The values "$slowly$" and "$rapidly$" are allowed for "$decreasing$" and "$increasing$", while the pattern movement "$peak$" requires a qualifying modifier to give the direction of the peak ("$upward$" or "$downward$"). We give examples for the in- and output of the rules in Section 5.4. The higher level abstractions have the benefit of making the results of the analysis more understandable for humans. Though the abstractions obtained from the SAX pattern extraction give a representation of the shape of the data within the time frame, the SAX strings are not intuitively understandable and still have to be interpreted. Furthermore, by taking into account the statistics of the time frame in the abstraction process we are able to represent more information in the higher level abstractions than just the shape.

$$\begin{aligned} < l >= &\quad [\text{“}low\text{”}, \text{“}medium\text{”}, \text{“}high\text{”}] \\ < f >= &\quad [\text{“}vehicleCount\text{”}, \text{“}avgSpeed\text{”}, \\ &\quad \text{“}tempm\text{”}, \text{“}wspdm\text{”}] \\ < mod >= &\quad [\text{“}slowly\text{”}, \text{“}rapidly\text{”}, \\ &\quad \text{“}upward\text{”}, \text{“}downward\text{”}] \\ < pm >= &\quad [\text{“}decreasing\text{”}, \text{“}increasing\text{”}, \\ &\quad \text{“}steady\text{”}, \text{“}peak\text{”}, \text{“}varying\text{”}] \end{aligned} \quad (4)$$

The generic rules can be applied to SAX patterns produced from any type of data. The shape of the pattern is analysed and given a human interpretable representation. The pattern "abc" for example will be an "increasing" traffic pattern within a specific time-frame from a certain location.

4.4 Virtual *Document* Generation

In order to use the higher level abstractions described in Section 4.3 as an input for LDA, we need to group them together. Since the original LDA method is designed for textual information, we will call these groups *documents*, to follow the naming conventions used in the existing LDA work in the textual domain literature and strengthen the intuition of how the method works. For the same reason we will from now on use *word* in reference to higher level abstractions.
We set a fixed time-frame as the *document* size. All higher level abstractions produced from the different data streams in this time-frame are put together into one document. This *document* will then be represented as a *bag-of-words (bow)* which leads to two things: each *word* is only present at most once in the *bow* representation (even if the *word* was present more than once in the original *document*) and the order of the words are no longer included. Although in general sequential information is very important (and one of the reasons why we used SAX instead of clustering to discretise the numerical data), in this case we are mainly concerned with extracting the topics (i.e. *words*) which give fair representations of the relatively small time windows.
The *document* generation specifically means that words produced in a certain time-frame (from now on referred to as the *document window size*) will be saved together in one *document*. Similar to the pre-processing step in LDA, the virtual *documents* are also represented by *bags-of-words*. The *bag-of-words* representation is described in Section 4.5 in more detail.
Furthermore, we consider some of the features as contextual information (e.g. features from weather streams like wind speed or temperature) while other features are treated as the main features we want to analyse. In our case study, we use traffic streams, containing information about the average speed and the number of cars within a segment of a street from a smart city dataset. If we have more than one stream producing the same features of interest (i.e. data type), within each *document window size* we save one *document* for each of the streams. In these documents all higher level abstractions produced from features of the data stream are saved as well as the higher level abstractions produced by the other streams such as the weather stream. This way we can find relations between the streams of interest and the complimentary information provided by the other streams.
During this step the choice of *document window size* has the most impact on the overall results. We need to pick a window size which is both long enough so that we can encapsulate enough different *words* into *documents* for the topic modelling but still short enough not to lose to much information. The size of the *document* window is not the only important factor. Another issue for applying the method to online classification is that during the observation time (i.e. window length), while sufficient data to form the *words* and *documents* is gathered, we can not make any assumptions about the state of the current environment. Only after the

new *document* has been created we can draw any conclusions about this time window.

4.5 Uncovering the Underlying Structures and Relations

LDA is a generative probabilistic model used to extract topic models from textual *documents* (for more information please refer to Section 2.2). It does so by finding the latent variables that describe the relations within the *words* and *documents*. Therefore by applying LDA to the virtual *documents* described in Section 4.4, we can uncover hidden relationships between the different features of the data stream. Initially the LDA model is trained for a period of time before its output is used for further analysis. In our experiments, described in Section 5 we train the model on 2400 virtual *documents* generated from 36000 data samples before folding in new documents. Depending on the task and the granularity of the data, the training period can range from a few hours to one or several days or even weeks. After the training phase, the model can be used to extract the latent topics from incoming documents. From now on we will use the term *corpus* to refer to the *documents* used for the training period and all *documents* which have been folded in the stream analysis process up until the current time-point.

