Revealing the Hidden Links in Content Networks: An Application to Event Discovery

Antonia Saravanou[1], Ioannis Katakis[1], George Valkanas[2], Vana Kalogeraki[3], Dimitrios Gunopulos[1]
[1] University of Athens, [2] Detectica, [3] Athens University of Economics and Business
{antoniasar,katak,dg}@di.uoa.gr,george@detectica.com,vana@aueb.gr

ABSTRACT

Social networks have become the *de facto* online resource for people to share, comment on and be informed about events pertinent to their interests and livelihood, ranging from road traffic or an illness to concerts and earthquakes, to economics and politics. This has been the driving force behind research endeavors that analyze such data. In this paper, we focus on how Content Networks can help us identify events effectively. Content Networks incorporate both structural and content-related information of a social network in a unified way, *at the same time*, bringing together two disparate lines of research: graph-based and content-based event discovery in social media. We model interactions of two types of nodes, *users* and *content*, and introduce an algorithm that builds heterogeneous, dynamic graphs, in addition to revealing content links in the network's structure. By linking similar content nodes and tracking connected components over time, we can effectively identify different types of events. Our evaluation on social media streaming data suggests that our approach outperforms state-of-the-art techniques, while showcasing the significance of hidden links to the quality of the results.

1 INTRODUCTION

Event detection in social media has attracted a lot of attention during the last few years due to the spread of social networks (e.g., Facebook, LinkedIn) and content sharing applications (e.g., Youtube, Flickr). To this end, a lot of effort has been put by the research community and a multitude of techniques have been developed targeting applications with social and commercial impact. Success stories of the application of event detection techniques can be found in a variety of domains [1, 3, 13, 15, 16].

Related Work. There are two main lines of research in the field: a) *content-based* detection, that tackles the problem by identifying novel topics in a stream of text [1, 6, 8, 9, 18], and b) *structure-based* techniques that model a social network as a graph based on the interactions between users and either track significant changes in time or look for very active sub-graphs [2, 4, 5, 10–12, 14]. On the one hand, content-based techniques perform poorly in case of events that cause increased interactions within the community. On the other hand, structure-based methods ignore content altogether (tweets, messages, articles) and may, therefore, miss some vital attributes like topics discussed and sentiment expressed.

Our approach, LiCNo[1], (Linking Content Nodes) fills this gap by utilizing a novel representation that considers both the content and the structure of a social network. More specifically, we model the streaming data as a dynamic, heterogeneous graph. Our approach treats large connected components of this graph as indicators of events. The **intuition** is that when there are a lot of users who either directly interact with each other (as seen with the network structure) or, indirectly, talk about similar topics (based on the text similarity), then these users form large connected components.

Our results show that LiCNo, taking advantage of the network structure and content, is able to track effectively and efficiently various types of events. On top of a better predictive performance, LiCNo detects multiple events per time window, instead of finding anomalous time windows [11]. Each of these events has a description and a duration. Additionally, LiCNo is independent of the way the stream is split into time windows because it does not only contain links between similar content nodes, but also between similar connected components over time. Finally, LiCNo is memory efficient since it tracks only graph's summary information and does not maintain node-specific statistics as in other approaches [4, 12].

The contributions of this paper can be summarized as follows:

A) **Network Representation:** We introduce a novel representation of a network as a *dynamic heterogeneous graph*, the *Content Network*. The utility of such a graph in other tasks has been recently discussed in the literature [4, 17].

B) **Revealing Hidden Links:** We provide an algorithm that identifies hidden links in Content Networks by connecting similar content nodes utilizing neural word embeddings.

C) **Event Detection:** We present an algorithm for detecting events by tracking large connected components of *Content Networks* over time. Our results demonstrate, that we are able to effectively identify events compared to widely used event detection techniques.

2 PROBLEM STATEMENT AND SOLUTION

In this section, we introduce some basic notation to help us present our research objective. We proceed with the description of our proposed technique to address event detection in social media.

