Information Systems 78 (2018) 68–80

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Invited Reproducibility paper

Discovering and learning sensational episodes of news events

Xiang Ao[a,*], Ping Luo[a], Chengkai Li[b], Fuzhen Zhuang[a], Qing He[a]

[a] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology,CAS, Beijing, China
[b] University of Texas at Arlington, USA

ARTICLE INFO

Article history:
Received 16 August 2016
Revised 14 October 2017
Accepted 15 May 2018
Available online 17 May 2018

ABSTRACT

In this paper, we study the problem of discovering and learning sensational episodes of news events. A sensational episode of news events is in the form of *lhs* → *rhs*, where *lhs* is an antecedent event, *rhs* is a consequent event, and *rhs* often happens shortly after *lhs*. Such pairs of co-occurring news events within short period, while not necessarily bearing causal relationship between each other, are possible essential to media since they deliberately seek and broadcast examples of uncommon events to fascinate crowd attentions. First, to find all frequent episodes, we propose an efficient algorithm, MEELO, which significantly outperforms conventional algorithms. There can be a large number of frequent episodes. We rank them by their sensational effect from the perspectives of news audience, through learning from manually labeled examples. Instead of limiting ourselves to any individual subjective measure of sensational effect, we utilize a learning-to-rank approach that exploits multiple features to capture the sensational effect of a news episode from various aspects. NLP tools combined with knowledge bases are used in extracting and aggregating news events from news text. Experiments on real data verify our approach's efficiency and effectiveness.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

Media, including news media, self media or gossip media, are skilled in utilizing availability heuristic [17]. One form of these is that they deliberately discover examples of uncommon "correlated" events, which may burst hot topics and wide spread, as hyping materials. What they will get is public's attentions and click rates. Two prominent examples in recent years are known as the Taiwanese pop star Jam Hsiao who deserves his nickname of "God of Rain" [1] and the "Ramsey's Curse" [2] in soccer circles. Hsiao's nickname was created as it heavily rained whenever and wherever Hsiao appeared at a public event. Ramsey was raised concern was because when he scored in a match, an international figure in politics or culture died within 21–72 h. Both of them gained unprecedented sensation in social networks.

Such example pairs of co-occurring news events, while not necessarily bearing causal relationship between each other, could be essential to media reporters for their potential value and sensation. We call such a pair of events a *sensational episode*. Specifically, a sensational episode is in the form of *lhs* → *rhs*, where *lhs* is an *antecedent event* and *rhs* is a *consequent event*. It suggests that *lhs* often happens before *rhs* in the real world. *Frequent episode mining* has been extensively studied [3,6,8,15,18,21,26–29,31,33,40,41,44]. The general notion of an episode (also known as *serial episode*) refers to a totally ordered set of events. The subject of this paper, sensational episode, refers to 2-episode [44]—a serial episode consisting of 2 events. We believe episodes in this simplest form are of particular interests among all episodes, as longer episodes may easily become convoluted.

Manually discovering sensational episodes is an undoubtedly difficult task since there are too many events can be combined as sequential pairs and only few of them bear sensation. In this paper, we try to discover and learn to rank sensational episodes of news events. This is accomplished by two steps. The first task is to find all *frequent 2-episodes* whose co-occurring frequencies are above a threshold. For this, we designed an efficient algorithm—MEELO. The frequency of an episode is the number of its minimal occurrences, i.e., those occurrences that do not subsume other occurrences (within shorter period) of the same episode. Conventional algorithms for frequent episode mining need to check, by post-processing, whether a detected occurrence is minimal or not. MEELO, instead, directly finds all minimal episode occurrences in two passes of sequential scan of an event sequence. This particular approach is enabled by a special data structure for bookkeeping the last occurrences of events in the latest time window. Our

* Corresponding author.
E-mail addresses: aoxiang@ict.ac.cn (X. Ao), luop@ict.ac.cn (P. Luo), cli@uta.edu (C. Li), zhuangfuzhen@ict.ac.cn (F. Zhuang), heqing@ict.ac.cn (Q. He).
[1] http://usa.chinadaily.com.cn/china/2013-08/21/content_16912171.htm
[2] http://www.youtube.com/watch?v=IFHaDDhYO0w

https://doi.org/10.1016/j.is.2018.05.003
0306-4379/© 2018 Elsevier Ltd. All rights reserved.

experimental results show that MEELO outperforms conventional methods on execution time in finding frequent 2-episodes.

There can be a large number of frequent 2-episodes of news events that satisfy a given frequency threshold. The second task is to rank all such candidate episodes by their *sensational effect* from the perspective of news audience. Instead of limiting ourselves to any individual subjective measure of sensational effect, we develop a *learning-to-rank* [25] approach that exploits multiple features to capture the sensational effect of a 2-episode from different aspects. In addition to traditional features such as support and Granger value [14], we designed novel features to capture bombshell value of news events and semantic relationships between entities based on a knowledge base. Given a set of manually labeled frequent 2-episodes and their features, we trained a random forests ranker [13]. Note that several works on frequent episode mining proposed episode interestingness measures such as *confidence* [18,28] and *utility* [42]. However, our learning-to-rank approach distinguishes itself by two ways: (1) It ranks episodes based on manually labeled examples from the perspective of news audience; (2) It combines multiple features instead of limiting itself to an individual measure.

Prior to discovering and learning sensational episodes of news events, we extract news events from news text (Tweets from news agency, to be specific). Natural Language Processing (NLP) tools, lexical database of English, as well as knowledge base built on Wikipedia concepts are exploited for event extraction. The reason we use Twitter as the source of news text here is because previous study [34] shows that Twitter reports the same events as newswire providers, and meanwhile it also reports minor events ignored by mainstream media. It means there might be more possible to discover potential sensational news episodes in news events posted in tweets.

Though our solution does not request episodes necessarily bearing causal relationship, this route can also find some news patterns with implicit causality in our experiments. Hence, we empirically demonstrate the proposed solution might also be helpful in peeping deep causation of news events.

Our previous work [3] roughly introduced the extended abstract for discovering sensational episodes of new events. In this paper, we detail every part of the solution especially on discovering frequent 2-episode and measuring episode sensational effect via a learning-to-rank method. In addition, we investigate the solution's effectiveness and efficiency by extensive experiments. In summary, this paper makes the following contributions:

- We developed a novel and efficient 2-episode mining algorithm, MEELO, which significantly outperforms conventional algorithms for frequent episode mining. We also formally prove the correctness and completeness of MEELO. (Section 3)
- We developed a learning-to-rank approach that ranks frequent episodes of news events by their sensational effect, based on a set of novel features that are predictive for sensational effect. (Sections 4 and 5)
- We conducted extensive experiments to evaluate the effectiveness and efficiency of our methods on real-world data set. The experiments on a 14-month Twitter data set shows the proposed features are effective for ranking news event episodes by their sensational effect and MEELO significantly outperforms the state-of-the-art frequent episode mining algorithms for discovering frequent 2-episodes. (Section 6)

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we explain the algorithm MEELO. In Sections 4 and 5, we describe the details of features for evaluating sensational effect of episodes and how to rank episodes with the proposed features. Section 6 presents experimental results. Section 7 concludes the paper.

2. Related work

There are several works related to this task including frequent episode mining and news event prediction. Hence we survey these two research fields in what follows.

First, frequent episode mining (FEM for short) on event sequences is an important data mining problem for different forms of data, such as alarm sequences in telecom networks [28], web navigation logs [8,28], time-stamped fault reports in car manufacturing plants [21], Wal-Mart sales transactions [6,42], stock data [2,4,18,31] and so on. According to different applications, various frequency measures are defined to discover different kinds of episodes, e.g., window-based occurrence [18,28], minimal occurrence [27], non-overlap occurrence [20] and so on. Among them, minimal occurrence, as it may unearth some implicit causal relationship, is one of widely used measures [1,26,27,42]. Given a particular frequency definition, FEM algorithms can be categorized into two types, namely breadth-first enumeration (known as apriori-like) methods and depth-first enumeration (known as prefix-growth) methods. The depth-first enumeration methods are fit to use minimal occurrence as frequency definition. However, most descriptions of such algorithms in the literature lack the details on how to detect a minimal occurrence of an expanded episode. To our best understanding, these algorithms usually consider the occurrences of a prefix as independent to each other while expanding to longer episodes. As a consequence, it requires a post-processing step for ensuring a detected occurrence is truly a minimal occurrence. However, MEELO can efficiently identify all the minimal occurrences of episodes by maintaining a compact data structure LO-List. The experiments also show the superiority of MEELO over the other methods in terms of efficiency. Besides, several works in FEM proposed episode interestingness measures, such as confidence [28] and utility [42] to rank the generated frequent episodes. However, because of the complexity of sensational effect for episodes, it is difficult to rank frequent episodes by an individual feature associated with sensational effect. Hence, we propose a learning-to-rank approach that exploits multiple features from different aspects to capture the sensational effect for any 2-episode.

Second, our study is also related to the work of news events detection in Twitter [5,11,23,32,37] and news event prediction based on the causal relationship [16,35,36]. For news event detection, classifiers are usually trained to recognize the target news events based on some NLP and spatiotemporal features. For example, Sakaki et al. [37] detected earthquake event from Twitter via training a classifier of tweets based on the keywords in a tweet, the number of words, and the their context as features. In [5], Aramaki et al. developed a SVM classifier to identify the tweets discussing influenza. Similarly, the system TEDAS [23] is built to monitor crime and disaster events in Twitter, and its key technique is also classifying based approach. Similarly, classifiers are still key technique of news prediction based on the causal relationship not surprisingly. In [35,36], Radinsky et al. extracted the causality pairs by the NLP-based templates, and then learned the classifier for causality predictions based on the proposed NLP-based features. As the case studies, they focused on identifying the events which cause the events of disease outbreaks, deaths, and riots. Hashimoto et al. [16] proposed a supervised method to extract event causalities from the web using semantic relation between nouns, context, and association features. In this study, we focus on the event pairs, not the event itself, with sensational effect. Apparently, it is different with event detection, and sensational we are interested here is a broader relationship than causality. In other words, some episodes with causal relationships can also be identified if they are sensational. Second, we formulate the problem of learning sensational episode of news events as a ranking problem, and solv-

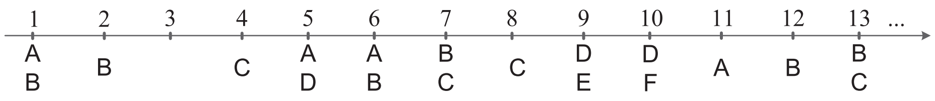


Fig. 1. The running example of event sequence.

ing the problem along the route of learning to rank techniques, which are different with the work we surveyed above. Additionally, the features used to learn the predictor of sensational effect are mainly based on the entity taxonomy and the sensation of event, which are different from these NLP-based features used in news event detection or news event prediction based on the causal relationship.