In the data discretisation step, we assign a different alphabet for each individual feature stream to distinguish the features from the data streams and translate them into higher level abstractions. By grouping the higher level abstractions together in *documents*, we can now analyse their relations to each other. If we have enough similarities within the individual *documents* of the *corpus*, then LDA will find the relations of associated higher level abstractions by grouping them together in a topic representation.

Since the topics represent probability distributions, we can apply similarity measures for probability distributions in order to find closely related topics. The topic model can generate synthesised *documents*, consisting of *words* based on the found relations within the different data streams. These synthesised *documents* could be used for predicting upcoming patterns in the data. At the same time, we can fold in singular patterns from one data source and predict the most likely co-occurring patterns in the co-related data streams.

4.6 Time Complexity

We discuss the time complexity of the algorithms used in our approach. PAA splits a time series subsequence into N equal sizes and then calculates the Section means of each data chunk. This can be done in one pass over the subsequence of length n, therefore the run time is in $\mathcal{O}(n)$. The resulting PAA coefficients are used as input for the SAX method.

The original SAX approach uses a lookup table to assign the SAX literals to each of the coefficients taking $\mathcal{O}(N)$ time. Because usually $N << n$, in practical application SAX runs in $\mathcal{O}(1)$. In our extension of SAX we introduce the use of the current PDF of a stream instead of a Gaussian distribution for calculating the breakpoints. Because the PDF evolves over time, we have to recalculate the breakpoints whenever there is a significant change in the distribution of the data. For the LDA analysis, we are using an online implementation of the algorithm by Hoffman *et al.* [26]. For the inference they are using an online Variational Bayes algorithm which like the batch Variational Bayes algorithm is in $\mathcal{O}(1)$, however with a much lower constant factor [26].

5 Experimental Results

In this Section we first describe the data sources and data sets that were used for the experiments. We present the different parametrisations used. Then we show how each of the components output looks like and discuss the impact of the different parametrisations.

5.1 Data-sets

We use two different kind of streams for our experiments:

(i) traffic data streams from the city of Aarhus[2]
(ii) a weather data stream from a weather station in Aarhus[3].

We use 100 different traffic sensors as our data streams. The data is provided by the open data Aarhus platform via the data management system Comprehensive Knowledge Archive Network (CKAN). The data included is the average speed and the number of cars measured in one segment of the city. New values are provided every 5 minutes. The weather data stream is used as the complimentary data stream. Here we use the wind speed and temperature data from the weather streams. The weather data was originally hosted on Weather Underground, a meteorological data service platform [27]. Every hour, 3 measurements are provided. Over the course of two months we gathered data from both sources and stored it in a csv format, to have reproducible experiments. At the same time we implemented a real-time version of our approach that automatically polls and analyses the data from the sources whenever new data is available.

5.2 Parameter Selection

During empirical experimentation we have investigated what parameters affect the results of our method the most. These parameters are the alphabet size a and word length w of the SAX pattern on the one hand and k, the number of topics used to create the LDA model. The word length is very much dependent on the granularity of the data set. On the dataset we are running our experiments we chose the fixed word length 3 for a window length of 1 hour based on the output speed of our data streams. This parameter needs to be adjusted accordingly to the data rate in the stream. We compare combinations of alphabet size $a \in \{3, 5\}$ and number of topics $k \in \{10, 20, 50, 100\}$. The differences in the outcome are described in Section 5.6. Below are the specific values we used for the individual parameters.

