Event Mining and Timeliness Analysis from Heterogeneous News Streams[☆]

Ida Mele[a], Seyed Ali Bahrainian[b], Fabio Crestani[b]

[a] *Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124 Pisa, Italy*
[b] *Faculty of Informatics, Università della Svizzera italiana, Via G. Buffi 13, 6904 Lugano, Switzerland*

Abstract

In this paper, we describe our research on detecting, tracking, and predicting events from multiple news streams. We also analyze the temporal publishing patterns of newswires and their timeliness in reporting the events.

We focus on news documents published by several newswires (e.g., BBC, CBC, CNN) on different platforms (e.g., Twitter, RSS news portals, and news websites). We first present an approach based on dynamic topic modeling and Hidden Markov Model for event detection and tracking. Then, we predict the events that would persist in the next time slices, which can be important for forecasting facts that would be popular in the future.

We leverage event detection for clustering news documents according to the events they describe. This allows us to determine which newswires published news about the same events and to analyze their temporal ordering in reporting events. Finally, we propose two scoring functions for ranking the newswires based on their timeliness in publishing news.

We tested our methodologies on different collections of news articles and tweets. Moreover, we built our own collection of heterogeneous news documents with event-document labels which were manually assessed using crowdsourcing.

[☆] This article is an extended version of the article *Linking News across Multiple Streams for Timeliness Analysis* published in *Proceedings of the 26th ACM International Conference on Information and Knowledge Management (CIKM'17)*, November 6–10, 2017, Singapore.

Email addresses: ida.mele@isti.cnr.it (Ida Mele), bahres@usi.ch (Seyed Ali Bahrainian), fabio.crestani@usi.ch (Fabio Crestani)

Keywords: news stream mining; event detection and tracking; temporal analysis

1. Introduction

News mining has recently attracted a lot of attention for event detection and tracking [21, 27], discovering novel stories [30], linking news articles to social-media posts [38], analyzing newswires' publishing patterns [15, 25], and detecting untrusted sources of news [48]. In this paper, we present approaches for mining events from multiple news streams, tracking their evolution over time, and finding those events that would persist in the future. We also cluster news documents based on the events they describe for cross-linking the news streams and analyzing their temporal relationships.

We focused on heterogeneous news streams. For us a news stream represents a sequence of news documents published by a news channel (e.g., BBC, CNN) on a platform (e.g., Twitter, RSS news portals, and news websites). Tweets, RSS feeds, and news articles published by the same news channel are considered as three separate streams and we refer to each of them with its channel name and publishing platform (e.g., $\texttt{cnn}_{\texttt{twt}}$, $\texttt{cnn}_{\texttt{rss}}$, and $\texttt{cnn}_{\texttt{news}}$). Hence, we denote with *news document* a tweet, an RSS feed, or a web article and with *event* a set of keywords that describe a specific fact appearing in a news document plus the time when it has happened.

Our research is motivated by the fact that millions of news are daily published online by different newswires using heterogeneous platforms. This huge amount of information is overwhelming for the users who would like to filter interesting news and, for example, be updated only on the evolution of some specific stories. Also, for news professionals and journalists, it is always more difficult to keep pace with the streams of information in order to early discover popular events which could be examined in depth or reported on the national newscast. One of the main challenges in event mining from news streams is to group news documents published by different channels that are about a same

event, such as *earthquake in Japan* or *Obama visit to Castro*. Traditional clustering approaches [15, 24] may help to divide the news documents into broad categories, but they fail in capturing low-level granularity details (e.g., crisp events which happened in a narrow time window). In particular, grouping news into broad topic clusters may result in large sets of documents describing related but distinct events.

Our work is related to the research area of Topic Detection and Tracking (TDT) which includes several tasks ranging from event segmentation of news streams to novel-event detection and event tracking. Most of the research has focused on extracting events from news [11, 16, 19], clustering news based on the events they describe [12], and on detecting events from tweets [21, 30, 31, 32, 35, 36]. These approaches have considered only one single stream of text, while we take into account multiple streams of news. This is important for two reasons: (1) to see which events were reported by most of the newswires (e.g., some events are world-wide newsworthy, others are reported only by local channels) and (2) to analyze the newswires' temporal ordering in reporting the events (e.g., which newswire reported an event before the others and on which platform). The only work that has already considered multiple news streams was the one by Gwadera and Crestani in [15] although with some limitations since they considered only one type of document (RSS feeds) and used a simple unsupervised clustering approach. We try to overcome these limitations by taking into account different types of documents (news articles, RSS news feeds, and tweets) and presenting a technique based on *discrete dynamic topic modeling* for discovering and tracking topics (events). We then leverage the event keywords for dividing the news documents into corresponding event clusters and providing a semantic interpretation of them. Finally, we analyze the temporal publishing patterns among different news streams and introduce two novel scoring functions for ranking them based on their timeliness.

Our research contributions can be summarised in (1) a discrete and dynamic topic modeling approach for detecting, tracking, and predicting events, (2) a topic-based clustering of news documents, (3) an analysis of the temporal

publishing patterns among news streams, and (4) a collection of heterogeneous news streams where news documents are labeled at the level of fine-granularity events using the crowdsourcing platform named CrowdFlower.

This journal is an extended version of [25], and the novel contributions are as follows:

- we provide a detailed literature review on event detection and tracking and a complete background on topic modeling and on dynamic topic modeling;
- we present a system for predicting persisting events whose performance was tested on a collection of heterogeneous news documents;
- we describe in detail the process for collecting and labeling the heterogeneous news documents. In particular, how we monitored different newswires on different platforms, our approach for automatic creating the event labels, and how we performed the evaluation in CrowdFlower. This collection of heterogenous news documents was important for testing our methodologies on different news streams;
- we present further experiments for testing the ability of our model to recognize novel topics quickly as well as to detect and track events over time. We used different datasets: (1) a collection made of news and blog articles, (2) a collection of tweets on innovation and technology, and (3) our own collection of news documents gathered from different news channels and platforms.

The paper is structured as follows: we review related work in Section 2 and provide the background on topic models in Section 3. Section 4 presents our methodology for event detection and tracking and, Section 5, for predicting persisting events. In Section 6 we show how to cluster news documents based on the events and to cross-link news streams. Section 7 presents techniques for mining temporal publishing patterns and for ranking the news channels according to their timeliness in reporting events. In Section 8, we describe our collection of heterogeneous news streams and our effort for labeling news

documents based on fine-granularity events. We report the experimental results in Section 9. Finally, Section 10 concludes the paper.

2. Related Work

2.1. Event Detection in News Documents

Event detection consists in extracting events from news documents and organizing these documents by the events they report. The approaches for detecting events can be divided into two broad categories: document- or feature-pivot approaches. Early works on event detection belong to the first category since the documents are clustered based on their content, then the event-based features are extracted from the clusters [1, 8, 45, 46, 47]. The second category is related to all those techniques where the news streams are analyzed with the purpose of discovering hidden features and then such features are clustered in order to identify the events [11, 12, 16, 19, 44]. We focus our literature review more on the feature-pivot approaches since our methodology is more similar to them as it first extracts event features and then clusters the news documents according to them. A pioneer work was conducted by Kleinberg et al. [19]. They proposed an approach for finding frequent patterns based on an infinite-state automation to model the changes in the word frequency, state transitions, and events. First, a set of related bursty features with similar frequency distributions are retrieved. Then, documents related to the bursty features are extracted and clustered into a hierarchy of events. Fung et al. [12] proposed to monitor unigrams and statistically model their frequency with a binomial distribution. The events are detected by maximizing the co-occurrences of bursty-frequency words among the documents, while the event timestamp is determined based on the periods of the features. In a further work, the authors applied such methodology to retrieve the documents that are about the same event and then organize them in a time-based hierarchy [11].

Other works treated features as signals that can be monitored over time [16, 44]. In [16], the authors proposed to extract bursty features applying the Dis-

crete Fourier Transformation (DFT). A signal for each feature is built using *document frequency-inverse document frequency* (*df x idf*) scheme along with the time domain. The signal is then transformed from the time domain into the frequency domain by DFT, and a spike in the frequency domain indicates a corresponding high-frequency signal source. Similar to [12], the bursty features are grouped into events by considering both feature co-occurrences and their distributions in the time domain. Weng and Lee [44] developed an approach based on *wavelet analysis* which provides measurements regarding how signals change over time. More in details, a signal is created for each individual word, then wavelet transformation is used to measure the energy of words. Only words with high energies are retained as event features, and the similarity between pairs of events is measured by cross-correlation. Using cross-correlation as a similarity measure may lead to results consisting of several distinct events which happened at the same period of time just by coincidence. Moreover, these approaches suffer from the scalability problem since the extraction of bursty features based on statistics may result in a prohibitive number of features, especially if unigrams are used. Also, the application of DFT or wavelet transformation is computationally expensive.

2.2. Event Detection in Twitter

Detecting events from tweets is even more challenging since traditional approaches developed for formal text, such as news articles, cannot be applied to tweets given their short and noisy nature. One of the main challenges in Twitter is discriminating the newsworthy events from mundane events (e.g., celebrities' status updates) [5, 31]. We did not do such a discrimination since we only monitored tweets published by news channels, hence we assume that all their tweets are newsworthy; rather we analyze the tweets to detect real-world events and divide tweets into clusters based on the events they describe. A similar problem was addressed by Petrovic et al. [30] who applied locality sensitive hashing (LSH) to measure the similarity among tweets and group together tweets representing the same events.