3. Mining frequent 2-episodes

In this section, we propose an efficient method for mining frequent 2-episodes from a sequence of event occurrences. We start by defining several important concepts and forming our problem statement, and then explain the details of the algorithm.

3.1. Concepts and problem statement

Event: An *event e* is a tuple ⟨*s, v, o*⟩, where *s* denotes a *subject entity, v* denotes a *verb predicate* and *o* denotes an *object entity*. Either *s* or *o* (but not both) can be absent (denoted as NULL). For example, the event representing the soccer player Ramsey scored in game is ⟨ *Aaron_Ramsey, score*, NULL⟩. The set of all distinct events, denoted $\mathcal{E}$, is called the *event set*.

Event Sequence: An *event sequence*, denoted as $\vec{S} = (E_1, T_1), \ldots, (E_n, T_n)$, is an ordered sequence of event sets, where each $E_i \subseteq \mathcal{E}$ consists of all events at time T_i. The event sets are chronologically ordered by their timestamps, i.e., $T_i < T_j$ for all $1 \le i < j \le n$. For example, Fig. 1 shows an event sequence $\vec{S}$ = (AB, 1), (B, 2), (C, 4), (AD, 5), (AB, 6), (BC, 7), (C, 8), (DE, 9), (DF, 10), (A, 11), (B, 12), (BC, 13). This is the running example throughout the paper.

2-Episode: A *2-episode* α consists of two events $e_1 \to e_2$ where $e_1 \ne e_2$, e_1 is the *antecedent event* and e_2 is the *consequent event*.

Occurrence: Given a 2-episode $\alpha = e_1 \to e_2$, a time window [T_1, T_2] is an *occurrence* of α if (1) e_1 occurs at time T_1; (2) e_2 occurs at time T_2; and (3) $0 < T_2 - T_1 < max\_intvl$, where *max_intvl* is a user-specified threshold called the *maximum time interval*. The set of all occurrences of α is denoted occ(α). For example, if $max\_intvl = 3$ in Fig. 1, occ(A → C) = {[5, 7], [6,7], [6,8], [11,13]}. [1,4] is not an occurrence of A → C because the span of the time window is not less than the maximum time interval threshold.

Minimal Occurrence: Consider two time windows [T_1, T_2] and [T_1', T_2']. [T_1', T_2'] is *subsumed* by [T_1, T_2] if $T_1 \le T_1'$ and $T_2' \le T_2$. An occurrence of 2-episode α, [T_1, T_2], is a *minimal occurrence* of α if no other occurrence [T_1', T_2'] of α is subsumed by [T_1, T_2]. We denote the minimal occurrences of α by mocc(α). For the running example mocc(A → C) = {[6, 7], [11, 13]}. Note that [5,7] and [6,8] are not minimal occurrences since they subsume [6,7].

Support: The *support* of a 2-episode α, denoted sp(α), is the number of its *minimal occurrences*, i.e., sp(α) = |mocc(α)|.

Frequent 2-episode: A 2-episode is *frequent* if and only if sp(α) ≥ min_*sup*, i.e., its support is no less than *min_sup*—a user-specified *minimum support threshold*. Otherwise, the episode is *infrequent*.

Problem statement of mining frequent 2-episodes: Given an event sequence $\vec{S}$, a minimum support threshold *min_sup* and a maximum time interval threshold *max_intvl*, our mining problem is to find all frequent 2-episodes in $\vec{S}$.

We emphasize that minimal occurrence, as the frequency measure, is preferred for our problem of mining sensational episodes of new events. Based on its definition there is no another occurrence of a same episode which is subsumed by a minimal episode occurrence. As a consequence, given two events minimal occurrence may demonstrate the most compact correlation among their occurrences in the event sequence. It may be possible that an occurrence time window of a 2-episode is minimal but its time interval is large. Such large time window, though it is the minimal one, could diminish the potential sensation of an episode because it may not be convincing if two events appear too far away from each other. Hence we utilize a maximum time interval threshold to control the maximum distance between the antecedent event and the consequent event of a 2-episode. Such parameter can filter out underlying minimal episode occurrences in which the antecedent event occurrence and the consequent event occurrence is distant. It is under the assumption that the events of a sensational episode candidate may be appeared close to each other without other substitutions.

Table 1
The notations and denotations.

Notations	Denotations
$\vec{S}$	a (frequent) event sequence
n	the length of $\vec{S}$
$\mathcal{L}$	the last occurrence list
q = (*q.event*: *q.time*)	a node in $\mathcal{L}$
C	the candidate episode set
F	the frequent episode set
E_i	the frequent event set occurring at time T_i

3.2. Algorithm MEELO

We propose an efficient algorithm MEELO (Mining frEquent Episode via Last Occurrence list). It finds all frequent 2-episodes in an event sequence $\vec{S} = (E_1, T_1), \ldots, (E_n, T_n)$ by two passes of sequential scan over $\vec{S}$. In the first pass, it scans $\vec{S}$ to obtain the frequencies of all distinct events. It then removes from $\vec{S}$ all occurrences of infrequent events (whose frequencies are less than *min_sup*). For the running example in Fig. 1, when $min\_sup = 3$, the resulting subsequence is shown in Fig. 2 which contains the occurrences of all four frequent events—A, B, C and D.

In the second pass, MEELO finds frequent 2-episodes by another sequential scan over the resulting subsequence from the first pass. It is straightforward that the subsequence suffices for finding all frequent 2-episodes, since infrequent events must not appear in any frequent 2-episode. The rest of this section thus focuses on explaining the second pass, and we will use $\vec{S}$ to denote the subsequence that contains only frequent events hereafter.

For clarity and convenience, the notations and denotations used in presenting the algorithm are summarized in Table 1.

3.2.1. Last occurrence list and its invariants

MEELO relies on a data structure, *last occurrence list* (LO-list for short, denoted $\mathcal{L}$), to find all minimal occurrences of 2-episodes. $\mathcal{L}$ is a doubly linked list. Each node q = *q.event*: *q.time* in $\mathcal{L}$ consists of two fields—*q.event* and *q.time*, which record an event and an occurrence time of the event, respectively. For example, q = A: 1 indicates that an occurrence of event A happened at a time with timestamp 1.

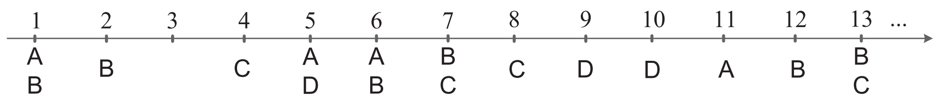


Fig. 2. The frequent event sequence, $min\_sup = 3$.

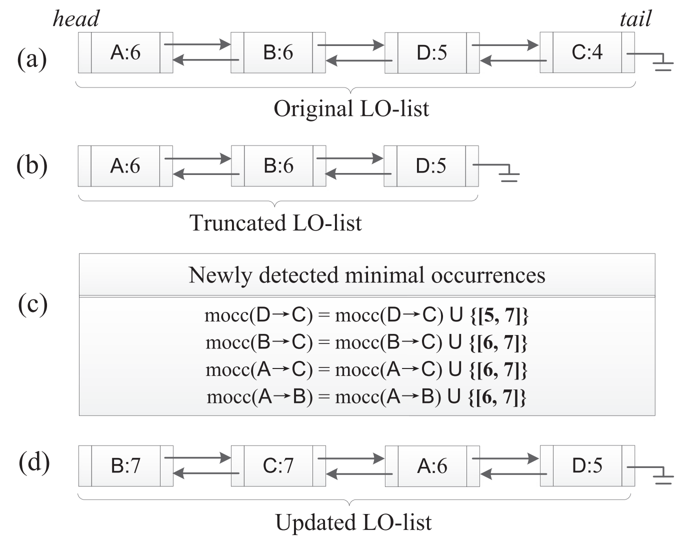

head
tail
(a)
A:6
B:6
D:5
C:4
Original LO-list
(b)
A:6
B:6
D:5
Truncated LO-list
(c)
Newly detected minimal occurrences
mocc(D→C) = mocc(D→C) U {[5, 7]}
mocc(B→C) = mocc(B→C) U {[6, 7]}
mocc(A→C) = mocc(A→C) U {[6, 7]}
mocc(A→B) = mocc(A→B) U {[6, 7]}
(d)
B:7
C:7
A:6
D:5
Updated LO-list

Fig. 3. Maintenance of LO-list by MEELO.

MEELO scans $\vec{S} = (E_1, T_1), \ldots, (E_n, T_n)$ from beginning (T_1) to end (T_n). At every timestamp T_i ($1 \le i \le n$), the following invariants on the content of LO-list $\mathcal{L}$ are guaranteed by the algorithm:

Invariant 1 For every node $q = q.event$: $q.time$ in $\mathcal{L}$, $q.time$ is the time of the latest occurrence of $q.event$ during time window $[T_p, T_i]$, where

$$T_p = \max(T_1, T_i - max\_intvl + 1) \tag{1}$$

Invariant 2 The nodes in $\mathcal{L}$ from head to tail are in reverse chronological order of their timestamps.

For example, at timestamp 6 in Fig. 2, the corresponding LO-list $\mathcal{L}$ is shown in Fig. 3 (a). Since $max\_intvl = 3$, we only consider the occurrences of frequent events in time window [4,6], which are C: 4, A: 5, D: 5, A: 6, B: 6. Note that A occurred twice in this window. Since $\mathcal{L}$ only records the latest occurrence of an event, A: 6 (instead of A: 5) is included into $\mathcal{L}$. From the head of $\mathcal{L}$ to its tail, the nodes are in reverse chronological order of their timestamps—A: 6, B: 6, D: 5, C: 4.

