EuroGames16: Evaluating Change Detection in Online Conversation

Cyril Goutte, Yunli Wang, Fangming Liao,[†] Zachary Zanussi,[‡] Samuel Larkin, Yuri Grinberg
National Research Council Canada
1200 Montreal Rd, Ottawa ON, Canada
{Cyril.Goutte, Yunli.Wang, Samuel.Larkin, Yuri.Grinberg}@nrc.ca, will.liao@mail.utoronto.ca, zzanussi@shaw.ca

Abstract

We introduce the challenging task of detecting changes from an online conversation. Our goal is to detect significant changes in, for example, sentiment or topic in a stream of messages that are part of an ongoing conversation. Our approach relies on first applying linguistic preprocessing or collecting simple statistics on the messages in the conversation in order to build a time series. Change point detection algorithms are then applied to identify the location of significant changes in the distribution of the underlying time series. We present a collection of sport events on which we can evaluate the performance of our change detection method. Our experiments, using several change point detection algorithms and several types of time series, show that it is possible to detect salient changes in an on-line conversation with relatively high accuracy.

Keywords: social media, change detection, online conversation

1. Overview

Social media, microblogs and news sources produce massive streams of textual data. Real world events and changes have an impact on these streams. For example, a significant action in a sport event will result in a flurry of positive or negative posts on twitter. Monitoring message streams to detect these changes requires automatic methods relying on a mixture of natural language processing and statistical modeling.

Social media analysis typically operates from two types of data: *raw* stream data and *filtered* stream data. Event detection detects either emerging events or specific events from raw stream data. In particular, the large amount of work done on *Topic Detection and Tracking* (TDT) attempts to detect emerging events. As a recent example, Laban and Hearst (2017) collected 4 million news articles, generated topics from the news, merged them into stories, and visualized the stories along a timeline. For *specific* event detection, Atkinson et al. (2017) created a corpus of security-related events extracted from news data by learning lexico-semantic patterns, and classified events into security-related categories.

Filtered stream data is usually generated by filtering social media using keywords related to public security, natural disaster etc. The TREC 2014 temporal summarization track focused on monitoring events by detecting sub-events, extracting relevant sentences and summarizing them, from a sequence of stream data (Aslam et al., 2014). Zhao et al. (2014) retrieved relevant documents, calculated the text similarity of sentences within the time period, and used k-means to cluster relevant sentences, using the cluster centers and the top sentences as summarization.

In our work, we do not address topic detection and tracking *per se*, but target the detection of significant changes, typically within an existing topic or event. We do not summarize changes like in the TREC 2014 temporal summarization task, but summarization can be used as a post-processing step. Earlier work used change point detection techniques to detect significant changes from sensor signals (Guralnik and Srivastava, 1999; James et al., 2014), but these do not use the textual content of message streams. Some recent studies focus on detecting changes within a storyline. For example, Bruggermann et al. (2016) used the dynamic topic model (Blei and Lafferty, 2006) to identify topics from news and the changes in the word distributions from the topic model were used to represent changes within the storyline. Also, Wang and Goutte (2017) detected changes within events from the temporal profile of hashtags in tweets and evaluated the resulting performance on two twitter datasets.

Our approach relates to some of this prior work, with clear distinctions. First, we target the detection of significant changes *within* an existing event or storyline, rather than detect and track events as in the TDT setup. Second, we focus on detecting the locations of significant changes from the message stream, rather than extract descriptive phrases from the text. Also, instead of detecting changes from external signals obtained from sensors or signals such as stock ticks, we use linguistically motivated signals obtained through text analysis pre-processing.

To establish a benchmark on detecting changes in online conversation, we collected a dataset of 16 sport events, with reference change points for each event. For the purpose of this paper, we will refer to a stream of messages related to a specific topic (e.g. a game, or a current event) as an *online conversation*. Our purpose is to detect, from that online conversation, the location of significant changes within the event. We do this by analyzing the content of the messages in order to produce one or several *time series* describing, for example, the sentiment or the topic of the conversation. We then use *change point detection* algorithms on these time series in order to detect locations where the underlying stochastic process changes. This typically is a change in mean, but also e.g. in variance or other distributional property. We apply this approach and benchmark it on the acquired collection of sport games conversations. In partic-

[†] from University of Toronto, ON, Canada
[‡] from Carleton University, ON, Canada

