Text-based Event Detection: Deciphering Date Information Using Graph Embeddings

Hilal Genc[1] and Burcu Yilmaz[2]

[1] Department of Computer Engineering
[2] Institute of Information Technologies
Gebze Technical University
{hgenc,byilmaz}@gtu.edu.tr

Abstract. Event detection is increasingly gaining attention within the fields of natural language processing and social network analysis. Graph models have always been integral to social media analysis literature. Owing to the long processing time and time complexities of graph-based algorithms, these models were initially very difficult to improve upon. Over the past few years, researchers proposed many approaches to create representations such as word2vec and doc2vec [11]. With the emergence of graph embedding techniques in recent years using deep learning techniques such as node2vec, it is possible to extract node embeddings that can be used to embed graph information into machine learning methods. We introduce SnakeGraph, a new model which uses the sequences of words making up each body of text along with key representations such as the user and the date. These representations can help us learn about the main ideas communicated via written language. However, our method not only looks at both the content of text and how it links to other key information, but also factors the relationship between words in our text as they appear in sequence and overlap as they appear across different bodies of text. We believe that date and user embeddings can especially shed light on event detection literature.

Keywords: Graph embeddings, extracting time embeddings, event detection, transfer learning.

1 Introduction

Time is a significant aspect in social network analysis. This is because it can not only help document individuals and their changing preferences and points of view but also the progress of events to a much larger scale. Information diffusion refers to the progress of events over time and it may manifest itself in different patterns. Attention given to an event may emerge in the beginning, peak and then fade away or it may continue to oscillate indefinitely. Time also has different levels of granularity ranging from seconds to months to years. Proposed models for information diffusion often consider time in units of days.

Recent years have seen an emergence in studies devoted to deep learning and language models that extract dense feature vectors called embeddings or representations. Graph representation learning has proven to be extremely useful for graph-based analysis and prediction. There has also been a rise in approaches that automatically encode graph structure (graph, subgraph, or node embeddings) into low and fixed dimensional embeddings.

In this study we represent social media data using a graph. The proposed graph embeds text information and key entities, such as date and user information corresponding to the tweets, and the relationships between them. Also we transferred a text-based news corpus and hierarchy of the entities to the training phase of the graph embedding extraction to extract the embeddings more accurately. The model that we used to extract vector representations from our graph is Node2Vec. The contributions of the paper are as follows:

1. **Gaining key information:** We can use the proposed model to extract date embeddings. To our knowledge, no graph embedding model currently exists for date embedding extraction. Although we test our method only for dates, the proposed model can be used for any other named entity.
2. **Understanding varying concepts:** The proposed method can extract varying levels of granularity for entities such as date, month, and year embeddings if we clearly define the hierarchy of these concepts. Extracting these date embeddings is the novel part of our study. It is also possible to extract village, city, and country embeddings.
3. **Graphs & Node2Vec:** The proposed model sheds light as to how graph data and the Node2Vec model can be used to model these concepts and extract embeddings.
4. **Enrichment of graph embeddings:** The time complexity is very high for graph mining algorithms. Thus we proposed to transfer non-graph data with the intention of making the model learn the embeddings more accurately. To our knowledge, transferring data to the graph domain from another domain to extract graph embeddings has never been done before. We believe this will pave the way for more advanced transfer learning methods to extract vector representations from graphs.
5. Although we did not propose a method specifically for event detection, we extracted related key concepts. We believe that these embeddings will shed light on event detection literature.

1.1 Literature Analysis

Event detection is identifying an event trigger (usually a single verb or noun [4]) from a body of text to determine what event(s) might be associated with it. An early study in event detection [8] clusters tweets through similarity measures and selects the most widely shared tweet to represent all other similar tweets. Each event trigger is introduced in [14] as a triple structure (arg_s, $verb$, arg_o) containing a verb phrase and two noun phrases representing the subject and object with respect to the action. The model filters the frame elements for noise via a probabilistic model.

The results from [9] show that convolutional neural networks alone can be very helpful for sentence classification. An LSTM model in [10] uses mean pooling to combine user embeddings, community embeddings, and word embeddings to determine whether a given post will initiate a conflict. [3] uses a bidirectional gated recurrent unit to generate the most likely hashtag that will appear in a given tweet while [15] detects the sentiment and similarity of tweets using a CNN-LSTM model. The model proposed by [1] looks at subword information on words belonging to morphologically rich languages such as Turkish and Finnish. [16] mentions the graph neural network (GNN) as a framework that computes embeddings of a node by recursively aggregating and transforming the embeddings of each node's neighbors. These neighbors have no natural ordering unlike the elements within a lattice used for image processing [7].