3.2.2. Maintenance of LO-list and identification of minimal occurrences

In order to guarantee the aforementioned invariants on LO-list $\mathcal{L}$ at every timestamp T_i while scanning the sequence $\vec{S}$, MEELO maintains $\mathcal{L}$ and finds the minimal occurrences of episodes. It performs the following three steps at every T_i: 1) *truncating LO-list by maximum time window*, 2) *finding minimal occurrences of 2-episodes*, and 3) *updating LO-list*. The pseudo code of MEELO is shown as Algorithm 1. The three steps are sub-procedures shown in Algorithms 2, 3 and 4, respectively.

Step 1: Truncating LO-list by maximum time window

When the scanning of $\vec{S}$ moves forward from T_{i-1} to T_i, MEELO removes from the LO-list the occurrences of events that are outside of time window $[T_p, T_i]$ (cf. Eq. (1)). In other words, it truncates all such nodes q that $T_i - q.time + 1 > \max\_intvl$. The truncation guarantees that the LO-list only contains event occurrences within the time window ending at the current timestamp T_i. For the example,

Algorithm 1 MEELO Algorithm.

```
Input:
    $\vec{S} = (E_1, T_1), \ldots, (E_n, T_n)$: frequent event sequence
    $\max\_intvl$: maximum time interval
    $min\_sup$: minimum support
Output:
    $F$: frequent 2-episodes

1: initialize an empty LO-list $\mathcal{L}$; candidate 2-episodes $C \leftarrow \emptyset$; frequent 2-episodes $F \leftarrow \emptyset$; $n \leftarrow$ length of $\vec{S}$;
2: for $i = 1$ to $n$ do
3:     $\mathcal{L} \leftarrow$ TruncateL($\mathcal{L}, \max\_intvl, T_i$);
4:     $C \leftarrow$ FindMOCC($\mathcal{L}, E_i$);
5:     $\mathcal{L} \leftarrow$ UpdateL($\mathcal{L}, E_i$);
6: end for
7: $F \leftarrow \{\alpha | \alpha \in C$ and $sp(\alpha) \ge \min\_sup\}$;
8: return $F$;
```

Algorithm 2 TruncateL: Truncating LO-list by Maximum Time Window.

```
Input:
    $\mathcal{L}$: LO-list
    $max\_intvl$: maximum time interval
    $T_i$: current timestamp
Output:
    $\mathcal{L}$: LO-list after truncation

1: $T_p \leftarrow \max(T_1, T_i - max\_intvl + 1)$;
2: $q \leftarrow \mathcal{L}.head$;
3: while $q$ ! = $\mathcal{L}.tail$ do
4:     if $q.time < T_p$ then
5:         $q \uparrow next \leftarrow$ NIL; break;
6:     end if
7:     $q \leftarrow q \uparrow next$;
8: end while
9: return $\mathcal{L}$;
```

Algorithm 3 FindMOCC: finding minimal occurrences.

```
Input:
    $\mathcal{L}$: LO-list
    $E_i$: the events occurred at time $T_i$
Output:
    $C$: candidate episode set

1: for all event $e \in E_i$ do
2:     partition nodes in $\mathcal{L}$ into $S1$ and $S2$ by Equations (2)~– (5);
3:     for all nodes $q \in S1$ in $\mathcal{L}$ do
4:         $\alpha \leftarrow q.event \rightarrow e$;
5:         $mocc(\alpha) \leftarrow mocc(\alpha) \cup \{[q.time, T_i]\}$;
6:         if $\alpha \notin C$ then
7:             $C \leftarrow C \cup \{\alpha\}$;
8:         end if
9:     end for
10: end for
11: retrun $C$;
```

Algorithm 4 UpdateL: updating LO-list.

```
Input:
    $\mathcal{L}$: LO-list
    $E_i$: the events occurred at time $T_i$
Output:
    $\mathcal{L}$: updated LO-list

1: for all event $e \in E_i$ do
2:     if $\mathcal{L}$ contains a $q$ such that $q.event$ = $e$ then
3:         delete $q$ from $\mathcal{L}$;
4:     end if
5:     insert a node $e$:$T_i$ at $\mathcal{L}$.head;
6: end for
7: return $\mathcal{L}$;
```

in Fig. 2, when the scanning reaches timestamp 7, MEELO prunes the last node C: 4, since it gets outside of the current time window [5,7]. After this step, the list becomes Fig. 3 (b).

Step 2: Finding minimal occurrences of 2-episodes

In Step 2, MEELO finds the minimal occurrences of 2-episodes within time window $[T_p, T_i]$, based on $\mathcal{L}$ and the event set E_i at the current timestamp T_i in sequence $\vec{S} = (E_1, T_1), \ldots, (E_n, T_n)$. For each event $e \in E_i$ and node $q \in \mathcal{L}$, a candidate occurrence is $[q.time, T_i]$, for corresponding 2-episode $q.event \rightarrow e$. However, not all such candidates are minimal occurrences. MEELO identifies only minimal occurrences of 2-episodes, as follows.

Given any event $e \in E_i$, MEELO identifies the node q' in $\mathcal{L}$ such that $q'.event = e$. It then partitions other nodes in $\mathcal{L}$ into the following two sets according to their relative positions to q':

$$S1 = \{q \in \mathcal{L} \mid q.time \geq q'.time \wedge q \neq q'\} \quad (2)$$

$$S2 = \{q \in \mathcal{L} \mid q.time < q'.time\} \quad (3)$$

If there is not such a node q' in $\mathcal{L}$ that $q'.event = e$, the two sets are defined as:

$$S1 = \{q \in \mathcal{L}\} \quad (4)$$

$$S2 = \emptyset \quad (5)$$

With the content of $S1$, $S2$, and the event set at the current timestamp T_i, MEELO finds all minimal occurrences of candidate 2-episodes, based on the following lemmas.

Lemma 1. *For every node $q \in S1$, $[q.time, T_i]$ is a minimal occurrence of the 2-episode $\alpha = q.event \rightarrow e$.*

Proof. We separate the proof into two cases.

Case 1: Suppose there is such a node q' in $\mathcal{L}$ that $q'.event = e$. For any $q \in S1$, consider occurrence $[q.time, T_i]$ of episode $\alpha = q.event \rightarrow e$. If $[q.time, T_i]$ is not a minimal occurrence, then there must be another occurrence of α that is subsumed by $[q.time, T_i]$. In other words, there must exist an E_j such that $q.time < T_j < T_i$ and $\exists e' \in E_j$ such that (1) $e' = q.event$ or (2) $e' = e$. Both are impossible. If there exists such an e' that $e' = q.event$, it contradicts with the fact that $q.time$ is the latest occurrence of $q.event$ before T_i. If there exists such an e' that $e' = e$, it contradicts with the fact that $q'.time$ is the latest occurrence of e before T_i (note that $q'.event = e$). Thus, the occurrence $[q.time, T_i]$ must be a minimal occurrence.

Case 2: Suppose there is not such a node q' in $\mathcal{L}$ that $q'.event = e$. The proof is very similar to that of Case 1 and is thus omitted. □

Lemma 2. *For every node $q \in S2$, $[q.time, T_i]$ is not a minimal occurrence of the 2-episode $\alpha = q.event \rightarrow e$.*

Proof. Suppose $S2 \neq \emptyset$, i.e., there exists such a node q' in $\mathcal{L}$ that $q'.event = e$. For any $q \in S2$, the 2-episode $\alpha = q.event \rightarrow e$ has an occurrence $[q.time, q'.time]$. Since $[q.time, T_i]$ subsumes $[q.time, q'.time]$, $[q.time, T_i]$ is not a minimal occurrence of α. □

For the example in Fig. 2, when the scanning reaches timestamp 7, $E_7 =$ BC. The $\mathcal{L}$ after the first step is shown in Fig. 3 (b). For B $\in E_7$, since there is a node B: 6 in $\mathcal{L}$, we have $S1 = \{$ A: 6$\}$ and $S2 = \{$ D: 5$\}$ based on Eqs. (2) and (3), respectively. Then, by Lemma 1, a new minimal occurrence of A → B is [6,7]. On the other hand, by Lemma 2, [5,7] is not a minimal occurrence of the 2-episode D → B. Note that [5,7] subsumes [5,6], which is an minimal occurrence of D → B detected at timestamp 6. For event C $\in E_7$, $S1$ is the whole LO-list and $S2$ is empty. Hence, every node q in $\mathcal{L}$ produces a minimal occurrence $[q.time, 7]$ for 2-episode $q.event \rightarrow$ C. All the new detected minimal occurrences of various 2-episodes are shown in Fig. 3(c).

At every timestamp T_i, the new detected minimal occurrence of 2-episode α is added into mocc(α). If α is a new detected episode, it is added into a candidate episode set C. After the algorithm finishes the whole event sequence at T_n, mocc(α) records all the minimal occurrences for every 2-episode $\alpha \in C$. According to the size of mocc(α) and the minimum support threshold *min_sup*, it is straightforward to determine which episodes are frequent. The frequent 2-episodes of running example in Fig. 1 are shown in Fig. 4.

Step 3: Updating LO-list

In Step 3, MEELO updates $\mathcal{L}$ with the events at time T_i. Specifically, for each event $e \in E_i$, if there is a node q in $\mathcal{L}$ such that $q.event = e$, MEELO removes q from $\mathcal{L}$. Regardless of the existence of such a q, MEELO inserts a new node e: T_i at the head of $\mathcal{L}$. For example, when the scanning of the sequence reaches $T_i = 7$, Fig. 3(d) shows the updated LO-list after Step 3. There are two events in E_7. For event C, C: 7 is directly inserted into $\mathcal{L}$ since the LO-list before insertion (shown in Fig. 3(b)) does not contain C. For event B, B: 6 must be removed before inserting B: 7 into $\mathcal{L}$.

3.2.3. Soundness and completeness of MEELO

We prove that MEELO is sound and complete, i.e., it finds all and only frequent 2-episodes.

