Linking News across Multiple Streams for Timeliness Analysis

Ida Mele, Seyed Ali Bahrainian, and Fabio Crestani
Faculty of Informatics, Università della Svizzera italiana (USI)
Lugano, Switzerland
{ida.mele,bahres,fabio.crestani}@usi.ch

ABSTRACT

Linking multiple news streams based on the reported events and analyzing the streams' temporal publishing patterns are two very important tasks for information analysis, discovering newsworthy stories, studying the event evolution, and detecting untrustworthy sources of information. In this paper, we propose techniques for cross-linking news streams based on the reported events with the purpose of analyzing the temporal dependencies among streams.

Our research tackles two main issues: (1) how news streams are connected as reporting an event or the evolution of the same event and (2) how timely the newswires report related events using different publishing platforms. Our approach is based on dynamic topic modeling for detecting and tracking events over the timeline and on clustering news according to the events. We leverage the event-based clustering to link news across different streams and present two scoring functions for ranking the streams based on their timeliness in publishing news about a specific event.

CCS CONCEPTS

• **Computing methodologies** → **Topic modeling**; • **Information systems** → *Data streams*; *Temporal data*;

KEYWORDS

news streams; event mining; dynamic topic modeling; temporal analysis

1 INTRODUCTION

In this paper we aim at cross-linking news streams based on the events they report and ranking the streams according to their timeliness. We focus on news published by several newswires on different platforms, and we propose techniques for finding semantic relations and temporal dependencies among these streams. A news stream for us is a sequence of news published by a newswire (e.g., Al Jazeera, BBC, CNN) using one of several publishing platforms (e.g., Twitter, RSS news portals, and news websites). Hence, tweets, RSS feeds, and news articles published, for example, by the CNN are considered as three separate streams even if the news channel is the same. In the rest of the paper, we use "stream" to refer to a news channel publishing on a platform, and we use "news document" to refer to a tweet, an RSS feed item, or a web article. Since our approach is based on topic modeling we use the terms "topic" and "event" interchangeably. In particular, an event is represented by a set of keywords that semantically describe it, plus the time when the event has happened (i.e., a time slice of the dataset).

Finding news documents which describe a specific event (e.g., Japan earthquake) or that are related to each other (e.g., terror attacks in Brussels followed by the arrest of some suspects) is challenging. First of all, we have to face the problem of automatic discovering the events and connecting them based on their temporal evolution. Secondly, we need to divide the news documents into clusters according to the events they describe. The first problem is related to event detection and tracking. Traditional approaches for event detection analyzed streams separately and focused only on news articles [9] or tweets [15]. Most of them are domain and query dependent [22, 25] and have scalability issues since they rely on monitoring the frequencies of words over time [9, 30]. The second problem is event-based clustering of news documents. Conventional approaches help to divide the documents into broad categories (e.g., politics, sports) [17] but fail in creating fine-granularity clusters representing crisp and coherent events. We propose an approach based on Dynamic Topic Modeling (DTM) [5] for discovering and tracking down events. Compared to other topic models (e.g., LDA), which are not concerned with the document sequence and cannot capture the topic evolution, DTM takes into account the sequential order of the documents and allows to monitor topic evolution and trends. We relax some of the assumptions of DTM in order to deal with highly-dynamic collections, detect fine-granularity topics (i.e., events), and chain them over the timeline. Differently from DTM, the evolution of the topics is modeled in a non-linear way which allows to promptly capture novel events. Moreover, we do not need to know the number of events in the temporal window of interest, since this is estimated by the approach, allowing more flexibility. Our approach is based on topic modeling, so it does not require any prior knowledge of the events (e.g., keywords for querying the collection). It can be used on dynamic and heterogenous streams where documents have different length and writing styles (e.g., news articles vs. tweets). Plus, we can use the document allocation in the topic space for clustering news based on the topics (events).

As further contribution we leverage the event-based clusters of news documents to link the streams of news and analyze their temporal publishing dependencies. To the best of our knowledge this is the first attempt to analyze multiple and heterogenous streams based on the reported events. The research paper that is most similar to ours is the one by Gwadera and Crestani [12]. It addresses the problem of mining and ranking multiple news streams, but it takes into account only homogeneous streams (i.e., RSS feeds) and applies a simple unsupervised clustering to divide the news into clusters. In addition, it does not provide any label for deciphering

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132988

the corresponding event and does not allow to track the events over the timeline. Moreover, the news channels are ranked based on a probabilistic approach which relies on news content similarity. We try to overcome these limitations and analyze different types of documents coming from different publishing platforms. We cluster news documents based on fine-granularity events, and each cluster is semantically described by keywords providing an understandable interpretation of the event. Furthermore, the events are chained over the timeline for reconstructing their evolution. Finally, we cross-link news streams that reported news on the same events and present two scoring functions for ranking these streams based on their timeliness in publishing the news.

Thus, the contribution of the paper is threefold:

- we present an approach based on dynamic topic modeling for detecting and tracking events over a timeline;
- we analyze temporal publishing patterns of news streams and provide two scoring functions for ranking them based on how promptly they report events;
- we created a collection of heterogenous news streams which is labeled at the level of fine-granularity events to prove the effectiveness of our methodology.