Definition 1 (*Snapshot Graph*) A snapshot graph is a static heterogeneous graph $G_t = (\mathcal{V}_t, E_t)$, at time t. $\mathcal{V}_t = \{V_{(0,t)} \cup \ldots \cup V_{(m-1,t)}\}$ is a set of $m \geq 1$ different types of nodes sets, where $V_{(j,t)}$ is the set of nodes of type j. E_t is a set of edges that connect nodes in $\mathcal{V}_t$, with $E_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$.

Figure 1 shows an example of a snapshot graph, with two ($m = 2$) types of nodes: *users* (yellow circles) and *content* (blue rectangles).

[1] ***Reproducibility Note:*** *Datasets and source code will be at: http://cgi.di.uoa.gr/~antoniasar/LiCNo/*

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

ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3133148

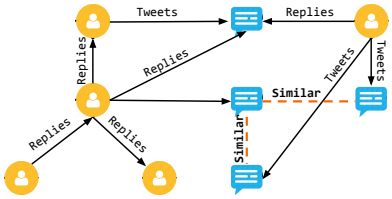

Figure 1: Example of a snapshot graph in Twitter.

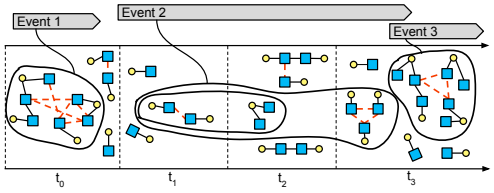

Figure 2: Tracking events over time. Let's assume that $|vlCC|$ = 9 and $|lCC|$ = 3. LiCNo tracks two $vlCC$s, Event 1 in t_0 and Event 3 in t_3. The three lCCs in t_1, t_2 and in t_3 are similar, with total size 13, and they form one more event (Event 2).

Edges between node types are labeled with their semantics. Nodes can have attributes, e.g., *language, age, location, etc*, with different node types having different attributes. We do not currently utilize that information, but plan to investigate its importance as part of our ongoing work.

Definition 2 (*Content Network*) A Content Network is a dynamic heterogeneous graph $\mathcal{G}$, comprised of a sequence of snapshot graphs over consecutive, discrete time windows. Formally, $\mathcal{G} = \{G_t | t = 1, \ldots, t_{max}\}$, where G_i is the i-th snapshot graph, observed during the i-th time window.

A time window can be as low as 1 second – or even lower – or as big as a year. An example of a Content Network in four time windows is shown in Figure 2. As we can see from the figure, edges are formed only between nodes of each individual snapshot graph.

Problem Definition. *(Event Detection in Content Networks)* Given a Content Network $\mathcal{G}$, our goal is to identify a set of M events $\mathcal{E} = e_0, \ldots, e_{M-1}$, where each event e_j is represented by its description d_j and its duration $t_{end,j} - t_{start,j}$. An event is a *topic* that attracts considerable attention compared to the norm.

Therefore, an event is a triplet, $e_j = (d_j, t_{end,j}, t_{start,j})$, and an event detection technique needs to report such triplets. The proposed definition supports multiple events during the same time window. The definition also covers events that are not *explosive* in nature, i.e., do not become spontaneously popular; rather, it allows for a topic to gradually gain popularity over time, supporting events that span across snapshot graphs.

2.1 LiCNo

LiCNo builds on the premise that important events will attract a lot of attention, proxied in a social setting by the coming together of multiple nodes. For this reason, it tracks connected components (CC) in the Content Network and treats them as event indicators. CCs are established by linking nodes that a) interact with each other, or, b) are similar.

Building the Content Network. The first step of LiCNo is to link nodes according to the domain's semantics. In a social network setting, we have two different node types: *content* and *user*. We connect a *user* node with a *content* node, if the user posted the content. We also add a link between two user nodes, if one mentioned / replied to the other. Other domains may follow different semantics under which nodes should be immediately connected [17].