- Configuration A = [$a = 3$, $k = 10$]
- Configuration B = [$a = 3$, $k = 20$]
- Configuration C = [$a = 3$, $k = 50$]
- Configuration D = [$a = 3$, $k = 100$]

2. Data set retrieved from: http://www.odaa.dk
3. Data set retrieved from: http://www.wunderground.com

- Configuration E = [$a = 5$, $k = 10$]
- Configuration F = [$a = 5$, $k = 20$]
- Configuration G = [$a = 5$, $k = 50$]

5.3 Data Discretisation and Pattern Recognition

We have implemented an extension of SAX which allows us to represent the data with patterns that more accurately reflect the original data. We compute the probability density function (PDF) using the kernel density estimation (KDE) from the recent time period. By dividing the PDF into equi-probable Sections we have a finer granularity in areas which are likely to happen. Values which are unlikely to happen are covered by larger ranges instead. This allows to assign a literal to segments. Figure 5 shows how a z-normalised Gaussian distribution (Figure 5a) and the PDFs of the *wind speed* data (Figure 5b and *temperature* data (Figure 5c) are segmented into equi-probable areas. We are aware that our extension could be criticised for ignoring the Gaussian distribution of time-series data in long-term analysis [25]. We use the following Section to justify our reasoning. First, it can be argued that in cases where we have a probability distributions with a strong centre such as the one in Figure 5b, all high outlier values would be assigned to the same literal and are not further differentiated.

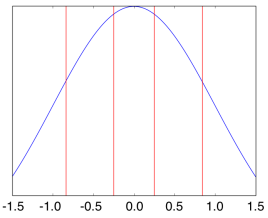
However, we argue that these outliers are both unlikely and very high relative to the other values of this segment; so it makes sense to assign them the same literal. To make this concept more clear, let us look at an example SAX word generation shown in Figure 6. As Figure 6a shows, if we use a Gaussian distribution, the SAX word ”cba” would be generated, indicating that the pattern has a shape which increases from intermediate values to very high values. Using the computed PDF (Figure 6b), the word ”aaa” is generated instead, which means that the pattern is of a shape in which constantly high values are expected. While this might seem counter-intuitive at first glance, it starts to make sense once you consider the fact that because of the current distribution, all values in this window are relatively high. This means that in this time-frame we certainly have a pattern which consists of high values.

The benefit of using a PDF based segmentation can be seen when we compare the word generation shown in Figure 6c to the one shown in Figure 6d. Using the Gaussian distribution, the pattern ”ddd” would be generated, indicating no movement in the pattern. Using the PDF, the word ”dcd” will be generated which shows that there is some change in the stream. Since it is more likely that we have values in this range, it makes more sense to have a more differentiated pattern here.

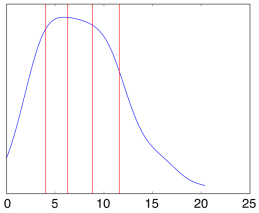
5.4 Higher Level Abstractions

In this section we show sample in- and outputs of the higher level abstraction rules described in Section 4.3 that were generated as part of our experiments in Equation 5 and 6. The complete data sets and results are available online at: https://github.com/UniSurreyIoT/SAX-LDA.

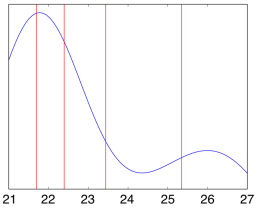
$$\{Pattern : \text{"}eed\text{"}, mean : 0.54\} = \{Low\ vehicleCount\ slowly\_decreasing\} \quad (5)$$



(a) Gaussian distribution

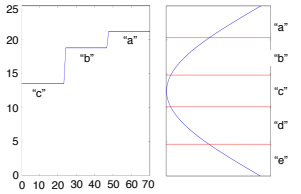


(b) Probability density function of wind speed

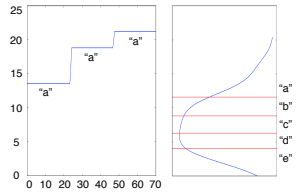


(c) Probability density function of temperature

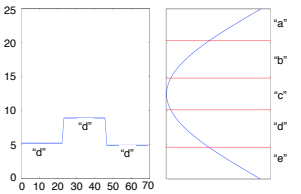
Fig. 5: Probability distribution functions split into equi-probable areas



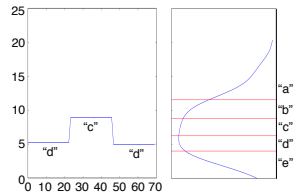
(a) SAX word generated by using Gaussian distribution (Windspeed)



(b) SAX word generated by using custom distribution (Windspeed)



(c) SAX word generated by using Gaussian distribution (Temperature)