Retrieving node embeddings from our proposed SnakeGraph to find correlations between written text and publication date is the focus of our study. We accomplish this with the help of an effective model that extracts embeddings for the nodes in our graph. Major kinds of embeddings identified in [2] are node embeddings, edge embeddings, subgraph embeddings, and graph embeddings. Node2Vec [6], Subgraph2Vec [12], DeepWalk [13], and Author2Vec [5] have successfully proposed models for extracting these node embeddings from already existing graphs. The differences in how these algorithms preserve graph properties affect how these algorithms will preserve distances between nodes in the embedding space.

Deepwalk learns node embeddings from random walks via a semi-supervised approach. It uses skipgram, which can essentially "skip" over words. Thus, skipgram models are unlike n-grams in that they are not necessarily consecutive. With little doubt, the skipgram method would enable us to capture the similarities between words that are not in sequence but still of relevance. Node2Vec is a variation of the DeepWalk approach combining breadth first search (BFS) and depth first search (DFS). Subgraph2Vec learns node embeddings and subgraph embeddings (generated through the embeddings of nodes and a small group of their neighboring nodes) with an unsupervised approach. Author2Vec uses a graph where a node represents each author and an edge represents each collaboration between two authors. The graph does not include text-based content. We have thus created a paradigm that builds connections between key information and text information which previous models such as doc2vec and word2vec cannot accomplish.

2 Our Approach

In this section we will discuss how we created our graph from a social media dataset and used the graph to extract node embeddings that will represent our key entities. We then introduce a transfer learning model to enrich the data.

2.1 Extraction of SnakeGraph

Here will will discuss how we created the *SnakeGraph*. To show the relationship between all the words and the key entities from the dataset, we decided to create a graph.

We define a graph as G=(V,E), where $v \in V$ is a node and $e \in E$ is an edge. The node mapping function of G is $f_v : V \rightarrow T^v$ and the edge mapping function of G is $f_e : E \rightarrow T^e$.

The set of node types and the set of edge types are given by T^v and T^e, respectively. Each node $v_i \in V$ belongs to one particular type, i.e., $f_v(v_i) \in T^v$. The same applies for each edge where for $e_{i,j} \in E, f_e(e_{i,j}) \in T^e$.

To incorporate the content of each tweet, we filtered the tweets through a preprocessing phase where we remove all stop words (punctuation marks) and any cluster of characters starting with a "@" or containing "http" or any other indication of a hyperlink. We then tokenized our preprocessed data and then added lowercase of these tokens to our graph as a node one by one, essentially in the form of a snake. Each word and the consecutive word after it is connected with an edge. Hence, we named the graph *SnakeGraph* for our proposed model. Each tweet from our dataset had a tweet id, given by a unique alphanumeric sequence to distinguish any tweet from all others that might have identical content. We added each tweet id to the graph as a node for every tweet we included in our graph. The node for each tweet id was attached to the the node for first token or word of the corresponding tweet with an edge. Then the rest of the words were added in a sequence.

No two nodes in our graph are alike. The "snakes" or sequences of words overlap when the tweets have a mutual word. The words appearing in our graph are linked only to either the tweet id (if said token is the first in the tweet or sequence) or to one or two other token(s).

Figure 1 shows the appearance of two different short bodies of text as they would appear on our comprehensive *SnakeGraph* graph. The bodies of text in our figure are "besiktas ve fenerbahçe de vergi indiriminden" and "özel haber - fenerbahçe de sasirtan advocaat yasaklari". They translate to "Besiktas (football team) and fenerbahce (football team) granted tax deductions" and "Special news: Advocaat (manager) announces surprising prohibitions for fenerbahce."

2.2 Extracting Graph Embeddings

To determine the embeddings of entities such as date from our *SnakeGraph*, we use the Node2Vec model which creates node embeddings from graphs [6]. We provide the formal definitions of graph and node embeddings below.

Definition 1. Graph Embedding: For a given graph $G = (V, E)$, the graph embedding extraction converts G into a d-dimensional vector space where d « |V|. The vector space preserves the graph attributes.