The paper is structured as follows. The background on topic models is presented in Section 2. Section 3 describes our methodology for event detection and news clustering. In Section 4 we present techniques for cross-linking news streams, mining their frequent temporal publishing patterns, and ranking them based on their timeliness. We report our experiments in Section 5, and review related work in Section 6. Finally, Section 7 concludes the paper.

2 BACKGROUND

Latent Dirichlet Allocation (LDA). This topic model assumes that the documents are mixture of topics, and the topics are represented by word distributions [6]. LDA is based on a probabilistic generative process where documents are generated one word at a time. At each step, a topic is selected from the topic distribution, θ, which is parametrized by the parameter α. Then, a word is randomly picked from a multinomial distribution of words, β, which is conditioned by the chosen topic. Inverting this process, it is possible to infer latent topics from the observed words. Statistical inference techniques (e.g., Gibbs sampling) are employed to learn the underlying topic distribution of each document as well as the word distribution of each topic based on word co-occurrences [10].

Dynamic Topic Model (DTM). LDA was conceived for working on non-sequential datasets and mixes up co-occurrence patterns from documents belonging to different historical periods. In sequential collections where, for example, documents are sorted by their timestamps, taking into account the ordering of documents is essential to better understand the underlying topics and to track their evolution over time. DTM [5] extends LDA for estimating the topic distributions at the different time instances.

Given a temporal-ordered collection of documents, DTM divides it into time slices and applies LDA to each of them. Since topics of the current time slice, t_i, evolve from the topics of the previous one, t_{i-1}, DTM chains the $\beta_{t,k}$ in a state space model that evolves with a Gaussian noise:

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

where $\mathcal{N}$ is a logistic normal distribution. It also uses a logistic normal with mean α to express uncertainty over per-document topic proportions:

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

discrete Dynamic Topic Model (dDTM). The sudden evolution of topics in dynamic collections of documents makes the detection of the emerging topics more challenging for DMT. Indeed, DTM is suitable only for datasets where topics change slowly (e.g., scientific articles) due to its assumption that topics evolve linearly over time. To overcome this limitation, in [2] we presented dDTM that models the topics of time slice t_i irrespective of those of t_{i-1}. Then, it chains the discovered topics using a Hidden Markov Model [23].

3 METHODOLOGY

We now present a variation of dDTM, called *dDTM-News*, to cope with highly-dynamic streams of news documents. We assume that the latent topics represent real-world events, so they change at a high pace (e.g., new topics may suddenly emerge and disappear or have skips in the timeline). Moreover, since topics are triggered by external events, their number varies depending on the time slice. Compared to the traditional dynamic topic model [5], our approach does not estimate the per-document topic proportions of one time slice based on those from the previous one. Moreover, it dynamically estimates the number of topics for each time slice.

Modeling topic chains. The graphical model of *dDTM-News* is illustrated in Figure 1. Similarly to DTM, it divides the dataset into time slices and applies LDA to each of them. The per-topic word distribution evolves with a Gaussian Noise $\mathcal{N}$, as shown in Equation 1. The parameter σ can be tuned for allowing topic variation over two subsequent time slices (e.g., a small value of σ ensures small evolution over two subsequent slices). Differently from DTM, α is fixed and does not depend on the previous time slice, and the number of topics K_i varies with the time slice t_i.

The topics discovered by LDA are chained over the timeline using a Hidden Markov Model (HMM) [23]. HMM is a Markov chain where the states, $S = \{s_1, s_2, ...\}$, are hidden and we only have observations of them, $O = \{o_1, o_2, ...\}$. In HMM there are *state transition probabilities*, $P(s_i|s_{i-1})$, and probabilities of having an observation o_i given the state s_i at the time i, $P(o_i|s_i)$, also called *emission probabilities*. HMMs are very powerful and can be used to represent several problems including estimating the optimal sequence of states given a sequence of observations. In HMMs, inference is done using a Forward-Backward algorithm. More formally, the probability of the state s_i given n observations is proportional to the joint probability as a function of s_i. Such joint probability can be computed as the product between the forward and backward probabilities:

$$P(s_i|O_{1:n}) \underset{s_i}{\propto} P(s_i, O_{1:n}) = P(s_i, O_{1:i}) \cdot P(O_{i+1:n}|s_i) \quad (3)$$

The forward probability, $P(s_i, O_{1:i})$, is the joint probability of the state s_i and the sequence of observations till time i ($O_{1:i}$). The backward probability, $P(O_{i+1:n}|s_i)$, is the probability of the future

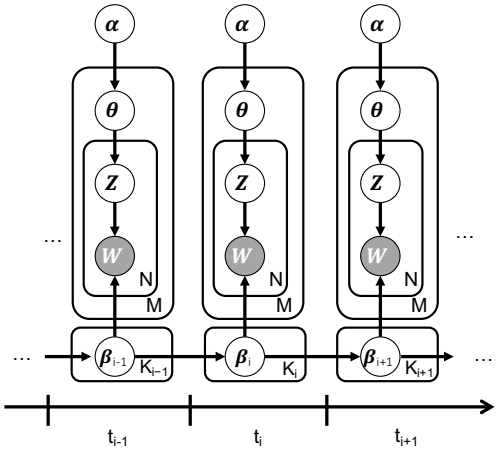


Figure 1: Graphical model of *dDTM-News*.

