Detecting Online Conversations Going Viral

Time-series aware evaluation of change detection algorithms

Matt Chapman
matthew.chapman@student.uva.nl

Spring 2017, 27 pages

Supervisor: Evangelos Kanoulas, Universiteit van Amsterdam
Host organisation: Buzzcapture International, http://www.buzzcapture.com

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Contents

Abstract

Write abstract

Chapter 1

Problem Statement & Motivation

1.1 Problem Statement

When asserting the veracity of an approach to a particular problem in software engineering, this is almost always achieved by utilising some sort of metric or measure. To give an example, when evaluating the quality of some new piece of software, this could be done through calculating measures such as McCabe Complexity [McC76] or unit test coverage.

The problem of detecting change points in a data stream is no different. There exists within this field a number of metrics that may be used to prove the accuracy and effectiveness of a given approach, and these often fall into one of three categories: binary classification (e.g. F1 score), clustering (e.g. BCubed, or the Rand Index), or information retrieval (utilising scoring systems such as those utilised in the TREC 2016 Real Time Summarisaion Track: [tre]).

Being that there are myriad ways to detect change points in a data stream, it is necessary to implement the application of some evaluation measure to back up an assertion that, for example, approach A is better than approach B for some data set y_s. The application of these measures can, in some cases, result in more questions than answers, especially in situations where two different measures may disagree with the results of the aforementioned approaches A and B.

However, for the number of change point detection methods that exist, there is an almost equal number of ways to evaluate the results of the methods. There is little agreement between researchers on the 'correct' method to use, and as this research will show, there are problems that exist when utilising measures that have generally been widely accepted for this purpose.

Outside of the 'normal' applications of change point detection (for example, spotting the onset of 'storm seasons' in oceanographic data [KEJ11] or the detection of past changes in the valuation of a currency, such as BitCoin [BNZ14], there is also an application for change point detection as an 'event detection' mechanism, when applied to data such as that sourced from online social media platforms.

There exists methods of event detection in social media data utilising approaches such as term frequency counting, lexicon-based recognition and context-free-grammar algorithms . However, these approaches rely on (sometimes computationally expensive) analysis of the content of messages being posted on social media platforms. Being that change point detection (and especially *online* change point detection algorithms can be utilised for the detection of past events, it stands to reason that these algorithms can also be utilised for event detection when applied to pre-computed data such as conversation volume or reach of a particular conversation. There certainly exists a requirement in the field of online reputation management to be able to inform businesses (in a timely manner) that a spike in conversation volume is occurring, and thus they may need to carry out some action to mitigate reputational damage if the conversation sentiment is negative.

cite yordi

Therefore, this thesis intends to answer the following research questions:

RQ1 Are existing change point detection algorithms effective at detecting changes/events in social media data in a timely fashion?

RQ2 Are measures not specifically defined for the purpose of change detection evaluation effective in this domain?

RQ3 Do the aforementioned measures agree with by-eye analysis of results from change detection methods?

RQ4 If existing measures are deficient in some way, what would an 'ideal' measure look like? reword

From these research questions, the following sub-questions have been formulated:

SQ1.1 Which algorithm, out of those being analysed, performs the 'best' when applied against real world data?

SQ2.1 In what way do established measures fail to behave correctly?

SQ2.2 Which evaluation measure provides the best evaluation, given the functional requirements set forth by the host organisation?

this needs expansion, I think

1.2 Motivation

This particular research is motivated specifically by the online reputation management sector. The business hosting this research project (Buzzcapture International [http://www.buzzcapture.com]) is a Dutch consultancy that provides online reputation management services to other businesses throughout Europe. Chief among these services is the BrandMonitor application, which, among other features, provides a rudimentary notification system for clients that is triggered once there is an absolute increase in conversation volume (conversation volume being defined as the number of tweets relevant to the client over a given time period).

Buzzcapture made a project available to students of the Universiteit van Amsterdam, wherein they would provide a method for more effectively providing these notifications, based on more than just an arbitrary threshold on conversation volume or some other computed metric. Upon accepting this project, research was carried out into the field of change detection algorithms, during which it was found that there was not a *single* accepted approach for evaluating measures. Indeed, for every publication that described some novel change point detection algorithm, there was a slightly different approach for evaluating it, and proving the veracity of algorithm in a certain situation. Most publications made use of some sort of binary classification measure (for example, [QAWZ15], [BNZ14], [PRG10]), while others had small variations on that theme, providing additional takes on the binary classification approach using methods such as *Receiver Operating Characteristic* curves (for example [FP99] & [DDD05]). Additionally, there were publications that made use of clustering measures, calculated using a segmentation of the time series based on computed change points, such as [MJ12].

It is the intention of this thesis to not only answer the research questions set out in § 1.1, but also to provide a robust recommendation and working prototype of a change detection methodology which could then eventually be implemented into the Brand Monitor tool to supply timely and relevant notifications to clients when a conversation concerning their brand exhibits a change in behaviour or otherwise 'goes viral'.

Chapter 2

Background & Context

2.1 Change Detection Algorithms

Change detection first came about as a quality control measure in manufacturing, and methods within this domain are generally referred to as *control charts*. Since the inception of approaches such as CUSUM that provide the possibility for on-line evaluation of continuous data streams, change detection has grown as a field. With applications such as epidemic detection, online reputation management and infrastructure error detection, change detection is hugely useful both as an academic problem and in production systems of myriad application.

2.1.1 Pruned Exact Linear Time

Pruned Exact Linear Time (henceforth referred to as *PELT* is a modern change point detection method proposed by R. Killick et. al in 'Optimal Detection Changepoints With a Linear Computational Cost', in 2012 [KFE11].

PELT is an *exact* method of computing change points, based on segmentation of data streams, in the same manner as Segment Neighbourhood and Binary Segmentation. PELT is a modification of the *Optimal Partitioning* method proposed by Brad Jackson et. al in 2003, in . cite

Jackson's method has a computational cost of $O(n^2)$, while Killick's PELT method improves on this with a computational complexity of $O(n)$. cite

The PELT method works similarly to Segment Neighbourhood and Binary Segmentation approaches, in that it attempts to fit a model of segmentation of a given data set, minimising the cost of the segmentation to achieve an optimal distribution of segments and therefore change points.