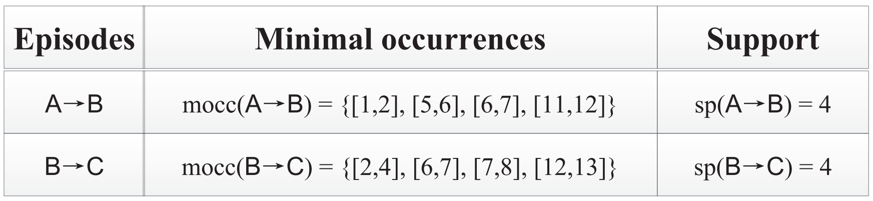
Theorem 1. *Given an event sequence $\vec{S}$, a minimum support threshold and a maximum time interval threshold, the MEELO algorithm finds all and only the minimal occurrences of frequent 2-episodes.*

Proof. At every timestamp T_i, MEELO considers all possible occurrences of 2-episodes within time window $[T_p, T_i]$ (cf. Eq. (1)). More specifically, it considers all 2-episodes $\alpha = q.event \rightarrow e$ for $e \in E_i$ and $q \in \mathcal{L}$. Lemma 1 shows that $[q.time, T_i]$ is a minimal occurrence of α if $q \in S1$. Lemma 2 shows that $[q.time, T_i]$ is not a minimal occurrence of α if $q \in S2$. Thus, MEELO finds all minimal occurrences within time window $[T_p, T_i]$.

As MEELO sequentially scans the whole (frequent) event sequence $\vec{S}$ from beginning to end, it finds all minimal occurrences of 2-episodes within every possible time window. After it finishes scanning $\vec{S}$, it identifies all frequent 2-episodes based on the minimum support threshold. □

4. Measuring sensational effect

There might be a large number of frequent episodes that exceed the minimum frequency thresholds and only few of them hold sensation. Hence the next problem we face is to measure the sensational effect of any custom frequent 2-episode. Such task is challenging because sensation could be subjective from perspective of audiences and difficult to give a succinct formal definition. As a result, we first discuss where the sensational effect come from.



Episodes	Minimal occurrences	Support
A→B	mocc(A→B) = {[1,2], [5,6], [6,7], [11,12]}	sp(A→B) = 4
B→C	mocc(B→C) = {[2,4], [6,7], [7,8], [12,13]}	sp(B→C) = 4

Fig. 4. All frequent 2-episodes in the running example.

4.1. Why episode be sensational?

To better investigate the sensational effect of an episode of news events, we derive some potential ingredients for sensational effect from the motivated examples and summarize as follows.

(a) *Probabilistic correlation between the events.* First, the probabilistic correlation between the antecedent event and the consequent event is essential for a sensational episode of news events. For example, in the motivated example of Jam Hsiao, the confidence (conditional probability) for the pattern is over 80%. A high probabilistic correlation might intuitively make an episode bear potential sensation.

(b) *The event conceptual deviation between the observation and the expectation of common sense.* Here the concept of an event refers to the semantic domain of such event. Consider the example of Jam Hsiao again, when he appeared at a public event, the place where he appeared apparently poured. Here the semantic concept of the antecedent event is about people and activity while that of the consequent event is about place and weather. The significant event concepts' distance results in its sensational effect. Usually the semantic domain is related to the entities inside an event. Hence we argue that the sensational effect of an episode might related to the "similarity" between the entities in the episodes.

(c) *The public impact value of the events/episode.* Third, the impact of the episode or the events which consists of the episode is related to the sensational effect. Recall that the example of "Ramsey's Curse", the public impact value of its consequent event is crucial since all the persons who appear in the consequent event are very famous. It further makes the whole episode widely spread and become a burst topic.

We argue that all these intuitive ingredients may contribute to discover sensational episodes, and they cannot be separated. For instance, it does be possible for frequent but unsensational episodes having a high probabilistic correlation between the events. As a result, we devise multiple features to capture the sensational effect of a 2-episode from the above mentioned aspects and adopt a learning-to-rank approach for evaluation instead of trapping ourselves to any individual subjective measure of sensational effect.

4.2. Information for measuring sensational effect

In what follows we introduce the information which can help design the features for measuring the sensational effect of the episodes.

4.2.1. Entity and event hierarchy

Recall that the semantic domain of an event is related to the entities inside. Meanwhile, we need to compute the similarities of the entities. Entity hierarchy then becomes a natural choice for such task since entities are connected by paths in the hierarchy and we can calculate the distance between any pair of entities based on these paths. As a consequence, we adopt the entity hierarchy and build an *event hierarchy* in this study. Specifically, we utilize DBPedia Ontology [22] [3] (an ontology based on most commonly used infoboxes within Wikipedia), to first acquire abstractions of entities. Based on the entity hierarchy, we further build a hierarchy of events with the entities in them. We show an example of event hierarchy in Fig. 5. In this Figure, the events in which entities are in the most specific form are located in the leaves of the tree, and the events whose entities are in form of abstractions are located in internal nodes.

We define an event located on an interval node of an event taxonomy tree as *super event*.

Super event: A *super event* e_s is defined as an event, which includes at least one entity that is in a form of abstraction.

For example, In Fig. 5 the event ⟨*SoccerPlayer, score*, NULL⟩ is a super event since *Aaron_Ramsey* generalizes to *SoccerPlayer*. We call the event ⟨*SoccerPlayer, score*, NULL⟩ is an *ancestor* of the event ⟨*Aaron_Ramsey, score*, NULL⟩, and ⟨*Aaron_Ramsey, score*, NULL⟩ is a *successor* of ⟨*SoccerPlayer, score*, NULL⟩. To be distinguished, we call an event which is not a super event as a basic event hereafter.

With the concept of super event, we might discover interesting episodes with high sensational effect. Recall that the example of "Ramsey's Curse", the episode could be ⟨*Aaron_Ramsey, score*, NULL⟩ → ⟨*Person, die*, NULL⟩. Here $\langle Person, die, \text{NULL}\rangle$ is a super event that supports the episode occurs enough times. However, not arbitrary episode with super events has a significant sensational effect. For instance, as shown in Fig. 5, though the episode ⟨*Person, score*, NULL⟩ → ⟨*Person, die*, NULL⟩ may have higher frequency (it consists of events in higher abstraction form), it might fail to be sensational since the antecedent and the consequent event happen nearly everyday. It cannot escape from common-sense facts to people. In another word, it has low deviation between the observation and the expectation of common sense. However, the episode ⟨*Aaron_Ramsey, score*, NULL⟩ → ⟨*Person, die*, NULL⟩ could be sensational since the antecedent event occurs with limited times (but greater than *min_sup*), and with a high probability when it happens, it is followed by the death of a big figure.

Hence with the super events, we aim to identify the sensational episodes with both basic and super events in this study. To this end, we add both basic and super events into the event sequence, and discover frequent episodes by the algorithm MEELO. All these episodes will be ranked by their sensational effect.

4.2.2. Event bombshell

Another helpful information for designing the features is the public impact value of events. In fact, every event occurrence itself has its public impact, and different occurrences may derive different values. How to capture such impact value of every event occurrence from the perspective of news audiences? We argue that a natural choice for acquiring such value of each event occurrence can be the re-tweet number of the tweet containing such event. We call it as the *event bombshell* in this study. The reason we adopt this number is under the intuitive assumption that people feel more obliged to re-tweet the events with bigger public im-

[3] DBPedia Ontology is actually a tree-like hierarchy, in which each higher level of abstraction encompasses a group of several entities or lower of abstractions.

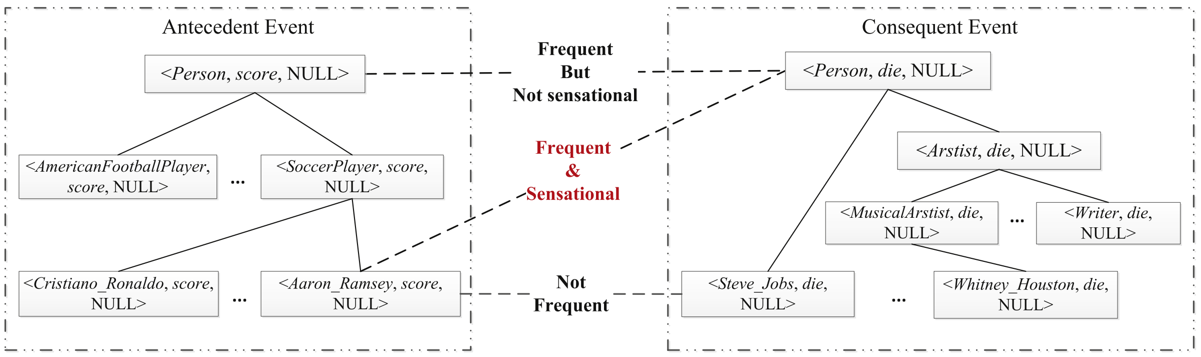

Antecedent Event
<Person, score, NULL>
<AmericanFootballPlayer, score, NULL>
<SoccerPlayer, score, NULL>
<Cristiano_Ronaldo, score, NULL>
<Aaron_Ramsey, score, NULL>
Frequent But Not sensational
Frequent & Sensational
Not Frequent
Consequent Event
<Person, die, NULL>
<Arstist, die, NULL>
<MusicalArstist, die, NULL>
<Writer, die, NULL>
<Steve_Jobs, die, NULL>
<Whitney_Houston, die, NULL>

Fig. 5. Event hierarchy and sensational effect of episodes corresponding to the hierarchy.

pact. For a super event, we can hardly acquire the bombshell value of its occurrence directly, but we can sum the bombshell values of all its successors occurring at the same time. Thus denoted by b_e^t as the bombshell value of an event e occurring at time t, and we will use this notion to exploit features.

5. Learning sensational relationship

With the information from the event hierarchy and the event bombshell, we develop the following features, which might be related to sensational effect. They can be grouped into the following three categories: features related to event hierarchy, features related to event bombshell and traditional features.

5.1. Features related to event hierarchy

In such group of features, we mainly expect the features can describe semantic as well as the probabilistic correlations between the events in an episode. To this end, the location of events in the hierarchy may contribute for calculation. Hence given an event hierarchy, we first define the depth of an episode $\alpha = e \rightarrow e'$ as follows:

$$\text{depth}(\alpha) = \frac{\text{dep}(e) + \text{dep}(e')}{2} \tag{6}$$

where dep(e) and dep(e') denotes the *depth* of the event e and e', respectively.

For an event $e = \langle s, v, o \rangle$, the depth of e, denoted by dep(e), can be computed as follows:

$$\text{dep}(e) = \frac{\rho_s d_s + \rho_o d_o}{\rho_s + \rho_o + 1} \tag{7}$$

where d_s and d_o denote the depth of the subject entity and the object entity of e in the entity hierarchy, respectively. ρ is an indicator such that it is $\rho = 1$ if the corresponding entity (the subject or object) is not absent. Here, the depth of an entity in the hierarchy is the number of nodes from the taxonomy root to the corresponding node of this entity.