Revealing Hidden Links. The next step of LiCNo is to *reveal hidden links*, by connecting similar content nodes. This is an important step, and novelty of our technique, because it brings together nodes that focus on the same topic but are disconnected (e.g., political opponents, opposing team fans, strangers with common tastes). In Fig. 1, hidden links correspond to the orange dashed links.

As we describe in Algorithm 1, we reveal hidden links by first extracting neural network embeddings [7] of a content node, based on terms of its text. Then, we augment the text of each content node with the most similar words of the node's initial terms. After that, we calculate the tf-idf embeddings and the pairwise cosine similarity between content nodes. If the similarity between two content nodes exceeds a predefined threshold we add an edge between them.

Identifying Events. At this stage, we have built our Content Network and revealed its hidden links. We now need to run our event detection approach, which must operate on a snapshot graph G_t. This is done as follows. We collect all distinct CCs found in G_t, i.e., $CC_i \in G_t, i \in \{1, p\}$. We then run each component CC_i through the function $h()$ shown below, which returns 1 if the component is anomalous, or 0 otherwise:

$$h(CC_i) = \begin{cases} 1, & \text{if } |CC_i| > \text{avg}(|CC_j|) + \theta \times \text{std}(|CC_j|), \forall CC_j \in G_t. \\ 0, & \text{otherwise.} \end{cases}$$

where $\text{avg}(|CC_j|)$ and $\text{std}(|CC_j|)$ are the (overall) average size and standard deviation, respectively, of *all* CCs in G_t, whereas θ is the detection parameter. In other words, CC_i is anomalous if it is θ standard deviations larger than the average CC in that snapshot.

Components that were marked anomalous by the previous step are termed *large* Connected Components (lCC). Using the set of

Algorithm 1 RevealHiddenLinks

Input: A snapshot graph $G_t = (V_t, E_t)$, where $V_t = V_{c,t} \cup V_{u,t}$ and $V_{c,t}$: content-nodes and $V_{u,t}$: user-nodes, a similarity threshold ϕ, a number of words to extend the text N.
Output: Snapshot Graph G_t with hidden links

Procedure enrich(text, N):
  w2v = model.Word2Vec(); textEnriched=$\emptyset$
  for token in text **do**
    //Add top N most similar words to text
    textEnriched += token+w2v.getSimilar(token, N)
  return textEnriched
Procedure revealHiddenLinks(G_t, N, ϕ):
  for c_i, c_j in $V_{c,t}$ **do**
    textEn_i = enrich(c_i.text, N), textEn_j = enrich(c_j.text, N)
    if cosineSim(tfidf(textEn_i), tfidf(textEn_j)) > ϕ **then**
      add edge (c_i, c_j) to G_t
  return G_t

*lCC*s, we rerun the anomaly detection step described above and obtain the components that are found anomalous when the baseline is set by the other *lCC*s. We call these *very large* Connected Components, denoted by *vlCC*. Such components are strong indicators of events, because of their abnormal size even when compared to the *lCC*s. As a result, they are reported immediately as *events*, with the start and end time being these of the oldest and most recent content nodes in the *vlCC*, respectively. The description of that event constitutes of the most impactful tokens in the content nodes, such as most frequent hashtags, mentions or entities.

Extending Events through Time. Due to our time discretization, it is likely for an event to cross time window boundaries. However, the corresponding *vlCC* may never form because the nodes of the component are not all part of the same G_t. To address this shortcoming, we focus on adjacent time windows and try to merge *lCC*s of the current G_t that did not become *vlCC*s with remaining *lCC*s from G_{t-1}. The process for merging is the same as when revealing hidden links, but now we add links between the similar content nodes of the candidate *lCC*s. Once *lCC*s are connected, they are checked against the threshold used to find *vlCC*s in the current G_t, and are flagged (or not) as events accordingly. This way, components of the current snapshot graph G_t are given priority in being reported, while we make up for events that were disadvantaged because of the time windows split. Those *lCC*s from G_{t-1} that were not merged with an *lCC* from G_t are discarded. An example of merged *lCC*s that spans 3 time windows (Event 2) is in Fig. 2.

