Datenbank Spektrum (2018) 18:79–87
https://doi.org/10.1007/s13222-018-0285-x

SCHWERPUNKTBEITRAG

Data Change Exploration Using Time Series Clustering

Leon Bornemann[1] · Tobias Bleifuß[1] · Dmitri Kalashnikov[2] · Felix Naumann[1] · Divesh Srivastava[2]

Received: 1 February 2018 / Accepted: 3 May 2018 / Published online: 25 May 2018

Abstract
Analysis of static data is one of the best studied research areas. However, data changes over time. These changes may reveal patterns or groups of similar values, properties, and entities. We study changes in large, publicly available data repositories by modelling them as time series and clustering these series by their similarity. In order to perform change exploration on real-world data we use the publicly available revision data of Wikipedia Infoboxes and weekly snapshots of IMDB.
The changes to the data are captured as events, which we call change records. In order to extract temporal behavior we count changes in time periods and propose a general transformation framework that aggregates groups of changes to numerical time series of different resolutions. We use these time series to study different application scenarios of unsupervised clustering. Our explorative results show that changes made to collaboratively edited data sources can help find characteristic behavior, distinguish entities or properties and provide insight into the respective domains.

1 Introduction

Nowadays, data is a resource that keeps growing to staggering numbers. Every day, databases store billions of social network interactions, transactions in e-commerce, or physical measurements. That being said, collected data is often not static. In fact, data changes frequently in almost all systems. Entire systems (such as databases) are built for the purpose of managing and documenting changes to the data.

However, the nature of changes is often unknown and can raise many questions for data analysts: Why does the data change this way? Are there regular patterns? Have any events caused large scale changes? Are there hidden data generation processes? It is clear that answering any of these questions can be of great interest as insights gained by such an analysis could help detect abnormal changes, assess data quality or discover useful correlations that can help to predict future changes.

While the dynamic changes of datasets from some specific domains are well understood, in many cases the data analyst has little prior intuition or knowledge about when, why or how the data might change. Often, the analyst does not even know what patterns he or she might be looking for. This is where data exploration comes into play. The analyst needs to be enabled to obtain a high-level understanding of the changes in the dataset in order to proceed with an in-depth analysis that might answer the questions raised above.

Clustering algorithms are a natural fit for such a task as they group similar data-objects. The results can serve a multitude of purposes, for example, data summarization, data categorization or outlier detection, all of which are relevant to get an understanding of previously unknown data. Additionally, the unsupervised nature of clustering algorithms makes them excellent tools for exploring unknown data since they do not require domain knowledge or ground truth, both of which are usually not available in an exploration scenario.

While exploration and analysis of historical data have been subject of many studies in different areas of research, these pieces of work are mostly disconnected. The different studies use different data formats, typically specific for a concrete task or application area. The *change-cube* [3] is an attempt to create a general data model that can represent both changes in data as well as changes in schema. The change-cube is a simple, yet powerful representation that stores changes as quadruples of the following format:

$$\langle \texttt{timestamp}, \texttt{id}, \texttt{property}, \texttt{value} \rangle$$

✉ Leon Bornemann
leon.bornemann@hpi.de

[1] Hasso-Plattner-Institut, Universität Potsdam,
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany

[2] AT&T Labs – Research, Bedminster, New Jersey, USA

We refer to one of these quadruples as a change record. A change record $\langle t, id, p, v \rangle$ describes that at a certain point of time t an entity with id id was changed in the property p which from then on has the new value v. As an example, consider:

Example 1

⟨20.01.2013, USA, President, Barack Obama⟩
⟨01.01.2014, USA, GDP Per capita, $52,839⟩
⟨01.01.2016, USA, GDP Per capita, $54,629⟩
⟨01.01.2018, USA, GDP Per capita, $57,220⟩
⟨20.01.2017, USA, President, Donald J. Trump⟩
⟨26.01.2018, Germany, GDP Per capita, $50,206⟩

This example change-cube consists of six change records, containing two entities, two different properties, and six distinct values. From the change records, we can see that the value of the property *GDP Per Capita* was constantly increasing for the entity *USA*, whereas for the entity *Germany* there is only one value present for this property.

The format can handle not only changes in the data but also changes to the schema, as entities being deleted can be represented by setting the values of all their properties to null and the first appearance of a property represents an insert. Due to its simplicity the change-cube can integrate data from various different formats or domains. Thus, change exploration or analysis frameworks that work on the change-cube can be applied to ubiquitous sources of data.

This paper extends the original work that proposed the change-cube format [3] by creating a framework for clustering data changes. The framework we propose groups changes by a user-supplied criterion and subsequently aggregates changes in the data to numerical time series by distributing change records into buckets and counting the number of change records in each bucket. The resulting time series are then clustered to provide the user with an overview over the different temporal behaviors in the data.