observations ($O_{i+1:n}$) given the state s_i. In an HMM, the Markov chain can be ergodic (in which every state can be reached from any other state in one single step) or left-right where at every time step the state index increases. We map our problem to an HMM based on a left-right Markov chain. We assume that the topics discovered by LDA are known (they are the HMM's observations) and we want to estimate the probability of transition from one topic to another (the sequence of states). For solving this problem, we apply a Viterbi algorithm [29] to find the sequence of states which maximizes the probability $p(S_{1:n}|O_{1:n})$. First, we need to know the parameters of the HMM which are: (i) the probability of the first state, $P(s_1)$, (ii) the transition probabilities, $P(s_i|s_{i-1})$, and (iii) the emission probabilities, $P(o_i|s_i)$. These parameters can be estimated with a Baum-Welch algorithm [3]. Thus, our approach for chaining topics can be summarized in two steps:

(1) Estimating the parameters of the HMM using a Baum-Welch algorithm. It 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 (i.e., $P(s_1)$, $P(s_i|s_{i-1})$, and $P(o_i|s_i)$) [27].

(2) Given a sequence of observations, applying the Viterbi algorithm to find the most likely sequence of states:

$$S^* = \underset{S}{argmax}\{p(S_{1:n}|O_{1:n})\} \tag{4}$$

Such maximization is done by recursion:

$$(1)\ \mu_1(s_1) = P(s_1, o_1) = P(s_1)P(o_1|s_1)$$

$$(2)\ \mu_i(s_i) = \underset{s_{i-1}}{max}[P(o_i|s_i)P(s_i|s_{i-1})\mu_{i-1}(s_{i-1})] \quad for\ i > 1$$

Optimal number of topics. In news streams where the latent topics are triggered by external events, one time slice is likely to have more or less topics compared to another one. Hence, the optimal number of topics over different time slices may vary largely depending on the number of newsworthy events that happened across the globe in the corresponding period of time. Following the work by Griffiths and Steyvers for modeling scientific topics [11], we estimate the number of topics that best represent the corpus. Given the time slice t_i, we create different LDA models by keeping α and β fixed and assigning different values to the number of topics. Then, we apply a Bayesian model selection to find out the best value of K_i. Model selection consists in computing the posterior probability of the set of models and integrating over all parameters in the model. In our case we need to compute the likelihood of $P(\mathbf{w}|K)$ where $\mathbf{w}$ represents the words in the vocabulary.

Optimal number of topic chains. The number of states in the HMM represents the number of topic chains to discover, and it is an unknown parameter that needs to be estimated. To determine the optimal value of the HMM's states we use the Bayesian Information Criterion (BIC) [26]. It is a well-known technique used for selecting the best model among a finite set of models. BIC is defined as: $BIC = -2 \cdot ln(L) + k \cdot ln(n)$, where n is the number of data points (inputs), k is the number of parameters to be estimated and L is the maximum value of likelihood of the model. Adding more hidden states to an HMM would lead to a higher likelihood and thus a better fit to the data, but after a certain point this could result in overfitting. Hence, BIC increases a penalty for each added state and 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 the 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.

Clustering news. After having detected and chained the events, we can cluster the news documents based on the events they describe. Document clustering and topic modeling are similar since they both allow the automatic organization of huge collections of documents. Topic modeling can be used as a dimensionality reduction technique to map the documents in a topic space and cluster them based on their closeness to the detected topics [17].

While *soft clustering* allows to have the same document assigned to different clusters, *hard clustering* assumes that a document can be assigned to only one cluster (in our case, the documents can cover only one topic). Since we deal with news documents which tend to cover one topic (event) at a time, we performed hard clustering by finding the most likely topic described by the document and assigning it to the corresponding cluster. More formally, a topic model represents each document as a distribution vector of topics, $\boldsymbol{\theta}$, and the document d can be assigned to the cluster k_i, if $k_i = argmax_j(\theta_j)$. In Section 5.3 (Table 6) we will show some examples of news documents belonging to different event clusters obtained with this approach.

4 LINKING AND RANKING NEWS STREAMS

After having detected and tracked topics (events) from news streams and having clustered the news documents based on the events they describe, we can leverage these clusters to cross-link news streams reporting the same event and then rank them based on their timeliness in publishing news.

Linking news streams. We assign to each news document a feature vector $\{ts, e, c, s\}$, where ts is the timestamp, e the event described in the news, c the event chain to which e belongs, and s the stream which published the news. Given this vector we can link news documents reporting the same event which are published

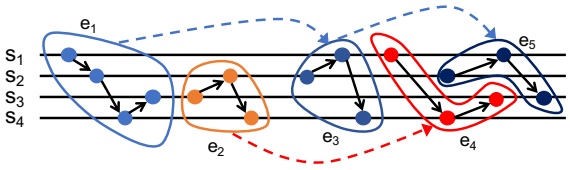


Figure 2: Cross-linking news streams: News documents (nodes) belonging to the same cluster report the same event and are linked across news streams with solid arrows representing the temporal publishing ordering. The events (clusters of nodes) are connected by dashed arrows showing the event evolution.

by different streams. We can also link events that are connected over time and belong to the same topic chain, as shown in Figure 2 by the dashed arrows. The picture depicts four streams, and each node represents a news document. News belonging to the same cluster describe the same event and are connected by direct links according to their temporal publishing ordering. 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 the news about e_i and s_j did it before s_k. Note that, for simplicity, we only show news about a same event which are reported by different streams. This is because we are interested in linking the news documents *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. For us a stream is made of news documents published by a newswire on a platform, and this allows us to analyze temporal dependencies not only among different newswires but also among the different platforms.