Filtering. For each time window, LiCNo *automatically* filters two types of *vlCC*s that are not events: a) components of *spam* messages (messages where a user posts a lot of information in a small time frame in order to draw attention), b) components representing *blacklist* incidents (messages that users post in a short period of time and do not provide useful information). For example, some *spam* messages are: *'please, check out my fashion blog'* or *'please follow me, i'm your biggest fan @JustinBieber'*. *CC*s that are formed by spam messages have a star-structure and can be easily filtered. Examples of *blacklist* messages are 'good morning' or 'happy birthday' wishes.

Complexity Analysis. The complexity of LiCNo is $O(n^2|\mathcal{V}|)$, where n is the size of V_t and $\mathcal{V}$ is the size of the vocabulary of terms in content nodes, which is constant. Here we omit the proofs due to lack of space. In Sec. 3.1, we demonstrate the efficiency of LiCNo in practice, despite the theoretical worst case bounds.

3 EXPERIMENTS

For our experiments, we collected tweets via the streaming Twitter API, between Nov 29 to Dec 09, 2013. We focused on tweets posted from London, UK, for which we used the API's bounding box filtering mechanism. Users and Tweets are the two node types of our Content Network. There are 69K unique users and 556K tweets over that period. We generate snapshot graphs from that data using a 15-minute time window, for a total of 940 snapshot graphs. Tweets fall in a snapshot graph according to their posting time, linked to the user who posted it. Additional edges are added between users who reply to each other within that snapshot. Following this process, and once hidden links are revealed with Algorithm 1, there are 559K edges in our Content Network.

Obtaining the Ground Truth. One main challenge in event detection is the lack of ground truth. In our case, we *automatically* obtained ground truth from Wikipedia[2] and other online sources for events such as Premier League games and popular TV shows, spanning the same period of time of our collected dataset.

Methods. We consider the following techniques:

- *Activity Detector.* This method identifies events when an unexpected volume of messages is observed. To account for differences in volume patterns throughout a day, we compute the average and standard deviation for each 3-hour segment of a 24-hour cycle. Then, an event is reported when the volume exceeds the corresponding average plus θ times the standard deviation.
- *Structure Components.* A graph-based version of LiCNo using the same technique of tracking *lCC*s but no similarity linking.
- *Content Components.* A content-based version of LiCNo using the same technique of tracking *lCC*s but using only content nodes.
- *SELECT-H.* A state of the art technique using ensembles of anomaly detectors [12]. Due to space, we consider only the best variant.
- *LiCNo*: The technique presented in Section 2.
- *LiCNo (tf-idf)*: A variation of LiCNo, using tf-idf vectors, but no neural network embeddings.

Evaluation Metrics. Our evaluation consists of two parts. The first focuses on the importance of the hidden links: We generate a vector GT of length equal to the number of time windows of our dataset. GT contains binary values, where $GT_i = 1$ means an event occurred during the i-th time window, and 0 otherwise. We opt for a binary vector as some baselines (e.g. structural ones) can only report the existence or absence of an event. Similarly, each evaluated method produces a binary vector M, $|M| = |GT|$, with similar semantics. Using GT and M, we compute for each method Precision, Recall and F-score. The second part compares LiCNo against a state-of-the-art technique that does event ranking. We provide more details for this comparison in Section 3.2.

Hardware & Software Setup. We run the experiments on a machine with Intel Core i7-5820K CPU, 16 GB RAM and 64-bit Linux OS. LiCNo is in Python 2.7. SELECT-H is available in Matlab.