Overall, the main contributions of this paper are:

- An extensible framework for clustering changes, built on the change-cube as a general data format. We make our implementation available as an open source project.[1]
- An empirical evaluation that demonstrates how our framework can be used to uncover user behavior and format changes.

The rest of the paper is organized as follows: Sect. 2 outlines related work and Sect. 3 follows up with a detailed description of the proposed framework. Sect. 4 then briefly describes the two datasets we study in the evaluation. Sect. 5 presents our exploratory results where we describe how the usage of the framework helped us uncover user behavior and events leading to systematic changes. Finally, Sect. 6 concludes the paper and presents future research opportunities.

[1] https://github.com/HPI-Information-Systems/ChangeTimeSeriesClustering

2 Related Work

The related work for this paper encompasses two areas: exploration and analysis of changes in large scale data sources as well as clustering time series.

2.1 Change Exploration and Analysis

Studying changes of data over time is no new research area. In fact many researchers have been studying database changes or update histories. The difference between our work and existing work lies in the fact that most related work focusses on the data and adds a temporal component, for example through temporal databases [19], while we focus solely on the changes made to the data.

Information about data changes has been used for many specific problems. Frequently, schema changes have been looked at separately from data changes. Researchers tried to detect changes in the schema [20], analyzed the impact of schema changes on object-oriented applications [16], or conceived methods to automatically create mappings when schema changes occur [26]. Temporal changes have also been studied in the research area of linked data. For example, Dividino et al. suggested a framework to analyze data dynamics in the Linked Open Data cloud [8]. Umbrich et al. introduce different notions of change and define change detection mechanisms in linked data [24].

Dasu et al. use changes to database dynamics to infer knowledge about the database [5]. The concept is similar to ours, but the approach differs from ours in that Dasu et al. assume limited knowledge about the database and want to find out more whereas we assume full knowledge and thus are more interested in things like the data generation process. In general, our approach differs from these earlier works because we do not focus on a specific kind of change (for example schema changes) or a specific data format (such as linked data).

Many researchers have focussed on changes in data while assuming a stable schema and mentioning them all would exceed the scope of this paper. Important research areas when mining data changes are sequential pattern and event mining [9] or workflow and process mining [25].

Data exploration in databases has many aspects that are nicely summarized by Idreos et al. [11]. Important aspects include user interaction and assistance [4] or visualiza-

tion [6, 13]. There is not yet an interactive tool dedicated to the exploration of changes in data, apart from the work by Bleifuss et al. [3].

2.2 Clustering Time Series

Clustering time series is a large area of research and covering the intricacies of all subtopics is beyond the scope of this paper. A literature review by Aghabozorgi et al. [1] nicely summarizes its different aspects and is thus a good source for up to date literature on specific problems.

The use cases of time series clustering are manyfold. Researchers frequently take advantage of the unsupervised nature of clustering algorithms and thus apply them in explorative scenarios. A common scenario is the discovery of common or regular patterns. Patterns can be used for different purposes, such as summarization or prediction, for example in the domains of energy [12] or finance [10]. In some domains, the obtained patterns themselves are the desired result, as they represent newly discovered knowledge, for example about muscle activity [17]. Time series clustering is also frequently used to detect outliers. Application areas include astronomy [23] and transport [14]. Yet another scenario is finding prototypical representations of groups, which has been used in robotics [22].

In our framework, we make use of existing clustering techniques for time series data.

3 From Change-Cubes to Time Series Clustering

To apply time series clustering to change-cubes, we transform a change-cube into multiple time series. Figure 1 gives an overview of the proposed transformation framework.

Each step gives the user input options in order to finely tune the clustering. The curly brackets give examples for possible user-supplied criteria. Optional steps are in square brackets.

The first step when exploring or analyzing changes in a change-cube is the grouping phase. Since a single change record is not a time series, changes need to be grouped by a certain criterion, which we refer to as the key. This step thereby determines the information in the final clustering since each key (whose group is not filtered) is one object in the final clustering. Choices for the key thus depend on the use case. A few typical examples:

- **Group by Entity** results in one group per distinct entity name. For the change-cube in Example 1, this would result in two groups, named *USA* and *Germany*. We apply grouping by entity when analyzing edits to Wikipedia infoboxes (see Sect. 5.2), which means that one object in our clustering is comprised of all changes to an infobox of a page.