Other works are based on detecting crime or disasters from tweets using predefined keywords as queries (e.g., "earthquake" or "shaking") [22, 36]. Similarly, Popescu et al. [32] proposed an approach for finding tweets that contain a specific named entity and then applied machine learning to predict whether or not the tweets constitute an event regarding the entity. These approaches rely on a predefined list of keywords or named entities, while Ritter et al. [35] presented an open-domain approach that does not need annotated data. They applied latent-variable models to discover the event types which are then used to classify and aggregate the events. In a similar way, Li et al. presented Twevent [21] which clusters tweets plus provides a semantic-meaningful description of the clusters. Their system extracts semantic phrases from tweets (segments) and detects bursty segments which represent the events. These segments are then clustered based on their frequency distribution and content similarity. Finally, Wikipedia is used to check the realistic events and to derive the most newsworthy segments that can be used to describe the events.

2.3. Analysis of News Streams

So far, most of the research works have focused on one single stream of news, and only a few of them have taken into account multiple streams. For example, Wang et al. [42] proposed a topic mining approach for discovering common topics from multiple textual streams. In a following work [43], they also presented a technique for reducing the asynchronism among the streams by exploring the correlation across them. Their approach consists of two alternate steps: (i) extracting the common topics, (ii) adjusting the timestamps, leveraging the temporal distribution of the discovered topics. This mutual-reinforcement process reduces the asynchronism among streams. Differently from their work, we do not aim at reducing the asynchronicity, we rather want to exploit it to analyze if there exist temporal publishing patterns among the news streams.

Del Corso et al. [9] tackled the problem of ranking the news articles and news sources based on their *decay of freshness* and *priority ranking*, respectively. Gwadera and Crestani [15] presented an approach for mining and ranking the

streams of news using cross-stream sequential patterns and content similarity. They monitored RSS news feeds coming from different channels and applied a simple streaming clustering technique to divide the stories based on the events. In this paper, we present a more effective approach for discovering and tracking events based on *dynamic topic modeling*. This allows to chain events over the timeline and, more importantly, to have semantic-meaningful descriptions of event-based clusters of news documents. Furthermore, we present a timeliness analysis of the news streams based on two ranking scores.

2.4. Topic Models

Understanding natural-language text is easy for people, who can get the meaning of words based on their context, but not for machines. Probabilistic topic models are trained on large text corpora (e.g., Wikipedia pages or news articles) and are able to infer the latent topics discussed in these corpora. They can capture the meaning of words by leveraging their co-occurrences. The first example of probabilistic topic model is the Latent Semantic Analysis (LSA). It models the word similarity based on the idea that the words appearing in similar contexts (i.e., the textual window around words) tend to have similar meanings. The model builds a term-document co-occurrence matrix and applies the Singular Value Decomposition (SVD) on it to get a low-dimensional representation of the documents. A probabilistic version of LSA (pLSA) was proposed by Hofmann [17], where observations are represented by the co-occurrences of the words in the documents. The probability of each observation is modeled as a mixture of conditionally independent multinomial distributions. Latent Dirichlet Allocation (LDA) [7] is similar to pLSA but the topic distribution is assumed to have a Dirichlet prior, which results in a more reasonable mixture of topics in a document, and each topic is characterized by a distribution over words. Anyway, LDA has some limitations when working on temporal-ordered datasets where the time can be used for better detecting the latent topics and for tracking their evolution over time. Temporal analysis can be applied after or before the topic extraction. In post-hoc temporal analysis, the topic model is fit with

no reference of time, then documents are sorted by time, sliced into time slices to examine the topic distributions in each of them [14]. This approach is very simple and does not exploit the temporal information for improving the topical discovery, while it would be better to pre-divide data into time slices to fit a separate topic model in each of them [3, 6, 40]. Given a sequential dataset sliced up based on the timestamps, *Dynamic Topic Model* (DTM) [6] computes the topics in each time slice and chains them over all the time slices to represent the topical evolution. The *continuous DTM* (cDTM) relaxes the requirement that time has to be discretized [40] to track topical changes even in short intervals of time. Although these models can capture the topical evolution, they rely on the assumption that each topic is present over all the time slices. This does not hold for highly-dynamic datasets where topics are constantly changing and may have skips in the timeline.

3. Background

3.1. Latent Dirichlet Allocation

Given a collection of M textual documents where each document has N words, Latent Dirichlet Allocation (LDA) [7] discovers the K latent topics discussed in these documents. It is a probabilistic model which represents documents as mixtures of topics and the topics are defined as probability distributions over words. The approach is based on a generative statistical model where documents are generated sequentially (one word at a time). At each step, a topic is selected from the per-document topic distribution, θ, which is parametrized by the hyper-parameter α. Then, a word is picked from a multinomial distribution of words, β, which is conditioned by the chosen topic. Using an analogy based on *buckets*, it is like having multiple buckets representing topics, and the number of instances of each bucket depends on the topic distribution. Each bucket contains topic-discriminative words, where the number of instances of a word depends on the probability of the word to appear in documents relevant to the corresponding topic. Words in the documents are generated by picking

a topic bucket and then extracting a word from it. Such process is repeated for each word position.

The graphical model of LDA is reported in Figure 1(a), and its generative process is as follows:

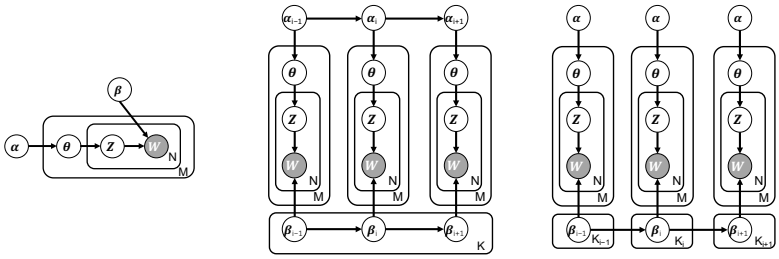
1. For each document d:
   1.1. Draw $\theta_d \sim \text{Dir}(\alpha)$
   1.2. For each word position n:
      1.2.1. Choose a topic $z_{d,n} \sim \text{Multinomial}(\theta_d)$
      1.2.2. Choose a word $w_{d,n} \sim \text{Multinomial}(\beta, z_{d,n})$

Inverting the generative process, it is possible to infer from the observed words the latent topics discussed in the documents. Statistical inference techniques, such as variational inference or Gibbs sampling, are employed to learn the underlying topic distribution of each document as well as the word distribution of each topic based on the word co-occurrences [13].

3.2. Dynamic Topic Models

In sequential datasets where the documents are, for example, sorted by their timestamps, LDA mixes co-occurrence patterns from documents belonging to different temporal periods. As observed in [41], running LDA on a collection of historical documents, it mixes the Mexican-American War (1846-1848) with World War I (1914-1918) since it is unaware of the 70-year separation between them. The Dynamic Topic Model (DTM) [6] takes into account the ordering of documents which is essential to better understand the underlying topics and to track the topical evolution over time. As we can see from Figure 1(b), DTM splits the sequential dataset into time slices, then it applies LDA to model the topics in each of them. The hyper-parameter α_i is estimated from the previous one, α_{i-1}, to ensure that the topics of time slice t_i evolve from the topics of t_{i-1}. While LDA draws the topic distributions, θ, from a Dirichlet distribution, DTM uses a logistic normal with mean α to express uncertainty over proportions:

$$\alpha_t | \alpha_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \delta^2 I) \tag{1}$$



(a) LDA (b) DTM (c) dDTM-News

Figure 1: Graphical models of LDA (a), DTM (b), and dDTM-News (c). The *plates* (boxes) represent replicates, e.g., the outer plate represents the M documents in the collection, while the inner plate represents the N word positions in the document (the assumption is that documents have same length). K is the number of latent topics and it is chosen by the user. In DTM (b), the dataset is split into temporal slices and LDA is applied to each of them. Its hyper-parameters α_i and β_i are estimated from the ones of the previous time slice (α_{i-1} and β_{i-1}) and the number of topics, K, is the same for all the time slices. Differently from DTM, dDTM-News (c) uses a fixed α and the number of topics K_i changes with the time slice t_i. These topics are then chained based on their evolution with an approach based on Hidden Markov Model.

The parameters are chained using a linear Kalman filter [18] over consecutive time slices. DTM also chains the parameters of the per-topic word distributions in a state space model which evolves with a Gaussian noise:

$$\beta_{t,k}|\beta_{t-1,k} \sim \mathcal{N}(\beta_{t-1,k}, \sigma^2 I) \tag{2}$$

where $\mathcal{N}$ is a logistic normal distribution and σ can be tuned to allow topic variation over two subsequent time slices (i.e., low values correspond to small variations).

DTM is based on the assumption that each topic is present over all the time slices which allows to model topics that evolve slowly over time (e.g., topics discussed in scientific articles) [6]. Such assumption is not suitable for highly-dynamic datasets (e.g., streams of tweets or news) where the topics constantly change over time because they are triggered by external events, hence they

appear, then disappear, and may appear again after some time. Moreover, when a topic suddenly emerges and disappears, DTM would not be able to identify it since it captures only small changes in the topical evolution and would detect only topics that persist over time. To overcome this limitation, we propose a discretized version of DTM which does not rely on a linear evolution of the topic proportions, allowing skips in the topical evolution.

4. Discrete Dynamic Topic Model for News Streams

We now present our model called *discrete Dynamic Topic Model for News Streams* (dDTM-News). This model is able to cope with news collections where the latent topics change at a high pace and they may suddenly emerge, disappear, or have skips in the timeline. A preliminary version of this model was presented in [3] to timely capture variations in the topic evolution by using independent per-document topic proportions. We propose a further modification of the model to dynamically estimate the number of topics for each time slice.

