Anomaly Detection using Kolmogorov Complexity for Website Activity Data

Pim Wijn

Leiden Institute of Advanced Computer Science
Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
p.wijn@umail.leidenunv.nl

Abstract. Time series data can be found everywhere. The nature of the time series makes them difficult to analyse, especially when the data is noisy. We will try to analyse posting activity on news websites using an approach based on information theory, more specifically Kolmogorov complexity. It uses compression to find structure in the data that can be used to find anomalies. Finally we will adapt this approach for use in real-time anomaly detection to see if we can detect anomalies in the noisy activity data as early as possible.

1 Introduction

Time series data is everywhere. Often data is logged to keep information over time. This data can be used to extract trends, find patterns or detect anomalies. The nature of the data makes analysis quite difficult. Many approaches have been developed to analyze time series, each with different use cases.

In this case want to know what is interesting in our data without forcing our algorithm to search for specific patterns. A novel way to do this is by using information theoretic principles. Using information theory makes it possible to find parts or structures in the data that contain the most information. In other words these parts contain as much information as possible in the most concise way. In this paper we will exploit this concept by using compression to find structure in the data, but also to find parts of the data that do not contain structure at all. Parts of the data that do not contain structure have low information content and are therefore incompressible.

A specific approach that uses these principles and tries to find variable length anomalies quickly is based on the method described by Senin et al. [1]. In their paper they describe a way to find anomalous subsequences in time series using a symbolic representation and a grammar-based compression. It is suggested that the approach could easily be extended to real-time anomaly detection.

The main goal of this paper is to implement this approach and extend it to be real-time. We will then analyze activity data of several Dutch websites. The specific activity we are interested in is the amount of news articles or blogs posted per day.

In the first part we will show how to pre-process the data. The preprocessed data can then be compressed, this compressed data gives us information that we can use to find anomalies.

To find anomalies we will apply both incremental and sliding window based approaches. Both will use information from the compression or the actual compressed length itself to evaluate points. Finally we will apply these approaches to see how they work on real world data.

2 Preliminaries

2.1 Time Series

A *Time Series* $T = t_1, t_2, \ldots, t_m$ is a set of scalar observations repeatedly measured over time. A *subsequence* consists of a contiguous sampling of a certain length from the time series $C = t_p, \ldots, t_{p+n-1}$ where n is the length and p an arbitrary position such that $1 \leq p \leq m - n + 1$. To analyze time series a *sliding window* is often used. A window is a subsequence of a predefined length. It is moved over the data to find all subsequences of a specific length.
The distance between two time series $C = c_1, \ldots, c_n$, $Q = q_1, \ldots, q_n$ is defined as

$$D(Q, C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \tag{1}$$

Efficiently analyzing Time series Analyzing time series data brings several new challenges. Adding time as an extra dimension makes the data more difficult to analyze. Standard algorithms for normal data are often not very useful, they often don't scale to the large dimensions time series provide and are often not applicable to values that change over time. When we specifically look at the anomaly detection problem these algorithms lead to a large number of false positives and negatives.

There are two directions from which to approach these challenges. First of all it is necessary to look at what is relevant for us in time series data. Often the data is more detailed than necessary. If this is the case we can reduce dimensionality and range. The result is much smaller, but in many applications still keeps all relevant information. The second approach is to develop new algorithms for data that has a time dimension. This could for example be done using a sliding window that moves forward in time.

Discretization with SAX First we will explain the approach used to discretize the time series data in linear time. This approach is called SAX or Symbolic Aggregate Approximation [2]. The aim of this algorithm is to build a symbolic representation of the data that can directly be used by many other algorithms. The algorithms has two parts. In the first step the resolution of the time series is made smaller and data points are aggregated, this is called PAA (Piecewise

Aggregate Approximation). When converting a new vector of length M for a time series of length n we have to calculate each element of the new vector using the following equation.

$$\bar{x}_i = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x_j \tag{2}$$

The second is to divide the values in several bands. Each band has a symbol from the alphabet attached to it. These bands are not uniformly distributed, instead they are based on a Gaussian distribution, which almost always the distribution of data points in time series data. Each point is assigned a character according to the pre-calculated cut-off points.

$$\hat{c} * i = alpha * j \iff \bar{c} * i \in [\beta_{j-1}, \beta_j) \tag{3}$$

The resulting representation will be the input for the next step of finding anomalies in the data.