Mining frequent sequential patterns. The problem of mining sequential patterns was originally introduced in [1] for studying customer transactions. An itemset is a non-empty set of items, and an itemset sequence is defined as an ordered list of these itemsets. More formally, let $\mathcal{A} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$ be a set of items and $s = \langle \mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n \rangle$ denote an itemset sequence, s is a *subsequence* of another itemset sequence s' if exist 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. Given a support threshold *minSup*, the itemset sequence is called *frequent sequential pattern* if its $sup_{\mathcal{S}}(s_j) \geq minSup$. Mining frequent sequential patterns consists in finding all the frequent sequences in $\mathcal{S}$ for a *minSup*.

As we have seen before, for each news document we have a publishing stream, s, and a timestamp, ts. News are sorted by their timestamps and can come from different news streams. Note that the pair (s, ts) is unique, which means that a stream (channel publishing on a platform) cannot publish more than one news document at the same time, but it is possible to have news with the same timestamp coming from different streams. We follow [12] and convert the time-ordered set of news into a sequence of publishing streams. Since only news documents from different streams may have the same timestamp, this conversion does not imply any loss of information. Moreover, since timestamps have resolution at the level of seconds, it is very unlikely that two streams would report an event exactly at the same time. Hence, in order to determine the temporal dependencies, we use a quantization-step parameter Q_t (e.g., 15 minutes). It introduces a time tolerance, so that if two or more channels have published news about the same event almost at the same time we can treat them as they have reported the event together. For example, the sequence $\langle (\text{abc}), (\text{bbc cnn}), (\text{cbc}) \rangle$ means that the event was reported by four channels: abc reported it before everybody else, followed by bbc and cnn, which reported the event at about the same time, and lastly by cbc. In Section 5.4 we will show how using the PrefixSpan algorithm [21] we can mine temporal sequential patterns among the different streams and discover which stream reports the event before the others.

Ranking news streams. Mining frequent temporal patterns gives us only an indication on which news stream publishes before the others, but it does not give any quantitative and cumulative measure on how promptly the stream publishes news. For ranking news streams based on their timeliness we propose two different scoring functions. The first one depends on the channel position in the time-ordered list of channels and the second one depends on the time distance with respect to the first news. More formally, we can 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 we organize the channels in a list sorted by their first timestamps, $l_e : \langle (s_1, ts_1), (s_2, ts_2), \ldots, (s_n, ts_n) \rangle$, where ts_i is the timestamp of the first news about e published by s_i.

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

$$timeliness(s) = 1 - \frac{\sum_{e \in E_s} delay(s, e)}{|E_s|} \tag{5}$$

where $delay(s, e)$ is the latency of s in reporting e. The lower is this value the more the stream is fast in reporting the news and its $timeliness(s)$ would be high. We propose two definitions for $delay(s, e)$, inspired by the work on the early-adopter graph for news recommendation [18]. The first one considers the *relative position* of s in the list l_e:

$$delay(s, e) = RP(s, e) = \frac{|pred(s, e)|}{|l_e|} \tag{6}$$

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:

$$delay(s, e) = TD(s, e) = \frac{ts(s, e) - ts_{first}(e)}{ts_{last}(e) - ts_{first}(e)} \tag{7}$$

where $ts(s, e)$ is the publishing timestamp for the event e by s, while ts_{first} and ts_{last} are the publishing timestamps of the first and last news about e, respectively. Lower values of $(ts_{last}(e) - ts_{first}(e))$

represent quick events for which the attention decreases fast, while higher values are typical of popular events which span a longer period of time.

5 EXPERIMENTAL RESULTS

5.1 Dataset

We created a dataset[1] consisting of news documents collected from 9 channels (listed in Table 1) and 3 platforms (Twitter, RSS news portals, and news websites) for a period of 4 months (from March 1 to June 30, 2016). Statistics about the dataset are reported in Table 1. News documents have a title (optional), content, link (optional), publishing timestamp, and source channel. The fields title and link are optional as they are not always available in tweets. Note that the time when a news document has been published reflects a different time zone, so we converted all the publishing timestamps to UTC.