5.1.1. Hierarchy based semantic correlation

Given an episode $\alpha = e \rightarrow e'$, we define the following feature to describe the semantic correlation between the events.

$$\text{HSC}(\alpha) = \text{SC}(e, e') \times \text{depth}(\alpha)^{\lambda} \tag{8}$$

where SC(e, e') is the semantic correlation of e and e' (defined later), depth(α) is the depth of this episode defined in Formula 6, λ can be considered as a hierarchy based normalized factor and we fix it as 1 in the experiments. Here, the basic idea for using depth(α)$^{\lambda}$ is to prefer the episode whose entities are close to the leave nodes in the taxonomy because a specific event may usually make more sense and meanwhile be easier to understand than a general one.

In the following, we introduce how to compute SC(e, e'), which is inspired by Shen et al. [38]. In [38], a method measuring semantic similarity between entities (wikipedia concepts) based on the taxonomy of knowledge base was proposed. Given two entities w_1 and w_2, the sets of their super categories are denoted as Φ_{w_1} and Φ_{w_2}, respectively, and the semantic similarity between Φ_{w_1} and Φ_{w_2} is computed first. For each category C_1 in the set Φ_{w_1}, the target category $\varepsilon(C_1)$ in another set Φ_{w_2} is defined as follows:

$$\varepsilon(C_1) = \underset{C_2 \in \Phi_{w_2}}{\arg\max}\, \text{sim}(C_1, C_2) \tag{9}$$

where sim(C_1, C_2) is the semantic similarity between two category C_1 and C_2 , and $\varepsilon(C_1)$ in the category in Φ_{w_2} which maximizes the semantic similarity between these two categories. Then, sim(C_1, C_2) is computed by a information-theoretic method[24]. In the tree-like taxonomy, the semantic similarity between two category C_1 and C_2 can be computed as follows:

$$\text{sim}(C_1, C_2) = \frac{2 \times \log(\text{P}(C_0))}{\log(\text{P}(C_1)) + \log(\text{P}(C_2))} \tag{10}$$

where C_0 is the root of the smallest subtree that contains both C_1 and C_2 in the taxonomy. P(C) is the probability that a randomly selected object belongs to the subtree with the root of C in the taxonomy. Next, the semantic similarity from one set of categories Φ_{w_1} to another set of categories Φ_{w_2} can be computed as follows:

$$\text{sim}(\Phi_{w_1} \rightarrow \Phi_{w_2}) = \frac{\sum_{C_1 \in \Phi_{w_1}} \text{sim}(C_1, \varepsilon(C_1))}{|\Phi_{w_1}|} \tag{11}$$

The semantic similarity from Φ_{w_2} to Φ_{w_1}, namely $sim(\Phi_{w_2} \rightarrow \Phi_{w_1})$ can be calculated in the similar way to that in Formula 11. The semantic similarity between two entities w_1 and w_2 is defined as the arithmetic mean of the semantic similarity from Φ_{w_1} to Φ_{w_2} and that from Φ_{w_2} to Φ_{w_1}:

$$\text{SmtSim}(w_1, w_2) = \frac{\text{sim}(\Phi_{w_1} \rightarrow \Phi_{w_2}) + \text{sim}(\Phi_{w_2} \rightarrow \Phi_{w_1})}{2} \tag{12}$$

Next, we can calculate the semantic correlation of two events e and e' as follows:

$$\text{SC}(e, e') = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \text{SmtSim}(w_i^e, w_j^{e'}) \tag{13}$$

where w^e and $w^{e'}$ denote any entities in event e and e' respectively, and m and n represent the number of such entities in e and e', respectively.

5.1.2. Hierarchy based lift

Next we define the hierarchy based lift of an episode $\alpha = e \rightarrow e'$ as follows. It describes the probabilistic correlation between e and e' by taking the event hierarchy information into account.

$$\mathrm{HLift}(\alpha) = \mathrm{depth}(\alpha)^{\lambda} \times \mathrm{lift}(e, e') \tag{14}$$

where depth(α) is the depth of the episode defined in Formula (6), λ is also a hierarchy based normalized factor and we fix it as −1 in the experiments. Here, the main idea for involving depth$(\alpha)^{\lambda}$ is to balance the bias brought by the event hierarchy. Two specific events may have higher correlations in statistic (such as confidence and lift) than general ones because of their lower support values.

lift can be calculated as follows:

$$\mathrm{lift}(e, e') = \frac{n \cdot \mathrm{sp}(\alpha)}{\mathrm{sp}(e) \times \mathrm{sp}(e')} \tag{15}$$

where sp(.) denotes the support of the item, and n denotes the length of event sequence.

5.2. Features related to the event bombshell

For the features related to the event bombshell, we mainly expect them to depict both the global impact of the whole episode and the local impact for every occurrence. Thus to design these features, we are given an episode $\alpha = e \rightarrow e'$ with all its minimal occurrences $\{[t_1, t_1'], [t_2, t_2'], \cdots, [t_k, t_k']\}$. For each occurrence of $[t_i, t_i']$, the bombshell values of e, e' are denoted as $b_e^{t_i}, b_{e'}^{t_i'}$. respectively.

5.2.1. Weighted local bombshell values

From the observation of the motivation episodes, we learn that the bombshell value of the consequent event may have more impact on the sensational effect of episode. Moreover, a local correlation of each episode occurrence is also important. Retrospect to the episode about Hsiao, every time the episode occurs, a persistently high correlation between the two events makes such episode really impressive to the public. Hence, we try to design a feature corresponding to both of them here.

Kulczynsky [43] is used to measure correlation of two variables, and we use such measure to calculate the local correlation of every episode occurrence. Given two events e and e', the Kulczynsky (denoted as Kulc) is defined as follows:

$$\mathrm{Kulc}(e, e') = \frac{\mathrm{P}(e'|e) + \mathrm{P}(e|e')}{2}, \tag{16}$$

where P($e'|e$) is defined as

$$\mathrm{P}(e'|e) = \frac{\mathrm{sp}(\alpha)}{\mathrm{sp}(e)} \tag{17}$$

where sp(.) denotes the support of the item. P($e|e'$) can also be calculated similarly.

Now, we introduce how to use Kulczynsky to calculate local correlation of every episode occurrence. The method is we truncate the whole event sequence $\vec{S}$ to a subsequence, denoted as $\vec{S}_{t_j'}$, of which the first timestamp is the starting point of $\vec{S}$ and the last timestamp is the end time of one minimal occurrence $[t_i', t_j']$ of a given frequent episode $\alpha = e \rightarrow e'$. Then, we compute the measure of Kulczynsky between e and e' over the sequence of $\vec{S}_{t_j'}$. The result value is considered as the local correlation of such episode occurrence in this paper. The Kulc(e, e') on $\vec{S}_{t_i'}$ can be calculated as follows (denoted by $\mathrm{Kulc}_{t_i'}(e, e')$):

$$\mathrm{Kulc}_{t_i'}(e, e') = \frac{\mathrm{P}_{t_i'}(e'|e) + \mathrm{P}_{t_i'}(e|e')}{2} \tag{18}$$

where

$$\mathrm{P}_{t_i'}(e'|e) = \frac{\mathrm{sp}_{t_i'}(\alpha)}{\mathrm{sp}_{t_i'}(e)}, \quad \mathrm{P}_{t_i'}(e|e') = \frac{\mathrm{sp}_{t_i'}(\alpha)}{\mathrm{sp}_{t_i'}(e')} \tag{19}$$

where $\mathrm{sp}_{t_i'}()$ denotes the support of item on $\vec{S}_{t_i'}$.

Then, the feature named *weighted bombshell value* of an episode is defined as follows:

$$\mathrm{WBV}(\alpha) = \frac{1}{k}\sum_{i=1}^{k} \mathrm{Kulc}_{t_i'}(e, e') \times \log\left(b_{e'}^{t_i'}\right)\Big) \tag{20}$$

5.2.2. Global bombshell values

We further define a group of straightforward features describing the global event bombshell of an episode. They include the bombshell of the antecedent event, the bombshell of the consequent event and the difference between them.

$$\Delta\mathrm{B}(\alpha) = \mathrm{Bomb}(e') - \mathrm{Bomb}(e), \tag{21}$$

where

$$\mathrm{Bomb}(e) = \frac{1}{k}\sum_{i=1}^{k} \log b_e^{t_i}, \mathrm{Bomb}(e') = \frac{1}{k}\sum_{i=1}^{k} \log b_{e'}^{t_i'} \tag{22}$$

These two values defined in Formula (22) are bombshell value of the antecedent and the consequent event, respectively. The value in Formula 21 is their difference. All the three values can be considered as features related to the global event bombshell.

5.3. Traditional features

Some traditional measures are also utilized here to measure the sensational effect, including the Granger value [14] and the support of a frequent episode α. The Granger causality test is a statistical hypothesis test for determining whether one time series is useful in forecasting another [14]. It is always be used to examine if two events have causality in relationship. Given an episode $\alpha = e \rightarrow e'$, the occurrences of each individual event can also be built as a sequence, and hence we can perform the Granger causality test on the occurrence sequence of e and e'. We adopt 1 minus the p-value of the test as one feature, which is denoted as Granger(α).

5.4. Learning to rank sensational episodes

With all these aforementioned features, we can generate a feature vector $\vec{F}(\alpha)$ for each frequent episode $\alpha = e \rightarrow e'$ where $\vec{F}(\alpha)$ = (HSC(α), HLift(α), WBV(α), ΔB(α), Bomb(e), Bomb(e'), Granger(α), sp(α)). By training a random forests ranker [13] implemented in RankLib [4] (an opensource library of learning-to-rank algorithms) on labeled examples with $\vec{F}$, we can give a predicted rankings for frequent episodes by their sensational effect. The random forests ranker is a "TreeBoost" point-wise model in which gradient boosted regression trees [13] are boosted in order to produce competitive, highly robust, interpretable procedures for rankings. Here we prefer a point-wise ranker instead of list-wise or pair-wise rankers is because we consider every episode is independent to each other. Meanwhile a global ranking based on ranking scores produced by point-wise ranker is more intuitive to verify the effectiveness of the proposed features. The experimental results show that the proposed features are effective for ranking sensational episodes higher.