1 describes a pseudocode implementation of the PELT algorithm, as described in [EFK11]

Algorithm 1: PELT Method for change point detection

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
A measure of fit $C(.)$ dependent on the data.
A penalty constant β which does not depend on the number or location of change points.
A constant K that satisfies $C(y_{(t+1):s}) + C(y_{(s+1):T}) + K \leq C(y_{t+1):T})$

Initialise: Let n = length of data and set $F(0) = -\beta$, $cp(0) = NULL$

for $\tau* = 1, \ldots, n$ **do**
  Calculate $F(\tau*) = min_{\tau \in R_{\tau*}}[F(\tau) + C(y_{\tau+1:\tau*}) + \beta]$
  Let $\tau^1 = arg\Big\{min_{\tau \in R_{\tau*}}[F(\tau) + C(y_{(\tau+1):\tau*}) + \beta]\Big\}$
  Set $cp(\tau*) = [cp(\tau^1), \tau^1]$
  Set $R_{\tau^*+1} = \{\tau \in R_{\tau^*} \cup \{\tau^*\} : F(\tau) + C(y_{\tau+1:\tau^*}) + K \leq F(\tau^*)\}$
end
Output: the change points recorded in $cp(n)$

2.1.2 Segment Neighbourhoods

Proposed by Auger and Lawrence in 1989, Segment Neighbourhood (*SegNeigh*) is another example of cite an exact segmentation algorithm for the detection of change points in data. SegNeigh utilises dynamic programming to search the segmentation space (defined as the maximum number of change points in a given data stream) and then compute the cost function for every possible segmentation of the data. In this way, the location and number of change points can be computed exactly by taking the segmentation that returns the lowest cost function result.

2 describes a pseudocode implementation of the SegNeigh algorithm, as described in [EFK11]

Algorithm 2: Generic Segment Neighbourhoods method for change point detection

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$
A measure of fit $R(.)$ dependent on the data which needs to be minimised.
An integer, $M - 1$ specifying the maximum number of change points to find.
Initialise: Let n = length of data.
Calculate $q^1_{i,j} = R(y_{i:j})$ for all $i, j \in [1, n]$ such that $i < j$.

for $m = 2, \ldots, M$ **do**
  for $j \in \{1, 2, \ldots, n\}$ **do**
    Calculate $q^m_{1,j} = min_{v \in [1,j]}(q^{m-1}_{1,v} + q^1_{v+1,j})$
  end
  Set $\tau_{m,1}$ to be the v that minimises $(q^{m-1}_{1,v} + q^1_{v+1,n})$
  for $i \in \{2, 3, \ldots, M\}$ **do**
    Let $\tau_{m,i}$ to be the v that minimises $(q^{m-i-1}_{1,v} + q^1_{v+1,cp_{m,i-1}})$
  end
end
Output: For $m = 1, \ldots, M$: the total measure of fit, $q^m_{1,n}$ for $m - 1$ change points and the location of the change points for that fit, $\tau_{m,1:m}$.

2.1.3 Binary Segmentation

Binary Segmentation is a popular method for change point detection, widely utilised and cited in change point related literature such as . Binary segmentation is a method that recursively applies a cite single change point detection method. On the first iteration, if a change point is detected, the data is split around that change point (resulting in two data sets) and the change point method is run

again on the two resulting data sets. This process is repeated such that many data set segments are created, and runs until no further change points are detected.

Binary Segmentation is an *approximate* change point detection approach, and returns estimated change point locations - unlike PELT and SegNeigh - which return exact change point locations.

3 describes a pseudocode implementation of the BinSeg algorithm, as described in [EFK11]

Algorithm 3: Generic Binary Segmentation method for change point detection

Input: A set of data of the form, $(y_1, y_2, \ldots, y_n)$
A test statistic $\Lambda(.)$ dependent on the data
An estimator of change point position $\hat{\tau}(.)$
A rejection threshold C.

Initialise: Let $C = \emptyset$, and $S = \{[1, n]\}$

while $S \neq \emptyset$ **do**
  Choose an element of S, denote element as $[s, t]$
  if $\Lambda(y_{s:t}) < C$ **then**
    remove $[s, t]$ from S
  end
  if $\Lambda(y_{s:t})$ **then**
    remove $[s, t]$ from S
    calculate $r = \hat{\tau}(y_{s:t}) + s - 1$, and add r to C
    if $r \neq s$ **then**
      add $[s, r]$ to S
    end
    if $r \neq t - 1$ **then**
      add $[r + 1, t]$ to S
    end
  end
end

Output: the set of change points recorded in C

2.2 Algorithm Configuration

Change detection is an *unbounded* problem. Left without some system of constraint, the algorithm could theoretically run to infinity. Indeed, one of the algorithms utilised in this research, PELT, when left unbounded, will detect every data point in the time-series as a change point. This result is *technically* correct, but not useful for our purposes. For this reason, the algorithms implement a penalty system, allowing for an optimal number of change points to be detected.

citation needed

2.2.1 Penalty Scores

Penalty scores operate as a mechanism for optimising an unbounded problem such as the one being addressed here. Haynes et al. define the problem as follows [HEF14]: Given time series data points $y_1, \ldots, y_n$, the series will contain m change points such that their locations $\tau_{1:m} = (\tau_1, \ldots, \tau_m)$, where $\{\tau_i \in \mathbb{Z} \mid 1 \leqslant \tau_i \leqslant n-1\}$. τ_0 is assumed to be 0, and τ_{m+1} is assumed to be n. In this way we can state that a given change detection algorithm will split the time series into $m+1$ segments such that segment i contains the points $y_{(\tau_{i-1}+1):\tau_i} = (y_{\tau_{i-1}+1}, \ldots, y_{\tau_i})$. A *cost function* for a segmentation of the data set y_s can be defined as $C = (y_{s+1:t})$. This cost function defines a cost for a given segmentation containing points $y_{s+1:t}$. To illustrate, the cost function C, in the context of binary classification, returns a value that increases for an 'incorrect' classification and decreases for a 'correct' classification.