(d) SAX word generated by using custom distribution (Temperature)

Fig. 6: SAX Pattern generation with Gaussian Distribution compared to using the computed PDF

$$\{Pattern : \text{"}bab\text{"}, mean : 20.0\} = \{Medium\ avgSpeed\ downward\_peak\} \quad (6)$$

5.5 Latent Dirichlet Allocation

Figure 8 shows a small snippet of the topics that were created by running the LDA on top of the *corpus*. It shows which higher level abstractions are correlated with each other. It can be seen from Figure 8 that higher level abstractions are not mutually exclusive for the topics (i.e. latent variables in the LDA model). For example, the abstraction Medium_tempm_steady is present in Topic 6 and Topic 14.

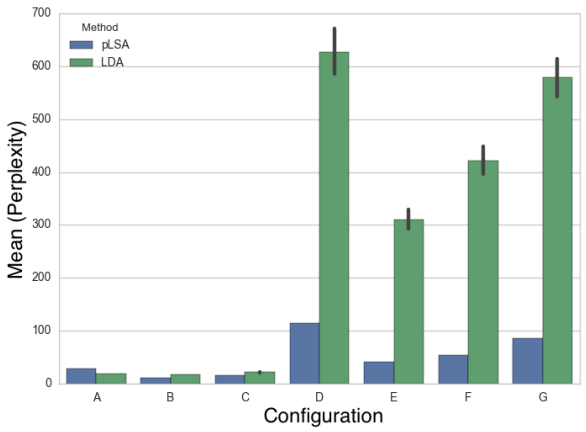


Fig. 7: Perplexity score of models regarding number of topics and alphabet size of SAX

Therefore if we want to use the topics for predicting a new document, the most likely one with the most similarities will be chosen.

In our approach we are using multiple alphabets, one alphabet for each kind of data to create the symbolic representations from the numerical data streams. While this transforms the heterogeneous data sources into a more homogeneous representation, the comparison between the patterns using different symbols is still not straightforward. Using LDA we cluster the virtual *documents* in an unsupervised fashion by identifying co-occurring higher level abstractions. The latent factor topic vectors z_k provide a common representation for the different data sources in a latent space. This model also allows us to fold in a single higher level abstraction created by one data source and predicting which other abstractions are most likely to be expected in the other data streams. By computing the cosine distance between different topic vectors, we can compute the similarity of different abstractions and their predicted co-occurence.

5.6 Analysis and discussion

In order to compare different latent models, in text processing, conventionally the perplexity score is used [7]. The formula to compute the perplexity score of a model is given in equation 7, where $\mathbf{D}_{test}$ is the held-out *corpus* used to test the model, $p(\mathbf{w}_d)$ is the probability the model assigned to seeing *word* $\mathbf{w}$ in *document* d and $\sum_{d=1}^{M} N_d$ is the total number of (non-unique) words in the test corpus.

$$perplexity(\mathbf{D}_{test}) = exp(-\frac{\sum_{d=1}^{M} log(p(\mathbf{w}_d))}{\sum_{d=1}^{M} N_d}) \quad (7)$$

High perplexity scores indicate an over-fitted model. Therefore low perplexity values are more desirable. Figure 7 shows how the alphabet size chosen to generate the SAX patterns (i.e. *words*) affect the perplexity score of the LDA model. The higher perplexity in the case of an alphabet size of 5 is explained by the fact that there are more possible permutations of SAX patterns, which leads to a much larger dictionary. At the same time this explains why the perplexity can be that low for the alphabet size 3. The number of topics also affects the perplexity once the alphabet size grows. The results of Figure 7 show that configurations A-C can be used with LDA as the topic model, whereas configurations D-G lead to too high perplexity scores in the model, which makes them unfeasible to be used. When we use pLSA as the topic model, the perplexity score still rises with configuration D-G, however, the perplexity score does not increase much higher than 100, which means it can still be used in practice.