2.2 Compression for Data mining

This approach to data mining is based on Kolmogorov complexity [3] and the Minimum Description Length principle [4]. Both try to find the shortest possible representation (or algorithm) of the data to extract meaning. The advantage of using information theory is that the amount of structure that is considered relevant and the residual data is automatically balanced. For MDL $L(M) + L(D|M)$ is optimized, which is the length of the model plus the length of the data given the model. The final model will not under-fit or over-fit the data. In MDL such a model is often represented represented by a probability distribution which directly gives the encoded lengths of each item.

In the next section we will see a grammar based compression algorithm, called Sequitur, which builds a grammar model. This model has a similar effect as an MDL model.

Sequitur The algorithm builds a grammar $G = \{r_0, \ldots, r_k\}$ which contains a set of rules r. It poses constraints on the form of the grammar. Once the algorithm finds a violation of these constraints the grammar is updated such that the violation is removed. The two constraints that are not allowed to be violated are:

1. No pair of adjacent symbols appears more than once in the grammar.
2. Every rule is used more than once.

Each symbol is added separately. After it is added the constraints are checked and the grammar is updated if necessary. Adding a rule to the grammar can also cause a new violation of the constraints. Therefore all existing rules have to be checked and updated. An efficient, linear time implementation of this algorithm can be found in [5].

Grammar Rule Density Using the grammar we can get an indication of the compressibility of each point by counting the grammar rules that span that data point. Calculating this value for each data point gives us a grammar rule density graph that can be used to quickly calculate the distance between two subsequences. So for density curve $D = d_0, \ldots, d_n$.

$$d_i = density(t_i, G) = \#\{r | r \text{ covers } t_i \wedge r \in G\} \tag{4}$$

2.3 Related Work

Other approaches to analyzing time series are the well known STL [6] which decomposes time series data in a seasonal, trend and residual component. Another paper that analyses time series is written by Vespier et al. [7] and uses the MDL principle to decompose a signal into multiple timescales.

3 Anomalies in Time Series

In this section we will look at the steps required to find the most anomalous subsequence in a time series. Even though that is not the goal of this paper, the basis needed is the same. Later the differences between the two approaches will be shown and we will take a look at the extensions needed for real-time anomaly detection.

3.1 Interesting Anomalies

Interesting anomalies in time series are not necessarily the same as interesting anomalies in other kinds of (non-temporal) data. In data without change over time anomalies are often single points that deviate from the norm. When we look at time series data single point anomalies are generally not relevant. Differences in the structure of the data over time or the breaking of patterns that occur with a certain frequency hold more meaning.

In time series the anomaly detection problem is described as: *Finding outliers relative to some standard or usual signal.*

Therefore we are trying to find subsequences that are either the most different from any other subsequence in the time series, or a subsequence that has a distance over a certain threshold with all other subsequences.

Examples of anomalies in time series that occur often are: Spikes (Fig. 1) or additive outliers, Drops, flatlines (Fig. 3) or temporal changes and Level shifts (Fig. 2), where the shape does not change but the overall value does

Since we do not know the size or the shape of anomalies it is important that the way we detect them is not limited to a predefined sequence length.