[4] http://people.cs.umass.edu/~vdang/ranklib.html

Table 2
The account information of news agencies.

NBC World News	CNN	CNN Breaking News
BBC News (World)	BBC Breaking News	The New York Times
A Googler	Sky Sports News	New York Post
Fox News	Reuters Real Time	Breaking News

6. Experiments

6.1. Event extraction from tweets

6.1.1. Data of event extraction

To demonstrate that our work can discover sensational episodes of news events, we first prepare a news text corpus from Twitter for experiments. We gathered 65,298 tweets of twelve news agency accounts in Twitter range from February 21, 2012 to March 17, 2013 (collected by the Twitter4J API[5]). Table 2 lists the account information used in the experiments.

Prior to discovering and learning to sensational episodes, we need to extract new events from the tweets we gathered. Natural Language Processing (NLP) tools combined with a knowledge base based on Wikipedia concepts are utilized to extract and aggregate new events. We perform it via the following steps, and we finally obtain the event set $\mathcal{E}$ including 23,692 basic events and 47,358 super events.

6.1.2. Sentence-level paraphrase detection

Identifying paraphrases is important for event extraction since different journalists may have diversities in representations referring to lexical and syntax to an same event. For example, the two different tweets as shown in follows are paraphrase in fact:

S1:*Neil Armstrong, the first man to walk on the moon, has died - he was 82.*

S2: *Neil Armstrong, the 1st man on the moon, has died at age 82 - @NBCNews.*

They all tell the astronaut Neil Armstrong died on his 82 in different ways. Such diversities may impact on the results of event extraction. Hence, in order to improve the quality of event extraction, we first merge sentences which are paraphrases before extracting events from them.

We adopt sentence paraphrase detection tool [39] to identify whether two tweets are paraphrase in sentence level after cleaning the URL links, hashtags and special symbols of tweets. If they are, we unify them into the same form.

6.1.3. Event extraction and consolidation

Next, we perform event extraction and conslidation from tweets. In this stage, The OpenIE tool Boxer [7,10], which can generate semantic representations of sentences, under C&C tools[6] was utilized. By parsing the outputs of the Boxer tool, we obtained the events defined in Section 3. When parsing outputs generated by Boxer, some detailed processes such as verb stemming based on WordNet [30] and entity recognition were done to improve the extraction results.

Followed by event extraction, we have to consolidate the events having the same meanings since it is possible for different events are still paraphrase. This step can reduce the sparsity of data, which will make the mining algorithm discover more frequent episodes. This consolidation process can be separated into two phrases, including *entity resolution* and *verb aggregation*.

Entity resolution is to give different entities with the same meaning a unified form. For example, the president of United States Barack Obama may have distinct forms appearing in news text like “Barack Obama”, “President Obama” or “Obama”, and we need to link each representation into a unified form such as “Barack Obama”. In this paper, we combine the Stanford Named Entity Recognizer (NER)[12] with DBPedia Ontology [22] to do entity resolution. In more detail, named entities identified by Stanford NER in events are searched in the DBPedia Ontology as queries. Then, we replace each named entity by the first instance returned by DBPedia Ontology in the corresponding event. Some trivial disambiguations are conducted manually in this work.

For the verbs, due to the existence of synonyms on verbs, aggregating verbs having similar meanings is also meaningful for event consolidation. In the paper, given a verb in an event, we consider the similarity relations in Wordnet [30] including *hypernyms, hyponyms, similar to* and *see also* relations, and replace all the related terms by a unified form. For example, the verb “defeat” and “beat” can be aggregated because “defeat” is a hypernym of “beat”. The second column of Table 3 shows the results after performing these stages to the tweets in the first column of the same table.

6.1.4. Super event generation

After that, we generate super events from all extracted and consolidated events. Since the entities are already linked to DBPedia Ontology in the aforementioned stages, they can be mapped to leaves in the entity taxonomy. Then, given an entity within an event, every time we replace such entity with an arbitrary interval node of it, we can generate an ancestor of such event, and it comes a super event. There may be some entities not covered by DBPedia Ontology, and hence we give them a simple abstraction, denoted by “THING”, when generating super events. The third column of Table 3 shows the examples of super event generation. For example, as we observe from the table, based on the basic event ⟨*Neil_Armstrong, die*, NULL⟩, we can generate two super events ⟨*Astronaut, die*, NULL⟩ and ⟨*Person, die*, NULL⟩ because “Neil_Armstrong” is an “Astronaut” and “Astronaut” is a “Person”. The basic event ⟨*Airbag problems, cause, Hyundai*⟩ generalizes to a super event ⟨*THING, cause, Hyundai*⟩ since the entity “Airbag problems” is not included in DBPedia Ontology.

6.1.5. Timestamps of event occurrences

Finally, we build the event sequence. So far, we have obtained the event set $\mathcal{E}$ via the above stages, and need to acquire the timestamps of event occurrences for building the event sequence. In this paper, we use days as timestamps unit and treat the publishing date of the tweets from which the event e is extracted as its timestamp. This treatment may not always accurately capture the occurrence timestamps of events. For example, a retrospective story might mention an event which happened earlier than the publishing time of the story. However, such inaccuracy does not raise serious concerns with daily news from news agency, since such news mostly only narrate the current events happening shortly before the publishing time. On the other hand, as we discover serial episodes specifically, our mining algorithm will not concatenate two events with a same timestamp as an episode.

6.1.6. Analysis on event extraction errors

During the process of event extraction, we observed possible cumulative errors happened on the verb aggregation. For example, the event extracted from the tweet “American judo player Nicholas Delpopolo expelled from Olympics for doping violation.” was “⟨*Nicholas Delpopolo, resign*, NULL⟩”. Though the verb “expel” and “resign” are related in semantic, the event may mislead as there exist other events describing people resign from some positions, e.g., “Pope Benedict XVI to resign at end of February”. We argue that such error is inevitable and it has less impact on

[5] http://twitter4j.org/en/index.html
[6] http://svn.ask.it.usyd.edu.au/trac/candc

Table 3
Examples of event extraction from tweets.

Tweet content	Basic events	Super events
Neil Armstrong, 1st man on moon, has died at age 82.	⟨*Neil_Armstrong, die*, NULL⟩	⟨*Astronaut, die*, NULL⟩ ⟨*Person, die*, NULL⟩ ⟨*Airbag problems, cause, Company*⟩
Airbag problems have caused Hyundai to recall 220,000 vehicles.	⟨*Airbag problems, cause, Hyundai*⟩ ⟨*Hyundai, recall, 220,000 vehicles*⟩	⟨*Airbag problems, cause, Organisation*⟩ ⟨*THING, cause, Hyundai*⟩ ⟨*Company, recall, 220,000 vehicles*⟩ …

Table 4
The distribution of ranking labels in labeled data set.

Ranking labels	# in fold 1	# in fold 2	# in fold 3
perfect-4	2	2	2
excellent-3	4	4	3
good-2	10	14	11
fair-1	11	8	6
bad-0	659	657	663

Table 5
Feature set effectiveness over labeled data set.

Feature Set	NDCG@5	NDCG@10	ERR@10
BF	$0.5499_{(\pm 0.0706)}$	$0.4777_{(\pm 0.053)}$	$0.9469_{(\pm 0.0068)}$
BF+HSC	$0.5687_{(\pm 0.0788)}$	$0.5423_{(\pm 0.0513)}$	$0.9506_{(\pm 0.0037)}$
BF+HSC +HLift	$0.5861_{(\pm \mathbf{0.0167})}$	$0.5326_{(\pm \mathbf{0.0182})}$	$0.952_{(\pm \mathbf{0.0012})}$
BF+HSC+ HLift+WBV	$\mathbf{0.6477}_{(\pm 0.1043)}$	$\mathbf{0.58}_{(\pm 0.0736)}$	$\mathbf{0.9544}_{(\pm 0.0031)}$

this study. Our study focuses on mining 2-episodes from event sequence of news events and learning sensational effect from masses of frequent episodes of new events. The NLP techniques for improving event extraction is out of the scope of this paper.

6.2. Experiment settings

We investigate the effectiveness of the proposed approach for ranking frequent episodes by their sensational effect. For this purpose, we randomly selected 2,056 frequent episodes, and they were labeled by ten volunteers (graduate students) with five-graded ("bad-0","fair-1","good-2","excellent-3","perfect-4") absolute relevance judgement. If an episode is "perfect sensational", it is labeled by 4, and the grade will decreases as the sensational effect decreases. Specifically, we provided a labeling system for the volunteers and they could check every frequent episode as well as each occurrence date of such episode with their original tweets and the retweet numbers. Every volunteer labeled all the selected episodes. We did not lure the volunteers to follow any specific conscious guidance, and they labeled all the data by their own individual judgement. Eventually the round-off average ranking score of each episode was considered as its final ranking rate. Since we randomly selected the candidates for labeling, the sensational episodes in the experimental set were sparse. To ensure the reliability of evaluation results, we divided the labeled examples through a stratified 3-fold cross-validation. That is the folds were made by preserving the percentage of samples for each rating scores. The data statistic of experimental data set is demonstrated on Table 4. We must claim that there are more than 200,000 frequent episodes be discovered in such tweet data set, and candidate episode set for the news audience is provided by a random selection. Hence they only observe a tip of the iceberg of all possible sensational episodes. For this comparison, the baseline feature set consists of sp(α), Granger(α), ΔB(α), Bomb(e) and Bomb(e'), which is denoted as BF hereafter. The feature HSC(α), Hlift(α), and WBV(α) will be added gradually to the BF feature set, and they are denoted as HSC, HLift and WBV, respectively. The parameters of random forests ranker are set as default values as implemented in RankLib.

6.3. Experimental results

We adopt NDCG@5, NDCG@10 [25] and ERR@10 [9] as the metrics for the effectiveness evaluation. Table 5 shows the mean as well as the standard deviation (in the brackets) of each metric on different feature sets based on the stratified 3-fold cross validation, and the highest value of each metric is marked in boldface.

Basically, as observed from the table, every feature has a positive impact on the effectiveness as it is added to the feature set gradually. The combination of all features obtains the best performance. The improvements achieved by adding HSC to {BF} are significant on NDCG@10 and ERR@10, which indicates that the feature is quite useful to detect sensational episodes since more sensational ones are ranked in top 10.