4.1. Modeling Topics and their Evolution

As we can see from Figure 1(c), dDTM-News first divides the dataset into time slices and then applies LDA to each of them. Differently from DTM, it keeps α fixed to have an independent evolution of the topics. These topics are chained over time by leveraging a Hidden Markov Model (HMM) [33] which is a Markov chain where the states $(s_1, s_2, ..., s_n)$ are hidden and we only have observations of them $(o_1, o_2, ..., o_n)$. In particular, we map the latent topics discovered by LDA to the observations of an HMM, and we want to estimate the probability of transition from one state to another (i.e., the evolution of topics). For solving this problem we need two steps:

1. *Estimating the parameters of the HMM.* We apply the *Baum-Welch algorithm* [4] for estimating the parameters of the HMM, i.e., the probability of the first state, $P(s_1)$, the transition probabilities, $P(s_i|s_{i-1})$, and the emission probabilities, $P(s_i|o_i)$. Baum-Welch algorithm is an *Expectation*

Maximization (EM) algorithm that, given a set of observations, applies the *Forward-Backward algorithm* to find the maximum likelihood estimate of the HMM's parameters.

2. *Maximizing the probability of a state sequence.* Given a sequence of observations, $O_{1:n}$, we maximize the probability of a sequence of states, $S_{1:n}$, applying the *Viterbi algorithm* [39]. It uses recursion for maximizing the following probability:

$$S^* = \underset{s}{argmax}\{p(S_{1:n}|O_{1:n})\} \tag{3}$$

The number of topic chains that we would like to discover is represented by the optimal number of states in the HMM. We estimate this parameter using the *Bayesian Information Criterion* (BIC) [37]. It increases a penalty for each added parameter (in our case a state added to the HMM), therefore it finds the best trade-off between the number of states and the model fitting the data. For our problem, we initialize the HMM with a number of topic chains equivalent to the number of topics n that we expect to find in a time slice. Then, the initialization of HMM is repeated with $n + 1$ topic chains until it reaches an upper bound of $n + 20$ (which is set empirically). For each run, the BIC value is measured and the model with the lowest value is selected.

Generative Process. Having showed how topics are chained with the HMM, we now present the generative process of our model:

1. From HMM
   1.1. Draw $\beta_{t,k}|\beta_{t-1,k} \sim \mathcal{N}(\beta_{t-1,k}, \sigma^2 I)$
2. For each document d_t at time slice t:
   2.1. Draw $\theta_{d_t} \sim \text{Dir}(\alpha)$
   2.2. For each word position n in the document d_t:
      2.2.1. Draw $z_{d_t,n} \sim \text{Multinomial}(\theta_{d_t})$
      2.2.2. Draw $w_{d_t,n} \sim \text{Multinomial}(\pi(\beta_{t,z_{d_t,n}}))$

where π is a function which maps the multinomial natural parameters to the mean parameters.

4.2. From Fixed Number of Topics to Dynamic Number of Topics

So far, we made the assumption that the number of topics K does not change over time. This represents a limitation when working on news streams since the latent topics represent events, and each time slice is likely to have more or less topics compared to the other ones. As shown in Figure 1(c), the number of topics should change with the time slice. For this reason, we estimated the number of topics by creating different LDA models with fixed α and variable K_i. Similarly to [14], the number of topics in the time slice t_i is determined by selecting the model that satisfies the following equation:

$$argmin_{K_i}\{log\ P(\ \mathbf{w}\ |\ K_i\ )\}, \tag{4}$$

where $\mathbf{w}$ represents the words in the vocabulary and $P(\mathbf{w}|K_i)$ is the probability of the words given the number of topics.

5. Modeling the Evolution of Persisting Topics

In this section, we present the problem of predicting topics that persist in the next time slice. Starting from the words appearing in the documents of the first n time slices, we can construct a word vector consisting of probability scores that are estimated based on the *recency* and the *establishment* effects. The former increases the weights of words representing the most recent topics, while the latter favors words belonging to stable topics.

More formally, the *recency reference vector* is computed according to the following equation:

$$P_{ref,recency} = \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{w_i \in t} \frac{P(w_i) * 2^n}{(n * t)} \tag{5}$$

where n is the sequence number of the time slice, t is one of the topics discussed in the time slice, and w_i is a word from that topic. 2^n is the rate with which higher weights are assigned to the recent topics.

On the other hand, the establishment effect assigns higher weights to the words related to topics which have persisted over time. The corresponding

word vector, called *establishment reference vector*, is computed according to the following equation:

$$P_{ref,establishment} = \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{w_i \in t} \frac{P(w_i) * 2^{-n}}{(n * t)} \tag{6}$$

where n, t, and w_i are defined as in the recency effect, and 2^{-n} is the rate with which higher weights are assigned to established topics.

We combine the recency and establishment effects using a *Kalman combination of Recency and Establishment* (K2RE) [2]. The linear interpolation for a time slice t is defined as:

$$K2RE_t = w_{E,t} * Score_{establishment} + w_{R,t} * Score_{recency} \tag{7}$$

where $Score_{establishment}$ and $Score_{recency}$ are computed by the establishment and the recency effects, respectively. The weights $w_{E,t}$ and $w_{R,t}$ are the establishment and recency weights and are computed dynamically by a Kalman filter [18], such that: $w_{E,t} + w_{R,t} = 1$.

We use dDTM-News for linking the topics discussed in the documents. The resulting topic keywords are then input of the Kalman filter which estimates the weights used for balancing the recency and establishment effects.

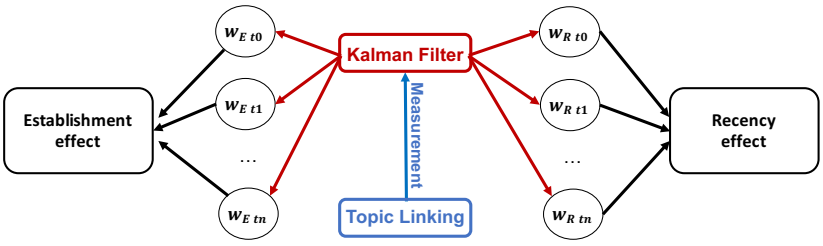


Figure 2: K2RE system: topic linking is performed by dDTM-News and the resulting topic keywords are used by the Kalman filter for estimating the recency and establishment weights.

6. News Clustering and Cross-Linking News Streams

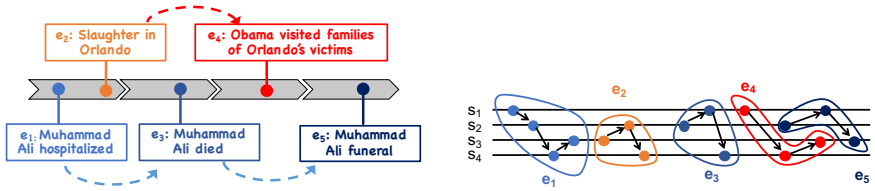
In Section 4 we have seen how to detect and track topics (events) from highly-dynamic streams of news using discrete dynamic topic modeling. We now describe our approach for event-based clustering of news documents and for cross-linking the news streams reporting the same events.

Clustering News Documents. Document clustering consists in the automatic organization of documents based on some criteria. For our problem, we assume that a news document would cover one event at the time, hence we opt for hard clustering and assign each document to one event cluster. We use topic modeling as a dimensionality reduction technique to map the documents in a topic space and cluster them based on their closeness to the detected topics. More formally, we apply the approach proposed in [24] which treats each topic as a cluster and each document is represented by a distribution vector of topics, $\boldsymbol{\theta}$. The document d is assigned to the cluster k_i, if:

$$k_i = argmax_j(\theta_j) \tag{8}$$

Cross-Linking News Streams. In Figure 3(a), we show an example of five events belonging to two different stories (the event evolution is shown by dashed arrows). The news documents (nodes) reporting on the same events are divided into event clusters as shown in Figure 3(b). As we can see, the streams of news that have reported the same event are linked to each other. Formally, let n_{ij} represent the news document about the event e_i published in the news streams s_j, the link (n_{ij}, n_{ik}) is drawn if both streams, s_j and s_k, published about e_i and s_j did it before s_k. This is helpful for analyzing the temporal dependencies among the newswires, e.g., which channel has published before the others and on which platform.

Note that, for simplicity, we only show news about a same event which are reported by different news streams. This is because we are interested in linking the news *across* streams, but it may happen that news on the same event are reported by the same stream multiple times. When this happened, we considered the first news reporting the event published by the stream.



(a) chaining events over the timeline (b) cross-linking news streams

Figure 3: Figure (a) shows two examples of event chains. Events are detected over the timeline and connected based on their evolution (see the dashed arrows). Figure (b) shows the four news streams which published news. The news documents (nodes) are clustered based on the events they describe and the news streams are cross-linked based on the publishing timestamp of the news documents (black arrows).

7. Timeliness Analysis and Ranking News Streams

Cross-linking news streams makes easier the analysis of the timeliness among different streams. In this section, we describe our effort in detecting temporal publishing patterns among the news streams. Moreover, we present two scoring functions for ranking news streams based on their timeliness in reporting events.

7.1. Mining Frequent Sequential Patterns

For mining frequent temporal patterns among the news streams we use the PrefixSpan algorithm [29]. It computes the most frequent subsequences present in an itemset sequence. Let $\mathcal{A} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$ denote a non-empty set of items and $s = \langle \mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n \rangle$ the itemset sequence, s is defined as a *subsequence* of another sequence s', if there are n integers, $i_1 \leq i_2 \leq ... \leq i_n$, such that $\mathcal{I}_1 \subseteq \mathcal{I'}_{i_1}$, $\mathcal{I}_2 \subseteq \mathcal{I'}_{i_2}$, ..., $\mathcal{I}_n \subseteq \mathcal{I'}_{i_n}$. For example, the sequence of itemsets $\langle (1), (2\ 3), (4) \rangle$ is contained in the sequence $\langle (5), (1\ 6), (7), (2\ 3\ 9), (4) \rangle$, because $(1) \subseteq (1\ 6)$, $(2\ 3) \subseteq (2\ 3\ 9)$, and $(4) \subseteq (4)$, while the sequence $\langle (1), (6) \rangle$ cannot be the subsequence of $\langle (1\ 6) \rangle$ and vice versa.