Game	Hashtag	# Eng.	# Total
Croatia-Spain	CROESP	52,953	115,328
England-Iceland	ENGISL	191,384	209,851
France-Albania	FRAALB	61,748	433,954
France-Ireland	FRAIRL	172,872	665,476
France-Iceland	FRAISL	158,457	720,771
Germany-France	GERFRA	273,074	496,498
Germany-Italy	GERITA	426,381	709,453
Germany-Poland	GERPOL	82,132	232,200
Poland-Portugal	POLPOR	128,079	663,612
Portugal-Austria	PORAUT	72,644	170,526
Portugal-France	FRAPOR	229,000	1,000,000
Portugal-Wales	PORWAL	287,417	461,343
Russia-Wales	RUSWAL	110,165	141,994
Switzerland-France	SUIFRA	36,507	468,043
Wales-Belgium	WALBEL	288,312	378,852
Wales-N. Ireland	WALNIR	95,679	114,723
Total	-	2.69M	7.04M

Table 1: Basic statistics on collection content.

ular, we test several change point detection algorithm and uncover their strengths and weaknesses.

In Section 2. we describe the collection that we acquired in order to benchmark change point detection algorithms. In Sections 3. and 4., we describe the linguistic preprocessing and change point detection algorithms, respectively. We then present experimental results, testing and validating this approach, in Section 5.

2. Collection

Our dataset contains messages related to 16 games from the 2016 UEFA European Championship[1], a continental football (soccer) competition held from 10 June to 10 July 2016. Dedicated hashtags comprised of three letter codes for each country were used to harvest messages from the twitter API, e.g. #FRAPOR or #PORFRA for the France-Portugal final game. Messages were also filtered by language, including English for all games. In the following, we only consider English messages, for which the relevant sentiment analysis tool is available. Statistics on the number of messages are given in Table 1. The number of messages per game is between 100k and one million, for a total of 7M messages. Most of the messages are posted over a period of about two hours, starting shortly before the game and ending shortly after.

In order to obtain a gold standard of reference events, we obtained game reports from sport websites, from which we semi-automatically extracted the main game events such as start/end of each period, goals, on-target attempts, substitutions, etc. The collection we release contains the multivariate sentiment time series, produced as described below, together with the reference labels for each of the 16 games. The processed dataset is available from: https://github.com/cyrilgoutte/EuroGames16

[1] https://en.wikipedia.org/wiki/UEFA_Euro_2016

3. Linguistic Preprocessing

The first stage in our approach is to turn the stream of messages into one or several time series. This can be done by using existing linguistic preprocessing methods such as sentiment analysis or topic models. As a running example below as well as in our experiments we use sentiment analysis for that purpose. Application to other linguistic preprocessing methods is similar.

We formalize a stream of N messages as a set $\mathcal{C} = \{(t_i, \mathbf{c}_i), i = 1 \dots N\}$, where t_i is the time associated with message i, and $\mathbf{c}_i$ is its textual content. We assume that the linguistic preprocessing produces, for each content $\mathbf{c}$, a set of scores describing the target linguistic properties. For example, we use the NRC sentiment analyzer (Kiritchenko et al., 2014) to produce, for each textual content $\mathbf{c}$, a set of three scores estimating the positive, negative and neutral polarity of the message: $\mathbf{c} \rightarrow \left(s^+, s^-, s^0\right)$. Processing the entire collection $\mathcal{C}$ results in a large table containing time and scores for each message, $\mathbf{S} = \left[t_i, s_i^+, s_i^-, s_i^0\right]_{i=1\dots N}$.

The posting times for messages are not uniformly distributed. In order to produce three times series for the positive, negative and neutral scores, we first *bin* the messages in time intervals of equal sizes, then average the scores within each bin. For simplicity and without loss of generalization, let us assume that posting times range from 0 to T, i.e. $0 \leq t_i \leq T, \forall i = 1 \dots N$. Dividing the range from 0 to T into $\lceil T/\Delta \rceil$ bins, each of width Δ, we build time series by averaging scores within each bin:[2]

$$s^+(t) = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} s_i^+, \quad \mathcal{B}_t = \{i, (t-1)\Delta \leq t_i < t\Delta\}, \tag{1}$$

and similarly for $s^-(t)$ and $s^0(t)$ for the negative and neutral scores. We therefore obtain three time series of sentiment scores, or alternatively a multivariate, three dimensional time series, on which we run the change point detection algorithms. We later compare it with detection from raw message counts. This corresponds to a trivial preprocessing where each message is associated with a single score of 1 (performing binning and averaging as before), resulting in the time series $n(t) = |\mathcal{B}_t|$. Keyword profiles (Sec. 5.4.) are obtained similarly by recording the number of keywords in each bin.