To demonstrate how the results of the presented approach can be interpreted and how the identified relations between the different data features can be verified, we have run further experiments. To have a comparison, we are using two different approaches, one using LDA as the topic extraction model and one using pLSA. Again we have an unsupervised training process, in which we obtain the initial model from the generated virtual documents. An excerpt of the resulting topics can be seen in Figure 8, higher level abstractions with a probaility less than 0.01 in the model are omitted from the topics. We can observe some of the relations that were found from the LDA model. Once the training is completed, the streaming part of the approach begins. To verify the relations that can be seen in the topics, we perform the following: whenever a new *document* is generated, we cover the higher level abstractions generated by one of the features, meaning that we just include the higher level abstractions generated by the other features in the covered document. We then estimate the topic that most likely has produced the covered document. The higher level abstraction of the missing feature with the highest probability is predicted by our model to be the one that was covered. If the correlations are correct, then most of the higher level abstractions that are predicted this way should be correct. In the case of weak correlations, the prediction is no better than random guessing; for this reason we do not include them in the identified correlations.

$$precision = \frac{|\{AHLA\}| \cap |\{PHLA\}|}{|\{PHLA\}|} \quad (8)$$

$$precision = \frac{|\{AHLA\}| \cap |\{PHLA\}|}{|\{AHLA\}|} \quad (9)$$

$$precision = 2 * \frac{precision * recall}{precision + recall} \quad (10)$$

We verify this by calculating the F-measure for each of the higher level abstractions. To compute the F-measure (equation 10) we need to calculate precision (equation 8) and recall (equation 9). AHLA stands for the higher level abstractions that have been actually extracted in the data processing and PHLA stands for the predicted higher level abstractions using the topic model as a predictor. After analysing the F-measures we could observe that for some of the higher level abstractions we have strong correlations to the other higher level abstractions, while for others we do not have these. To give an example, we show the F-measures of the higher level abstractions generated from the feature vehicle count in Figure 9 and average speed in Figure 10.

Because the names of the higher level abstractions are to long to be included as the labels of the figures, we map them to capital letters. This mapping can be found below, along

Topic 6

Label	Probability
Low_avgSpeed_steady	0.457
Medium_tempm_steady	0.184
Low_vehicleCount_steady	0.17
Low_wspdm_steady	0.079
Medium_wspm_stead	0.066
Low_wspdm_downward_peak	0.018

Topic 14

Label	Probability
Low_vehicleCount_slowly_increasing	0.265
Medium_tempm_steady	0.26
Low_vehicleCount_downward_peak	0.237
Medium_avgSpeed_steady	0.203
High_avgSpeed_steady	
Low_wspdm_upward_peak	
Medium_wspdm_steady	

Fig. 8: Excerpt from the initial topics of the LDA model

with the information how often the higher level abstraction actually appeared during the data processing.
Vehicle count:

- A = Medium_vehicleCount_steady (Appeared 19968 times)
- B = Low_vehicleCount_upward_peak (Appeared 1739 times)
- C = Low_vehicleCount_steady (Appeared 33448 times)
- D = Low_vehicleCount_downward_peak (Appeared 1839 times)
- E = High_vehicleCount_steady (Appeared 4998 times)
- D = Low_vehicleCount_slowly_increasing (Appeared 3733 times)
- E = Low_vehicleCount_slowly_decreasing (Appeared 5706 times)

Average speed:

- A = Low_avgSpeed_steady (Appeared 11768 times)
- B = Medium_avgSpeed_steady (Appeared 43811 times)
- C = Medium_avgSpeed_downward_peak (Appeared 168 times)
- D = High_avgSpeed_steady (Appeared 15623 times)
- E = High_avgSpeed_downward_peak (Appeared 131 times)

In Figure 9 we can see that for three of the higher level abstractions, the pLSA approach is outperforming the LDA approach. This is the case because the pLSA model only predicts these three higher level abstractions throughout the experiment, leading to almost perfect recall scores, which result in the high F-measure. The pLSA approach fails to predict the other higher level abstractions within the experiment. The approach using LDA is able to identify all frequently appearing higher level abstractions, meaning that the model more accurately represents the relations found in the data stream overall.
In the case of the average speed, we can see that both models perform similar for most abstractions. However, the approach using LDA provides more consistent results for all higher level abstractions.
A further look at the results of the experiments reveals that the higher level abstractions that are generally more present in the data set can be better predicted with our model. This is because they have a stronger effect on the resulting topics. This is especially the case for the three labels that pLSA is able to predict.
Running the experiment in replay mode, we could process two months of data from 100 different traffic sensors within less than a day. In the live mode where we poll the data continuously from the traffic sensors, our approach can provide results in (near) real-time.