At this point it can be stated that the *constrained minimisation problem* is as follows:

$$Q_m(y_{1:n}) = \min_{\tau_{1:m}} \left\{ \sum_{i=1}^{m+1} [C(y_{(\tau_{i-1}+1):\tau_i})] \right\} \tag{2.1}$$

Equation 2.1 is an equation showing the optimisation problem that change point detection presents. If the equation is taken intuitively, it shows that for a dataset with a *known* number of change points, the segmentation of the data around the change points can be effectively estimated by obtaining the segmentation that returns the minimum cost based on the cost function C.

However, the problem being solved by many change detection algorithms involves an unknown number of change points, at unknown locations. In this case we can estimate the following to obtain the number and location of the change points:

$$\min_m \left\{ Q_m(y_{1:n}) + f(m) \right\} \tag{2.2}$$

Equation 2.2 includes a *penalty function* $f(m)$ that increases proportionally with the number of change points m. Essentially, as the segmentation calculated as Q_m increases in size, so does the result of the penalty function $f(m)$. These two elements are at-odds with each other, presenting a problem that requires optimisation to correctly estimate the number and location of change points in the data set y_s. If $f(m)$ increases in a linear fashion with m, we can define a *penalised minimisation problem* in the following manner:

$$Q_m(y_{1:n}, \beta) = \min_{m, \tau_{1:m}} \left\{ \sum_{i=1}^{m+1} [C(y_{(\tau_{i-1}+1):\tau_i}) + \beta] \right\} \tag{2.3}$$

In this equation, β is the penalty function defined previously as $f(m)$.

There are a number of established approaches to calculating penalty values for unbounded problems, chiefly among which are Schwarz Information Criterion (SIC)/Bayesian Information Criterion (BIC) [Sch78], Akaike Information Criterion (AIC) [Aka74] and Hannan-Quinn [HQ79]. Of these approaches, it is necessary to experiment to find the scheme that produces the 'correct' number of change points for a given dataset. The penalty schemes are defined as follows:

$$\text{SIC} = \ln(n)k - 2\ln(\hat{L}) \tag{2.4}$$

$$\text{AIC} = 2k - 2\ln(\hat{L}) \tag{2.5}$$

$$\text{HQC} = -2L_{max} + 2k\ln(\ln(n)) \tag{2.6}$$

Where n is the total number of observations, k is the number of *free parameters*[1] and $\hat{L}$ is the maximum value of a likelihood function.

reword, wikipedia

The penalty function returns a value which is represented by β in Equation 2.3, which then limits the number of possible segmentations returned by Q. In this way, varying the penalty function β can alter the results given by a change detection algorithm by increasing or decreasing the number of change points detected.

The `changepoint` package requires the provision of a penalty function for each algorithm, which has been arbitrarily chosen as 'SIC' for this research. This decision is based upon advice published in [EFK11], in which it is stated that AIC (while popular in the field of change point detection as a penalty term) tends to asymptotically overestimate k.

While the selection of an optimal penalty term for a given data set and change point detection algorithm is an open research question in itself,

2.2.2 Distribution Assumptions

It is also important to note that many change point detection methods make an assumption about the underlying distribution of the data being analysed. This distribution can be, for example, a normal distribution, poisson

[1] variables that are unable to be predicted by a given model

2.3 Evaluation Measures

As briefly discussed in the introduction to this thesis, there are a number of pre-existing approaches for the evaluation of change detection methods. Here, the measures being evaluated are briefly explained:

2.3.1 F1 Score

This measure is utilised for testing accuracy in problems of binary classification. It considers two different measures, *precision* and *recall*, and takes a harmonic mean of the two measures to compute the final score. cite this?

To calculate Precision and Recall, a *confusion matrix* is first constructed to provide values for the total number of true positives, false negatives, false positives and true negatives:

Prediction Outcome

Actual Value	**p**	**n**	**Totals**
p′	True Positive	False Negative	P′
n′	False Positive	True Negative	N′
Totals	P	N	

Recall is computed as the number of correct positive results, divided by the number of positive results that should have been detected. *Precision* is computed as the number of correct positive results divided by the number of all possible positive results:

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

The F1 score calculation (the harmonic mean of both recall and precision) can be described in general terms as follows:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.9}$$

As the F1 score is a binary classification measure, it can only be used to test the precision of an algorithm in a single domain, that is, was the change detected or not. Precision, Recall and F1 Score all provide a score s such that $\{s \in \mathbb{R} \mid 0 \leq s \leq 1\}$

2.3.2 Rand Index

This measure is for computing the similarity between two clusters of data points. It is used for calculating the overall accuracy of a given clustering approach, when compared against a set of ground truth clusters. The Rand Index is defined as:

$$R = \frac{a + b}{a + b + c + d} \tag{2.10}$$

Given a set of data points S, partitioned through two different methods, which we shall refer to as X and Y. Based on this knowledge the following can be defined:

- a = total number of pairs that were partitioned into the *same* subset by both X and Y
- b = total number of pairs that were partitioned into *different* subsets by both X and Y
- c = total number of pairs that were partitioned into the *same* subset by X and into a different subset by Y
- d = total number of pairs that were partitioned into the *same* subset by Y and into a different subset by X

reword, wikipedia

Intuitively, it can be stated that $a+b$ is the total number of agreements between methods X and Y, while $c+d$ is the total number of disagreements. This calculation will return 0 for completely different clusters and 1 for identical clusters. The Rand Index provides a score s such that $\{s \in \mathbb{R} \mid 0 \leq s \leq 1\}$

2.3.3 Adjusted Rand Index

Similar to the Rand Index, but adjusted to take into account the random chance of pairs being assigned to the same cluster by both approaches being compared. While the Rand Index is suited for comparing a segmentation method against a known-good *oracle* method, the adjusted index is more suited to comparing two differing approaches [MJ12]. It is defined as:

$$R_{adjusted} = \frac{R - R_{expected}}{R_{max} - R_{expected}} \tag{2.11}$$

where $R_{expected}$ is defined as the expected Rand Index score for two completely random classifications of the given dataand R_{max} is defined as the maximum Rand Index value - generally 1.

cite, https://arxiv.o

The Adjusted Rand Index provides a score s such that $\{s \in \mathbb{R} \mid -1 \leq s \leq 1\}$. This is different to the score returned by the 'basic' Rand Index, in that it is possible for a negative value to be returned.