3.1 Event Detection Results

In Table 1(top section), we present the performance of all detection methods on the Twitter dataset. LiCNo demonstrates the best Precision and F-score overall. It is marginally second in Recall to Structure Components, which has a much lower Precision, and is different from LiCNo in that it does not consider hidden links. This clearly validates our idea for linking similar nodes.

Fig. 3 shows how well LiCNo's reported events align with the ground truth across time. This qualitatively shows the performance of LiCNo beyond the binary vector evaluation. Also, we observe that methods output larger duration for events such as soccer games, since post-game TV shows discuss the highlights of the game.

Scalability. Our theoretical analysis showed a high, worst-case complexity for LiCNo. For this reason, we evaluate its efficiency in a more realistic setup. From our data, we randomly sample 720 nodes over a very short duration of time (<1min), which we replicate in powers of two (2×, 4×, 8×). We then run LiCNo and measure the time taken to process all the data (Fig. 4 (left)). We also performed

[2] http://en.wikipedia.org/wiki/Portal:Current_events

Event Detection			
Method	Precision	Recall	F-score
Activity Detector	0.33	0.70	0.45
Structure Components	0.29	0.74	0.41
Content Components	0.39	0.49	0.43
LiCNo	0.46	0.73	0.57

Event Ranking				
Method	APrec	ARec	AF-Score	Run Time
LiCNo (tf-idf)	0.65	0.69	0.67	259 sec
LiCNo (w2v)	0.5	0.61	0.54	22733 sec
SELECT-H	0.3	0.31	0.30	24746 sec

Table 1: Predictive performance in the Twitter data

a long-running experiment, by selecting a full week, replicating it and appending it to the previous one, for a total duration of up to a year. The gray bars in Fig. 4 (right) show the time needed to process the dataset, while the red-dotted line shows the number of content nodes (tweets). These plots indicate that LiCNo has a more linear-like behavior, rather than the theoretical quadratic one.

Sensitivity Analysis. In Figure 5, we report the F-measure while varying the three main parameters of LiCNo, the similarity threshold (ϕ), the detection threshold (θ), and the number of words by which we augment the content nodes (N). We observe that the number of words N plays little role to the performance LiCNo, except for larger similarity thresholds ϕ. The tf-idf variant is better for low ϕ and θ values, but is also a lot more unstable and quickly drops in performance. On the contrary, the w2v embeddings are far more stable and outperform tf-idf in the long run.

3.2 Event Ranking Results

In this section, we compare LiCNo to the state-of-the-art SELECT-H in the task of Event Ranking. To do so, we used a ranking variation of LiCNo that sorts time windows according to the size of the largest CC, when an event is discovered. Table 1 (bottom) reports Average Precision, Recall and F-score (APrec, ARec, AF-Score) of the techniques. We observe that both versions of LiCNo outperform SELECT-H. Unlike LiCNo, which reports ranked lists only when it detects an event, SELECT-H ranks all time windows by how

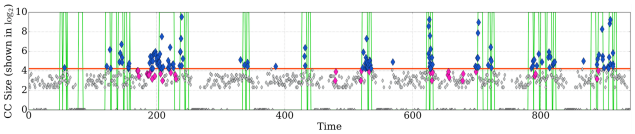

Figure 3: LiCNo's event triggers and their correlation with the timestamps of ground truth events (green lines). Events are noted as blue diamonds (exceed $vlCC$ threshold - orange horizontal line) and red diamonds (exceed lCC threshold and connected in sequential time windows). Gray colored diamonds are CCs that are not considered as events by LiCNo.