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

News Labeling. We labelled news documents at the level of fine-granularity events to evaluate the performance of the clustering. The approaches proposed in the literature for clustering news documents are often evaluated by manually selecting keywords representing some popular events which happened in the period of interest. The events are, for example, found in Wikipedia and manually-picked keywords are used as queries to retrieve the documents [20]. In this way, the evaluation may be biased by the manual selection of events and keywords. To avoid this, we mined most frequent *event keywords* from news streams using the approach presented in [19]. In Table 2 we report some examples of labeled events with the corresponding date. We manually checked from Wikipedia the event veracity (i.e., whether they really occurred) and when they happened, then we retrieved the news documents that are potentially-relevant to them by computing their cosine similarity with the event keywords. The documents are filtered by time to make sure to have only the ones published in the temporal window of the event, and we evaluated the news documents based on their *aboutness* to an event using crowdsourcing. Given the large number of documents, we randomly selected a subset of 60 events and used CrowdFlower[2] to collect human judgements for the pairs {event, news document}. The former is represented by the event keywords and the date of the event, while the latter is a potentially-relevant document (i.e., a tweet, an RSS feed item, or a news article) with its timestamp (i.e., when it was published). The task consisted in showing the pair {event, news document} to the evaluators that were asked to determine whether the news document was about the event or not. If unsure about the answer, they could select "I can't decide" and move to the next pair.

[1]Please contact the first author for the dataset
[2]http://www.crowdflower.com/

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

To guarantee high quality results, we created 200 *gold questions* (i.e., questions with known answers) which were randomly shown during the evaluation to detect low-quality answers, sloppy workers, and malicious activities (e.g., spam). Some gold questions were also used as a quiz for the training phase. The evaluators had to pass the training phase and successfully complete at least 3 out of 5 gold questions. Once the evaluation started, the evaluators 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 collection. We collected relevance judgements for 4.3K pairs {event, news}. For each of them we gathered 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 Fleiss' Kappa [8]. 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 [14], corresponds to 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. It is defined as the level of agreement between multiple contributors weighted by the contributors' trust scores. For our task, we registered an average confidence of 0.91.

5.2 Event Detection and Tracking

In this subsection we present the experimental results on detecting the events and chaining them over the timeline. We compare our *dDTM-News* approach against the original version of DTM [5] and other event-detection approaches.

Event Detection. We detected the events with the dynamic-topic-modeling approaches (DTM and *dDTM-News*) and with two other techniques proposed in the literature for event detection: *Unigram Co-Occurrences* [9] and Event Detection with Clustering of Wavelet-based signals (*EDCoW*) [30]. The former finds the events by analyzing the co-occurrences of bursty features (unigrams) in non-overlapping time windows. The latter creates a signal for each

Table 3: Events detected in the different time windows. Due to the space limit we show only one event (if present) per window. 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	–

individual word, then it applies *wavelet transformation* and *auto-correlation* to measure the bursty "energy" of the words. Words with high energies are used as event features and the similarity between pairs of events is measured using *cross-correlation*. We set the time windows to 7 days for all the approaches.

Some examples of events detected by our approach and by *Unigram Co-Occurrences* and *EDCoW* are reported in Table 3. We show one event (if detected) for each window. As we can see, *dDTM-News* was able to discover more events compared to *EDCoW* which found no events in some of the time windows. Moreover, *EDCoW* uses cross-correlation as similarity measure which can lead to broad events that actually consists of several distinct events happened in the same period of time just by chance. Also, we could observe that *Unigram Co-Occurrences* produces keywords too difficult to interpret without a manual inspection of the news in the dataset. For the lack of space we do not show the events detected by the original version of DTM which performs similarly to our model in term of keyword clarity.

To quantify the performance of the different approaches, we computed the *event coverage* with respect to the pool of 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 *dDTM-News* detects more events compared to the other approaches. Furthermore, 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.

Event Tracking. We now report the performance of DTM and *dDTM-News* on chaining related events over the timeline. Note that we could not chain events with the other two approaches used for event detection (i.e., *Unigram Co-Occurrences* and *EDCoW*) since they only discover events and do not connect them over the timeline. In Table 4 we report some of the event chains created by our model. These chains were computed with $\alpha = 0.01$, which was determined empirically, while the other parameters (β, K, and the number of event chains) are estimated automatically by the 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 which allows to properly capture real-world events in weekly temporal windows.

As we can see from Table 4, our *dDTM-News* approach is able to discover and chain some popular events, for example, related to Brexit ($c = 31$), Zika ($c = 66$), terror attacks in Belgium ($c = 80$), and multiple launch attempts of nuclear missiles from North Korea ($c = 4$). We could also 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 preferred to involve 10 people among PhDs and PhD students from different universities and research institutes that are familiar with the concepts of event tracking and topic modeling. 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 represent a good fit, and fp (false

Table 4: 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

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 [14]. Values of coherence and informativeness are reported in Table 5. As we can see, both models achieve high level of coherence (> 0.7) but our model has a larger fraction of informative chains.

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 5). This confirms how our approach can detect novel events in a more timely manner by relaxing the assumptions of DTM (e.g., a topic is present over all the time slices).

As last experiment, we followed the approach used in [5] for evaluating the topic-prediction performance. We trained the two models on the first t (e.g., $t = 5$) time slices, then we 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 from all the channels 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 5, 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 can timely adjust its topic prediction, while DTM tends to be more conservative and predicts topics that have become obsolete.