4. Change Point Detection

In time series analysis, a change point is a location where the underlying stochastic process changes. Although it may seem superficially related, this is a different problem than anomaly detection, where the purpose is to identify observations that do not conform to an expected pattern or distribution in the data. In change point detection, we assume that data before the change point conforms to one distribution, while data after the change point comes from a second, different distribution. In our case, we are interested in identifying where the change has occurred, as soon as possible after it occurs.

Many algorithms have been proposed to detect change points. Most work on univariate time series, and use the

[2] When no message falls within a bin, it yields a missing value.

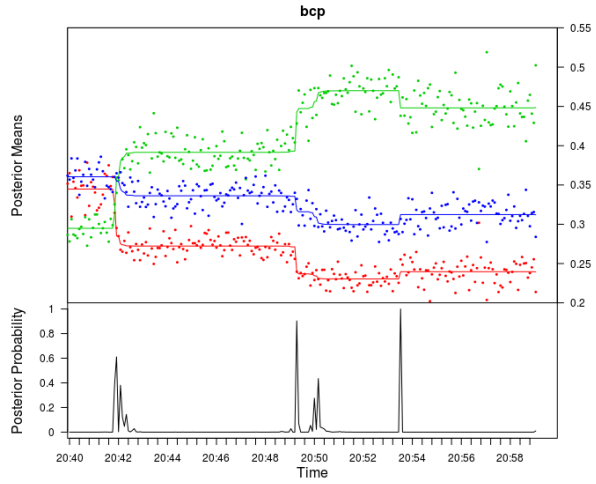

Figure 1: Top: Sentiment time series for the Wales v.s Belgium game in green/blue/red (positive/neutral/negative); Bottom: Posterior probability of change from `bcp`. At 20:42, Wales scores their third goal; at 20:50, Belgium is awarded three consecutive corner kicks, then injury time starts.

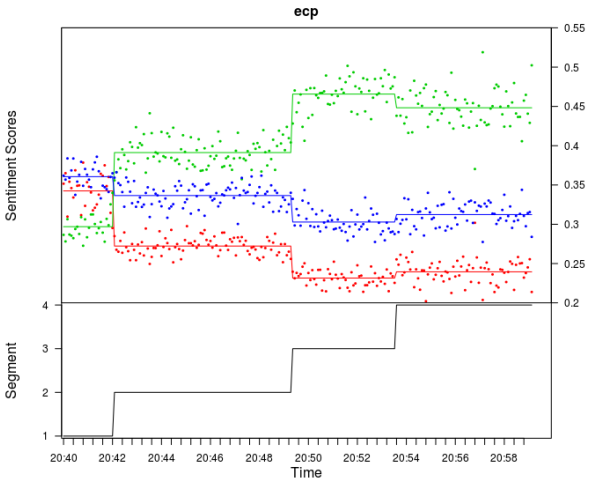

Figure 2: Top: Sentiment time series for the Wales v.s Belgium game in green/blue/red (positive/neutral/negative); Bottom: Segments output by `ecp`. At 20:42, Wales scores their third goal; at 20:50, Belgium is awarded three consecutive corner kicks, then injury time starts.

entire time series to detect change points. We will focus on techniques that can be used with multivariate time series:

bcp

The Bayesian change point detection of Barry and Hartigan (1993) assumes that each block between two change points arises from a (multivariate) normal distribution. It outputs the posterior probability that a change occurred at each point in the time series. Figure 1 illustrates this on a short extract from the sentiment time series for one of the games in the collection. We use the implementation from the `R` package `bcp` (Erdman and Emerson, 2007). The `bcp` algorithm runs fast: it is linear in the length of the time series, and handles multivariate time series. The biggest limitations are that it is designed to detect changes in the mean of independent Gaussian observations, and that it works off-line, once the entire time series is available.

ecp

The nonparametric, hierarchical divisive algorithm of James and Matteson (2015) uses recursive bisections, identifying change points using a non-parametric divergence measure from Székely and Rizzo (2005). As the divergence measure is non-parametric, this makes `ecp` suitable to detect changes with minimal assumptions on the underlying distributions. The divisive approach by recursive bisections returns a number of consecutive *segments* between change points, without knowing the number of change points *a priori*. In addition, the implementation from the `R` package `ecp` handles multivariate time series, as illustrated in Figure 2 on the same data as Fig. 1. One remaining limitation is that it works only in off-line mode, once the entire time series are available.