2.3.4 BCubed

BCubed is similar to the Adjusted Rand Index, in that it is adjusted to be appropriate for all constraints in a clustering problem. It was developed by Bagga and Baldwinin 1998.

cite

BCubed operates in a similar way to the Recall and Precision classification metrics, in that it computes precision and recall values, before calculating a harmonic mean of the two metrics. BCubed also makes use of an additional *correctness* function defined as follows, where e and e' are equivalent elements in a ground truth and computed clustering respectively, L is a computed label and C is a ground truth category:

$$\text{Correctness}(e, e') = \begin{cases} 1 & \text{iff } L(e) = L(e') \leftrightarrow C(e) = C(e') \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

Equation 2.12 can be stated intuitively as returning 1 if and only if both the ground truth and computed clusterings place the element e into the same cluster, returning 0 in all other cases.

Precision and Recall can then be calculated as follows:

$$\text{Precision} = Avg_e[Avg_{e'.C(e)=C(e')}[Correctness(e, e')]] \tag{2.13}$$
$$\text{Recall} = Avg_e[Avg_{e'.L(e)=L(e')}[Correctness(e, e')]] \tag{2.14}$$

To obtain the F score, a harmonic mean of Precision and Recall is taken in the same manner as the F1 score for binary classification:

$$F_{bc} = 2 \cdot \frac{1}{\frac{1}{Recall_{bc}} + \frac{1}{Precision_{bc}}} \tag{2.15}$$

Chapter 3

Research Method

3.1 Introduction

The purpose of this thesis is twofold: firstly, to conduct a meta analysis of evaluation measures, and how they perform in the domain of change point detection problems. Being that these measures are metrics designed for other problems that are not necessarily change point detection, it stands to reason that there are perhaps situations where families of metrics will disagree with each-other, or disagree with by-eye evaluation by domain experts. The second purpose is to evaluate the veracity of a number of existing change point detection algorithms when they are applied to data from social media.

This thesis consists of two distinct parts:

- Meta-analysis of evaluation measures, fulfilling RQ2 and associated sub-questions.
- Application of change detection approaches to real world data and evaluation based on requirements elicitation and evaluation measures.

3.2 Evaluation Pipeline Construction

This thesis will analyse three different change detection algorithms: *Pruned Exact Linear Time*, *Binary Segmentation*and *Segment Neighbourhoods*. These will be referred to as *PELT*, *BinSeg* and *SegNeigh* respectively. The algorithms will be briefly discussed in this thesis, along with the chosen critical values such as *minimum segment length*, *penalty scoring* and *assumed underlying distribution*.

cite

cite

cite

The algorithms will then be applied to a collection of datasets falling into two categories: real-world conversation volume data taken from Twitter, and simulated data generated according to certain constraints which will also be discussed here.

The results provided by each technique will then be evaluated according to the following measures: Precision, Recall, F1 score, Rand Index, Adjusted Rand Index, Bcubed Precision, Bcubed Recall and Bcubed F-Score. A meta-analysis will take place to evaluate the effectiveness of these methods when compared with each other and by-eye analysis from social media domain experts.

maybe not

The method of evaluating the approaches will be developed using a combination of Python and R. R is a combined language and environment created for the purpose of statistical computing [R C15].

Python is being utilised also due to the availability of relevant software packages for the purposes of evaluation measure calculation.

3.2.1 Calculation of Changepoint Locations

`changepoint` is a powerful R package that provides a number of different change detection algorithms, along with various approaches to penalty values. `changepoint` offers change detection in mean, variance and combinations of the two, using the AMOC, PELT, Binary Segmentation and Segment Neighbourhood algorithms.

cite these

`Changepoint` was developed by Rebecca Killick and Idris A. Eckley and is provided free of charge under the GNU general public license [KE14].

3.2.2 Calculation of Evaluation Measures

For the calculation of Precision, Recall and F1 Score, the R package `caret` is being utilised. `caret` is an acronym for 'Classification and Regression Training, and provides a number of tools for data manipulation, model creation and tuning, and performance measurements. Through the use of this package, it is possible to generate a *confusion matrix* based on algorithm output, and generate various performance measures based upon this. `caret` was written by Max Kuhn, and is provided free of charge under the GPL license [**?**].

For the calculation of the Rand Index and Adjusted Rand Index the R package `phyclust` is being used. This package was developed by Wei-Chen Chen, for the purposes of providing a phyloclustering implementation. While this approach is not something being examined in this thesis, the package does provide an implementation of the Rand Index and Adjusted Rand Index metrics, which are relevant to this research. This package is also provided free of charge under the GPL license.

cite

Calculating the BCubed precision, recall and f-score metrics is being carried out in Python, using the `python-bcubed` project. This is a small utility library for the calculation of BCubed metrics, developed by Hugo Hromic and provided under the MIT license [**?**]. In order to interface with this library via R, the package `rPython` is being used. This package provides functionality for running Python code and retrieving results in R, and was developed by Carlos J. Gil Bellosta. It is provided under a GPL-2 license [**?**].

Additionally, the `ROCR` package was utilised during the research phase of this project to generate Receiver Operating Characteristic (ROC) curves - a method used by some publications to evaluate change point detection methods. While ROC curves are not being utilised to produce the results published in this thesis, it was nonetheless an important part of the research, and provided useful insights into binary classification as a field and as a method of evaluation. `ROCR` was developed by Tobias Sing et. al. and is provided free of charge under a GPL license [**?**]

3.3 Measures Meta-Analysis

The experiments hereunder were developed to fulfil the requirement to answer research question 2, and it's associated sub-questions:

- Are measures not specifically defined for the purpose of change detection evaluation effective in this domain?
  - In what way do established measures fail to behave correctly?
  - Which evaluation measure provides the best evaluation, given the functional requirements set forth by the host organisation?