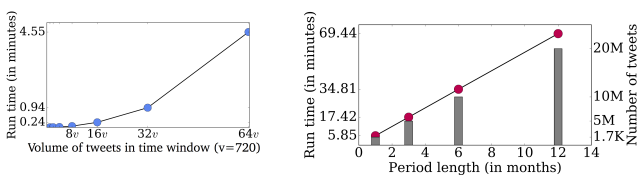

Figure 4: Scalability: left: varying volume per time window, right: varying time period (constant volume per window)

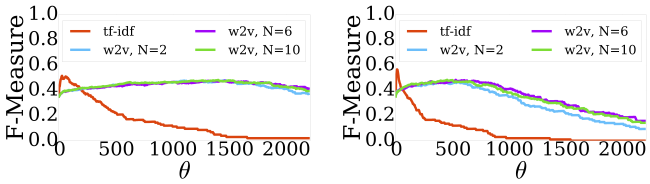

Figure 5: F-measure when varying θ for similarity threshold $\phi = 0.5$ (left) and $\phi = 0.6$ (right)

anomalous they are, even when there is nothing to report. Such behavior affects both recall and precision of SELECT-H negatively. By excluding all time segments with no ground truth events, we obtain an upper bound for the performance of SELECT-H: 0.57 AF-Score, which is still lower than the best performing LiCNo variant.

4 CONCLUSIONS AND FUTURE WORK

We presented LiCNo, a method for event detection that considers both the structure and the content information of a social network. It tracks connected components through time and utilizes them as event indicators. Our experimental evaluation on twitter data shows that LiCNo outperforms baselines and state-of-the-art approaches. For future work we intend to work on sub-event discovery and to study alternative techniques for semantic similarity in content.

ACKNOWLEDGEMENTS

This research has been financed by the European Union through the FP7 ERC IDEAS 308019 NGHCS project, the Horizon2020 688380 VaVeL project and a Google Faculty Research Award.

REFERENCES

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. EvenTweet: Online Localized Event Detection from Twitter. In *VLDB '13*.
[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery* (2015).
[3] Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter Catches the Flu: Detecting Influenza Epidemics Using Twitter. In *EMNLP '11*.
[4] Feng Chen and Daniel B. Neill. 2014. Non-parametric Scan Statistics for Event Detection and Forecasting in Heterogeneous Social Media Graphs. In *KDD '14*.
[5] F. Chierichetti, J. M. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. 2014. Event Detection via Communication Pattern Analysis. In *ICWSM 2014*.
[6] R. Kempter, V. Sintsova, C. C. Musat, and P. Pu. 2014. EmotionWatch: Visualizing Fine-Grained Emotions in Event-Related Tweets. In *ICWSM 2014*.
[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*.
[8] Miles Osborne and S Petrovic. Bieber no more: First story detection using Twitter and Wikipedia. In *SIGIR 2012 Workshop on Time-aware Information Access*.
[9] Nikolaos Panagiotou, Ioannis Katakis, and Dimitrios Gunopulos. 2016. On Event Detection from Spatial Time series for Urban Traffic Applications. In *Solving Large Scale Learning Tasks: Challenges and Algorithms*. Springer.
[10] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sanchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *KDD '14*.
[11] S. Rayana and L. Akoglu. Less is More: Building Selective Anomaly Ensembles with Application to Event Detection in Temporal Graphs. In *SDM '15*.
[12] S. Rayana and L. Akoglu. 2016. Less is More: Building Selective Anomaly Ensembles. In *Transactions on Knowledge Discovery from Data (TKDD)*.
[13] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open Domain Event Extraction from Twitter. In *KDD '12*.
[14] Polina Rozenshtein, Aris Anagnostopoulos, Aristides Gionis, and Nikolaj Tatti. 2014. Event detection in activity networks. In *KDD '14*.
[15] Takeshi Sakaki, M Okazaki, and Y Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *WWW '10*. ACM.
[16] A. Saravanou, G. Valkanas, D. Gunopulos, and G. L. Andrienko. Twitter Floods when it Rains: A Case Study of the UK Floods in early 2014. In *WWW '15*.
[17] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *WSDM '12*.
[18] George Valkanas and Dimitrios Gunopulos. How the Live Web Feels About Events. In *CIKM '13*.