ocpd

The Bayesian online change point detection algorithm of Adams and MacKay (2007) is designed to update the detection of change points sequentially as new data points are acquired, rather than wait until the entire times series are available. It relies on two components: a probabilistic model $P(r_t|\mathbf{s}(1\dots t))$ of the length of a *run* during which the underlying distribution is stable, given observations until time t; and an underlying predictive model (UPM) $P(\mathbf{s}(t+1)|\mathbf{s}(1\dots t), r_t)$ governing the stochastic generation of new data in each run. Our basic implementation, available in the `R` package `onlineCPD`, uses a multivariate Gaussian UPM. Figure 3 shows the results obtained from `ocpd` on the same sentiment time series as before.

ocpd+

We extend the basic `ocpd` algorithm beyond the simple Gaussian assumption by using a more flexible UPM. We model the linear trends within each run, using a multivariate linear regression with additive Gaussian noise. This allows modeling drifts in the time series without forcing multiple change points. Our implementation in `R` will shortly be included in the `onlineCPD` package.

4.1. Online vs. offline

Despite different modeling inspirations and assumptions, the key differentiating feature of `ocpd`/`ocpd+` is that they work in *online* mode, updating the model and the detection at each step. When analyzing short events such as sports event in our collection, the difference may seem contrived. However, many real-life monitoring situations span days or months. This is the case when tracking changes in the days after a terror attack (Wang and Goutte, 2017) or in public health when following events related to epidemics over months or years. In those cases, it is clearly impractical to wait until all data acquisition is finished before running the

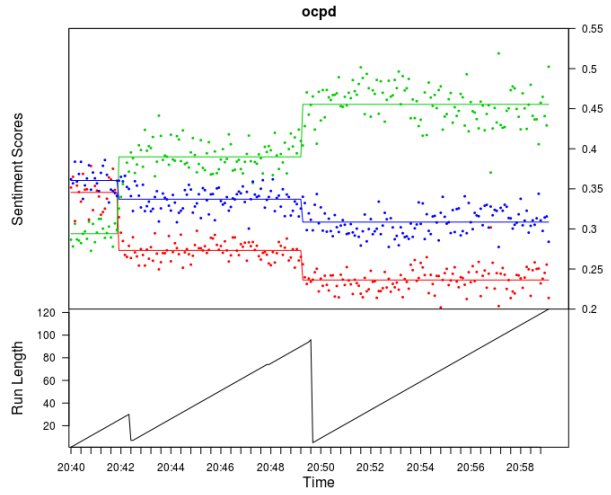

Figure 3: Top: Sentiment time series for the Wales v.s Belgium game in green/blue/red (positive/neutral/negative); Bottom: Maximum run length probability from `ocpd`. At 20:42, Wales scores their third goal; at 20:50, Belgium is awarded three consecutive corner kicks, then injury time starts.

change detection algorithm. Section 5.5. discusses this in more details.

4.2. Finding the number of change points

Both `bcp` and `ecp` can be used for univariate and multivariate time series without a priori knowledge of the number of change points. The posterior probability of change output by `bcp` may be thresholded to tune the output of the method, and in `ecp`, similarly, a threshold may be applied to the p-value estimating the significance of a divergence in distributions. However, in our experience, both methods work much better when a target number of changes is provided. We investigate the impact of this in our experimental section (Section 5.6.). On the other hand, `ocpd` and `ocpd+` provide a posterior distribution on the run length, from which it is possible to automatically detect the number of change points. Light post-processing may be used to avoid multiple detections around the same change.

5. Results

Each of the games listed above (Section 2.) was pre-processed (as described in Section 3.) in order to produce multivariate time series estimating the positive, negative and neutral sentiment[3] during each game. We use a bin size of $\Delta = 15s$ for these experiments, which yields good performance. We first look at the results from a particular game (Sec. 5.1.) before presenting our systematic evaluation in Sections 5.2. to 5.7..