The research questions above are to be answered using a series of experiments carried out using simulated data and algorithms. The experiments intend to test the following criteria:

1. Dependence on sample size (otherwise referred to here as time series length)
2. Impact of data preceding a known change point and its detection
3. Impact of data following a known change point and its detection
4. Ability to provide a temporal penalty for late, early or exact detections
5. Ability to penalise correctly for false positives
6. Ability to penalise correctly for false negatives

3.3.1 Experiment Listings

For each experiment description, the sample size (otherwise known as time series length) is denoted by n, the true change point in the time series is denoted as τ_n and any 'detected' change points are denoted by τ'_n. For each experiment, the changing variable (the iterator in the R source code) is denoted as i.

Experiment 1

This experiment involves increasing the 'head' (sample size prior to a single known change point) of a time series, prepending to and thus lengthening the time series. The time series contains a single known change point, and a detected change point provided by a pseudo-algorithm is detected 6 points before the true change point.

For each prepended point, all of the evaluation metrics are calculated and plotted. The experiment evaluates both criteria 1 and 2.

The experiment runs such that $n = 55$ on commencement, and $n = 500$ upon completion, with each iteration adding a single data point such that $i = 445$. For each iteration, $\tau = i + 1$ and $\tau' = n - 4$.

Experiment 2

This experiment involves increasing the 'tail' (sample size after a single known change point) of a time series, appending to and thus lengthening the time series. The time series contains a single known change point and a detected change point provided by a pseudo-algorithm is detected 4 points before the true change point.

For each appended point, all of the evaluation metrics are calculated and plotted. The experiment evaluates both criteria 1 and 3.

The experiment runs such that $n = 55$ on commencement, and $n = 550$ upon completion, with each iteration adding a single data point such that i runs from 2 to 500. For each iteration, $\tau = 51$ and $\tau' = 45$.

Experiment 3

This experiment involves moving a known change point and a computed change point through a fixed length time series. The time series contains a single known change point and a detected change point provided by a pseudo-algorithm that is detected 5 data points before the true change point.

For each iteration of the experiment, all of the evaluation metrics are calculated and plotted. The experiment evaluates both criteria 1 and 2.

The experiment runs such that n is constant at $n = 501$. For each iteration, $\tau = i$ and $\tau' = i - 3$, for values of i from 5 to 500.

Experiment 4

This experiment involves moving a detected change point provided by a 'pseudo-algorithm' through a time series of fixed length, thus evaluating the ability of a measure to provide a temporal penalty for late, early, or exact detections.

For each iteration of the experiment, all of the evaluation metrics are calculated and plotted. The experiment evaluates criterion 4.

The experiment runs such that n is constant at $n = 500$. For each iteration $\tau = 51$ and $\tau' = i$, where $i = 51$ at the commencement (such that $\tau = \tau'$) and $i = 500$ at completion.

Experiment 5

This experiment involves adding 'false positive' change point detections to a time series of fixed length with a single known change point location.

For each iteration of the experiment, all of the evaluation metrics are calculated and plotted. The Experiment evaluates criterion 4.

The experiment runs such that n is constant at $n = 500$. For each iteration $\tau = 51$ and $\tau' = i$, where $i = 51$ at the commencement of the experiment, $i = 55$ at the second iteration, and increases in increments of 5 until $i = 500$. This serves to move the detected change point through the time series, starting at the true change point and progressing to the end of the time series, ending at the final point of the time series.

Experiment 6

Removing 'false negative' results (by way of explanation, adding correct detections) to a fixed length time series, thus increasing the number of correctly detected change points.

Experiment 7

Adding both true change points and detected change points from a 'pseudo algorithm' that provides detections n points to a time series, by appending to it, and thus increasing it's length. check this

Experiment 8

Adding both true change points and detected change points from a 'pseudo algorithm' that provides detections n points to a time series of fixed length. check this

3.3.2 Illustrative Algorithms for Experiments

Algorithm 4: Experiment to plot the effect of increasing 'head' length of a dataset on metrics

```
1 begin
2     foreach iteration do
3         Add 1 point to head of data, increasing length
4         Calculate metrics for current τ, τ′ against time series
5         Store results
6     end
7 end
```

Algorithm 5: Experiment to plot the effect of increasing 'tail' length of a dataset on metrics

```
1 begin
2     foreach iteration do
3         Add 1 point to tail of data, increasing length
4         Calculate metrics for current τ, τ′ against time series
5         Store results
6     end
7 end
```

Algorithm 6: Experiment to plot the effect of increasing 'tail' length of a dataset on metrics

```
1 begin
2     foreach iteration do
3         Move τ and τ′ 1 point further in time series
4         Calculate metrics for current τ, τ′ against time series
5         Store results
6     end
7 end
```

Algorithm 7: Experiment to plot the effect of temporal distance between τ and τ' on metrics

begin
  foreach *iteration* **do**
    Leave τ unchanged
    $\tau' \leftarrow \tau' + 1$
    Calculate metrics for current τ, τ' against time series
    Store results
  end
end

Algorithm 8: Experiment to plot the effect of increasing the number of false positive detections on metrics

begin
  foreach *iteration* **do**
    Add 1 additional false positive, such that $\tau'_{n+1} = \tau'_n + 5$
    Calculate metrics for current τ, τ' against time series
    Store results
  end
end

Algorithm 9: Experiment to plot the effect of decreasing the number of false negative detections on metrics

begin
  foreach *iteration* **do**
    Add one additional τ' such that $\tau'_n = \tau_n$
    Calculate metrics for current τ, τ' against time series
    Store results
  end
end

Algorithm 10: Experiment to plot the effect of the number of change points on metrics, by appending

begin
  new_point $\leftarrow$ time series segment with 1 changepoint
  foreach *iteration* **do**
    Append *new_point* to data, increasing length
    Add additional τ at tau_{new_point}
    Add additional τ' at $tau'new\_point$
    Calculate metrics for current τ, τ' against time series
    Store results
  end
end

Algorithm 11: Experiment to plot the effect of the number of change points on metrics, by adding 1 to slices of the time series & maintaining constant time series length

```
begin
    foreach iteration do
        Add 1 to next slice of time series
        Add τ and τ′ such that τ′ = τ + 2
        Calculate metrics for current τ, τ′ against time series
        Store results
    end
end
```