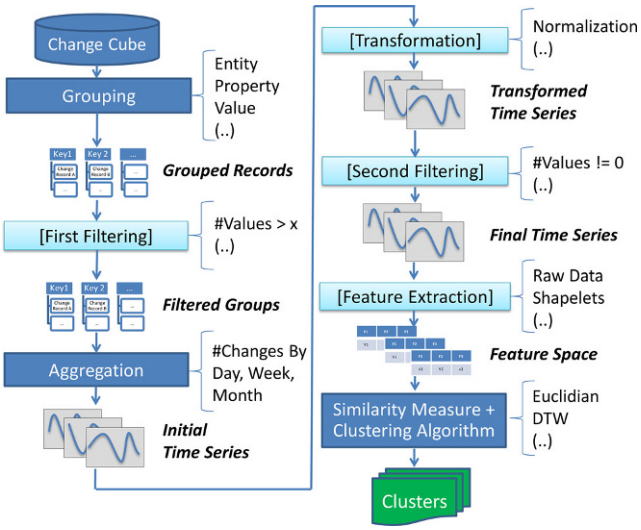

Fig. 1 Transforming a change-cube into clusters based on temporal behavior

- **Group by Property** results in one group per distinct property name. For the change-cube in Example 1 this would result in two groups with two and four members respectively. A possible use case is the search for similarly behaving properties across all entities.
- **Group by Value** – This will result in one group per distinct value. For the change-cube in Example 1 this would result in as many groups as there are change records. A possible use case is the search for events that cause inserts of certain values, for example, the election of US-presidents.
- **Group by {Entity, Property}** results in one group for each property of every entity. For the change-cube in Example 1 this would result in three groups. This grouping strategy is useful when comparing semantically identical properties of different entities. We used this strategy when searching for patterns in the user-votes for episodes of different TV shows in IMDB (see Sect. 5.1).

The results of this first grouping step are groups of change records. The next step is a filtering step, which might become necessary if there are many groups that could obscure the result. Because we transform the groups to time series, it is usually desirable to discard groups with a very small number of records, since those will have little temporal behavior to be analyzed. This is especially prevalent in data taken from the Wikipedia Infobox dataset (see Sect. 4), as it contains many articles that rarely change. In our experiments we discarded groups with fewer than 50 change records.

When ordered by timestamp, each resulting group is a sequence of change events $S = [(e_1, t_1), ..., (e_n, t_n)]$, where e_i are the change events and t_i are the timestamps ($t_i \leq t_{i+1}$). In order to obtain time series data, these

Table 1 Basic statistics about the IMDB change cube

# Change Records	85 Million
# Distinct Entities	16 Million
# Distinct Properties	20,000
# Distinct Values	5 Million

Table 2 Basic statistics about the WI change cube

# Change Records	122 Million
# Distinct Entities	99,000
# Distinct Properties	141,000
# Distinct Values	20 Million

categorical sequences need to be aggregated to numerical time series. We employ an approach that is essentially Piecewise Aggregate Approximation (PAA) [15]. Using a user-determined bucket size B we split the sequence into $M = \lceil \frac{t_n - t_1}{B} \rceil$ equal parts, where each part encompasses the same duration. We then simply sum up the number of events in each bucket, which gives us a time series of length M. In our experiments with the Wikipedia infobox dataset we used varying bucket sizes between 15 days and two months (see Sect. 5.2). The aggregation procedure can also be customized to the data at hand. Depending on the data it might make sense to use distance functions between adjacent values or transform the individual values to a numerical value.

The next step is time series transformation. Since the result of the previous step is now a set of time series, it is possible to apply established methods of time series analysis [1]. There are many possible transformations, and it is important to keep in mind that they should serve the desired result. If one is interested in structural similarity, without considering absolute values, the time series should be z-normalized. To z-normalize a time series $y = [y_1, ..., y_n]$, one calculates the new time series $y' = [y'_1, ..., y'_n]$ with $y'_i = \frac{y_i - \mu}{\sigma}$ where μ is the mean and σ is the standard deviation of y. This results in a time series, whose mean is approximately zero and whose standard deviation is close to one. If absolute values matter but their impact should be reduced, applying transforming functions like the square root or the logarithm might be a good choice. Several transformation steps can also be chained in order to account for many influencing factors. If the behavior of the raw time series is of interest one can skip this step.

We suggest a second and last filtering step after the transformation step to once again filter out time series that might prove to be uninteresting.

The next phase is the feature extraction phase, in which all input time series are mapped to equal-length numerical vectors, based on some procedure. A large amount of literature exists on feature extraction from time series [7, 18, 27] . This step is also optional: the raw data of equal-length time series (potentially padded with zeroes if there are no values present) can also be used as is. In practice, the appropriate feature extraction method heavily depends on the user's intention, so we do not give any default suggestion here.

In the last step, it is necessary to choose a similarity measure and a clustering algorithm. Once again the choice heavily depends on the goal of the analysis. A few examples include:

- **Euclidean Distance** – Absolute comparison of shape
- **Dynamic Time Warping (DTW)** – Shape comparison that allows time shifts and time stretching
- **Pearson's Correlation Coefficient** – Linear correlation between the time series.

When choosing the distance measure, it is important to chose a suitable clustering algorithm. For example, the k-means algorithm implicitly uses the Euclidean distance, whereas k-medoids or its variants can work with any distance measure.

Since our framework mainly consists of grouping, mapping and filtering operations, it can easily be implemented as a Map-Reduce program. As a proof of concept, we implemented the framework in Apache Spark. The implementation already supports basic possibilities for each step in the framework and can be easily customized by implementing the individual steps of the framework as Scala-functions.