5.1. Example

We use the Wales vs. Belgium game that took place on July 1st, 2016 to illustrate what the data and the output of the change point detection algorithm look like. Figure 4 shows the timeline from a few minutes before the game starts to minutes after it stopped. The shaded area in the background represents the volume of tweets, and shows high variability. Most spikes are associated to significant events in the game (light green, labeled lines), but this is not always the case (e.g. first and third yellow cards). Changes in the positive and negative scores (blue and red curves, smoothed) during the game also tend to match peaks in the tweet volumes, but are quite variable everywhere.

The detections produced by `ocpd+` (blue ticks) and `ecp` (bottom red ticks) show very different behaviours. `ocpd+` yields high precision: the detections are usually close to reference events; but it also misses several. On the other hand, `ecp` detects too many changes. As a consequence it yields high recall, but lower precision. Note that there is a qualitative difference between changes. Those related to predictable events such as half time or end of game tend to be detected early, while unpredictable game plays such as goals or yellow cards tend to be detected with a slight delay. This makes sense, as people may start tweeting about predictable events in anticipation, before they actually occur, while unpredictable events, by definition, can not be anticipated. More analysis would be required to check, e.g. whether false positives correspond to notable game plays that may not be recorded in our gold standard.

5.2. Off-line CPD

A systematic evaluation was carried out using precision, recall and F-score (Goutte and Gaussier, 2005) to evaluate the performance of change point detection algorithms on the 16 games. A detected change point was considered as a true positive if it falls into the time window 180 seconds on either sides of a reference change point. This allows to take into account that tweets need to be written and posted, as well as slight inaccuracies in the real timing of the reference game plays.

In these experiments, we run all algorithms *off-line*, on the entire dataset, which is the standard mode of operation for `bcp` and `ecp`. The `ocpd` and `ocpd+` algorithm are run one data point at a time, as designed, simulating an on-line operation over the entire dataset.

Table 2 shows the performance on all games for all change point detection algorithms applied to the sentiment time series. We see that `bcp` performs poorly overall. This may be due to the strong underlying Gaussian assumption, and the fact that it is very sensitive to non normally distributed noise, a problem we confirmed using simulated data (not reported here). The `ecp` algorithm performs well, obtaining the best F-score for 7 out of 16 games, often by a small margin (e.g. ENGISL, WALBEL, WALNIR). The `ocpd` algorithm behaves sometimes quite poorly, maybe due to the Gaussian assumption again, but it is typically close to (and sometimes better than) `ecp`. Finally, the more flexible assumptions underlying the `ocpd+` algorithm allows it to get the best results for 7 games and overall (Tables 3, 4).

5.3. Detection from Message Counts

We have run the change point detection algorithms on the three sentiment signals (positive/negative/neutral), taking advantage of the fact that they handle multivariate data. However, it is simple to run them on the univariate tweet

[3]Time series are not independent: sentiment scores sum to one.