Definition 2. Node embedding: Node embedding provides an embedding vector (or feature vector) $ev(u)$ as a representation for each node u. These vectors appear in a low dimensional space and nodes that have similar characteristics have similar vector representations.

We may use the node2vec algorithm to generate the embeddings $e(w)$, $e(tw)$, $e(us)$, and $e(d)$ for the words w, tweet ids tw, user ids us, and/or dates d, respectively.

Each entity (word, tweet id, user id, and/or date) in our dataset is represented by a node in our graph. The node2vec algorithm enable us to generate the embeddings for each node appearing on our graph. These embeddings allow us to find similarities between words and other particular entities appearing on our *SnakeGraph*.

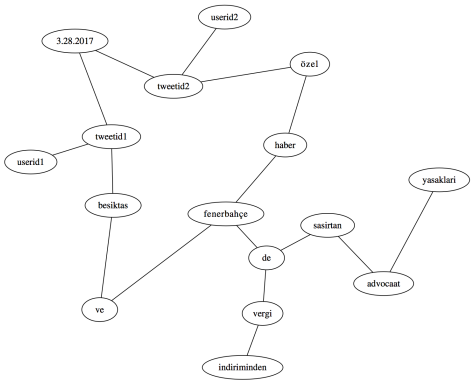


Fig. 1. Relationships between a given date on the graph and an id for a body of text, the words in it, and the author's userid.

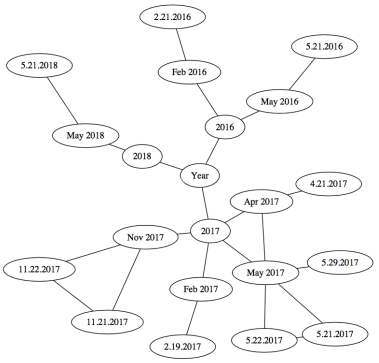


Fig. 2. Relationships between the dates as they appear on our SnakeGraph.

We developed a biased random walk procedure to explore the neighborhoods of each node with a technique using both breath first search and depth first search. We created random walks in a similar way as given by the node2vec algorithm. For a node u we created random walks with a fixed length l. In a random walk, let c_i be the ith node starting with node c_0=u. The following distribution is used to generate the random walks.

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & if(v,x) \in E \\ 0 & otherwise \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant. Transition probability π_{vx} on an edge (v, x) is used to determine the next node x after node v in the walk. $\pi_{vx} = \alpha_{pq}(t, x).w_{vx}$ where

$$\alpha_{pq}(t,x).w_{tx} = \begin{cases} \frac{1}{p} & d_{tx} = 0 \\ 1 & d_{tx} = 1 \\ \frac{1}{q} & d_{tx} = 2 \end{cases}$$

and d_{tx} denotes the shortest path distance $\in \{0, 1, 2\}$ between nodes t and x. Parameters p and q determines the random walk using an interpolation of BFS and DFS.

We created a number of random walks $rw_u \in RW$ for each node u in the *SnakeGraph* with the methodology mentioned above where the list of random walks is given by RW.

To extract an embedding $ev(u)$ for each node u, we decided to use the Node2Vec methodology which is based on the skipgram method that extracts word embeddings in natural language processing (NLP). We seek to optimize the objective mentioned as mentioned below. The objective function maximizes the log-probability of neighbourhood $N(s)$ of a node u.

$$\max_{f} \quad \sum_{n=1} logPr(N(u)|f(u))$$

2.3 Enriching Graph Embeddings with Transfer Learning

Language models have limited performance when developing models for morphologically rich languages such as Turkish, Finnish, and Czech. As expected, the Node2Vec model has longer processing time as the graph grows. The time complexity of the Node2vec model depends on the number of random walks created.

Nonetheless, we need more data to learn word embeddings. Our text information embedded into our *SnakeGraph* will not be enough to learn text embeddings accurately enough. Thus, we decided to use a news text corpus C which includes sentences $s \in S$. Each sentence includes a number of words w. We transfer the text information in the skipgram model trained on C to the Node2Vec model trained on the random walks RW extracted from the graph information. This is known as transfer learning. After training the word2vec model with a text corpus, we used the parameters of the model and use them in the Node2Vec model. Then we fine-tuned the Node2Vec model with random walks RW extracted from the graph. In the softmax layer, the model uses the tokens in the last trained corpus. This means that the model extracts the embeddings from these tokens that are derived from the random walks. In order to construct sentences for event detection, we will need more words than the ones in the *SnakeGraph*. Thus we extended it to extract embeddings $ev(u)$ of both nodes in the last trained model and the embeddings $ev(w)$ of words w in the corpus C. We used a concatenation of the corpus C and random walks RW extracted from the graph. Then we give this data $C \cup RW$ to the embedding extraction step of the Node2Vec model and expect the model to learn the relationships between the tokens extracted from the graph and to learn the semantics of the corpus. Because we trained all of the nodes and words together, the Node2vec model will extract the relationships between them using the common words across both datasets. When we get the node embeddings $ev(n \in C)$, accuracy will also be higher for embeddings such as date embeddings.