Given a collection of sequences $\mathcal{S} = [s_1, s_2, ..., s_{|\mathcal{S}|}]$, the *support* of an itemset sequence s_j, denoted by $sup_{\mathcal{S}}(s_j)$, is the number of itemset sequences $s_i \in \mathcal{S}$ with $i \neq j$ that contain s_j as a subsequence. Let $minSup$ be the support thresh-

old, the itemset sequence is called *frequent sequential pattern* if its $sup_{\mathcal{S}}(s_j) \geq minSup$. The PrefixSpan algorithm receives as input $\mathcal{S}$ and $minSup$ and computes the frequent sequential patterns, namely, all the sequences in $\mathcal{S}$ whose frequency is greater than $minSup$.

In our analysis, we want to find frequent temporal patterns in the time-ordered sequences of news streams. We sort the news documents by time and convert this time-ordered set into a sequence of publishing streams. This conversion does not cause any loss of information since the pair stream/timestamp (s, ts) is unique, i.e., a stream cannot publish more than one news document at the same time. Since the timestamps have resolution at the level of seconds, we use a quantization-step parameter $Q_t = 15$ minutes to introduce a time tolerance, namely, when two or more channels have published news documents about the same event within the window of 15 minutes, we treat them as they have reported the event at the same time.

7.2. Ranking News Streams using Timeliness Scores

PrefixSpan discovers frequent temporal patterns but does not provide any way to rank the news streams based on their timeliness. Hence, we define two scoring functions that capture the delay of a stream in publishing news about real-world events and allow to rank the news streams based on their timeliness.

We denote with $E_s \subseteq E$ the set of events published by the stream s and with $S_e \subseteq S$ the set of streams which reported the event $e \in E$. For each event, the channels are organized in a list sorted by the timestamp of the first news, $l_e : \langle (s_1, ts_1), (s_2, ts_2), \ldots, (s_n, ts_n) \rangle$, where ts_i is the timestamp of the first news about e published by s_i.

Given a stream s, let the score $timeliness(s) \in [0, 1]$ represent how promptly s publishes. It is computed as the average over all the events reported by s:

$$timeliness(s) = 1 - \frac{\sum_{e \in E_s} delay(s, e)}{|E_s|} \tag{9}$$

where $delay(s, e)$ is the latency of s in reporting e. The lower is the delay, the higher is the value of $timeliness(s)$. Following [26], we adopt two definitions for

$delay(s, e)$. The first one considers only the *relative position* of s in the list l_e:

$$delay(s, e) = RP(s, e) = \frac{|pred(s, e)|}{|l_e|} \tag{10}$$

where $pred(s, e)$ is the number of streams that preceded s in reporting the event e, and $|l_e|$ is the number of streams which have published about the event. The second one considers the *time distance* between s and the very first stream which reported the event:

$$delay(s, e) = TD(s, e) = \frac{ts(s, e) - ts_{first}(e)}{ts_{last}(e) - ts_{first}(e)} \tag{11}$$

where $ts(s, e)$ is the publishing timestamp for the event e by s, while ts_{first} and ts_{last} are the publishing timestamps for the first and last news about e, respectively. Notice that popular events that span a long period of time (e.g., Brexit) have a high value of $(ts_{last}(e) - ts_{first}(e))$, while less popular events have lower values since the attention for them decreases fast.

8. A Test Collection of News Streams

We collected news documents published by different news streams for 4 months. The documents were labeled at the level of fine-granularity events using crowdsourcing. To the best of our knowledge, this is the first dataset gathering news documents from multiple and heterogeneous sources and enriched with event labels.

We created our own collection as other publicly available ones (e.g., *20 Newsgroups*, *Temporalia*, and *SignalMedia*) are mostly made of homogeneous documents (e.g., only news and blog articles). Enriching the existing collections with RSS feeds and tweets published in the same period of time of the articles was impossible, because they were not available anymore. In fact, the Twitter REST APIs can retrieve the tweets containing some specific words, but it goes back up to 7 days. Regarding the RSS feeds, they cannot be downloaded because they are updated at a high speed, so the old feeds disappear soon to be replaced by the new ones. Moreover, news collections lack of fine-granularity labels on the

relevance of a news document to a specific event since most of the labels are about general topics (e.g., politics or sport). Lastly, we need data which spans a few months allowing to track the evolution of events (e.g., Muhammed Ali's death followed by his funeral or the slaughter in an Orlando's nightclub followed by the killing of the shooter).

8.1. Data Gathering

We built a dataset of heterogeneous news documents by monitoring 9 different news channels (the list is reported in Table 1) over 3 different platforms (Twitter, RSS news portals, news websites) for a total of 27 news streams. Our collection spans 4 months (from March 1 to June 30, 2016) and consists of around $147K$ news documents. Other statistics are reported in Table 1.

Each document has a *title* (optional), *content*, *link* (optional), *publishing timestamp*, and *source channel*. The fields *title* and *link* are optional since they may be missing in the tweets. Also, since the time when a news has been published can reflect a different time zone, we converted all the publishing timestamps to UTC. We downloaded the RSS feeds and news articles every 15 minutes to get up-to-date news. Regarding the tweets, we monitored the accounts of different newswires using the `getUserTimeline` method provided by the Twitter REST APIs [1]. It collects the latest tweets (author, text, timestamp, etc.) posted or retweeted by a user in JSON format. We run the download every 24 hours, removing the repeated tweets when needed.

The distribution of news documents per week is shown in Figure 4. Note that the last week has less news documents since it consisted of only 4 days (from June 27 to 30, 2016).

8.2. Extraction of Event Labels

The approaches proposed in the literature for news clustering are often evaluated by manually selecting keywords representing some popular events which

[1] `https://dev.twitter.com/rest/public`

Table 1: Channels and number of news documents (from March 1 to June 30, 2016).

Channel name	News Articles	RSS Feeds	Tweets	Total
American Broadcasting Company (abc)	2,561	5,661	20,039	28,261
Al Jazeera (alj)	2,823	3,763	5,414	12,000
British Broadcast (bbc)	4,715	6,318	1,867	12,900
Canadian Broadcast (cbc)	1,747	2,680	5,874	10,301
Cable News Network (cnn)	4,197	11,969	10,433	26,599
NBC News (nbc)	3,261	5,789	10,050	19,100
Reuters (reu)	2,271	4,527	12,072	18,870
United Press International (upi)	1,520	1,547	3,479	6,546
Xinhua China Agency (xin)	1,061	1,126	10,906	13,093
TOTAL	24,156	43,380	80,134	147,670

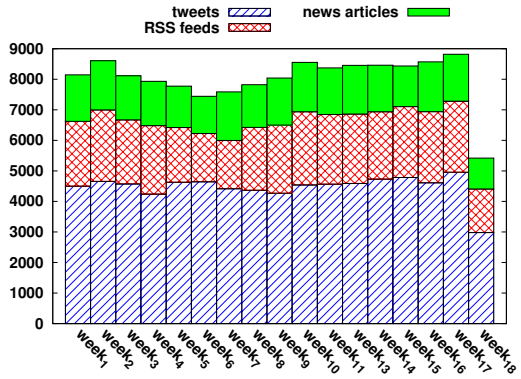


Figure 4: Distribution per week of the news documents (tweets, RSS feeds, and news articles).

happened in the period of interest. These events are, for example, found in Wikipedia while the keywords are manually picked and used as queries to retrieve the documents [28]. In this way, the evaluation relies on the manual selection of events and keywords, so it may be biased by very popular events. For our labeling, we automatically detected the event labels, then we retrieved the corresponding news documents and asked human assessors to evaluate their relevance to the events.

To extract the events occurring in the 4 months of our data, we used an automatic approach for mining labels from the document content and detecting the temporal span of the event [27]. Our approach first splits the news collec-

Table 2: Some labeled events from our collection. Each event is characterized by the keywords and the date of when it happened.

Labeled Events	Date
north-korea, ballistic, missile, nuclear, test, U.S.	March 6, 2016
castro, cuba, havana, obama, visit	March 21, 2016
attack, brussels-airport, isis, maelbeek-metro	March 22, 2016
japan, kumamoto, earthquake, damage, victims	April 14, 2016
earthquake, ecuador, strikes	April 16, 2016
obama, visit, asia, vietnam, embargo	May 20, 2016
boxing, died, louisville, muhammad-ali	June 3, 2016
nightclub, orlando, shooting, victims	June 13, 2016

tion into time buckets of fixed size (i.e., 24 hours), then it computes the most frequent co-occurrences of named entities and event-descriptive keywords which are detected using an NLP tool developed by Ritter et al. [34]. To determine the time span of an event, we computed the keyword overlapping in consecutive time buckets to see whether the detected event is new or can be merged with a previous one. This overlapping depends on the length of the list of keywords, if the percentage of common keywords is high (i.e., 70%), then the two events are merged. Iterating this operation results in events with flexible temporal windows. The reason why we decided to check only consecutive time buckets (δ = 24 hours) is that if the two buckets are distant we can assume that the events are distinct although they have high-overlapping keywords.

Some of the extracted event keywords with the corresponding date of the event are shown in Table 2. After having detected the events, we manually checked on Wikipedia whether they really occurred and whether the estimated time window was correct.

8.3. CrowdFlower Evaluation

After having generated the event labels, we conducted a manual assessment on the relevance of a document to an event using the CrowdFlower[2] crowdsourc-

[2] `http://www.crowdflower.com/`

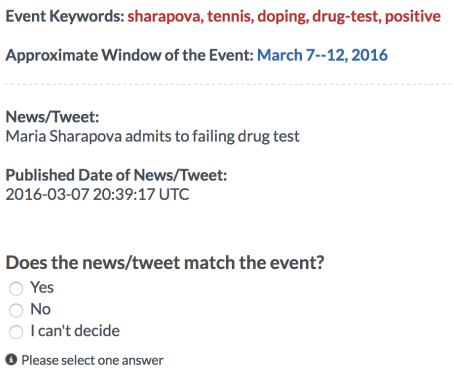


Figure 5: An example of evaluation task in CrowdFlower.

ing platform. Given the large number of news documents, we could not collect relevance labels for all of them, so we randomly selected 60 events and retrieved the potentially-relevant news documents using their cosine similarity with the event keywords. Such selection was done taking into account the popularity of the event: we sorted the events based on the number of potentially-relevant documents and pick the events homogeneously in the distribution (this allowed to have popular events as well as "long-tail" events).