3.4 Comparison of Measures Based Upon Functional Requirements

As part of the research being carried out, a set of functional requirements are elicited from the host organisation. Based on these functional requirements the measures are evaluated and compared in such a manner that we can choose the 'best' measure for the use case of the host organisation. The previous research section (§ 3.3) provides the results necessary to

3.5 Application of Algorithms to Real World Data

3.5.1 Data Preparation

The experiment is being carried out using conversation volume data taken from Buzzcapture's Brand Monitor application. Brand Monitor harvests tweets via the Twitter API and makes them searchable through a bespoke interface as well as providing a number of useful metrics for reputation management. In order to gain data for this experiment, Social Media Analysts from Buzzcapture were requested to provide query strings for the Brand Monitor application, showing a situation where they feel a client should have been informed of a distinct change in conversation volume. The analysts were then asked to manually annotate the data with the points at which they believe a change point detection algorithm should detect a change. In this way, bias is avoided when establishing the ground truth for the data - the author was not involved in this annotation process other than providing instructions. The result of this exercise is then a set of time series' showing the twitter conversation volume over time for a given brand. This is accompanied by a separate data set containing indices at which changes *should* be detected - thus serving as the ground truth for this part of the experiment.

Once the query strings provided were executed, the corpus of test data included the following data sets:

- Dirk
- Bol.com
- Connexion
- DAP
- Jumbo
- Kamer van Koophandel
- Rabobank
- Tele2
- UWV

- Ziggo

All data sets were trimmed such that the sample size equals 60 for every set. The sets were exported from the BrandMonitor application in CSV format, with daily conversation volume totals.

3.5.2 Execution

To perform the experiment, an R script is executed that reads the CSV file containing the data points being analysed, and reads them into a data frame object. At this point, the data frame is passed to `changepoint` method calls, running the following analysis methods:

- Mean, using PELT
- Mean, using SegNeigh
- Mean, using BinSeg
- Variance, using PELT
- Variance, using SegNeigh
- Variance, using BinSeg
- Mean & Variance, using PELT
- Mean & Variance, using SegNeigh
- Mean & Variance using BinSeg

Once all of the analysis methods have been executed successfully, the measures are calculated based on a comparison between the computed change points and the manually annotated ground truth. These results are then arranged in a tabular format for analysis and discussion.

Chapter 4

Research

4.1 Introduction

This section of the thesis describes the execution, results and conclusions of each individual study carried out. It is divided into three sections:

- Simulation studies - taking simulated data and simulated algorithm results to analyse the impact of various cases on calculated measures.
- Analysis of measures - taking the results of the simulation studies and testing their ability to match the functional requirements of the host organisation.
- Real-World Data - taking a set of real world twitter data and analysing the ability of change detection algorithms to detect changes in the data compared with established ground truth change points.

4.2 Simulation Studies

The simulation studies carried out are intended to compare each of the measures against the set of criteria described in § 3.3. The simulations are carried out according to the experiment descriptions previously explained, and the execution and results are presented here.

4.2.1 Dependence on Sample Size

Two of the experiments carried out in this thesis take particular note of sample size. Both involve increasing the sample size of the data, appending or prepending a number of points both prior to and following the ground truth change point. In each situation, a simulated algorithm detects a change n points after the ground truth, simulating detection that is relevant, though slightly late, to ensure that there that the baseline metric value at $t = 0$ is not equal to 1.

Figures here

Figures 1 and 2show the results of this simulation study. Figure 1 demonstrates the value of metrics as the 'head' of the data is increased, while Figure 2 demonstrates the value of metrics as the 'tail' of the data is increased.

add figures

In this situation, we can see that both studies show a variation in metric score as the length of the data stream increases. For all of the clustering measures there is a clear increase in metric value as points are added. At $t = 0$, the clustering metrics appear to correctly penalise for a late detection, though this penalty decreases as t increases.

The Adjusted Rand Index shows a far more extreme temporal penalty for the change point detection, which increases sharply at the beginning of the experiment, before maintaining a rate of increase comparable to the other clustering measures.

The binary classification measures of Precision, Recall and F1 maintain a constant value of 0 throughout the experiment, classifying the change point detection as a 'missed' detection and penalising accordingly.

4.2.2 Impact of Data Preceding and Following a Known Change Point

The experiment run for this criteria is similar to those run in § 4.2.1, but differs in one important manner: the size of the time series is maintained at a constant value throughout. As points are added to the 'head' of the data set, points are removed from the 'tail'. In practice, this serves to move both the true and computed change points through the time series.

add figures

Figure x shows the results of the experiment. As with the previous experiment, the binary classification measures maintain a constant value of 0, with variation only being shown by the clustering measures. It is interesting to note that the change in all clustering measures aside from Adjusted Rand Index is minimal, with an almost constant value being held throughout the experiment, aside from when the change points are at the extreme ends of the time series.

The Adjusted Rand Index exhibits behaviour unique to itself, with a value approaching 0 as the change points approach either end of time series. From position 150 to 350, the Adjusted Rand Index holds an almost constant value.

From these results it can be inferred that it is only the Adjusted Rand Index that is effected in any meaningful way by the position of the change point in a fixed length data set.

4.2.3 Temporal Penalty

This is experiment makes use of a fixed length data set with a single, static, known change point. A computed change point is moved through the time series, beginning at the known change point, and ending at the end of the time series. Figure x shows a plot of the results of this experiment:

add figure

At the start of the experiment, when the ground truth and the computed change point are equal, all of the measures return a value of 1 as expected. Once the computed change point is moved through the time series, the binary classification measures immediately return a value of 0, showing that none of the change points in the series (a single change point, in this case) have been detected.

All of the clustering measure values decrease as the computed change point moves away from the ground truth change point, though exhibit some strange behaviour as the computed change point approaches the end of the time series. In all cases, the clustering measures show an increase in score as the computed point approaches the end of the time series, with the temporal penalty for a late detection reducing.