To enrich the node embeddings we used a technique to embed the hierarchy of dates. We initialize the graph with the node "Year" in order to connect all the nodes representing each year on which the contents of our dataset were published. Then we add the nodes representing the months for each year, connecting these month nodes to their corresponding years. In this step, we also connect the nodes of each consecutive month. We then add the nodes representing each relevant date by connecting each date to its corresponding month. As we connected the nodes for each consecutive month we also connect the nodes for each consecutive date.

We created a node for each date and each date node is connected to a tweet node. To embed the date hierarchy of dates we added edges between consecutive dates. That way, date embeddings will also include the relationships between

the dates. We provide a toy example in Figure 2. In this paper we only focused on date embeddings, although it is possible to similarly enrich user embeddings and embeddings for other entities. It is also possible for us to add edges between two people who are mutual followers. This way the embeddings will also take into account the relationships between users as well.

2.4 Detecting Events Using the Embeddings

In this section we try to determine the events. For each date embedding $e(d)$, we get the set of k nearest words w in the embedding space. These words are likely to have close semantic relationships between the events that happened on that day. Because there is more than one event in a day, we clustered the embeddings of words $ev(w)$. Thus, each cluster EC which is called the **event cluster** will include the words $W_{EC} \in EC$ used to mention an event. To define what these words of events correspond to we used the cosine similarity metric to get the most related words to the words in the event cluster.

3 Experiments

For our experiments, we initially created a graph *SnakeGraph* which contains the words appearing in each body of text, the tweet ids, the user ids, and the dates. The dataset from which we extracted our graph nodes contained 23,801 tweets published between February 15, 2017 and May 31, 2017. The tweets are primarily published in Turkish, but there is some English content as many of the authors are either bilingual or use English words and phrases that are well known among Turkish speakers. Our dataset contains information about each tweet such as the date on which it was published, the unique user id of the author who published it, and the unique corresponding tweet id. We chose these particular dates because our total dataset had the largest density of tweets within this time period and because these months are among those during which events planned in advance, such as football championships, are most likely to take place. Our comprehensive graph contains all of this information because we wanted a framework for which we could find connections between different kinds of entities on our dataset, i.e. dates and topics. Our comprehensive graph G_s has 74,839 nodes and 254,073 edges. Every node on the graph is unique.

We extracted the embeddings from our graphs in two ways. The first method uses "non-enriched embeddings" and we refer to it as $SnakeGraph_{nonEE}$. The second method we refer to as having "enriched embeddings" because it involves using a text-based news corpus. We refer to it as $SnakeGraph_{EE}$. For the former method, we extracted embeddings using only the contents of the graph. We define the embedding of node u as $e(u)$. This algorithm walks only through the nodes of our graph using the Node2Vec method. It also includes a hierarchy of dates. The other method involves the extraction of embeddings using the contents of the graph including hierarchy of dates along with the text from a large news corpus entirely in the Turkish language. We essentially "transfer" the information from a Turkish text corpus onto our dataset containing tweets (most of which were in Turkish). This algorithm appends the walks from our graph to the contents of the news text corpus, the input being first the text corpus and then the walks from our graph. Thus, we extracted the embeddings from the data including both the graph contents and the text corpus. We define the embedding of node u for this method as $e_T(u)$, indicating that the method involves a transfer of data.

For our experiments, we use the gensim implementations of skipgram models. For the method involving non-enriched embeddings, we use a window of 5 and 10 and for the method involving enriched embeddings we use a window of 5 for

a reasonable run time. For both methods, we set the size as 128. Our node2vec implementation is the one presented in [10].

To evaluate the extracted embeddings, we conducted several experiments.