We collected human judgements for the pairs {event, news document} where the former is made of event keywords and the approximate time of the event (i.e., the day of the peak in the keyword frequency), while the latter is the document information. The evaluators were asked to determine whether the news document was about the event or not. They could also select "I can't decide" and move to the next pair if unsure about the answer. Figure 5 shows an example of the CrowdFlower interface used for the assessment.

Before starting, the evaluators were instructed on the scope of the evaluation with also examples, e.g., the tweet "A 6.4 earthquake hits southern Japan with reports of collapsed buildings and people injured" is relevant to the event *Japan earthquake* but not to *5th anniversary from Japan Tsunami*.

To guarantee high quality results, we created 200 *gold questions* (i.e., questions with known answers). These questions were randomly shown during the

evaluation to detect low-quality answers due to sloppy or malicious workers and were used as a *quiz* for the training phase. In particular, the evaluators could pass the training phase if they successfully completed at least 3 out of 5 gold questions. Once the evaluators started the evaluation, they had to maintain a minimum accuracy of 70% to be considered "trusted evaluators" and allowed to continue the task. Only labels from trusted evaluators were included in our dataset. We collected relevance judgements for $4,3K$ labeled pairs {event, news document}. For each of them we collected judgements from at least 3 different trusted evaluators, and the news document was considered truly relevant to the event if at least 2 out of 3 evaluators agreed. To measure the inter-annotator agreement we computed the Fleiss' Kappa [10]. It gives a measure on how consistent are the assessors' ratings and is computed as $\kappa = \frac{\bar{P}-\bar{P}_e}{1-\bar{P}_e}$. In a nutshell, $1-\bar{P}_e$ gives the degree of agreement that is attainable above chance, and $\bar{P}-\bar{P}_e$ gives the degree of agreement actually achieved above chance. If $\kappa = 1$, the agreement is complete, while $\kappa \leq 0$ means no agreement. We registered a value of 0.45 which, according to the table for interpreting κ values provided in [20], corresponds to a moderate agreement. Note that Fleiss' Kappa does not take into account the trustworthiness of the assessors. A criterion that considers the trust scores of assessors is the CrowdFlower's confidence which is defined as the level of agreement between multiple contributors weighted by the contributors' trust scores. For our task, we had an average confidence of 0.91.

9. Experimental Results

In this section, we describe the evaluations we conducted to assess the effectiveness of our techniques using different datasets. Our first experiment focused on assessing the ability of the dynamic topic models in detecting emerging topics. We also inspected the topic keywords to see how qualitatively they describe the topics and capture their temporal variations. As second experiment, we tested our techniques for the tasks of detecting and tracking the events over time. Then, we show the results on predicting persisting topics (events) in a

collection of news documents. The last part of this section focuses on clustering news documents based on the detected events and on the timeliness analysis.

9.1. Datasets

We evaluated the effectiveness of dDTM-News in discovering latent topics on three temporal-ordered datasets. The first dataset is made of news articles from `Signal Media`[3], and it contains over one million of news articles collected from September 1 to September 30, 2015. This collection is made of articles from different sources including major news channels (e.g., Reuters) and local news and blog websites. For our experiments, all the news articles were preprocessed by removing stopwords, URLs, tokens not starting with alphabet letters, punctuation marks, and less-frequent words (i.e., words occurring less than 5).

Since `SignalMedia` spans a period of time relatively short (1 month), we created another dataset with tweets posted by the `NFCWorld` Twitter channel[4] over three years (from April 15, 2013 to April 12, 2016). These tweets report on emerging technologies and were collected using the Twitter API[5] for a total of $3K$ tweets. For our experiments, tweets were preprocessed by removing stopwords, URLs, hashtags, tokens not starting with alphabet letters, punctuation marks, and less-frequent words (i.e., words with frequency less than 5).

The last dataset is our news collection made of different types of documents (news articles, RSS news feeds, tweets) gathered from 27 news streams as described in Section 8. Differently from other news datasets, our collection has news documents labeled at the level of fine-granularity events, allowing to assess the performance of the event detection and tracking tasks.

9.2. Modeling Emerging Topics

For assessing the performance of approaches based on dynamic topic modeling, a well-known technique consists in computing the log-likelihood of the

[3] http://research.signalmedia.co/newsir16/signal-dataset.html
[4] https://twitter.com/nfcw
[5] https://dev.twitter.com/rest/public

Model	Signal Media	NFCWorld
DTM	$276,473,612.34$	$431,603.59$
dDTM-News	$206,390,120.21$	$379,685.63$

Table 3: Average negative log-likelihood of DTM and dDTM-News using two datasets made of news and blog articles (Signal Media) and of tweets (NFCWorld). The performance shows the ability in predicting the topics of the following time slice (the lower, the better).

model on held-out data [6]. More in detail, we can train the model using the first n time slices (e.g., $n = 5$) and compute the log-likelihood of the topics modeled for the next one. This is done iteratively with sliding windows of size 5. As baseline, we used the original version of DTM [6]. As we can see from Table 3, our model gives better results in terms of negative log-likelihood since it can capture topics which change at a high pace.

To verify the quality of the extracted keywords and informativeness of the topical skips, we manually inspect some of the topics in the timeline. As we can see from Tables 4 and 5, the keywords are highly descriptive and they do not appear in all the temporal windows continuously. We split the SignalMedia dataset, which spans one month, in daily slices. One example is about *health care* which had skips in the timeline as it was discussed during the whole month but not in all the days. Regarding the NFCWorld dataset, it spans a longer period of time, so we divided the three year of data in time slices of three months each. This dataset is made of tweets which talk about technology and innovation, hence we could observe a very dynamic behavior of the discussed topics. As an example, the topic about the *apple pay service* was not present at the beginning of the dataset, but its popularity increases around October 2014 (5th time slice). We checked on Wikipedia and could see that this was when the service was launched in the USA. The topic kept to be popular for a few months, then its popularity decreased to grow again in February 2016 when the service was presented in China (captured in the 12th time slice).

Table 4: Example of a topic which evolves over time and has skips in the timeline. The topic is about *health care* and is extracted from the `SignalMedia` dataset which covers 1 month and was divided in daily time slices.

Time Slice	Time Window	Keywords
w_0	Sep. 1st	health, patients, medical, care, cancer, dr, treatment, disease, clinical, drug
w_3	Sep. 4th	patients, cancer, study, cell, treatment, disease, clinical, drug, drugs, therapy
w_7	Sep. 8th	cancer, patients, clinical, treatment, drug, hospital, tues, lung, dr, patient
w_8	Sep. 9th	health, patients, care, study, cancer, medical, dr, treatment, disease, patient
w_9	Sep. 10th	patients, treatment, study, drug, clinical, disease, heath, cancer, phase, researchers
w_{10}	Sep. 11th	health, study, patients, blood, cancer, medical, treatment, care, hearth, dr
w_{13}	Sep. 14th	patients, study, health, studies, medicine, clinical, medical, drug, treatment, disease
w_{16}	Sep. 17th	patients, medical, health, study, cancer, treatment, clinical, disease, dr, drug
w_{18}	Sep. 19th	cancer, patients, clinical, treatment, drug, hospital, tues, lung, dr, patient
w_{20}	Sep. 21st	health, study, care, medical, patients, treatment, system, cancer, disease, drug
w_{21}	Sep. 22th	health, study, medical, patients, treatment, system, cancer, disease, drug
w_{22}	Sep. 23th	study, cancer, patients, treatment, dr, disease, view, researchers, clinical, service
w_{24}	Sep. 25th	care, dr, medical, physician, patient, weight, infected, healthcare, health, doctors
w_{27}	Sep. 28th	cancer, drug, medical, patients, clinical, care, treatment, health, patient, dr
w_{29}	Sep. 30th	health, cancer, patients, medical, care, treatment, study, dr, clinical, patient

Table 5: Example of a topic which evolves over time and has skips in the timeline. The topic is about the *apple pay service* and is from the `NFCWorld` dataset which covers 3 years and was divided in time slices of 3 months each.

Time Slice	Time Window	Keywords
w_6	Oct.–Dec. 2014	payments, apple, launches, loyality, wearable, retail, wallet, applepay, visa, contactless
w_7	Jan.–Mar. 2015	pay, apple, payments, applepay, ble, hce, biometrics, launch, marketing, wallet
w_8	Apr.–Jun. 2015	hce, payments, tokenization, apple, applepay, pay, contactless, mobilewallet, biometrics, industry
w_9	Jul.–Sept. 2015	pay, apple, android, payments, contactless, androidpay, beacons, mobilewallet, biometrics
w_{10}	Oct.–Dec. 2015	mobilewallet, applepay, adds, pay, payments, emv, launch, concatless, apple, androidpay
w_{12}	Apr. 2016	contactless, payments, pay, mobilewallet, hce, china, applepay, apple, retail, launch

9.3. Event Detection

We now present our results on event detection from news streams. For this experiments, we used our heterogenous collection of news documents downloaded from different streams. As baselines we used the original version of DTM [6] and other event-detection approaches: *Unigram Co-Occurrences* [12] and Event Detection with Clustering of Wavelet-based signals (*EDCoW*) [44]. The former analyzes the co-occurrences of bursty features (unigrams) in non-overlapping time windows, while the latter creates a signal for each individual word and applies *wavelet transformation* to find bursty words.