4 The Datasets

We designed our framework as a structured approach of clustering data changes. For our experiments, we focused on two datasets for which the history of changes is publicly available: the International Movie Database (IMDB) and Wikipedia Infoboxes.

4.1 International Movie Database (IMDB)

IMDB gathers information about movies, TV shows and their episodes, but also about involved persons, such as actors and directors. The dataset is publicly available[2] as a snapshot of the current version and diffs of previous versions. By applying the diffs in reverse order to the current snapshot, we were able to reconstruct the data for roughly three years and five months (2014-02-21 until 2017-07-15). Due to the weekly resolution of the updates and some missing or broken updates, the dataset comprises data for 174 distinct timestamps. Some basic statistics about the dataset are given in Table 1.

IMDB has a relational architecture. However, the snapshots available to us are in a semi-structured textual format,

[2] ftp://ftp.fu-berlin.de/pub/misc/movies/database/frozendata/

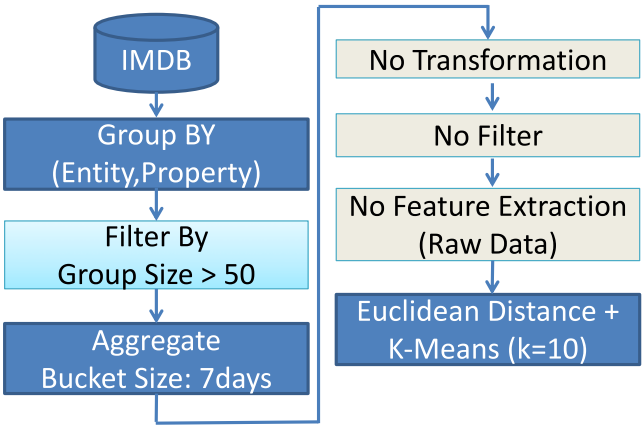

Fig. 2 Framework configuration for the first clustering run

which introduces difficulties when trying to reconstruct the relational tables. In particular, the reconstruction of column headers poses some difficulty, since these are often not given in the text files. Frequently, attribute information is simply appended to a tuple in brackets (example: ‘‘(as Monkey)”). For some attributes, we manually craft column names; we heuristically generate others. For example, inspection showed us that the roles of an actor in a movie were always surrounded by square brackets, so we were able to assign a useful name to this attribute. Whenever we were not able to infer such rules, we took the first word of the content as the attribute name. For example the token *(as Monkey)* would result in a property called ‘as’ with the value ‘Monkey’. We wrote a parser that parses the central tables and recursively executes patch commands in order to obtain the change records[3]. We adopt the following rules to extract change records from a tuple:

- Entities correspond to keys of the table schema. If keys cover more than one column, the two values are concatenated and separated by an unambiguous separator.
- Every attribute in the tuple that is not part of a key is a value of a property. The property name is the corresponding column name.

The resulting entities are either individual works, such as movies, TV shows, or episodes, or pairs of a person (like directors, actors or editors) and a movie, TV show or episode.

4.2 Wikipedia Infoboxes (WI)

In Wikipedia, infoboxes are concise, structured summarizations of key facts regarding an article, usually located at its top-right corner. This dataset contains all changes made to infoboxes in the English Wikipedia. The changes are parsed from the revisions of Wikipedia pages, which are publicly available[4]. Earlier works have also used this as a data source to extract infobox data [2].

Each change record represents the insert, update or deletion of one property of an infobox. In the quadruple, the Wikipedia page name corresponds to the entity, the key in the infobox (for example $\langle t, id, p, v\rangle$ or $\langle t, id, p, v\rangle$) corresponds to the property. In order to narrow the focus of the dataset, we only consider pages with infoboxes that were created using a popular infobox template. Infobox templates are standardised infobox schemata for a certain kind of entity, such as settlements, actors or companies. Templates are hierarchically organized since there can be specialized sub-templates. When assigning the labels, we applied the following procedure: By default, a page is assigned to the template that was present across the majority of revisions. If a page is associated with both a template and a sub-template, we pick the more specific template as the label. For our scope, we include infoboxes of templates that have at least 10.000 instances. There are 33 such templates.

The data encompasses roughly sixteen years and seven months (2001-01-18 until 2017-08-02) at a resolution of one second. Some basic statistics about this dataset are given in Table 2.

5 Framework Application

We demonstrate the usefulness of the framework by showing how its application to the two datasets helped uncover interesting properties and correlations in the them.

5.1 IMDB

Since we didn’t know much about the data, but were interested in the behavior of entity-specific properties we configured the framework as shown in Fig. 2.