Table 5: 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 in predicting topics of the next time slice 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**

5.3 News Clustering

After having detected the events with *dDTM-News*, we can cross-link the news streams based on the events they reported. To do so, we need to create the event-based clusters of news documents, as described in Section 3 (Clustering news), and then link the news as shown in Section 4. We compared our clustering against the ones obtained with *k-means*, *k-means+time*, and another topic model approach, called *hierarchical LDA* (*hLDA*) [10].

k-means is an unsupervised clustering algorithm which is commonly used for hard clustering (i.e., each document is member of exactly one cluster). *k-means* clusters together similar documents by applying the cosine similarity as distance measure. 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. Moreover, *k-means* does not consider the timestamps of the news documents, so we implemented a time-aware clustering approach called *k-means+time* which applies *k-means* and then filters out the news documents outside the window of the event of interest.

hLDA allows to create a hierarchy of topics and has been used for text clustering [16]. Similarly to our model, *hLDA* has the advantage that one does not need to specify the number of topics because, using a relatively high number of levels, the model automatically discovers it. We constructed the hierarchical topic model using different depths. We could notice that with depth 6 and more, *hLDA* returns the same number of topics returned with depth 5. We also observed that the best trade-off between number of events and the size of the news clusters can be achieved with 5 because with lower values the sets are too big and group news documents describing different events. For these reasons, we set the number of levels to 5. As for the other DTM approaches, we used the *hLDA*'s bottom-level topics and clustered the corresponding news documents based on the allocation of the documents in the topic space.

Table 6: 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

Clustering performance. For evaluating the clustering performance there are internal and external criteria. The former are based on intra- and inter-cluster similarities and used for unlabelled clustering. The latter can be applied when clusters have labels as in our case. We used the news labeled as described in Section 5.1 (News Labeling) and computed the *recall*, *precision*, and *F-Score*. Given the cluster of news documents corresponding to the event e, the *recall* and *precision* are defined as follows:

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

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

and they 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*. To quantifies the trade-off between precision and recall we used the macro-average *F-Score* = $2 \cdot \frac{P \cdot R}{P+R}$.

Table 7: Comparison of the different 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

Table 7 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 other approaches.

Table 6 reports some examples of news documents for some of the event clusters created by our approach. As we can see, the documents are coherent and well representative 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. Nevertheless, for *dDTM-News* we could observe many true positives and only few false positives, as corroborated by the high value of precision. On the other hand, *k-means* and *hLDA* have low precisions due to many false positives. We manually inspected the news clusters to see what caused them and could see that these two approaches did not separate news documents about distinct events that present similar keywords. For example, they mixed the news about the earthquake in Japan happened around mid April 2016 with the ones about the Japan tsunami's 5th anniversary. Documents about wildfires in Canada were together with those about wildfires in California. Also, news about the singer Alejandro Fuentes who was shot in Chicago were mixed with the ones about Christina Grimmie that was shot during her concert in Orlando.

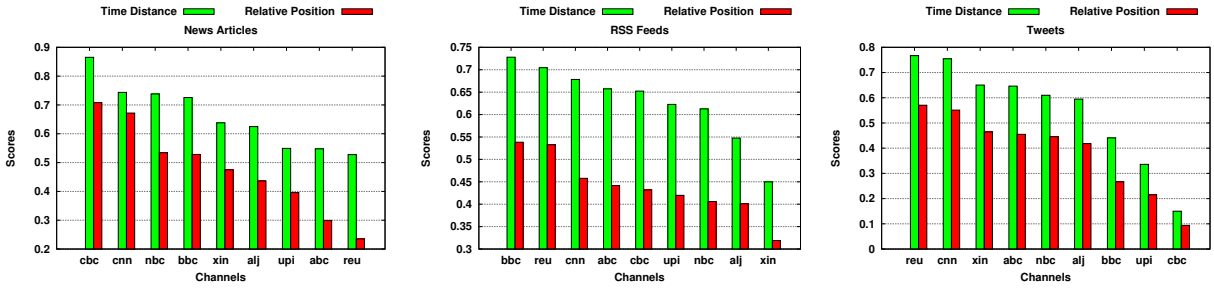
5.4 Timeliness Analysis

We now present the results on mining temporal patterns and ranking news streams based on their timeliness in reporting real-world events. We used PrefixSpan [21] for mining the frequent temporal patterns. Some patterns, obtained with *minSup* = 4, are reported in Table 8. We could observe that `reu` reported the news before `abc` and `xin`. Also, `bbc` and `cnn` preceded `abc`, `nbc`, and `xin`. Other interesting patterns concern how the channels used the 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. We will see that these patterns are confirmed by the ranking of the channels shown in Figure 3.

Table 8: 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

Concerning the timeliness of the news streams in reporting the events, Figure 3 shows the rankings of the streams based on the scores described in Section 4 (Ranking news streams). As we can see, the two scoring functions (relative position and time distance)



(a) News Articles (b) RSS Feeds (c) Tweets

Figure 3: Channels' 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.

behave more or less in the same way. The relative position is less informative because if the events are promptly reported by all the channels, the differences between the first and second positions or the second and third ones are insignificant and what matters is the time distance. From the rankings we observed that cbc and cnn published the news very quick on their websites and then spread them with RSS feeds and tweets. This is due to the fact that these channels used RSS platforms and Twitter for advertising their news articles (i.e., the RSS feed items and tweets oftentimes have links to the articles of their official websites). Reuters reported the events in Twitter in a very timely way and was followed by cnn and xin. The bbc published RSS feeds and news articles in a more timely way compared to tweets. Finally, for alj and upi we registered similar scores for all the platforms.