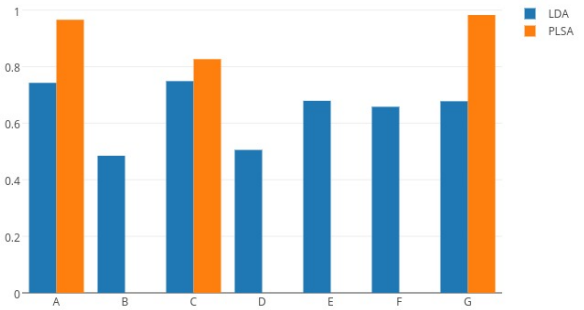


Fig. 9: F-measures of vehicle count

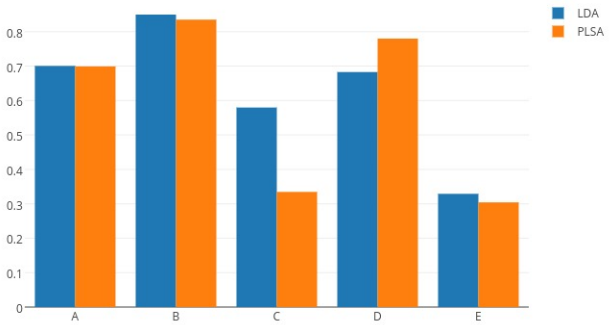


Fig. 10: F-measures of feature average speed

5.7 Reproducibility

We are aware that testing a novel approach on a number of data-sets does not provide enough evidence for the effectiveness of the method. Therefore we have developed a configurable data generator and a flexible experimental set-up. This way our experiments can be reproduced and they can be also easily applied on any kind of data-sets. Using the GUI all parameters for the different parts of our approach can be set. The system and datasets are available via: https://github.com/UniSurreyIoT/SAX-LDA.
A data stream is represented by at least one CSV file. Each file represents the measurements of one sensor. One column of the CSV file needs to be called *TIMESTAMP*, representing the time the measurement was taken at. The other columns each represent one feature of the data stream. Files produced by different sensors of the same kind (e.g. traffic sensors) have to contain the exact same features.
Two folders have to be chosen. The first called *Main Data Stream* is the data stream of interest, which is intended to be enhanced by contextual information, provided by the files in the folder set at *Correlated Data Stream*.

6 Conclusion

We have introduced a novel approach which is designed to identify structures and relations within heterogeneous data sources. We have created a solution which looks at different sensor data streams in an unsupervised fashion and extracts patterns from the data streams, translates them into human understandable and machine interpretable higher level abstractions and identifies cross-stream relations and correlations between different features. The proposed solution is domain independent and can be applied to any IoT data stream.

We have presented the individual components of the approach by applying it to traffic data and weather data streams. We have shown how our extension of the data discretisation method SAX, by using a computed PDF instead of a Gaussian distribution, affects the pattern creation process on real data. We have also shown how the higher level abstractions created in a time window are used to generate virtual documents. The use of higher level abstractions and virtual documents allowed us to apply a technique, which is usually used for text analysis on numerical data streams. We have also shown which of the extracted relationships of the higher level abstractions by the LDA model are correct and which ones are incidental by applying a prediction mechanism on partially hidden data.

For future work, we plan to introduce dynamic time windows, which can automatically decrease to capture times of high interest in a finer granularity and adjust again in times of low interest. Since our work allows us to analyse numerical data in the latent factor space, we plan to combine this work with social media analysis to automatically interpret and label the data in an unsupervised fashion.

Acknowledgment

The research leading to these results has received funding from the European Commissions Seventh Framework Programme for the CityPulse project under grant agreement no. 609035.

References

[1] IBM, "IBM Annual Report 2015," IBM, Tech. Rep., 2015.

[2] L. Hernández, C. Baladrón, J. M. Aguiar, L. Calavia, B. Carro, A. Sánchez-Esguevillas, D. J. Cook, D. Chinarro, and J. Gómez, "A study of the relationship between weather variables and electric power demand inside a smart grid/smart world framework," *Sensors*, vol. 12, no. 9, pp. 11 571–11 591, 2012.

[3] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *IJCAI International Joint Conference on Artificial Intelligence*, no. 9, 2009, pp. 1420–1426.