We group by (Entity, Property)-pairs, because we wanted to explore the specific behavior of different properties over time, especially how ratings, and votes (the number of users that gave the entity a rating) change over time. We require a group to have at least 50 change records in order to filter out large amounts of barely known movies or shows that would obscure the clustering. Since the updates to IMDB occurred weekly, we chose a bucket size of 7 days. This setting resulted in binary time series (either the value of the property changed or not), which means that we maximize the information for the clustering algorithm while still obtaining time series of manageable dimensionality. Inspection reveals that the only property that was changed more than

[3] The parser is available at: https://github.com/HPI-Information-Systems/IMDBParser

[4] https://dumps.wikimedia.org/

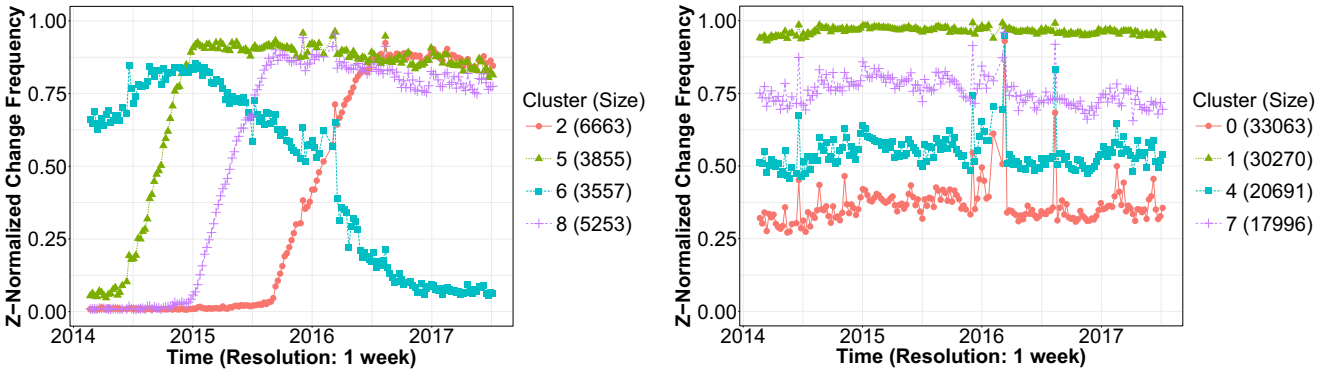

Fig. 3 Different cluster centers of the first clustering

Cluster 2	Cluster 5	Cluster 6	Cluster 8
2015: 1351	2014: 1213	2013: 317	2015: 1393
2016: 958	2013: 113	2012: 201	2014: 376
2014: 138	2011: 135	2014: 188	2013: 89
2013: 63	2012: 93	2011: 135	2012: 49
2012: 45	2010: 42	2010: 93	2011: 48

Fig. 4 Value distribution of the $\langle t, id, p, v \rangle$ property for the cluster centers shown in Fig. 3

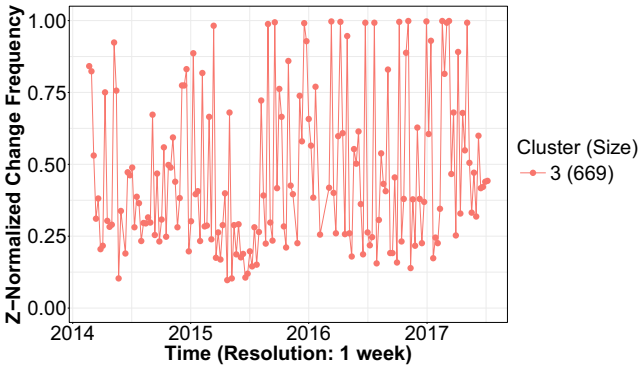

Fig. 5 Cluster containing 669 objects of which 659 are episodes of Doctor Who

50 times for each entity was $\langle t, id, p, v \rangle$. This means that in the following we are exploring whether there were new ratings given to a TV show, TV episode, or movie. It is also important to note that in the aggregation phase (which creates the time series) the framework only registers whether the value changed. It does not consider the actual numeric value of the attribute, since this would require prior domain knowledge that the property $\langle t, id, p, v \rangle$ is always numeric and thus would not be fitting for a general framework.

Figure 3 shows the different cluster centers obtained by the k-means algorithm. The left diagram shows selected cluster centers whose members show a strong increase or decrease in popularity. Inspection of the clusters revealed that this correlates strongly with the release dates of these works as shown by the value distributions for the $\langle t, id, p, v \rangle$ property in Fig. 4.

Cluster 1 is what we refer to as the popular cluster since it encompasses all entities that were changed almost every time. Clusters 0, 4 and 7 look like randomly distributed noise with different probabilities of noise occurrence. Apparently, the clustering algorithm was not able to find significant patterns for these entities. The clustering also produced two other cluster centers: one outlier cluster that contains just 2 objects and another cluster containing 669 objects, whose center is shown in Fig. 5.