The Adjusted Rand Index once again exhibits behaviour unique to itself, dropping below 0 at approximately $t = 275$. This behaviour is expected from the Adjusted Rand Index, being that the adjusted nature of the metric (allowing for cluster classification occurring by chance) results in a metric capable of returning a value < 0.

get the proper value

4.2.4 Penalisation for False Positives

This experiment utilised a fixed length time series with a single ground truth change point, and a single computed change point that equals the ground truth change point. Successive false positives are added to the time series, moving from left to right across the time series.

Figure x shows the results of the experiment, with the score provided by each metric plotted against the number of false positives present in the data stream.

add figure

The results of this experiment show that almost all of the metrics behave in a similar fashion. The binary classification measure of Recall maintains a value of 1, while all of the other measures show a precipitous drop in score as the initial false positives are added. Precision and F1 scores for binary classification drop sharply, while the measures for clustering show a more gentle decrease in score.

check this

The Rand Index and BCubed F-Score end the experiment at approximately the same value, while the Adjusted Rand Index, F1 Score and Binary Classification Precision metrics also terminate at approximately the same value.

4.2.5 Penalisation for False Negatives

For this criteria evaluation, the experiment utilised a fixed length time series with multiple ground truth change points. As the experiment progressed, computed change points were added, equal to each of the ground truth change points. Figure x shows the results of the experiment, with the metric scores plotted against the number of false negatives present in the data stream.

Add figure

The vast majority of measures see a consistent decrease in returned score as the number of false negatives increases. Indeed, once all of the ground truth change points have been correctly detected, all of the metrics return a score of 1. Only the Adjusted Rand Index returns a score of 0 after 10 false negatives are found in the data stream. As Expected, BCubed Recall remained at a value of 1 for the duration of the experiment.

4.2.6 Number of Change Points in Fixed Length Data Stream

This experiment adds change points to a data stream by 'lifting' slices of the data stream - adding a value to them to create two change points. This is carried out multiple times, each time adding two ground truth change points, and two computed change points 3 time units after the ground truth change points. Figure x shows the results of this experiment, plotting the score returned by the metric against the number of change points in the stream.

add figure

The results of this experiment show a drop in score provided by the Adjusted Rand Index and BCubed F-Score. The BCubed F-Score shows a linear rate of decline as change points are added to the stream, while the Adjusted Rand Index shows an increase in downward velocity towards the end of the experiment. All of the other metrics maintained a constant value of either 1 or 0, as expected.

4.2.7 Number of Change Points in Variable Length Data Stream

This final experiment is similar to the previous one, in that it increases the number of change points in a data stream. However, the difference here is that in this case, data is appended to the data stream, also increasing the length of the stream as each successive change point is added. The results of this experiment are shown in figure x below:

add figure

The results of the experiment show some interesting behaviour in the Rand Index, Adjusted Rand Index and BCubed F-Score metrics. The Rand index and the Adjusted Rand Index increase in value as data points and change points are added to the time series, while the BCubed F-Score decreases in value. The Adjusted Rand Index begins at a slightly lower level than the Rand Index and BCubed F-Score metrics, while the Rand Index and BCubed F-Score metrics diverge almost immediately.

Aside from the large change in score value at the beginning of the experiment, the change in value for the Rand Index, Adjusted Rand Index and BCubed F-Score metrics is small.

4.3 Real-World Data Analysis

Chapter 5

Results

Chapter 6

Analysis & Conclusions

Bibliography

[AF13] Leman Akoglu and Christos Faloutsos. Anomaly, event, and fraud detection in large network datasets. *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, page 773, 2013. URL: http://dl.acm.org/citation.cfm?id=2433396.2433496, doi:10.1145/2433396.2433496.

[AGAV09] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009. doi:10.1007/s10791-008-9066-8.

[Aka74] Hirotugu Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. arXiv:arXiv:1011.1669v3, doi:10.1109/TAC.1974.1100705.

[AMRW11] Foteini Alvanaki, Sebastian Michel, Krithi Ramamritham, and Gerhard Weikum. EnBlogue: emergent topic detection in Web 2.0 streams. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 1271–1274, 2011. URL: http://doi.acm.org/10.1145/1989323.1989473{%}5Cnpapers2://publication/uuid/44E009D1-8574-49C1-A0AC-C2B247FE9043, doi:10.1145/1989323.1989473.

[BN93] M Basseville and Igor V Nikiforov. *Detection of Abrupt Changes: Theory and Application*. 1993. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.6896{&}rep=rep1{&}type=pdf, doi:10.1016/0967-0661(94)90196-1.

[BNZ14] Cody Buntain, Christopher Natoli, and Miroslav Zivkovic. A Brief Comparison of Algorithms for Detecting Change Points in Data. In *Supercomputing*, 2014. URL: https://github.com/cbuntain/ChangePointDetection.

[BPP07] S. Bersimis, S. Psarakis, and J. Panaretos. Multivariate statistical process control charts: An overview. *Quality and Reliability Engineering International*, 23(5):517–543, 2007. doi:10.1002/qre.829.

[DDD05] Frédéric Desobry, Manuel Davy, and Christian Doncarli. An online Kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 53(8):2961–2974, 2005. doi:10.1109/TSP.2005.851098.

[DKL+09] Tamraparni Dasu, Shankar Krishnan, Dongyu Lin, Suresh Venkatasubramanian, and Kevin Yi. Change (detection) you can believe in: Finding distributional shifts in data streams. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5772 LCNS:21–34, 2009. doi:10.1007/978-3-642-03915-7_3.

[Dow08] Allen B. Downey. A novel changepoint detection algorithm. *Applied Microbiology and Biotechnology*, pages 1–11, 2008. URL: http://arxiv.org/abs/0812.1237, arXiv:0812.1237.

[EFK11] I.A. Eckley, P. Fearnhead, and R. Killick. Analysis of Changepoint Models. *Bayesian Time Series Models*, (January):205–224, 2011. doi:10.1017/CBO9780511984679.011.

[FP99] Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1(212):53–62, 1999. URL: http://portal.acm.org/citation.cfm?id=312195, doi:10.1016/j.ecoleng.2010.11.031.

[GMP+09] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4, 2009. URL: http://www.ncbi.nlm.nih.gov/pubmed/19020500, doi:10.1038/nature07634.