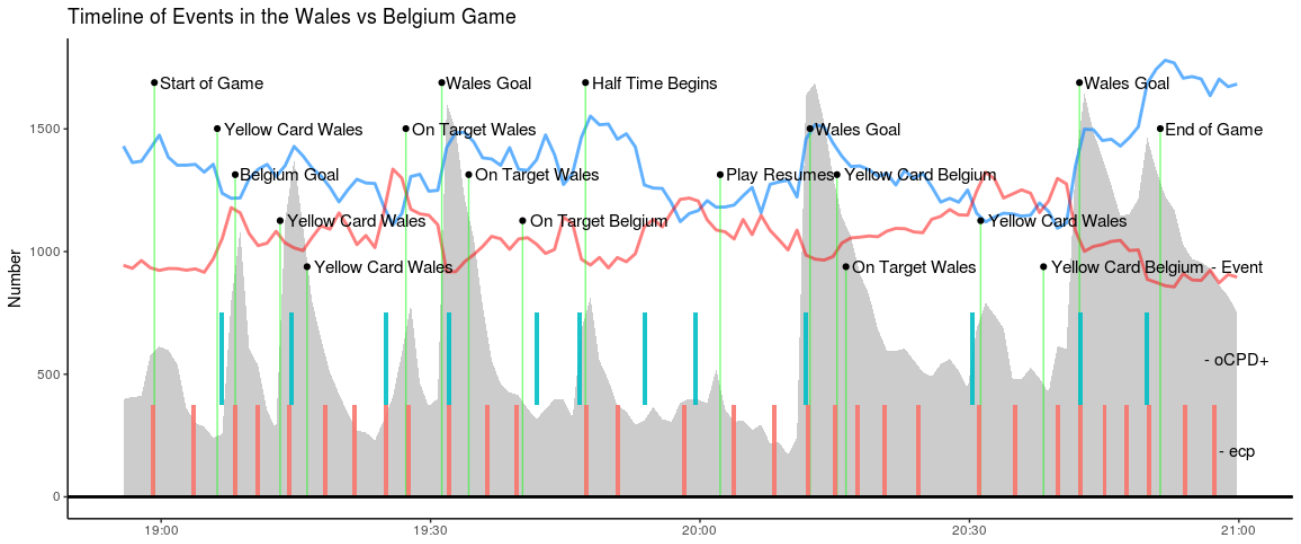

Figure 4: Timeline for the Wales v.s Belgium game. Tweet volume is in grey shade, and +/- sentiment time series in blue/red; reference events in light green, detected events in blue (ocpd+) and red (ecp).

Table 2: The F-scores of four algorithms on sentiment of Euro games in 15 seconds time interval

Game	bcp	ecp	ocpd	ocpd+
CROESP	0.267	**0.643**	0.444	0.500
ENGISL	0.085	**0.429**	0.333	0.414
FRAALB	0.390	0.480	0.111	**0.583**
FRAIRL	0.290	0.240	0.250	**0.400**
FRAISL	0.225	0.370	**0.538**	0.400
GERFRA	0.186	0.462	0.231	**0.467**
GERITA	0.217	**0.571**	0.300	0.516
GERPOL	0.211	0.462	0.400	**0.552**
POLPOR	0.212	**0.500**	0.174	0.370
PORAUT	0.069	0.267	0.160	**0.400**
FRAPOR	0.143	0.182	0.154	**0.303**
PORWAL	0.367	**0.563**	0.435	0.389
RUSWAL	0.266	0.417	**0.571**	0.444
SUIFRA	0.333	0.435	0.167	**0.500**
WALBEL	0.419	**0.703**	0.370	0.684
WALNIR	0.086	**0.190**	0.111	0.174
Average	.2354	.4315	.3047	**.4428**

frequency signal $n(t)$. On datasets such as sports games where tweet volume is correlated with significant changes, this works surprisingly well, cf. first row of Table 3. Pairwise comparisons (Table 4) also shows that ecp and ocpd+ are overwhelmingly more effective than the other two algorithms. As algorithms handle multivariate signal, we can also *add* counts as a fourth time series, in addition to the sentiment scores. In that case (last row, Table 3), performance jumps above what we obtained on counts and sentiments separately, and reaches 50.6% F-score using ocpd+.

Table 3: Performance on raw counts (top), sentiment (middle, with/without providing the target number of change points) and combined (bottom).

Input	bcp	ecp	ocpd	ocpd+
Count	.3434	.4645	.4250	**.4725**
Sentiment	.2354	.4315	.3047	**.4428**
+ # references	.3918	**.4860**	.3047	.4380
Count+Sentiment	.4251	.4645	.4010	**.5062**

Table 4: Pairwise comparisons of four algorithms on counts and sentiment scores, with/without providing the target number of change points. Cells report how many games the row method wins/ties/loses vs. the column method.

Counts	ecp	ocpd	ocpd+
bcp	2/0/14	5/0/11	2/0/14
ecp	-	11/1/4	**7/1/8**
ocpd	-	-	4/1/11
Sentiment	**ecp**	**ocpd**	**ocpd+**
bcp	1/0/15	5/1/11	0/0/16
ecp	-	12/0/4	**7/1/8**
ocpd	-	-	3/0/13
+ references	**ecp**	**ocpd**	**ocpd+**
bcp	0/0/16	4/0/16	0/0/16
ecp	-	14/0/2	**10/0/6**
ocpd	-	-	3/0/13

5.4. Detection from Keyword Usage

Another simple comparison is with the detection of change points from temporal profiles of keywords such as "goals", "begin", "end", etc. and hope that changes in the usage of these words can be captured by change point detection algorithm. There are two serious issues with this approach, however. One is obviously that it requires specific domain knowledge to pick the appropriate keywords: obvi-

Table 5: Performance of change point detection algorithms on temporal profiles of specific keyword usage.

Time Interval	bcp	ecp	ocpd	ocpd+
5 mins	**0.3748**	0.2413	0.1692	0.1773
1 min	0.1495	**0.2046**	0.1107	0.112

Table 6: Performance of change point detection algorithms in on-line mode, with different time intervals for binning.