Table 6: Events detected in the different time windows. We show one event (if present) per window and, when possible, the rows are aligned based on the keywords in order to show the same real-word event.

Time Window	dDTM-News	Unigram Co-Occurrences	EDCoW
w_0: Mar. 1–7	allegations, abuse, pell, cardinal, commission	pell, abuse	cuba, damage, killed, told, vatican
w_1: Mar. 8–14	cuba, president, cuban, government, castro	cuba, obama	cuba, left, meet
w_2: Mar. 15–21	korea, north, south, missile, ballistic	north-korea, test	–
w_3: Mar. 22–28	brussels, attacks, belgian, paris, isis	brussels, attacks	meet, obama, visit
w_4: Mar. 29–Apr. 4	plane, cyprus, egyptair, hijacked, hijacker	plane, hijacker	–
w_5: Apr. 5–11	ukraine, dutch, referendum, vote, deal	ukraine, dutch	–
w_6: Apr. 12–18	people, earthquake, japan, quake, area	quake, japan	damage, death toll, hit, missing, working
w_7: Apr. 19–25	obama, president, eu, castro, europe	obama, british	–
w_8: Apr. 26–May 2	cuba, cruise, people, sail, photography	cuba, cruise	–
w_9: May 3–9	fire, city, mcmurray, fort, killed	mcmurray, fire	fire, fort-mcmurray, started
w_{10}: May 10–16	bangladesh, people, lightning, kill, rain	lightning, bangladesh	fire, flooding, rain, reported
w_{11}: May 17–23	egyptair, flight, plane, cairo, egypt	flight, egyptair	–
w_{12}: May 24–30	obama, war, president, visit, hiroshima	vietnam, obama	–
w_{13}: May 31–Jun. 6	ali, muhammad, family, boxing, police	boxing, ali	funeral, kentucky, muhammad-ali, vietnam
w_{14}: Jun. 7–13	grimmie, pool, voice, christina, shot	grimmie, christina	–
w_{15}: Jun. 14–20	orlando, mateen, people, nightclub, shooting	orlando, mateen	gun control
w_{16}: Jun. 21–27	eu, uk, european, britain, leave	britain, brexit	–
w_{17}: Jun. 28–30	zika, virus, transmission, california, committee	zika, rio	–

Table 6 shows the detected events with time windows of 7 days. Each column represents a different methodology. As we can see, some of the events detected by *EDCoW* are noisy because this approach uses cross-correlation as similarity measure and may merge distinct events which have happened in the same period of time just by chance. On the other hand, *Unigram Co-Occurrences* extracts too general keywords that are difficult to interpret without a manual inspection of the news documents.

To quantify the performance of the different techniques, we computed the *event coverage* with respect to the 60 events labeled in the collection. We could observe that our model achieved a coverage of 55/60 and the original DTM of 44/60. While *Unigram Co-Occurrences*'s and *EDCoW*'s coverage were 40/60 and 8/60, respectively. Although these results depend on the labeled events, they confirm how topic models detect more events. Note also that determining the coverage (recall) of an event-detection task is tricky due to an unknown number of events that may have happened in the four months of our data.

9.4. Event Tracking

Event tracking consists in chaining related events over the timeline. Note that we could not chain events with *Unigram Co-Occurrences* and *EDCoW* since they only discover events. In Table 7, we show some of the event chains computed with dDTM-News using $\alpha = 0.01$, which was determined empirically, while the other parameters (β, K, and the number of event chains) are estimated automatically by our model. In particular, we obtained 90 chains with at least 30 and at most 40 events, and an average of 37 events over all the time slices. While in our model the number of topics is estimated dynamically based on the time slice, in DTM the number of topics is fixed and must be specified up front. To have a fair comparison between the two models, we used $K = 40$ for DTM, which is the maximum number of events estimated by dDTM-News in a time window. Also, we determined experimentally that 40 events per time slice is a good value for modeling events in weekly temporal windows.

We observed that our dDTM-News approach is able to track some popular events related to Brexit ($c = 31$), Zika ($c = 66$), terror attacks in Belgium ($c = 80$), and multiple launch attempts of nuclear missiles from North Korea ($c = 4$). While we could notice that the original DTM approach returns events very similar to each other for different time slices, making more difficult the detection of novel events and, as expected, it does not work well with highly-dynamic collections.

To quantify the performance of the two models in tracking the events, we measured the *coherence* and *informativeness* of the chains obtained with DTM and dDTM-News. We asked human assessors to label the chains. Given the complexity of this manual labeling, we did not rely on crowdsourcing workers but we rather involved 10 assessors among PhDs and PhD students from different universities and research institutes that are familiar with the concepts of event tracking. For the coherence labeling, the assessors had to verify using the Web (e.g., search results and Wikipedia) whether the events appearing in a given chain represented a *good fit* for the chain or not. We quantify the chain coherence using precision: $\frac{tp}{tp+fp}$ where tp (true positives) is the number of events that

Table 7: Each block of the table corresponds to a chain of events. In particular, we report the id of the chain (c), the event id (e), the temporal window, and the event keywords.

c	e	**Time Window**	**Keywords**
4	10	w_2: Mar. 15–21	korea, north, south, missile, ballistic
	29	w_8: Apr. 26–May 2	north, korea, missile, kim, test
	9	w_{13}: May 31–Jun. 6	north, korea, missile, korean, nuclear
	9	w_{16}: Jun. 21–27	north, korea, test, launch, musudan
31	37	w_{10}: May 10–16	eu, station, germany, munich, german
	10	w_{11}: May 17–23	eu, britain, uk, european, union
	21	w_{15}: Jun. 14–20	eu, court, britain, leave, brexit
	7	w_{16}: Jun. 21–27	brexit, cameron, virginia, west, eu
	25	w_{16}: Jun. 21–27	eu, uk, european, britain, leave
	20	w_{17}: Jun. 28–30	eu, brexit, leave, vote, britain
	31	w_{17}: Jun. 28–30	eu, european, brexit, britain, union
38	24	w_{13}: May 31–Jun. 6	ali, muhammad, family, boxing, police
	0	w_{14}: Jun. 7–13	ali, muhammad, boxing, louisville, funeral
66	13	w_{11}: May 17–23	zika, health, gonzales, cases, exposed
	18	w_{15}: Jun. 14–20	zika, health, virus, women, art
	16	w_{17}: Jun. 28–30	zika, virus, transmission, california, committee
80	15	w_3: Mar. 22–28	brussels, attacks, belgian, paris, isis
	11	w_4: Mar. 29–Apr. 4	police, attacks, brussels, belgian, suspect
	30	w_5: Apr. 5–11	authorities, france, police, people, investigators
	7	w_8: Apr. 26–May 2	official, paris, attacks, islamic, abdeslam
86	20	w_{14}: Jun. 7–13	orlando, police, shooting, nightclub, gay
	30	w_{15}: Jun. 14–20	orlando, mateen, people, nightclub, shooting

represent a good fit, and fp (false positives) the number of events that are not a good fit for the chain. The coherence does not give any indication on how useful the chain itself is (i.e., it may have a high coherence but be not informative because reports the same keywords in all the time slices). Hence, we also asked the assessors to evaluate whether the chain was useful and made sense for a human (informativeness). For these labeling tasks, we registered a Fleiss' Kappa of 0.67 for the coherence and of 0.84 for the informativeness, corresponding to a substantial and almost perfect agreement, respectively [20]. Values of coherence and informativeness are reported in Table 8. As we can see, both models achieve high level of coherence (> 0.7) but our model has a larger fraction of informative chains.

Table 8: Evaluation of the topic chains created by DTM and dDTM-News in terms of coherence, informativeness, and delay (in days). The last column shows the performance of topic prediction in term of average negative log-likelihood (the lower, the better).

Model	Coherence	Infor.	Delay	Log-likelihood	
				All tweets	bbc & cnn
DTM	0.71	0.34	5.3	79,537,557.16	219,352,622.09
dDTM-News	**0.76**	**0.60**	**1.75**	**75,214,149.14**	**142,032,353.42**

We also considered the *delay* of the two models in capturing novel events (i.e., events that are not present in the previous time slices). Since for the 60 labeled events we know the dates when they occurred, we could check how much was the delay of the event detection. More formally, let ts_e^* be the timestamp of the event e and ts_e^i be the first timestamp of the temporal window in which e was detected, we can assume that $ts_e^i > ts_e^*$ and define the latency for the event e as $l(e) = ts_e^i - ts_e^*$. The average latency over the events was 1.75 days for dDTM-News and 5.3 days for DTM (as reported in Table 8), confirming that dDTM-News, relaxing the assumptions of DTM, can timely detect novel events.

Lastly, we trained the two models on the first t (e.g., $t = 5$) time slices and evaluated their ability to predict the topics of the next time slice $(t + 1)$ by computing the variational bound on the negative log-likelihood of the news in the time slice $t + 1$ under the resulting models. This process is iteratively repeated till the last time slice. Running these experiments on the whole dataset is prohibitive, so we created two samples of data. The first one is made of all the tweets and has size 80K. The second one contains all the news documents from two popular channels (`bbc` and `cnn`) for a total of 39K news documents. As we can see from Table 8, our model performs better than DTM. Note that, since the negative log-likelihood is used, lower values are better. The improvement in the log-likelihood is due to the fact that our model promptly adjusts its topic prediction, while DTM tends to keep older topics.

9.5. Predicting Persisting Events

We now present our experiments on predicting the events (topics) that would persist in the future in a collection of news documents. This is particularly important, for example, when one wishes to know in advance which events would attract attention for a longer period of time. Indeed, journalists may want to discover the emerging events that are likely to remain popular for the next weeks in order to give them more importance or to examine them more in depth (e.g., collecting detailed information, organizing interviews with experts).