[4] F. Ganz, P. Barnaghi, and F. Carrez, "Information Abstraction for Heterogeneous Real World Internet Data," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3793–3805, 2013.

[5] W. Wang, P. M. Barnaghi, and A. Bargiela, "Probabilistic topic models for learning terminological ontologies," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 1028–1040, 2010.

[6] F. Ganz, D. Puschmann, P. Barnaghi, and F. Carrez, "A Practical Evaluation of Information Processing and Abstraction Techniques for the Internet of Things," *IEEE Internet of Things Journal*, vol. 2, no. 4, pp. 340–354, 2015.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 3, pp. 993–1022, 2003.

[8] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A Symbolic Representation of Time Series , with Implications for Streaming Algorithms," in *8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[10] A. A. Wanigatunga, P. V. Nickerson, T. M. Manini, and P. Rashidi, "Using symbolic aggregate approximation (SAX) to visualize activity transitions among older adults." *Physiological Measurement*, vol. 37, no. 11, pp. 1981–1992, 2016.

[11] J. A. Reggia and Y. Peng, "Modeling diagnostic reasoning: a summary of parsimonious covering theory," *Computer Methods and Programs in Biomedicine*, vol. 25, no. 2, pp. 125–134, 1987.

[12] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[13] T. Hofmann, "Probabilistic latent semantic analysis," in *15th Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 289–296.

[14] P. Hu, W. Liu, W. Jiang, and Z. Yang, "Latent Topic Model Based on Gaussian-LDA," in *Chinese Conference on Pattern Recognition*, 2012, pp. 556–563.

[15] P. Quelhas, F. Monay, J. M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. Van Gool, "Modeling scenes with local descriptors and latent aspects," *10th IEEE International Conference on Computer Vision*, vol. 1, pp. 883–890, 2005.

[16] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their location in images," *10th IEEE International Conference on Computer Vision*, vol. 1, no. 1, pp. 370–377, 2005.

[17] N. Rasiwasia and N. Vasconcelos, "Latent Dirichlet Allocation Models for Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2665–2679, 2013.

[18] J. Bian, K. Yoshigoe, A. Hicks, J. Yuan, Z. He, M. Xie, Y. Guo, M. Prosperi, R. Salloum, and F. Modave, "Mining twitter to assess the public perception of the "internet of things"," *PLoS ONE*, vol. 11, no. 7, pp. 1–14, 2016.

[19] J. Xin, Z. Cui, S. Zhang, T. He, C. Li, and H. Huang, "Constructing topic models of internet of things for information processing," *Scientific World Journal*, vol. 2014, 2014.

[20] N. Lathia, D. Quercia, and J. Crowcroft, "The hidden image of the city: Sensing community well-being from urban mobility," in *International Conference on Pervasive Computing*, vol. 7319 LNCS, 2012, pp. 91–98.

[21] A. J. Jara, D. Genoud, and Y. Bocchi, "Big data for smart cities with KNIME a real experience in the SmartSantander testbetd," *Software: Practice and Experience*, no. 8, pp. 1145–1160, 2015.

[22] F. Lécué, S. Tallevi-diotallevi, J. Hayes, R. Tucker, V. Bicer, M. Sbodio, and P. Tommasi, "Smart traffic analytics in the semantic web with STAR-CITY : Scenarios , system and lessons learned in Dublin City," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 27-28, pp. 26–33, 2014.

[23] F. Lécué, R. Tucker, S. Tallevi-Diotallevi, R. Nair, Y. Gkoufas, G. Liguori, M. Borioni, A. Rademaker, and L. Barbosa, "Semantic Traffic Diagnosis with STAR-CITY: Architecture and Lessons Learned from Deployment in Dublin, Bologna, Miami and Rio," in *International SemanticWebConference*, 2014, pp. 292–307.

[24] T. L. Griffiths and M. Steyvers, "Finding scientific topics." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. suppl 1, pp. 5228–35, 2004.

[25] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks : A Survey and Empirical," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[26] M. Hoffman, F. R. Bach, and D. M. Blei, "Online Learning for Latent Dirichlet Allocation," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 856–864.

[27] J. J. Celenza and R. W. Carver, "A community supported, Internet-based surface meterological network, the Wunderground experience," in *25th Conferemce on International Interactive Information and Processing Systems*, 2009.