Date	Enriched				Not Enriched							
	Window = 5				Window = 5				Window = 10			
	Topic	Meaning	Similarity Measure	Event	Topic	Meaning	Similarity Measure	Event	Topic	Meaning	Similarity Measure	Event
2.19.2017	kalinic	Nikola Kalinic	0.5151	Football team transfer	soranın	Questioner's	0.5665	Football game	kalinic	Nikola Kalinic	0.6227	Football team transfer
	adıyamandayız	At Adiyaman	0.5032	Presidential visit	adıyamandayız	At Adiyaman	0.5595	Presidential visit	adıyamandayız	At Adiyaman	0.6011	Presidential visit
	teşekkürleradıyaman	Thank you Adiyaman	0.4845	Presidential visit	kalinic	Nikola Kalinic	0.5589	Football team transfer	teşekkürleradıyaman	Thank you Adiyaman	0.5646	Presidential visit
	tolunay	Tolunay Kafkas	0.4652	Football game	teşekkürlerkahta	Thank you Kahta	0.5353	Presidential visit	tolunay	Tolunay Kafkas	0.5319	Football game
	kararımızınet	Our decision is certain	0.4424	Referendum	öncememlekettabikicvet	Nation first absolutely yes	0.5124	Referendum	poldiiii	Lukas Podolski	0.5229	Football game
2.23.2017	türkvatanının	Turkish citizen's	0.6720	Referendum	türkvatanının	Turkish citizen's	0.7247	Referendum	türkvatanının	Turkish citizen's	0.7224	Referendum
	bekasıiçnevet	Say yes for perpetuity	0.6356	Referendum	bekasıiçnevet	Say yes for perpetuity	0.7013	Referendum	bekasıiçnevet	Say yes for perpetuity	0.7002	Referendum
	nuno	Nuno Espírito Santo	0.6000	Football news update	nuno	Nuno Espírito Santo	0.6332	Football news update	nuno	Nuno Espírito Santo	0.6816	Football news update
	bundesbank	Bundesbank	0.5736	Financial statement	bundesbank	Bundesbank	0.6300	Financial statement	bundesbank	Bundesbank	0.6733	Financial statement
	türkbesd	TÜRKBESD Association	0.5700	Financial statement	haklısın	You are right	0.6207	Referendum	türkbesd	TÜRKBESD Association	0.6372	Financial statement
5.7.2017	emenikenin	Emmanuel Emenike	0.5677	Football game	emenikenin	Emmanuel Emenike	0.6194	Football game	emenikenin	Emmanuel Emenike	0.6559	Football game
	hayatlarının	To their lives	0.5411	Football game	kaptanımı	My captain	0.5953	Football game	takima	To the team	0.6206	Football game
	dabo	Basketball federation	0.5232	Outreach program	hayatlarının	To their lives	0.5914	Football game	hayatlarının	To their lives	0.5977	Football game
	takima	To the team	0.5048	Football game	dabo	Basketball federation	0.5709	Outreach program	yensen	If you win	0.5792	Football game
	yensen	If you win	0.4918	Football game	yensen	If you win	0.5562	Football game	dabo	Basketball federation	0.5763	Outreach program

Table 1. Major event for three selected dates within our dataset.

3.1 Neighborhood Embeddings of Dates

We initialize the comprehensive $SnakeGraph_{nonEE}$ including dates by hierarchically organizing the relevant date information. The core node connects all other nodes representing each year. Each node representing a year is then connected to each month corresponding to that year and each month node is connected to each date corresponding to that month. The consecutive days, months, and years are connected to an edge. We construct a graph using the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique that is used for representing date embeddings extracted from $SnakeGraph_{nonEE}$ defined above. The t-SNE is a technique for dimensionality reduction that is particularly used for the visualization of high-dimensional datasets [17]. The method through which we extracted the date embeddings on our t-SNE outputs non-enriched embeddings, has a window of ten, and sets the size to 128. Our t-SNE graph shows the neighborhood of the date embeddings spanning the months of June, July, and August of 2017. The dataset here contains 88 dates and 35,020 tweets from June 2017 to August 2017. We used a separate dataset than the one described above for event detection because we wanted our t-SNE graph to span three full months.

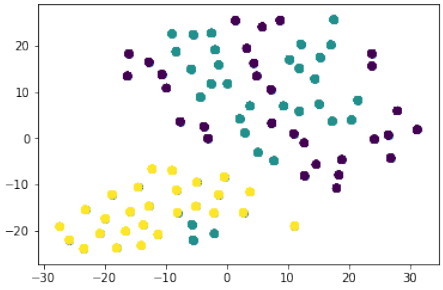


Fig. 3. The t-SNE graphic representing the dates appearing on our graph. The dates of June correspond to the indigo coordinates, the dates of July to the teal coordinates, and the dates of August to the yellow coordinates.