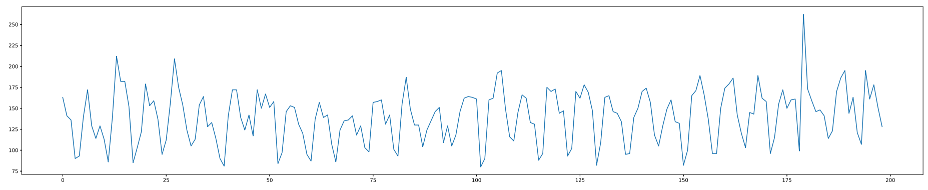


Fig. 1: A spike anomaly

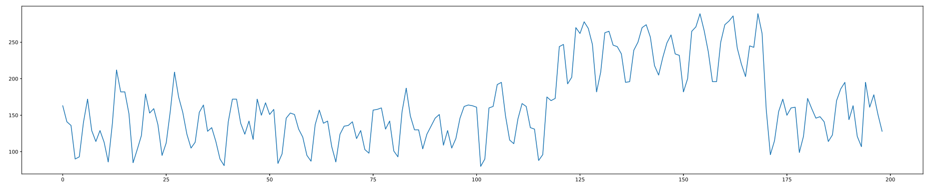


Fig. 2: A level change anomaly

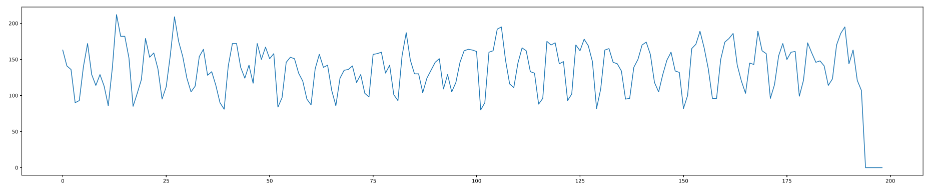


Fig. 3: A flatline anomaly

3.2 Anomalies and Incompressibility

This idea of recurring structures in time series data is closely related to compression. Subsequences of a time series that have structure can be compressed. A lack of structure in a certain subsequence means that it is (almost) incompressible. This makes the goal of anomaly detection to find, given these patterns, what subsequences of the data do not compress as well as subsequences and are therefore anomalous.

We have a symbolic representation of the time series data in the form of a string of symbols from a certain alphabet. From this we have created a grammar G that compresses the data. In this grammar recurring structures have been compressed. Subsequences that have no structure do not have grammar rules applied to them. From this we can conclude that the sequence is incompressible and therefore likely to part of an anomaly.

4 Real-time Anomaly Detection

We have seen the preprocessing and compression steps that can be used to find structure. In this section we will extend these principles to apply them to real-time anomaly detection. The goal of real-time anomaly detection is to find upcoming anomalies as early as possible. Unlike in the original approach described in [1] we do not necessarily want to find the most anomalous subsequence. However we want all new subsequences that are anomalous enough.

For real-time anomaly detection we will try two main approaches. One is based on sliding windows, the second adds points incrementally.

Sliding Window The simplest way of finding real-time anomalies is by using a sliding window and recalculating for each window the grammar and rule density curve. It is also possible to calculate measures for the whole window, such as compressed size of grammar G_i which compresses window $C_i = t_{i-n}, \ldots, t_i$ and grammar rule density of the last point in the window.

The density of the last point in a window is

$$d_i = density(t_i, G_i) \tag{5}$$

When using compression we define the density as the size of G_i

$$d_i = L(G_i) \tag{6}$$

Incremental We can also add each point individually to the whole time series. The grammar density and compression for that point can then be calculated.

Both SAX and Sequitur are incremental in nature and new data points can easily be added.

Anomaly Score The anomaly score of a subsequence is related to the area of the grammar density curve that is under the average density. This means that both short and incompressible and longer and less compressible increase the anomaly score. However both do so at different rates.

$$a_i = a_{i-1} + \frac{1}{n} \sum_{d \in D} d - d_i \tag{7}$$

Anomaly Delta We will also look at the changes in compressed size of each window to see what effect adding a point has on the compression ratio.

$$\delta_i = \begin{cases} d_i - d_{i-1} & \text{if } i \geq 1 \\ 0 & \text{if } i = 0 \end{cases} \tag{8}$$

5 Experimental Results

5.1 Experimental Data

The dataset used for evaluation contains the posting activity per day measured over one year for Dutch websites. The data is based on the amount of posted news articles and blogs, and will therefore be quite noisy. Most of the time series in the datasets do not contain relevant anomalies.

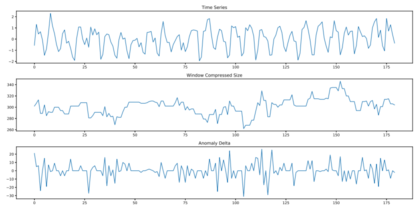
Anomalies that do exist can have multiple causes. Either real changes in the activity on the website (broken website?) or problems with data crawling or extraction (broken spiders or new website layout). Any of these anomalies will have to be evaluated for their cause.

5.2 Experimental Results

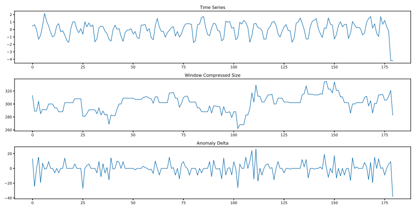
First we will look at the results of using a sliding window to compute the anomaly scores and compressed sizes. Then we will look at the results of the incremental approach. We look at the plots of only one website. The results for other websites lead to the same observations.