Time Interval	bcp	ecp	ocpd	ocpd+
60 seconds	**0.4799**	0.3274	0.2012	0.3826
30 seconds	0.4693	0.4260	0.2996	**0.4810**
15 seconds	0.4391	0.4313	0.3625	**0.4624**

ously "goal" of "red card" would not be very useful keywords when tracking health- or security-related document streams. Another is that even domain-related keywords will be sparse in the collection. This may require to use larger bin sizes in order to capture any statistics at all on keyword usage patterns, leading to short temporal profiles and little information for the CPD algorithms to work with. We will see below (Sec. 5.5.) that this tends to yield inferior performance for most algorithms. Despite these objections, we benchmarked this baseline on the same 16 games. Table 5 presents the results using 1min and 5min time intervals. The pattern in the result is similar to what we saw in Table 6: bcp performs better on the larger time interval, ecp performs better on the shorter time interval, while ocpd and ocpd+ do not perform well on these large intervals and short time series. Overall, the performance is clearly much lower than what we obtained with our approach.

5.5. On-line CPD

In practice, it is often more useful to run the change point detection on-line. As discussed above, it would be inconvenient to wait until all data is acquired before the analysis is run. Since bcp and ecp are off-line change point detection methods, we simulate on-line operation by running sequentially over a sliding window on the time series: We do a first run on the first 50 minutes, then offset the window by 25 minutes and run the algorithms again on the data from minute 25 to minute 75, and so on until the end of the time series. In principle, this provides change point detections with at most a 25 minute delay. Although the ocpd and ocpd+ algorithms are on-line by design, we run them over the same sliding window in order to have a fair comparison and evaluate the impact on performance.

Further, we investigate the impact of the time interval used for binning messages in order to produce the time series. In addition to the 15 seconds interval used earlier, we experiment with 30 seconds and 60 seconds. Note that this has a direct impact on the length of the time series: with a 15s interval, times series have 240 points per hour, versus 60 with a 60s interval. Experiments were run on the Count+Sentiment time series, which produced the best performance in Table 3.

Table 6 suggests that the performance of bcp is impacted favourably by running in a sliding window. Its performance increases slightly with larger time window. The three other

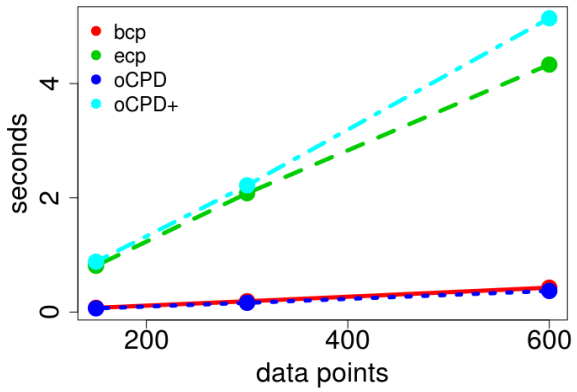

Figure 5: Average computational time of change point detection algorithms in on-line mode over 16 games, for 60s, 30s and 15s intervals (150, 300 and 600 data points).

algorithm suffer when the large time window is used, possibly because this reduces the length of the time series and limits the amount of statistics the underlying model can work with. On the other hand, bcp does better on larger time intervals, possibly because averaging scores over more messages makes the resulting data points more Gaussian (according to the central limit theorem). The performance of ecp improves with decreasing time interval. This may be due to more robust permutation tests when more data is available. ocpd+ performs best on the 30s and 15s time interval, and the best F-score of 48.1% is obtained at 30s. It also does better than ocpd (see also Tables 2, 4), which is consistent with the fact that the underlying predictive model is more flexible in ocpd+. All algorithm apart from bcp do somewhat worse with the sliding window than in off-line mode. This suggest that it is beneficial to run in true on-line mode, one data point at a time, as ocpd and ocpd+ are designed to do.

5.6. Number of Changepoints

Although changepoint detection algorithms try to guess the correct number of changes, they usually work better if the target number of changes is given. We evaluate this effect by running experiments in which we provide the correct number of reference changes. Results reported in the 3rd line of Table 3 should be compared to the average of Table 2 (also second line in Table 3). bcp and ecp clearly benefit from knowing how many changes to detect, and ecp now yields the best results, and wins over ocpd+ in the majority of cases (Table 4). It is understood, however, that this is an unrealistic scenario: in practice, we do not know the correct number of changes. In addition, ocpd and ocpd+provide a key functionality that the other two do not have: they process the data *online*, one point at a time, and can detect changes as soon as (or soon after) they occur instead of waiting for the entire collection to be acquired.