We observed that some of the events were not reported by all the channels using all the platforms, so we computed the fraction of events reported by the newswires using the different platforms in each time slice. For example, if 35 events are detected in a time slice and the channel reports only 30 of them, the fraction of reported events for that channel in that time slice is $30/35 = 0.85$. Figure 4 shows the fraction of reported events averaged over all the time slices. As we can see, 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.

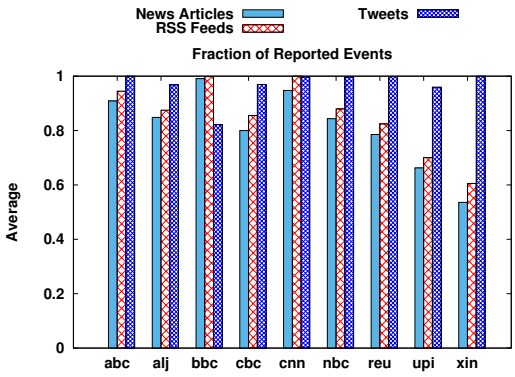


Figure 4: Fraction of events reported by different channels on different platforms averaged over all the time slices.

6 RELATED WORK

Topic Models. Topic models (e.g., pLSI, LDA) exploit word co-occurrences to capture the semantic relatedness of words and can effectively discover latent topics from large document collections. LDA is based on the assumption that all documents are exchangeable in the entire collection (i.e., the probability of documents in the dataset is invariant with respect to permutation). This holds for non-sequential collections of documents but not for streams, where the temporal information is very important and can be leveraged for better detecting the topics and for tracking their evolution over time. To address this limitation, Blei and Lafferty proposed Dynamic Topic Model (DTM) [5], while in [2] we presented a discretized version of it for dealing with non-linear evolutions of topics.

Event detection. Event detection and tracking is based on monitoring a stream of news and automatically organizing them by the events they describe. Research has mostly focused on detecting events from news articles and tweets, separately. Regarding news articles, Fung et al. [9] extracted features by statistically modeling the frequency of each unigram with a binomial distribution. Then, the events are detected by maximizing the co-occurrences of bursty-frequency words among the documents.

The traditional approaches developed for formal text (e.g., news articles) cannot be applied to tweets since these are short, noisy, and published at a high-speed rate. In Twitter, a first challenge consists in discriminating newsworthy events from trends or mundane events which can attract a large number of users but are not of general interests [4]. For our research we do not need to do such a discrimination, because we only monitor news channels so their tweets are all newsworthy. Popescu et al. [22] used an entity-based event-detection approach where a set of tweets containing the predefined target entity are processed and machine learning techniques are used to predict whether the tweets constitute an event regarding the entity. Many works, such as [25], detect crime or disasters from tweets, but the events must be known a priori and should be easily represented by keywords (e.g., "earthquake" or "shaking"). Ritter et al. [24] overcame this limitation by designing

an open-domain system for extracting a calendar of events and categorize them. Similarly, Twevent [15] clusters tweets representing an event, plus it provides a semantic-meaningful description of the clusters. Other works treated features as signals and applied wavelet analysis [30].

Analysis of news streams. Past research has focused on one single stream of news, and only a few works have taken into account multiple streams [7, 12, 13]. Del Corso et al. [7] proposed techniques for ranking news stories and news sources based on their decay of freshness and on priority ranking. Gwadera and Crestani [12] ranked the streams of RSS feeds using cross-stream sequential patterns and content similarity. Their approach monitored news feeds coming from different channels and applied a simple stream-clustering technique to divide the stories based on the events. Differently from [12], we use heterogenous sources (i.e., not only RSS feeds but also tweets and news articles). We also introduce a more sophisticated approach for discovering events based on dynamic topic modeling which makes possible to describe the events with semantic-meaningful topic keywords and also to show the event evolution over time. In [13] the authors proposed a unified framework for modeling temporal dynamics of multiple text streams. Their correlated-text-stream model captures both local and shared topics from multiple streams, and it is integrated with a temporal model which assigns to each topic a time-dependent function representing the topic's popularity over time.

Another line of research has studied the timeliness of different social media in reporting newsworthy events. In [20], Osborne and Dredze studied coverage and latency of Facebook, Google Plus, and Twitter. The authors used a list of events manually selected from Wikipedia to discriminate posts reporting major events, and they observed that Twitter usually dominates other social media but lags newswires. An orthogonal line of research have addressed the problem of linking news to social media posts, for example, using keywords extracted from news to find social media utterances that implicitly refer to the news articles [28].

7 CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach for cross-linking multiple and heterogenous news streams based on the events they reported. We first proposed an approach based on dynamic topic modeling for finding the events and tracking their evolution over time. Then, we analyzed the temporal publishing patterns of news streams and presented two functions for ranking streams based on their timeliness in reporting the events.