[GP04] Pedro Galeano and Daniel Peña. Variance Changes Detection in Multivariate Time Series. 2004.

[HEF14] Kaylea Haynes, Idris A. Eckley, and Paul Fearnhead. Efficient penalty search for multiple changepoint problems. pages 1–23, 2014. URL: http://arxiv.org/abs/1412.3617, arXiv:1412.3617.

[HQ79] E J Hannan and B G Quinn. The Determination of the Order of an Autoregression. *Journal of the Royal Statistical Society*, 41(2):190–195, 1979. URL: http://www.jstor.org/stable/2985032, doi:10.2307/2985032.

[KBdG04] Daniel Kifer, Shai Ben-david, and Johannes Gehrke. Detecting Change in Data Streams. *Proceedings of the 30th VLDB Conference*, pages 180–191, 2004. arXiv:9310008, doi:10.1016/0378-4371(94)90421-9.

[KE14] Rebecca Killick and Idris A. Eckley. changepoint: An R Package for Changepoint Analysis. *Journal of Statistical Software*, 58(3):1–19, 2014. URL: http://www.jstatsoft.org/v58/i03/, doi:10.18637/jss.v058.i03.

[KEJ11] Rebecca Killick, Idris Eckley, and Philip Jonathan. Efficient Detection Of Multiple Changepoints Within An Oceanographic Time Series. pages 4137–4142, 2011. URL: http://www.lancaster.ac.uk/{~}jonathan/2011{_}ISI{_}ChangePoints{_}Talk.pdfhttp://eprints.lancs.ac.uk/56978/.

[KFE11] R Killick, P Fearnhead, and I A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107:500–1590, 2011. URL: http://amstat.tandfonline.com/action/journalInformation?journalCode=uasa20http://dx.doi.org/10.1080/01621459.2012.737745http://arxiv.org/abs/1101.1438{%}0Ahttp://dx.doi.org/10.1080/01621459.2012.737745, arXiv:1101.1438, doi:10.1080/01621459.2012.737745.

[KHH+05] Martin Kulldorff, Richard Heffernan, Jessica Hartman, Renato Assun????o, and Farzad Mostashari. A space-time permutation scan statistic for disease outbreak detection. *PLoS Medicine*, 2(3):0216–0224, 2005. doi:10.1371/journal.pmed.0020059.

[KS09] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 389–400, 2009.

[LS99] Tze Leung Lai and Jerry Zhaolin Shan. Efficient recursive algorithms for detection of abrupt changes in signals and control systems. *IEEE Transactions on Automatic Control*, 44(5):952–966, 1999. doi:10.1109/9.763211.

[McC76] Thomas J. McCabe. A Complexity Measure. *IEEE Trans. Softw. Eng.*, 2(4):308–320, 1976. URL: http://juacompe.mrchoke.com/natty/thesis/FrameworkComparison/Acomplexitymeasure.pdf.

[MJ12] David S. Matteson and Nicholas A. James. A nonparametric approach for multiple change point analysis of multivariate data. *Submitted*, 14853:1–29, 2012. arXiv:1306.4933, doi:10.1080/01621459.2013.849605.

[PRG10] Anita M Pelecanos, Peter a Ryan, and Michelle L Gatton. Outbreak detection algorithms for seasonal disease data: a case study using Ross River virus disease. *BMC medical informatics and decision making*, 10(1):74, 2010. URL: http://www.biomedcentral.com/1472-6947/10/74, doi:10.1186/1472-6947-10-74.

[QAWZ15] Abdulhakim A. Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A PCA-Based Change Detection Framework for Multidimensional Data Streams. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pages 935–944, 2015. URL: http://dl.acm.org/citation.cfm?id=2783258.2783359, doi:10.1145/2783258.2783359.

[R C15] R Core Development Team. *R: a language and environment for statistical computing, 3.2.1*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL: https://www.r-project.org/, arXiv:arXiv:1011.1669v3, doi:10.1017/CBO9781107415324.004.

[Sch78] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978. URL: http://projecteuclid.org/euclid.aos/1176344136, doi:10.1214/aos/1176344136.

[SSBL05] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. ROCR: visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, oct 2005. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti623, doi:10.1093/bioinformatics/bti623.

[SV95] D. Siegmund and E. S. Venkatraman. Using the generalized likelihood ratio statistic for sequential detection of a change-point. *The Annals of Statistics*, 23(1):255–271, 1995. doi:10.1214/aos/1176324466.

[TGS14] Dang-Hoan Tran, Mohamed Medhat Gaber, and Kai-Uwe Sattler. Change Detection in Streaming Data in the Era of Big Data: Models and Issues. *ACM SIGKDD Explorations Newsletter - Special issue on big data*, (1):30–38, 2014. doi:10.1145/2674026.2674031.

[TR05] Alexander G Tartakovsky and Boris L Rozovskii. A Nonparametric Multichart CUSUM Test for Rapid Intrusion Detection. *Proceedings of Joint Statistical Meetings*, pages 7–11, 2005.

[TRBK06] Alexander G. Tartakovsky, Boris L. Rozovskii, Rudolf B. Blažek, and Hongjoong Kim. Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology*, 3(3):252–293, 2006. doi:10.1016/j.stamet.2005.05.003.

[tre] TREC 2016 Evaluation Guidelines. URL: https://trecrts.github.io/TREC2016-RTS-guidelines.html.

[Wic09] Hadley Wickham. *Elegant Graphics for Data Analysis*, volume 35. Springer-Verlag New York, 2009. URL: http://had.co.nz/ggplot2/book, arXiv:arXiv:1011.1669v3, doi:10.1007/978-0-387-98141-3.

[WJ76] Alan S. Willsky and Harold L. Jones. A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems. *IEEE Transactions on Automatic Control*, 21(1):108–112, 1976. doi:10.1109/TAC.1976.1101146.

[XZYL11] Yi Xu, Zhongfei Zhang, Philips Yu, and Bo Long. Pattern change discovery between high dimensional data sets. *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, page 1097, 2011. URL: http://dl.acm.org/citation.cfm?doid=2063576.2063735, doi:10.1145/2063576.2063735.

ToDo Notes