5.7. Computational Time

The computational time of online change point detection algorithms is another important factor for real-time CPD. We

use the same sliding windows as before (50 minute windows in steps of 25 minutes) to assess the computational time of the four CPD algorithms. The average computation time needed to analyze all 16 games is shown in Figure 5 for increasing time series lengths corresponding to decreasing time intervals. This shows that `ocpd` and `bcp` are much faster than `ocpd+` and `ecp`. This reflects the computational cost of running permutation tests repeatedly in `ecp`, although computational effort can be adjusted by lowering the number of permutations. For `ocpd+`, this reflects that the added flexibility in the linear model fitting the trend comes with the increase in computational cost. Note that despite theoretical upper bounds suggesting quadratic runtime for some of these algorithm, actual runtime seems to increase linearly.

6. Conclusion

We introduced a framework for detecting significant changes from on-line streams of messages. It relies on linguistic preprocessing producing semantically or linguistically relevant times series, which we run through a multivariate change point detection algorithm. The EuroGame16 collection was used to benchmark this approach, showing that we can detect around half the significant game plays in sports events, from the content of the twitter messages alone. The collection is made available to allow researchers to try other approaches to improve on our results. In addition, we contribute two variants of change point detection, `ocpd` and `ocpd+`. They show competitive performance with the state of the art `ecp` package. In addition, they can be used in a fully *on-line* mode, which allows the detection of changes soon after they occur instead of waiting until the entire time series can be processed. This is a key feature when monitoring social media streams in real time.

7. Bibliographical References

Adams, R. P. and MacKay, D. J. (2007). Bayesian online changepoint detection. arXiv:0710.3742.

Aslam, J. A., Ekstrand-Abueg, M., Pavlu, V., Diaz, F., McCreadie, R., and Sakai, T. (2014). TREC 2014 temporal summarization track overview. In Voorhees and Ellis (Voorhees and Ellis, 2014).

Atkinson, M., Piskorski, J., Tanev, H., and Zavarella, V. (2017). On the creation of a security-related event corpus. In *Proceedings of the Events and Stories in the News Workshop*, pages 59–65. Association for Computational Linguistics.

Barry, D. and Hartigan, J. (1993). A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 35(3):309–319.

Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA. ACM.

Bruggermann, D., Hermey, Y., Orth, C., Schneider, D., Selzer, S., and Spanakis, G. (2016). Storyline detection and tracking using dynamic latent dirichlet allocation. In *Proceedings of 2nd Workshop on Computing News Storylines*, pages 9–19. Association for Computational Linguistics.

Erdman, C. and Emerson, J. (2007). bcp: An R package for performing a bayesian analysis of change point problems. *Journal of Statistical Software*, 23(1):1–13.

Goutte, C. and Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In D.E. Losada et al., editors, *Advances in Information Retrieval - 27th European Conference on IR Research*, pages 345–359.

Guralnik, V. and Srivastava, J. (1999). Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 33–42, New York, NY, USA. ACM.

James, N. A. and Matteson, D. (2015). ecp: An R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(1):1–25.

James, N. A., Kejariwal, A., and Matteson, D. S. (2014). Leveraging cloud data to mitigate user experience from "breaking bad". *eprint arXiv:1411.7955*.

Kiritchenko, S., Zhu, X., and Mohammad, S. M. (2014). Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Laban, P. and Hearst, M. (2017). newslens: building and visualizing long-ranging news stories. In *Proceedings of the Events and Stories in the News Workshop*, pages 1–9. Association for Computational Linguistics.

Székely, G. J. and Rizzo, M. L. (2005). Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of Classification*, 22(2):151–183.

Ellen M. Voorhees et al., editors. (2014). *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*, volume Special Publication 500-308. National Institute of Standards and Technology (NIST).

Wang, Y. and Goutte, C. (2017). Detecting changes in twitter streams using temporal clusters of hashtags. In *Proceedings of the Events and Stories in the News Workshop*, pages 10–14. Association for Computational Linguistics.

Zhao, Y., Yao, F., Sun, H., and Yang, Z. (2014). BJUT at TREC 2014 temporal summarization track. In Voorhees and Ellis (Voorhees and Ellis, 2014).