Inspection of the cluster showed that 98% of its members are „Doctor Who“ episodes. Seeing such a strong correlation between the temporal behavior and TV show membership is unexpected and warrants an explanation. Automatically generated votes (by bots) seem unlikely, because the shape of the cluster center is rather irregular. We originally had the hypothesis of users being able to rate all episodes at once, which would explain the common peaks in the second half of the graph, but there are inconsistencies contradicting this hypothesis. We have contacted the IMDB administrators for possible explanations but have received no reply yet. The fact that the Doctor Who episodes showed such a specific pattern sparked the hypothesis that this could also be the case for episodes of other TV shows. To explore this new hypothesis we used the same setting as shown in Fig. 2 but pre-filtered the change-cube to only contain episodes of ten preselected TV shows. The TV shows now give us class labels, which means that the supervised clustering evaluation metric of purity can be used. To calculate purity one assigns each cluster c_i the label l_j that the majority of its members have. Subsequently purity is calculated by dividing the number of correctly assigned objects (having the same label as the cluster) by the total number of objects. The clustering achieved a purity of 62.0%, which is suprisingly high if you consider that we do the clustering just based on whether the number of votes for an episode changed or not. The complete confusion matrix is given in Table 3. The confusion matrix shows the labels in the columns and the

Table 3 Confusion matrix of the attempted clustering according to TV shows.

	0	1	2	3	4	5	6	7	8	9
Arrow	0	46	25	0	0	1	0	0	45	0
Doctor Who	0	15	49	219	208	0	55	71	0	127
Friends	0	233	8	0	0	1	0	0	2	0
House of Cards	0	26	13	0	0	0	0	0	26	0
Lost	0	116	2	0	0	2	0	0	0	0
Star Trek	0	79	1	0	0	1	0	0	0	0
The Big Bang Theory	0	163	26	0	0	0	0	0	45	0
The Fresh Prince of Bel Air	145	2	1	0	0	0	0	0	1	0
The Simpsons	0	526	65	0	0	26	0	0	44	0
Walker, Texas Ranger	0	0	200	0	0	0	0	0	0	0

Table 4 Purity for the different clustering attempts

Configuration	Purity
N	0.433
L_N	0.440
L_N_30	0.469
L_N_15	0.481
N_DBA_60	0.321

assigned clusters in the rows, which means it gives information about the distribution of the episodes of each TV show among the clusters. A clustering with purity 1.0 would have exactly one number greater zero in each column.

The confusion matrix shows that the clustering produces mostly pure clusters with the exception of Clusters 1, 2 and 8. The cluster centers of these are shown in Fig. 6. Similarly to the previous clustering, there is once more a cluster that contains all popular entities (Cluster 1). Cluster 2 represents the other end of that spectrum which are those episodes whose vote count rarely changes. Additional inspection showed that Cluster 8 contains the episodes that were released in the time period we observe which leads to the fact that they are clustered together (since they share zeros in the beginning).

It is also notable that the Doctor Who episodes are diverse enough in their behavior to be assigned into five different clusters. Overall the clustering shows that the frequency of user actions concerning individual episodes over a longer period of time can help to identify the corresponding TV show.

5.2 Wikipedia Infoboxes

Now we turn to the exploration process in the WI dataset. For this dataset we were especially interested in uncovering the data generation process. Specifically we wanted to look for (potentially automated) update patterns. As we have 33 different templates (which we refer to as categories) that categorize infoboxes appearing on pages, we can use these as class labels and see if the temporal behavior identifies them. If we are able to find an at least partially successful clustering, the cluster centroids can hint at patterns in the data. We used the configuration as shown in Fig. 7 as a basic configuration, which we subsequently modified.

In comparison to IMDB, we increased the bucket size to avoid the curse of dimensionality for the resulting time series. Additionally, we z-normalized the time series. This is necessary because in each category there are expected to be entities that change more frequently than others of the same category. If absolute values were to be compared, these entities would be assigned to the same cluster. Additionally, outliers (for example caused by vandalism or edit wars) could dominate the clustering. In contrast, for the use case of outlier detection, taking the absolute values would be beneficial. The configuration shown in Fig. 7 is our first base configuration, which we refer to as *N*. We modified this configuration by exploring other transformation sequences:

- **N** – Basic configuration, time series z-normalized, bucket size of 60 days.
- **L_N** – Logarithm of time series, then z-normalized (to dampen the effect of outliers within a time series).
- **L_N_30** – Same as L_N but with a bucket size of 30 days instead of 60.
- **L_N_15** – Same as L_N but with a bucket size of 15 days instead of 60.
- **N_DBA_60** – The same preprocessing steps as N, but with DTW as a distance measure and a modified version of k-means that uses the DTW Barycenter Averaging method to find the average time series with respect to the DTW distance measure for the centroid computation [21].

Table 4 gives an overview of the results of the different approaches.