Figure 3 shows the t-SNE graph of the dates appearing in the dataset between June 2017 and August 2017 and extracted from $SnakeGraph_{nonEE}$. Rather than

showing three separate clusters for the three different months, the t-SNE graph shows clusters where each cluster corresponds to a group of days with similar characteristics. The overlap between some of the months, most notably between the months of June and July, most likely results from the repetition of events such as football matches.

To detect how the similarity between a specific day and its consecutive days changes we got 16 days and drew the similarity curves for each day. A similarity curve for a specific day is the cosine similarity of the day and the 33 consecutive days after it. The results of two days are given by figure 4(a). We then get the average of the similarity curves for 16 consecutive days. The average similarity curve is presented in figure 4(b). We observed that events have different trends. While some of the events can boost and diminish over time by having an oscillating behavior. This corresponds to the instant events. Some of the events can have slight oscillating behavior indefinitely. This may correspond to periodic events. For similar reasons the similarity curve may have an oscillating behavior. Football matches occurring repeatedly might cause the days of each of the match to have similar embeddings. As we see in figure 4(b) the trendline is decreasing which shows that, on the average, the events fade slightly.

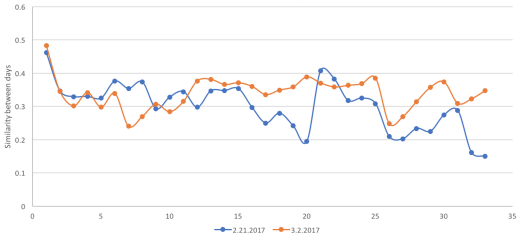
3.2 Event Detection with Human Evaluation

Because we trained both the dates and the words appearing in our dataset on the same comprehensive graph, we map both the date and word embeddings onto the same embedding space. Thus, we find the cosine similarity between the date and word embeddings. For each date, we extract the N most similar entities (words, user ids, tweet ids, and/or dates) along with their respective cosine similarity measures, with N being the number of entities to extract. After getting the most similar entities, we filtered out all the entities that were not word tokens appearing in our dataset. We extracted the embeddings for each most similar word by using the graph data and, for the method involving enriched embeddings, the Turkish news text corpus. Our dataset for event detection used the dataset of tweets published between February 15, 2017 and May 31, 2017.

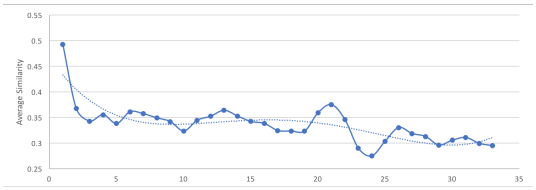
In our dataset, we selected 17 days and extracted the most similar words for each given date. The days that we selected were days on which one of Turkey's major football teams was competing as reported by the UEFA. Table 1 shows results for three different days on which some of the most similar words directly correlated to an event associated with that day or during that time period. Most of the events we found occurred on the same day as the day users mentioned or referenced them. A human evaluator helped to confirm the correlation between events and the similar words. All events in Table 1 occurred on the same date except for the referendum, which occurred on April 16, 2017. Nonetheless, the days associated with the referendum were days on which users mentioned the upcoming event.

Despite having a small dataset of only 23,801, our model made it possible for us to perform event detection by identifying crucial nouns or verbs appearing in our dataset based on the dates on which users mentioned them. These words were all associated with an event that occurred during or near the time on which they were mentioned. Although we had to look up each date on twitter in order to directly understand the correlation between these words and an event, a familiarity with the meanings and relevant topics of the words can give one an idea of what kind of event might have occurred that day. It appears that a large dataset will give us very strong results for detecting relevant events occurring on a given day.

In information retrieval the two most common evaluation techniques are precision, the percentage of extracted results which are relevant, and recall, the percentage of total relevant results correctly classified by the algorithm. We only



(a) The similarities between a day and its consecutive 33 days for two specific days.



(b) The average similarities of all days and their consecutive 33 days.