In Section 5 we have presented the K2RE system for predicting persisting topics. This system was already tested on scientific articles [2], while here we use a news collection and check the performance of the K2RE system in predicting events, which is more challenging due to their unpredictable changes.

Given the events of the first n time slices, $\{e_{(0,0)}, ..., e_{(0,k)}, ..., e_{(n,0)}, ..., e_{(n,k)}\}$, and those of the $(n+1)$-th time slice, $\{e_{(n+1,0)}, ..., e_{(n+1,k)}\}$, we asked human assessors to manually annotate the events in the time slice $(n+1)$ as "continued" or "not continued." To help the annotation, we first clustered the events using a k-nearest-neighbors algorithm, then for each $e_{(n+1,i)} \in \{e_{(n+1,0)}, ..., e_{(n+1,k)}\}$, we showed to the assessor its top-5 neighbors among the events of the previous time slices. The assessors were asked to check whether $e_{(n+1,i)}$ is a continuation of the events appearing in the first n time slices or not. We compared the K2RE prediction performance against dDTM-News. In Figure 6, we report the behavior of precision and recall. As we can see from the curves, the K2RE system outperforms the discrete dynamic topic model, for example, keeping the recall around 20%, the precision is about 80% while dDTM-News has precision of 70%. Of course, when the recall increases, a lower precision is recorded, but the K2RE's precision is still higher compared to the other one.

We manually inspected the events returned by the system to find some examples of persisting events. We could notice that K2RE correctly predicts events, such as *Brexit* or *Zika virus*, which persist till the last time slice as they were popular for long time.

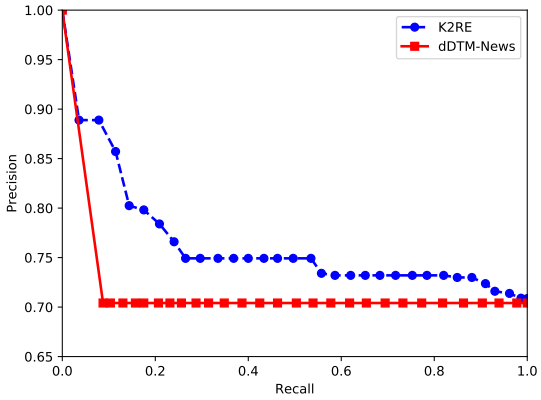


Figure 6: Precision-recall curve: K2RE vs. dDTM-News.

9.6. News Clustering

After having detected the events with dDTM-News (Section 9.3), we can create the event-based clusters of news documents. We compare our clustering against the ones obtained with *k-means*, *k-means+time*, and another topic modeling approach, called *hierarchical LDA* (*hLDA*) [13].

The *k-means* algorithm assigns each document to one cluster based on a distance measure such as cosine similarity. Since the number of clusters (events) must be known up front, we run *k-means* with different values of k and observed a good trade-off between precisions and cluster sizes with $k = 500$. These clusters do not have any label, so we manually checked the content of them to figure out the corresponding events. We could observe that *k-means* tends to mix similar events happened in different time windows, hence we implemented a time-aware version of it which filters out the news documents outside the window of the event of interest and we call it *k-means+time*.

The *hierarchical LDA* creates a hierarchy of topics and has the advantage that we do not need to know the number of topics because, using a high number of levels, the model automatically discovers it. This topic model has been also used for text clustering [23] since the documents can be clustered based on their allocation in a topic space representing the bottom-level topics. We run *hLDA*

Table 9: Comparison of clusterings.

Approach	Avg. Precision		Avg. Recall	F-Score
	Micro	Macro		
dDTM-News	**0.87**	0.80	**0.87**	**0.83**
hLDA	0.53	0.60	0.54	0.56
k-means	0.60	0.51	0.41	0.45
k-means+time	0.81	**0.82**	0.66	0.73

using different depths and could notice that the best trade-off between number of events and the size of the clusters can be achieved with depths equal to 5 as lower values tend to group together different events while higher values tend to create singleton clusters.

Clustering performance. To compute the clustering performance, we can apply the standard information-retrieval metrics. In particular, for each event e, we use the labels of the documents in the corresponding event cluster, C_e, to compute the *recall* and the *precision* as follows:

$$R_e = \frac{|\{\text{documents relevant to } e\} \cap \{\text{documents in } e\text{'s cluster}\}|}{|\{\text{documents relevant to } e\}|}$$

$$P_e = \frac{|\{\text{documents relevant to } e\} \cap \{\text{documents in } e\text{'s cluster}\}|}{|\{\text{documents in } e\text{'s cluster}\}|}$$

These are averaged over all the event-based clusters of news. The *macro-average precision* gives an idea on how the approaches perform overall. Since the clusters of news documents may largely vary in size depending on the event, we computed also the *micro-average precision*. Finally, we quantified the trade-off between precision and recall with the *macro-average F-Score* $= 2 \cdot \frac{P \cdot R}{P+R}$

Table 9 shows the results on the clustering performance. As we can see, dDTM-News has high value of micro-precision followed by *k-means+time* which has slightly better macro-precision. Also, the recall achieved with dDTM-News is higher and consequently the F-Score is better compared to the others.

Table 10 reports a few examples of news documents for some of the event clusters created by our approach. The documents are coherent and well rep-

Table 10: Examples of news documents for some of the event clusters.

Event	News Document
North Korea Nuclear Missile (c=4, w=16, e=9)	North Korea claims successful test of new midrange ballistic missile North Korea missiles 'a serious threat' after new tests Kim Jong-un says Musudan missile gives North Korea ability to attack US
Japan Earthquake (c=9, w=6, e=16)	Magnitude 7.4 quake hits near Japan's Kumamoto; tsunami advisory issued Three dead, over 100 taken to hospitals after strong quakes hit Japan's Kumamoto Powerful back-to-back earthquakes in Japan kill at least 41
Muhammad Ali Death (c=38, w=13, e=24)	Muhammad Ali dies aged 74. Ali is regarded as the greatest professional... Muhammad Ali, the three-time heavyweight boxing champion, has died at 74 Boxing legend Muhammad Ali died of septic shock, family spokesman says...
Orlando Shooting (c=86, w=14, e=20)	Police: 20 dead in act of terrorism at Orlando's Pulse nightclub Fifty people were massacred at a gay club in Florida, the worst shooting... BREAKING: Orlando mayor says 48 of 49 Orlando victims have been identified

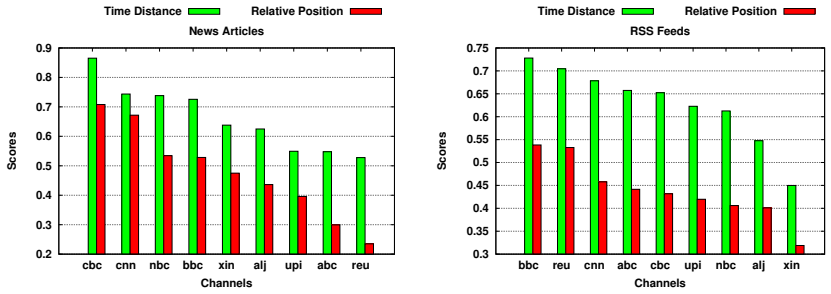
resentative of the events. Analyzing the data, we noticed that false positives are mainly due to some minor events that were poorly represented by the word co-occurrences. The low precisions achieved by *k-means* and *hLDA* are due to many false positives caused by documents which have similar words yet describe different events (e.g., Japan earthquake vs. Japan tsunami's 5th anniversary).

9.7. Timeliness Analysis

Our analysis on mining frequent temporal patterns was performed using PrefixSpan [29]. Some of the patterns discovered with a minimum support threshold of 4 ($minSup = 4$) are reported in Table 11.

Table 11: Temporal patterns sorted by their frequency.

Rank	Channels' Patterns	# Events
1.	($\texttt{reu}_{\texttt{rss}}$ $\texttt{reu}_{\texttt{twt}}$), ($\texttt{abc}_{\texttt{rss}}$ $\texttt{xin}_{\texttt{twt}}$)	13
2.	($\texttt{cnn}_{\texttt{rss}}$ $\texttt{upi}_{\texttt{rss}}$)	12
3.	($\texttt{bbc}_{\texttt{rss}}$), ($\texttt{nbc}_{\texttt{rss}}$ $\texttt{nbc}_{\texttt{twt}}$)	11
4.	($\texttt{reu}_{\texttt{twt}}$ $\texttt{reu}_{\texttt{rss}}$)	10
5.	($\texttt{nbc}_{\texttt{rss}}$ $\texttt{nbc}_{\texttt{twt}}$)	9
6.	($\texttt{bbc}_{\texttt{rss}}$), ($\texttt{abc}_{\texttt{rss}}$ $\texttt{xin}_{\texttt{twt}}$)	9
7.	($\texttt{abc}_{\texttt{rss}}$ $\texttt{abc}_{\texttt{twt}}$), ($\texttt{upi}_{\texttt{rss}}$)	9
8.	($\texttt{abc}_{\texttt{twt}}$ $\texttt{cnn}_{\texttt{rss}}$), ($\texttt{nbc}_{\texttt{rss}}$ $\texttt{nbc}_{\texttt{twt}}$)	8
9.	($\texttt{cnn}_{\texttt{rss}}$), ($\texttt{abc}_{\texttt{rss}}$ $\texttt{xin}_{\texttt{twt}}$)	8
10.	($\texttt{reu}_{\texttt{rss}}$ $\texttt{reu}_{\texttt{twt}}$), ($\texttt{abc}_{\texttt{twt}}$ $\texttt{cnn}_{\texttt{rss}}$)	8



(a) News Articles

(b) RSS Feeds

(c) Tweets

Figure 7: Streams' ranking using time-distance and relative-position scores. Each plot is for a different platform, and the channels are sorted by decreasing values of time-distance scores, so that faster channels are on the left.