Though it degrades the NDCG@10 slightly when adding HLift to the set of {BF, HSC}, it achieves improvements on the other two metrics. Moreover, the standard deviations between different folds are significantly reduced compared with the feature sets without it, which indicates such feature is useful to acquire a stable performance.

The improvements achieved by adding WBV to the feature set {BF, HSC, HLift} are significant on both NDCG@5 and NDCG@10, which indicates this feature describing the local impact on the event bombshell is an essential factor for giving sensational episodes higher rankings.

In summary, the *hierarchy based semantic correlation* is a predictable feature for discovering sensational episodes, the *hierarchy based lift* is an important factor of acquiring stable performance, and the *weighted bombshell values* is an essential factor for ranking sensational episodes much higher.

6.4. Feature study

We further investigate the correlations between the features and the ranking scores via extensive experiments. Fig. 6 shows the arithmetic mean and the standard deviation of each feature on labeled examples associated with their ranking labels. In the figure, the mean of some features, such as HLift(α) and WBV(α), increases significantly as the ranking label increases. While the mean of sp(α) decreases when the ranking label increases instead. The other features like Granger(α) and Bomb(e), fail to show significant changes as the ranking label varies from "bad-0" to "perfect-4".

We additionally perform a linear regression of the mean of features for the ranking labels and examine the Spearman correlation [19] between them as well. Table 6 shows the R^2 value of the regression model and the obtained correlation coefficients with their corresponding p-values. We can find that the feature HLift(α) and WBV(α) proposed in this paper have a strong positive correlation with the ranking labels, and the R^2 value of such two features

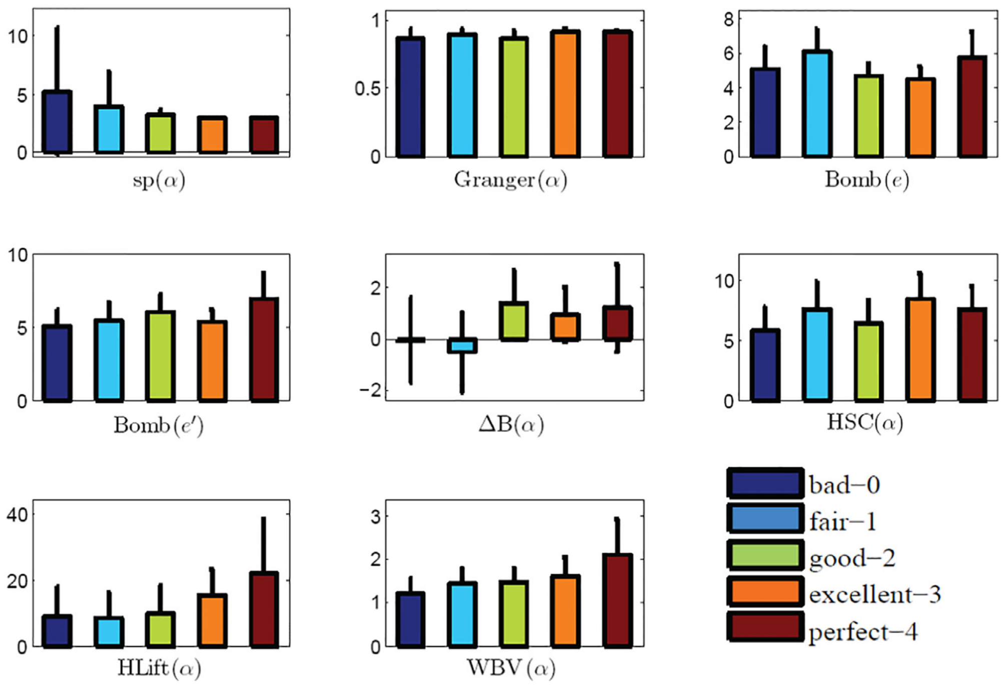


Fig. 6. The arithmetic mean and standard deviation of features on labeled data set.

Table 6
The correlation between the arithmetic mean of features and the ranking labels.

Features	Correlation	p-value	R^2
sp(α)	−0.975	0.0333	0.806
Granger(α)	0.9	0.0833	0.637
Bomb(e)	−0.2	0.7833	0.007
Bomb(e')	0.7	0.2333	0.611
ΔB(α)	0.6	0.35	0.611
HSC(α)	0.8	0.1333	0.483
HLift(α)	0.9	0.0833	0.811
WBV(α)	1	0.0167	0.878

Table 7
Statistical information on different sequences.

Data set	# Timestamp	# Events	Avg. # Events per Timestamp
Retail	88,162	16,470	10.3
Chess	3,197	75	37.0
Tweet	393	71,050	245.1

are relatively high as well, which indicates that these features are predictive for finding sensational episodes. Though the correlation coefficient between the feature HSC(α) with the ranking labels is moderately positive, the R^2 value is low, indicating that such a feature are not very predictive of the sensational episodes. We conjecture it may be the prominent reason for failing to significantly improve the metric of NDCG@5 when it is added to the baseline set as the only supplement. In addition, the support of episodes also has a strong correlation (in negative) with the ranking labels, and meanwhile its R^2 value is relatively high. We argue that that is why a small amount of sensational episodes can be detected merely via the baseline feature set.

6.5. Examples of interesting sensational episodes

We exhibit some examples of interesting sensational episodes discovered in the data set in this subsection. Among the discovered sensational episodes, we observe that some event pairs have implicit causality, which makes such episodes have significant sensational effect.

For example, we discovered ⟨ *Barack_Obama, lead,Mitt_Romney*⟩ → ⟨ *Dow_Jones_Indexes, fall*, NULL⟩. That is when the President of United States Barack Obama lead his opponent Mitt Romney in poll or in the final election, the United States stock, e.g. Dow Jones Industrial, would slip on next trading day. Especially on November 7, 2012, Dow slipped more than 300 points after Obama won the election. Due to different background between the two candidates, the financial community supports Romney than Obama during the period of election 2012. Hence, there may be some implicit causality between the two events.

By traditional causality analysis techniques based on content, it is difficult to discover this kind of implicit causality between two events. However, because of the sensational effect of such episode, it can be produced via the proposed routine in this paper.

6.6. Evaluation on efficiency

Next we evaluate the efficiency of the proposed MEELO algorithm for discovering frequent 2-episodes. All experiments on this investigation were performed on a PC with a 3.30 GHz Intel Core i5-2500 Processor with 8 gigabytes memory, running on Windows 7. All of the algorithms were implemented in Java.

Since there are two parameters in our problem of mining frequent 2-episodes, we focus the impact of two parameters (*min_sup* and *max_intvl*) on the efficiency. The baseline algorithms we compare with include MINEPI [28], MINEPI+ [18] and UP-Span [42]. MINEPI is a state-of-art of breadth-first enumeration algorithm which is designed for FEM in *simple sequence*. Simple sequence contains only one event in any single timestamp. MINEPI+ is a state-of-art in FEM in *complex sequence* containing multiple simultaneous events in the same timestamp (same with the event sequence in this paper). UP-Span was designed for mining high utility episodes also in complex sequence. It first detects minimal occurrences of episodes and then calculates utility value of each minimal occurrence and keeps these minimal occurrences whose utility value are bigger than a user-specified threshold. To

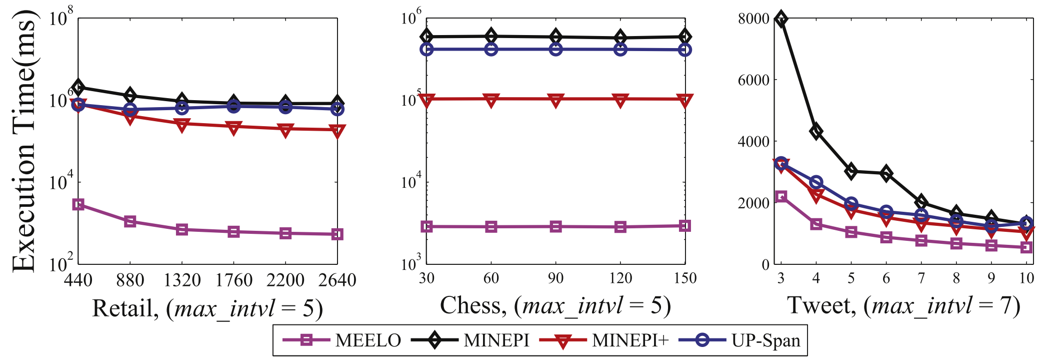


Fig. 7. The execution time vs. *min_sup*.

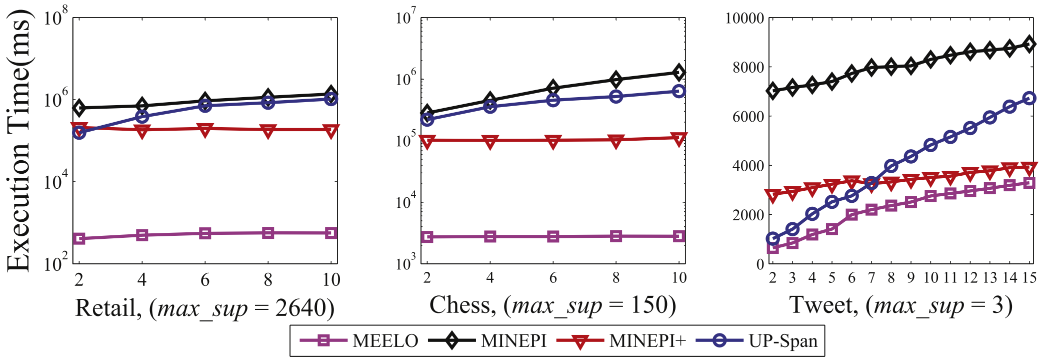


Fig. 8. The execution time vs. *min_intvl*.

make UP-Span satisfy our scenario, we try our best to modify it. Both MINEPI+ and UP-Span apply depth-first strategy on the event sequence for searching efficiency improvement. In this comparison, we merely post the time of finding frequent 2-episode including finding minimal occurrences of episodes on the frequent event sequence. The elapsed time of prior scanning for finding frequent events and other pre or post-processing are overlooked.