Fig. 4.

calculated the precision because of difficulties meeting the hardware requirements of processing all the tweets in our designated period of time. From the dates within the span of our dataset, we selected the same 17 days as mentioned earlier and collected the top ten relevant words for each date. We calculated the percentage of words which correlated to an event either occurring on that day or a very publicized upcoming event. Our calculated precision measure is given by the percentage of our top ten words for each of 17 dates correlated to an event. A human evaluator determined whether or not each of the 170 words was relevant to any event. We calculated the precisions for the model with enriched embeddings with a window of 5, the model without enriched embeddings with a window of 5, and the model without enriched embeddings with a window of 10. Because we could not train our model with enriched embeddings for a window of 10 within a reasonable amount of time, we did not calculate the corresponding precision results. We observed that the average precision value is 72.9% for the model with enriched embeddings and window of 5, 71.2% for the model without enriched embeddings and window of 5, and 71.2% for the model without enriched embeddings and window of 10. Table 2 shows our results.

Evaluation Results		
Enriched Embeddings	Window Size: 5	Window Size: 10
With	72.9% (124/170)	-
Without	71.2% (121/170)	71.2% (121/170)

Table 2. Precision measures of top ten tokens for 17 selected dates in our dataset.

With a larger dataset we can more automatically find the events by also looking for the closest words to our potential events. For example, if a given date is most closely related to the word “basketball” then that would be a potential event. The most similar words to “basketball” would give better insight on what kind of event occurred on that given date or in recent days or what kind of event is projected to occur.

4 Conclusion

We introduced a new paradigm for extracting key information from a text-based dataset of social media posts. Our contribution is a novel way of creating a graph to map the relationships between the entities within our text-based dataset. The embeddings for these graphs, along with a new way of enriching our embeddings, have helped us successfully identify correlations between dates and major indicators of events. Our model works on even a small dataset, and we demonstrate

that graphical models are powerful for building correlations between bodies of text, events, and dates. We hope that our research can pave the way for further studies in event detection.

Acknowledgment. This project (No. 117E566) is funded by the Scientific and Technological Research Council of Turkey (TUBITAK).

References

1. Bojanowski, Piotr, et al. *Enriching word vectors with subword information.* Transactions of the Association for Computational Linguistics 5 (2017): 135-146.
2. Cai, Hongyun, Vincent W. Zheng, and Kevin Chen-Chuan Chang. *A comprehensive survey of graph embedding: Problems, techniques, and applications.* IEEE Transactions on Knowledge and Data Engineering 30.9 (2018): 1616-1637.
3. Dhingra, Bhuwan, et al. *Tweet2vec: Character-based distributed representations for social media.* The 54th Annual Meeting of the Association for Computational Linguistics. 2016.
4. Feng, Xiaocheng, et al. *A language-independent neural network for event detection.* Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Vol. 2. 2016.
5. Ganguly, Soumyajit, et al. *Author2vec: Learning author representations by combining content and link information.* Proceedings of the 25th International Conference Companion on World Wide Web. International World Wide Web Conferences Steering Committee, 2016.
6. Grover, Aditya, and Jure Leskovec. *node2vec: Scalable feature learning for networks.* Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016.
7. Hamilton, Will, Zhitao Ying, and Jure Leskovec. *Inductive representation learning on large graphs.* Advances in Neural Information Processing Systems. 2017.
8. Ifrim, Georgiana, Bichen Shi, and Igor Brigadir. *Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering.* SNOW-DC@ WWW. 2014.
9. Kim, Yoon. *Convolutional Neural Networks for Sentence Classification.* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.
10. Kumar, Srijan, et al. *Community Interaction and Conflict on the Web.* Proceedings of The Web Conference (WWW). 2018.
11. Liu, Yang, et al. *Topical word embeddings.* Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.
12. Narayanan, Annamalai, et al. *subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs.* arXiv preprint arXiv:1606.08928 (2016).
13. Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. *Deepwalk: Online learning of social representations.* Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
14. Qin, Yanxia, et al. *Frame-Based Representation for Event Detection on Twitter.* IEICE TRANSACTIONS on Information and Systems 101.4 (2018): 1180-1188.
15. Vosoughi, Soroush, Prashanth Vijayaraghavan, and Deb Roy. *Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder.* Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.
16. Xu, Keyulu, et al. *How Powerful are Graph Neural Networks?* arXiv preprint arXiv:1810.00826 (2018).
17. L. van der Maaten, and G. Hinton. *Visualizing Data using t-SNE* Journal of Machine Learning Research. 2008.