As we can see, *Reuters* (`reu`) frequently reported news before `abc` and the Chinese channel (`xin`). Also, `bbc` and `cnn` preceded `abc`, `nbc`, and `xin`. Other patterns show how the channels used the different platforms. For example, `abc`, `nbc`, and `reu` published news on RSS platforms and Twitter mostly at the same time, while `bbc` favored the RSS platform. These patterns are confirmed by the ranking of the channels (shown in Figure 7).

Concerning the timeliness in reporting the events, Figure 7 shows the streams ranked with the scores described in Section 7.2. Although the two functions behave similarly, time distance is a better indicator of timeliness since the difference between the first and second positions in the time-sorted list of streams is significant only inspecting the time distance between the two publishing times-

tamps. We observed that some channels, such as `cbc` and `cnn`, publish very timely on their own website and then spread information using RSS feeds and tweets with links to the official websites. Reuters timely reported the events in Twitter and was followed by `cnn` and `xin`. The `bbc` published RSS feeds and news articles before the tweets. Finally, for `alj` and `upi` we registered similar scores for all the platforms.

Since some of the events were not reported by all the streams, in Figure 8 we show the fraction of events reported by the newswires using the different platforms averaged over all the time slices. Most of the newswires (`abc`, `cnn`, `nbc`, `reu`, and `xin`) reported all the events in Twitter, while `bbc` favored the news articles and RSS feeds. Moreover, some channels (e.g., `upi` and `xin`) reported most of the events using Twitter, but only some of them were also reported in RSS feeds and news articles.

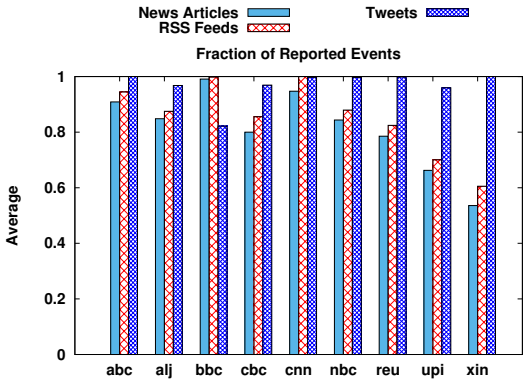


Figure 8: Fraction of events reported by different channels on different platforms averaged over all the time slices.

10. Conclusions

In this paper, we presented our research on mining events from heterogeneous news streams, tracking their evolution, and finding events that persist over time. We also showed how to create event-based clusters of news documents and how to use these clusters for cross-linking news streams that report on

the same events. We then analyzed the temporal publishing patterns of the streams and ranked the news channels based on their timeliness in reporting the events. We believe that the presented approaches could be of great value for news analytics and for discovering newsworthy stories as well as for studying and predicting the evolution of events over time. As future work, we plan to investigate the application of these results to the fields of fake-news detection and trustworthiness of online sources of information.

11. Acknowledgements

This research was partially supported by the Secrétariat d'Etat à la formation, à la recherche et à l'innovation (SEFRI) under the project AHTOM (Asynchronous and Heterogeneous TOpic Mining).

References

[1] J. Allan, R. Papka, and V. Lavrenko. On-line New Event Detection and Tracking. In *SIGIR '98*, pages 37–45, New York, NY, USA, 1998. ACM.

[2] S. A. Bahrainian, I. Mele, and F. Crestani. Predicting topics in scholarly papers. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval*, pages 16–28, Cham, 2018. Springer International Publishing.

[3] S.A. Bahrainian, I. Mele, and F. Crestani. Modeling discrete dynamic topics. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 858–865, New York, NY, USA, 2017. ACM.

[4] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 1967.

[5] H. Becker, M. Naaman, and L. Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. *ICWSM*, 11:438–441, 2011.

[6] D. M. Blei and J. D. Lafferty. Dynamic Topic Models. In *ICML'06*, pages 113–120, 2006.

[7] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[8] T. Brants, F. Chen, and A. Farahat. A System for New Event Detection. In *SIGIR '03*, pages 330–337, New York, NY, USA, 2003. ACM.

[9] G.M. Del Corso, A. Gullí, and F. Romani. Ranking a Stream of News. In *WWW '05*, pages 97–106, New York, NY, USA, 2005. ACM.

[10] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[11] G.P.C. Fung, J.X. Hu, H. Liu, and P.S. Yu. Time-dependent Event Hierarchy Construction. In *KDD '07*, pages 300–309, New York, NY, USA, 2007. ACM.

[12] G.P.C. Fung, J.X. Yu, P.S. Yu, and H. Lu. Parameter Free Bursty Events Detection in Text Streams. In *VLDB '05*, pages 181–192. VLDB Endowment, 2005.

[13] D.M. Griffiths and M.I. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[14] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[15] R. Gwadera and F. Crestani. Mining and Ranking Streams of News Stories Using Cross-stream Sequential Patterns. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1709–1712, New York, NY, USA, 2009. ACM.

[16] Q. He, K. Chang, and E. Lim. Analyzing Feature Trajectories for Event Detection. In *SIGIR '07*, pages 207–214, New York, NY, USA, 2007. ACM.

[17] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'99, pages 50–57, New York, NY, USA, 1999. ACM.

[18] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[19] J. Kleinberg. Bursty and Hierarchical Structure in Streams. In *KDD '02*, pages 91–101, New York, NY, USA, 2002. ACM.

[20] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[21] C. Li, A. Sun, and A. Datta. Twevent: Segment-based Event Detection from Tweets. In *CIKM '12*, pages 155–164, New York, NY, USA, 2012. ACM.

[22] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B. Lee. TwiNER: Named Entity Recognition in Targeted Twitter Stream. In *SIGIR '12*, pages 721–730, New York, NY, USA, 2012. ACM.

[23] P. Liu, L. Li, W. Heng, and B. Wang. HLDA based Text Clustering. In *2012 IEEE 2nd Int. Conf. on Cloud Computing and Intelligence Systems*, volume 3, pages 1465–1469. IEEE, 2012.

[24] Y. Lu, Q. Mei, and C. Zhai. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14(2):178–203, 2011.

[25] I. Mele, S.A. Bahrainian, and F. Crestani. Linking news across multiple streams for timeliness analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 767–776, New York, NY, USA, 2017. ACM.

[26] I. Mele, F. Bonchi, and A. Gionis. The early-adopter graph and its application to web-page recommendation. In *CIKM '12*, pages 1682–1686, New York, NY, USA, 2012. ACM.

[27] I. Mele and F. Crestani. Event detection for heterogeneous news streams. In *Natural Language Processing and Information Systems: 22nd International Conference on Applications of Natural Language to Information Systems*, NLDB '17, pages 110–123, Cham, 2017. Springer International Publishing.

[28] M. Osborne and M. Dredze. Facebook, twitter and google plus for breaking news: Is there a winner? In *ICWSM*, 2014.

[29] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *Proc. of the 17th Int. Conf. on Data Engineering*, pages 215–224, Washington, DC, USA, 2001. IEEE Computer Society.

[30] S. Petrović, M. Osborne, and V. Lavrenko. Streaming First Story Detection with Application to Twitter. In *HLT '10*, pages 181–189, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[31] S. Phuvipadawat and T. Murata. Breaking News Detection and Tracking in Twitter. In *WI-IAT '10*, pages 120–123, Washington, DC, USA, 2010. IEEE Computer Society.

[32] A. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting Events and Event Descriptions from Twitter. In *WWW '11*, pages 105–106, New York, NY, USA, 2011. ACM.

[33] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[34] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named Entity Recognition in Tweets: An Experimental Study. In *EMNLP '11*, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[35] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open Domain Event Extraction from Twitter. In *KDD '12*, pages 1104–1112, New York, NY, USA, 2012. ACM.

[36] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *WWW '10*, pages 851–860, New York, NY, USA, 2010. ACM.

[37] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[38] M. Tsagkias, M. de Rijke, and W. Weerkamp. Linking Online News and Social Media. In *WSDM '11*, pages 565–574, New York, NY, USA, 2011. ACM.

[39] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, Sep 2006.

[40] C. Wang, D. Blei, and D. Heckerman. Continuous Time Dynamic Topic Models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pages 579–586, Arlington, Virginia, United States, 2008. AUAI Press.

[41] X. Wang and A. McCallum. Topics over Time: A non-Markov Continuous-time Model of Topical Trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'06, pages 424–433, New York, NY, USA, 2006. ACM.

[42] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining Correlated Bursty Topic Patterns from Coordinated Text Streams. In *KDD '07*, pages 784–793, New York, NY, USA, 2007. ACM.

[43] X. Wang, K. Zhang, X. Jin, and D. Shen. Mining Common Topics from Multiple Asynchronous Text Streams. In *WSDM '09*, pages 192–201, New York, NY, USA, 2009. ACM.

[44] J. Weng and B. Lee. Event Detection in Twitter. *ICWSM*, 11:401–408, 2011.

[45] Y. Yang, T. Ault, T. Pierce, and C.W. Lattimer. Improving Text Categorization Methods for Event Tracking. In *SIGIR '00*, pages 65–72, New York, NY, USA, 2000. ACM.

[46] Y. Yang, J.G. Carbonell, R.D. Brown, T. Pierce, B.T. Archibald, and X. Liu. Learning Approaches for Detecting and Tracking News Events. *IEEE Intelligent Systems*, 14(4):32–43, July 1999.

[47] Y. Yang, T. Pierce, and J. Carbonell. A Study of Retrospective and On-line Event Detection. In *SIGIR '98*, pages 28–36, New York, NY, USA, 1998. ACM.

[48] X. Zhou, J. Cao, Z. Jin, F. Xie, Y. Su, D. Chu, Z. Cao, and J. Zhang. Real-Time News Certification System on Sina Weibo. In *WWW '15 Companion*, pages 983–988, New York, NY, USA, 2015. ACM.