We verified the compared methods on three real-world datasets including our tweet dataset and two classic benchmarks for sequential pattern mining. The statistical information about the datasets is demonstrated in Table 7. We followed the preprocessing methods in [42] to make the benchmark data sets fit for our mining problem.

Fig. 7 demonstrates the execution time on the three data sets under varied minimum support and fixed maximum time interval threshold. We can clearly observe that the proposed algorithm MEELO always has the lowest execution time compared with other baselines under different *min_sup* settings. The advantage of MEELO is significant on Retail and Chess that it outperforms the baselines by at least one order of the magnitude. For example, MEELO achieves a speedup of $40\times$ compared with MINEPI+ when $min\_sup = 150$. On the Tweet data set, though the advantage of MEELO is clearly less than the other two, MEELO is the fastest approach for mining frequnet 2-episodes. Compared with the two benchmarks, the scale of our Tweet data set is much smaller, and we conjecture that may be the reason that the time cost of different algorithms ranges from 2s to 8s. For the baselines, two depth-first enumeration algorithms, MINEPI+ and UP-Span, hold a similar performance on Retail and Tweet, and they almost outperform the breadth-first enumeration baseline MINEPI. While on the Chess data set, MINEPI+ clearly outperforms UP-Span. MINEPI, as a representative bread-first enumeration FEM algorithm, runs slower than the other methods except some unchallenged parameter settings. We also observe the execution time of MINEPI has a significant increase when the *min_sup* is set less than 7 on the Tweet data set. We conjecture the reason is that there are much candidates be generated by its breadth-first enumeration strategy. As depth-first enumeration algorithms avoid the step of candidate generation, they have a better performance especially on the situation that the minimum support threshold is set to a small value. Further, because MEELO avoids the post-processing on checking minimal occurrences of generated episodes, it derives a clearly better performance in execution time compared with the two compared depth-first algorithms.

Fig. 8 demonstrates the execution time on the three data sets under varied maximum time interval thresholds and fixed minimum support. As shown in Fig. 8, MEELO always has the lowest execution time compared with other baselines under different *max_intvl* settings. Similar with the previous comparison, MEELO holds significant speedup on the Retail and Chess data sets. It outperforms the other methods at least two orders of magnitude in such comparison. For instance, MEELO achieves a speedup of $2400\times$, $330\times$ and $1800\times$ compared with MINEPI, MINEPI+ and UP-Span, respectively when $min\_intvl = 10$ on the Retail data set. For the baselines, MINEPI+ is overall faster than MINEPI and UP-Span, and UP-Span holds a vigorous growth among all the algorithms in such comparison. Meanwhile MINEPI+ and our MEELO seems insensitive for the increase of the maximum time interval. It indicates these two approaches may have a better scalability than the others. Though the execution time of MINEPI+ might surpass that of MEELO in bigger settings of *max_intvl* on Tweet data set, we are more concern about the performance in the two larger-scale datasets because the Tweet dataset is small for evaluating efficiency.

By the above observations, we demonstrate that the overall performances of MEELO significantly outperforms the other three baselines.

7. Conclusion

In this paper, we studied the problem of discovering and learning sensational episodes of news events which were extracted from short text from Twitter. First, to find all frequent episodes, we proposed an efficient algorithm, MEELO, where the minimal occurrences of episodes can be identified directly with the support of the compact data structure LO-list. The correctness as well as the completeness was formally proved. Second, to rank the generated frequent episodes by their sensational effect from the perspectives of news audience, we proposed a learning-to-rank approach that exploits multiple features to capture the sensational effect of an episode from various aspects. Experiments on real data verified our approach's efficiency and effectiveness, and we discovered examples of new event episodes with implicit causality because of their valid sensational effects.

References

[1] A. Ibrahim, A. Avinash, P. S. Sastry, Pattern-growth based frequent serial episode discovery, DKE 87 (2013) 91–108.
[2] X. Ao, P. Luo, C. Li, F. Zhuang, Q. He, Online frequent episode mining, in: IEEE ICDE, 2015, pp. 891–902.
[3] X. Ao, P. Luo, C. Li, F. Zhuang, Q. He, Z. Shi, Discovering and learning sensational episodes of news events, in: WWW Companion, 2014, pp. 217–218.
[4] X. Ao, P. Luo, J. Wang, F. Zhuang, Q. He, Mining precise-positioning episode rules from event sequences, in: IEEE ICDE, 2017, pp. 83–86.
[5] E. Aramaki, S. Maskawa, M. Morita, Twitter catches the flu: detecting influenza epidemics using twitter, in: EMNLP, 2011, pp. 1568–1576.
[6] M. Atallah, W. Szpankowski, R. Gwadera, Detection of significant sets of episodes in event sequences, in: ICDM, 2004, pp. 3–10.
[7] J. Bos, Wide-coverage semantic analysis with Boxer, in: STEP, 2008, pp. 277–286.
[8] G. Casas-Garriga, Discovering unbounded episodes in sequential data, in: PKDD, 2003, pp. 83–94.
[9] O. Chapelle, D. Metlzer, Y. Zhang, P. Grinspan, Expected reciprocal rank for graded relevance, in: CIKM, 2009, pp. 621–630.
[10] J. Curran, S. Clark, J. Bos, Linguistically motivated large-scale NLP with C&C and Boxer, in: ACL, 2007, pp. 33–36.
[11] A. Das Sarma, A. Jain, C. Yu, Dynamic relationship and event discovery, in: WSDM, 2011, pp. 207–216.
[12] J.R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by Gibbs sampling, in: ACL, 2005, pp. 363–370.
[13] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. (2001) 1189–1232.
[14] C.W.J. Granger, Investigating causal relations by econometric models and cross-spectral methods, Econometr. (1969) 424–438.
[15] R. Gwadera, M.J. Atallah, W. Szpankowski, Reliable detection of episodes in event sequences, KAIS 7 (4) (2005) 415–437.
[16] C. Hashimoto, K. Torisawa, J. Kloetzer, M. Sano, I. Varga, J.H. Oh, Y. Kidawara, Toward future scenario generation: extracting event causality exploiting semantic relation, context, and association features, ACL, 2014.
[17] Availability Heuristic. http://en.wikipedia.org/wiki/Availability_heuristic(2014).
[18] K.-Y. Huang, C.-H. Chang, Efficient mining of frequent episodes from complex sequences, Inf. Syst. 33 (1) (2008) 96–114.
[19] M.G. Kendall, Rank Correlation Methods, 1948.
[20] S. Laxman, P.S. Sastry, K.P. Unnikrishnan, Discovering frequent episodes and learning hidden Markov models: a formal connection, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1505–1517.
[21] S. Laxman, P.S. Sastry, K.P. Unnikrishnan, A fast algorithm for finding frequent episodes in event streams, in: KDD, 2007, pp. 410–419.
[22] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semant. Web (2014).
[23] R. Li, K.H. Lei, R. Khadiwala, K.C.-C. Chang, Tedas: A twitter-based event detection and analysis system, in: ICDE, 2012, pp. 1273–1276.
[24] D. Lin, An information-theoretic definition of similarity., in: ICML, 98, 1998, pp. 296–304.
[25] T.-Y. Liu, Learning to rank for information retrieval, Found. Trends Inf. Retr. 3 (3) (2009) 225–331.
[26] X. Ma, H. Pang, K.-L. Tan, Finding constrained frequent episodes using minimal occurrences, in: ICDM, 2004, pp. 471–474.
[27] H. Mannila, H. Toivonen, Discovering generalized episodes using minimal occurrences., in: KDD, 96, 1996, pp. 146–151.
[28] H. Mannila, H. Toivonen, A. Inkeri Verkamo, Discovery of frequent episodes in event sequences, DMKD 1 (3) (1997) 259–289.
[29] N. Meger, C. Leschi, N. Lucas, C. Rigotti, Mining episode rules in stulong dataset, ECML/PKDD, 2004.
[30] G.A. Miller, Wordnet: a lexical database for english, Commun. ACM 38 (11) (1995) 39–41.
[31] A. Ng, A.W.-C. Fu, Mining frequent episodes for relating financial events and stock trends, in: PAKDD, 2003, pp. 27–39.
[32] J. Nichols, J. Mahmud, C. Drews, Summarizing sporting events using twitter, in: IUI, 2012, pp. 189–198.
[33] D. Patnaik, P. Butler, N. Ramakrishnan, L. Parida, B.J. Keller, D.A. Hanauer, Experiences with mining temporal event sequences from electronic medical records: initial successes and some challenges, in: KDD, 2011, pp. 360–368.
[34] S. Petrovic, M. Osborne, R. McCreadie, C. Macdonald, I. Ounis, Can twitter replace newswire for breaking news? International AAAI conference on weblogs and social media, 2013.
[35] K. Radinsky, S. Davidovich, S. Markovitch, Learning causality for news events prediction, in: WWW, 2012, pp. 909–918.
[36] K. Radinsky, E. Horvitz, Mining the web to predict future events, in: WSDM, 2013, pp. 255–264.
[37] T. Sakaki, M. Okazaki, Y. Matsuo, Earthquake shakes twitter users: real-time event detection by social sensors, in: WWW, 2010, pp. 851–860.
[38] W. Shen, J. Wang, P. Luo, M. Wang, Linden: linking named entities with knowledge base via semantic knowledge, in: WWW, 2012, pp. 449–458.
[39] R. Socher, E.H. Huang, J. Pennin, C.D. Manning, A. Ng, Dynamic pooling and unfolding recursive autoencoders for paraphrase detection, in: NIPS, 2011, pp. 801–809.
[40] N. Tatti, B. Cule, Mining closed episodes with simultaneous events, in: KDD, 2011, pp. 1172–1180.
[41] N. Tatti, J. Vreeken, The long and the short of it: Summarising event sequences with serial episodes, in: KDD, 2012, pp. 462–470.
[42] C.-W. Wu, Y.-F. Lin, S.Y. Philip, V.S. Tseng, Mining high utility episodes in complex event sequences, KDD, 2013.
[43] T. Wu, Y. Chen, J. Han, Re-examination of interestingness measures in pattern mining: a unified framework, DMKD 21 (3) (2010) 371–397.
[44] H. Zhu, P. Wang, X. He, Y. Li, W. Wang, B. Shi, Efficient episode mining with minimal and non-overlapping occurrences, in: ICDM, 2010, pp. 1211–1216.