Sliding Window For these experiments a window size of 60 was used. Lower window sizes lead to decreased efficiency of compression and are therefore less useful.

Window Compression Figure 4 shows the compressed sizes of the windows for both data without any real anomalies and for data that has an artificially added flatline. As can be seen the the compressed size actually goes down once the flatline occurs. The bottom graphs show the change in compressed size, this however does not seem to give any interesting patterns.



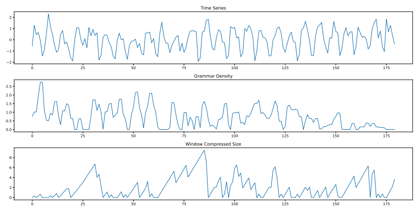
(a) No Anomalies



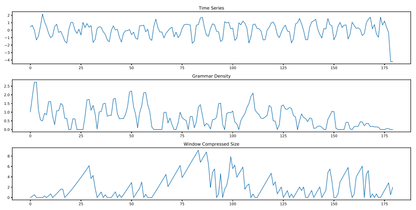
(b) Flatline Anomaly

Fig. 4: Tweakers Window Compression Score

Window Anomaly Score The window anomaly score plotted in Fig. 5 is based on the rule density of the last point in the window. The middle plot shows the average grammar rule density for a point for each window it occurs in. We see high anomaly scores and low grammar density for many points. Again adding an artificial flatline seems to decrease the anomaly score at the end.



(a) No Anomalies

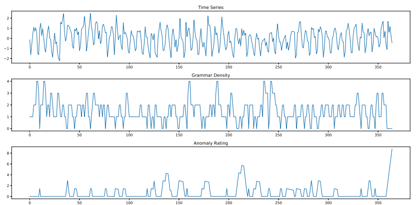


(b) Flatline Anomaly

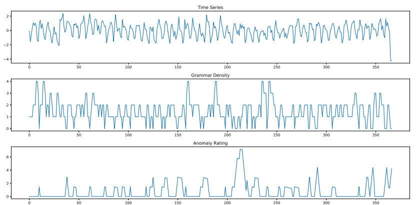
Fig. 5: Tweakers Window Anomaly Score

Incremental Finally we look at the incremental approach. In Fig. 6 we see the original plot which has no real anomalies and a plot that has a flatline added. Similar to the sliding window based approach, adding a flatline decreases the anomaly rating and it improves the grammar rule density curve.

For all of the approaches we see that they are not very effective at finding anomalies in real-time. I suspect that the lack of compressibility both in the incremental approach and even worse in the sliding window approaches prevent finding real anomalies. As can be seen in the density plots, for a many points the density is 0. This means that they are incompressible using Sequitur. This does not seem to be a problem in the implementation, since the application of the authors of [1] gives similar results for the grammar density plot.



(a) No Anomalies



(b) Flatline Anomaly

Fig. 6: Tweakers Incremental Anomaly Score

6 Conclusion

In this paper we have seen how to process time series data in such a way that structure can be easily extracted. Using the processed data and concepts from Kolmogorov complexity and the MDL principle we have tried to extract anomalies from the time series. The focus was to analyze the data in real-time and detect anomalies as early as possible. For this both a sliding window based approach and a incremental approach were developed and evaluated. The data used for this was the post activity of Dutch website. During the experiments we have seen that these neither of these approaches seem to detect useful anomalies. Which seems to be a problem with the compressibility of the data in general (using a grammar based compression). For future work it would be interesting to look at more structured data that is more compressible for finding anomalies using this approach. Another option would be to use the data with a different compression mechanism, for example the one used by Vespier et al. in [7].

References

1. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., P Boedihardjo, A., Chen, C., Frankenstein, S.: Time series anomaly discovery with grammar-based compression. (March 2015)
2. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. DMKD '03, New York, NY, USA, ACM (2003) 2–11
3. Li, M., Vitányi, P., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer (2008)
4. Grünwald, P.: The Minimum Description Length Principle. MIT Press (2007)
5. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. CoRR **cs.AI/9709102** (1997)
6. B Cleveland, R., S Cleveland, W., E McRae, J., Terpenning, I.: Stl: A seasonal-trend decomposition procedure based on loess. **6** (January 1990) 3–33
7. Vespier, U., Knobbe, A., Nijssen, S., Vanschoren, J. In: MDL-Based Analysis of Time Series at Multiple Time-Scales. Springer Berlin Heidelberg, Berlin, Heidelberg (2012) 371–386