The results show that smaller bucket sizes (and thus higher time series resolution) can help the clustering al-

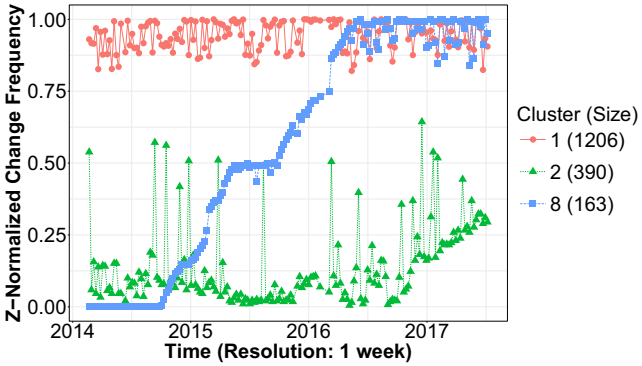

Fig. 6 Cluster centers of the impure clusters

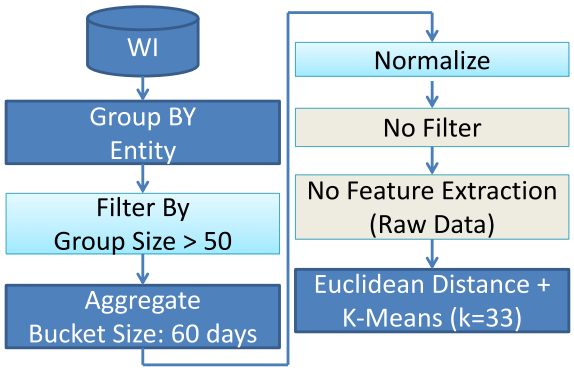

Fig. 7 Framework configuration for clustering the WI dataset

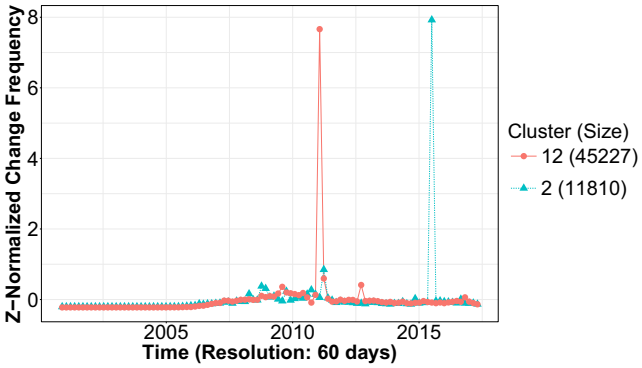

Fig. 8 Cluster centers of run N with high F-Measure

gorithms to achieve a higher purity. It is interesting to note that shape comparison with *N_DBA_60*, which allows for time shifts and time stretching, achieves lower purity than direct comparison of the time series via Euclidean distance. This hints at the fact that in contrast to many other application areas of time series similarity matching, absolute points of time play a large role in this dataset.

It is important to note that it is not necessarily our goal to achieve a clustering with a very high purity. High purity simply shows that many infoboxes of the same template actually get clustered together, thus meaning that they behave similarly. This means that there must be some characteristic changes for most of the instances of a template, which hints at automatic changes. After the clustering, we assigned each cluster a label by majority voting of its members and subsequently calculated the F-measure for each cluster of every clustering. Then we further investigated clusters with a high F-measure. Run N, for example, produced a cluster with an F-Measure of 0.820 (for the category baseball biography). This cluster (Cluster 2) is also discovered by *L_N*, *L_N_30* and *L_N_15* with similar F-measures. Plotting the cluster center (see Fig. 8) reveals a characteristic spike in the bucket that contains changes between 2015-09-02 and 2015-11-01.

Reviewing the actual revision history of sample pages of baseball players as well as the history of the template revealed that on the 14th September, 2015 the template *baseball biography* was merged with the template *mlb player* causing many automated changes in the pages of baseball players. We were able to uncover similar events by looking at the spikes of other cluster centers, for example a merge between the two templates *football biography* and *football biography 2* between 2011-03-27 and 2011-05-26 (see Cluster 12 in Fig. 8).

Overall, the clusterings helped us to identify specific events that caused automated changes to many data entries.

6 Conclusion

We presented a clustering framework for data change exploration, which can transform change-cubes, a recently suggested representation for data changes, to time series. The time series are then subsequently transformed, and clustered using user-defined transformations and clustering algorithms.

Our framework offers a flexible way to obtain high-level information about datasets, while being Map-Reduce friendly. Our initial experiments make us believe that it scales well. We demonstrated the usefulness of our framework on two examples from our current research. Using the framework, we were able to discover previously unknown patterns in voting behavior of users of IMDB. In the dataset consisting of changes made to Wikipedia infoboxes the clustering enabled us to discover events that caused automated changes for several templates.