As future work, we would like to include the trustworthiness of the news sources in our score definitions for ranking the news streams. This would be important especially for Twitter, since tweets may be the result of retweeting coming from untrusted sources of information. A further investigation would be to use the event chains for news summarization and link recommendation.

ACKNOWLEDGMENTS

This research was partially funded by the the Secrétariat d'Etat à la formation, à la recherche et à l'innovation (SEFRI) under the project AHTOM (Asynchronous and Heterogeneous TOpic Mining). We would like to thank the anonymous reviewers for their comments.

REFERENCES

[1] R. Agrawal and R. Srikant. 1995. Mining Sequential Patterns. In *ICDE '95*. IEEE Computer Society, Washington, DC, USA, 3–14.
[2] S. A. Bahrainian, I. Mele, and F. Crestani. 2017. Modeling Discrete Dynamic Topics. In *SAC '17*. ACM, New York, NY, USA, 858–865.
[3] L. E. Baum and J. A. Eagon. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.* 73, 3 (1967), 360–363.
[4] H. Becker, M. Naaman, and L. Gravano. 2011. Beyond Trending Topics: Real-World Event Identification on Twitter. In *5th Int. AAAI Conference on Weblogs and Social Media*. 438–441.
[5] D. M. Blei and J. D. Lafferty. 2006. Dynamic Topic Models. In *ICML'06*. ACM, New York, NY, USA, 113–120.
[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022.
[7] G. M. Del Corso, A. Gullí, and F. Romani. 2005. Ranking a Stream of News. In *WWW '05*. ACM, New York, NY, USA, 97–106.
[8] J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76, 5 (1971), 378–382.
[9] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. 2005. Parameter Free Bursty Events Detection in Text Streams. In *VLDB '05*. VLDB Endowment, 181–192.
[10] D. M. Griffiths and M. I. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems* 16 (2004), 17.
[11] T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101, suppl 1 (2004), 5228–5235.
[12] R. Gwadera and F. Crestani. 2009. Mining and Ranking Streams of News Stories Using Cross-stream Sequential Patterns. In *CIKM'09*. ACM, New York, NY, USA, 1709–1712.
[13] L. Hong, B. Dom, S. Gurumurthy, and K. Tsioutsiouliklis. 2011. A Time-dependent Topic Model for Multiple Text Streams. In *KDD '11*. ACM, New York, NY, USA, 832–840.
[14] J. R. Landis and G. G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174.
[15] C. Li, A. Sun, and A. Datta. 2012. Twevent: Segment-based Event Detection from Tweets. In *CIKM '12*. ACM, New York, NY, USA, 155–164.
[16] P. Liu, L. Li, W. Heng, and B. Wang. 2012. HLDA based Text Clustering. In *IEEE 2nd Int. Conf. on Cloud Computing and Intelligence Systems*, Vol. 3. IEEE Computer Society, Washington, DC, USA, 1465–1469.
[17] Y. Lu, Q. Mei, and C. Zhai. 2011. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval* 14, 2 (2011), 178–203.
[18] I. Mele, F. Bonchi, and A. Gionis. 2012. The early-adopter graph and its application to web-page recommendation. In *CIKM '12*. ACM, New York, NY, USA, 1682–1686.
[19] I. Mele and F. Crestani. 2017. Event Detection for Heterogeneous News Streams. In *NLDB '17*. Springer International Publishing, Cham, 110–123.
[20] M. Osborne and M. Dredze. 2014. Facebook, Twitter and Google Plus for Breaking News: Is There a Winner?. In *18th Int. AAAI Conference on Weblogs and Social Media*.
[21] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. 2001. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *Proc. of the 17th Int. Conf. on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 215–224.
[22] A. Popescu, M. Pennacchiotti, and D. Paranjpe. 2011. Extracting Events and Event Descriptions from Twitter. In *WWW '11*. ACM, New York, NY, USA, 105–106.
[23] L. R. Rabiner. 1990. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Readings in Speech Recognition*, Alex Waibel and Kai-Fu Lee (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 267–296.
[24] A. Ritter, Mausam, O. Etzioni, and S. Clark. 2012. Open Domain Event Extraction from Twitter. In *KDD '12*. ACM, New York, NY, USA, 1104–1112.
[25] T. Sakaki, M. Okazaki, and Y. Matsuo. 2010. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *WWW '10*. ACM, New York, NY, USA, 851–860.
[26] G. Schwarz. 1978. Estimating the Dimension of a Model. *The Annals of Statistics* 6, 2 (1978), 461–464.
[27] P. Smyth, D. Heckerman, and M. I. Jordan. 1997. Probabilistic Independence Networks for Hidden Markov Probability Models. *Neural Comput.* 9, 2 (Feb. 1997), 227–269.
[28] M. Tsagkias, M. de Rijke, and W. Weerkamp. 2011. Linking Online News and Social Media. In *WSDM '11*. ACM, New York, NY, USA, 565–574.
[29] A. Viterbi. 2006. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Trans. Inf. Theor.* 13, 2 (Sep 2006), 260–269.
[30] J. Weng and B. Lee. 2011. Event Detection in Twitter. In *5th Int. AAAI Conference on Weblogs and Social Media*. 401–408.