In the future, we plan to integrate the clustering framework into the existing change exploration tool DBChEx[5] in order to create a smooth exploration process, which can enable the user to quickly discover patterns in the dataset. Apart from additional tool functionality, there are many more future work opportunities in this area. We plan to assemble the datasets we used and publish them together with

[5] https://hpi.de/naumann/projects/data-profiling-and-analytics/dbchex.html

an assortment of tools in a later stage of our research. In terms of clustering methods, there are many more clustering algorithms or approaches that can be used to cluster objects according to their temporal behavior. These could be tried out on new or existing data sets to see if they yield new exploration results. Furthermore, it is clear that exploration is just the first step when confronted with a change-cube. After exploring initial patterns in the data, a next logical step is to perform change analytics. Use cases for analyzing changes include data cleansing, prediction or process mining.

References

1. Aghabozorgi S, Shirkhorshidi AS, Wah TY (2015) Time-series clustering–a decade review. Inf Syst 53:16–38
2. Alfonseca E, Garrido G, Delort J, Peñas A (2013) WHAD: Wikipedia historical attributes data – historical structured data extraction and vandalism detection from the Wikipedia edit history. Lang Resour Eval 47(4):1163–1190
3. Bleifuss T, Johnson T, Kalashnikov DV, Naumann F, Shkapenyuk V, Srivastava D (2017) Enabling change exploration (vision). Fourth International Workshop on Exploratory Search in Databases and the Web (ExploreDB), pp 1–3
4. Cetintemel U, Cherniack M, DeBrabant J, Diao Y, Dimitriadou K, Kalinin A, Papaemmanouil O, Zdonik SB (2013) Query steering for interactive data exploration. Conference on Innovative Data Systems Research (CIDR).
5. Dasu T, Johnson T, Marathe A (2006) Database exploration using database dynamics. IEEE Data Eng Bull 29(2):43–59
6. Deligiannidis L, Kochut KJ, Sheth AP (2007) Rdf data exploration and visualization. ACM first workshop on CyberInfrastructure: information management in eScience, pp 39–46
7. Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci (Ny) 239:142–153
8. Dividino RQ, Gottron T, Scherp A, Gröner G (2014) From changes to dynamics: dynamics analysis of linked open data sources. Proceedings of the Extended Semantic Web Conference (ESWC).
9. Fournier-Viger P, Lin JCW, Kiran RU, Koh YS, Thomas R (2017) A survey of sequential pattern mining. Data Sci Pattern Recognit 1(1):54–77
10. Fu T-C, Chung F-L, Luk R, Ng V (2001) Pattern discovery from stock time series using self-organizing maps. Workshop Notes of KDD 2001 Workshop on Temporal Data Mining, pp 26–29
11. Idreos S, Papaemmanouil O, Chaudhuri S (2015) Overview of data exploration techniques. International Conference on Management of Data (SIGMOD), pp 277–281
12. Iglesias F, Kastner W (2013) Analysis of similarity measures in times series clustering for the discovery of building energy patterns. Energies 6(2):579–597
13. Keim DA, Kriegel HP (1994) VisDB: database exploration using multidimensional visualization. IEEE Comput Graph Appl 14(5):40–49
14. Li X, Li Z, Han J, Lee JG (2009) Temporal outlier detection in vehicle traffic data. International Conference on Data Engineering (ICDE), pp 1319–1322
15. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
16. Maule A, Emmerich W, Rosenblum DS (2008) Impact analysis of database schema changes. International Conference on Software Engineering (ICSE). ACM, New York, pp 451–460
17. Mörchen F, Ultsch A, Hoos O (2005) Extracting interpretable muscle activation patterns with time series knowledge mining. Int J Knowledgebased Intell Eng Syst 9(3):197–208
18. Olszewski RT (2001) Generalized feature extraction for structural pattern recognition in time-series data. Tech. rep. Carnegie-Mellon University, School of Computer Science, Pittsburgh
19. Özsoyoglu G, Snodgrass RT (1995) Temporal and real-time databases: a survey. IEEE Trans Knowl Data Eng 7(4):513–532
20. Papavassiliou V, Flouris G, Fundulaki I, Kotzinos D, Christophides V (2009) On detecting high-level changes in RDF/S KBs. International Semantic Web Conference (ISWC), pp 473–488
21. Petitjean F, Ketterlin A, Gançarski P (2011) A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognit 44(3):678–693
22. Ramoni M, Sebastiani P, Cohen P (2000) Multivariate clustering by dynamics. National Conference on Artificial Intelligence (AAAI), pp 633–638
23. Rebbapragada U, Protopapas P, Brodley CE, Alcock C (2009) Finding anomalous periodic time series. Mach Learn 74(3):281–313
24. Umbrich J, Decker S, Hausenblas M, Polleres A, Hogan A (2010) Towards dataset dynamics: change frequency of linked open data sources. International Workshop on Linked Data on the Web.
25. Van Der Aalst W (2012) Process mining: overview and opportunities. ACM Trans Manag Inf Syst 3(2):7
26. Velegrakis Y, Miller J, Popa L (2004) Preserving mapping consistency under schema changes. VLDB J 13(3):274–293
27. Xing Z, Pei J, Yu PS, Wang K (2011) Extracting interpretable features for early classification on time series. SIAM International Conference on Data Mining, pp 247–258