CAS2VEC: Network-Agnostic Cascade Prediction in Online Social Networks

Zekarias T. Kefato[1], Nasrullah Sheikh[1], Leila Bahri[2], Amira Soliman[2],Alberto Montresor[1] and Sarunas Girdzijauskas[2]
[1] University of Trento
Trento, Italy
Email: {zekarias.kefato,nasrullah.sheikh,alberto.montresor}@unitn.it
[2] KTH - Royal Institute of Technology
Stockholm, Sweden
Email: {lbahri,aaeh,sarunasg}@kth.se

***Abstract*—Effectively predicting whether a given post or tweet is going to become viral in online social networks is of paramount importance for several applications, such as trend and break-out forecasting. While several attempts towards this end exist, most of the current approaches rely on features extracted from the underlying network structure over which the content spreads. Recent studies have shown, however, that prediction can be effectively performed with very little structural information about the network, or even with no structural information at all.**

In this study we propose a novel network-agnostic approach called CAS2VEC, that models information cascades as time series and discretizes them using time slices. For the actual prediction task we have adopted a technique from the natural language processing community. The particular choice of the technique is mainly inspired by an empirical observation on the strong similarity between the distribution of discretized values occurrence in cascades and words occurrence in natural language documents. Thus, thanks to such a technique for sentence classification using convolutional neural networks, CAS2VEC can predict whether a cascade is going to become viral or not. We have performed extensive experiments on two widely used real-world datasets for cascade prediction, that demonstrate the effectiveness of our algorithm against strong baselines.

I. INTRODUCTION

In online social networks, it is common to see posts or tweets that start from a few sources and then suddenly spread like a wildfire. Just to mention a recent example, the post celebrating the landing of the Falcon-Heavy rocket sent from the SpaceX[1] Twitter account on February 6th, 2018, has been retweeted more than 75k times within the same day of posting. Such diffusion events are called *viral cascades*. Predicting cascades virality is vital for different applications, for example to forecast trends and rumor break-outs [1]. However, it is challenging to effectively predict the virality of such kinds of events as early as possible, especially when little supporting information is available. Many research works have dedicated effort and attention to the prediction of content popularity with the focus of achieving good predictions in the shortest possible time, with the least information possible about the underlying network structure.

[1]https://twitter.com/SpaceX

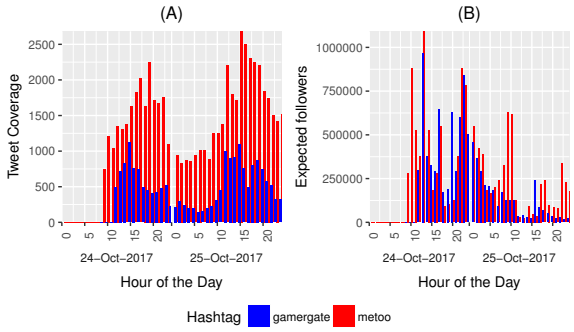

Fig. 1. Examples of two recent hashtag campaigns. (A) The tweeting frequency of each hashtag; `#metoo` achieved more spread compared to `#gamergate`. (B) The network properties of the participating nodes in each hashtag in terms of average number of followers; the nodes engaged in the first 12 hours almost achieve similar reachability in both hashtags.

On social networks platforms, content is usually diffused over the underlying social graph that represents the connections among the users in these frameworks. Early research on predicting cascade virality assumed strong correlations between the spread of content and structural properties of users who started these events. Therefore, most of the early attempts towards predicting the virality of cascades have relied on manually extracted features from the underlying network structure and the cascade itself [2]–[7]. Information such as the number of followers/followees that engaged users have, users connections and community structure, activity level, etc., have been exploited.

This, however, poses two kinds of issues. First, manual feature crafting is an expensive and challenging task. In most cases, domain knowledge and external information about the content in question is required. For instance, content popularity may be linked to several parameters, such as event topic, external events or the content relevance to given periods of time (e.g., posting about football during world cup period),

978-1-5386-9588-3/18/$31.00 ©2018 IEEE

etc. Besides, the optimal number and relevance of features that need to be extracted is not obvious, making it difficult to decide when to stop looking for additional ones [8]. Furthermore, some recent cascade examples show different spread patterns even when showing similar network properties of the engaged nodes in the underlying social graph; thus, network properties may not be the best or the only indicator for virality. For example, Fig. 1(A) shows the spread patterns of two hashtag campaigns `#metoo` and `#gamergate` that happened almost at the same time[2]. As shown, `#metoo` went viral in the first two days. The hashtag *#metoo* was tweeted more than 200k times by the end of October 15, 2017[3]. On the other hand, `#gamergate` did not become viral like `#metoo` even though they have reasonably similar network properties such as the expected number of followers (indicator for potential spread in the future), as shown in Fig. 1(B).

In addition to that, acquiring information about the social network structure is usually very expensive for those who work outside the companies hosting the data. For example, for popular social networks such as Twitter and Facebook, it may take several months to extract just a portion of the network. Moreover, due to privacy constraints and policies of such systems, the extracted network is usually lacking a significant amount of structural information, such as edges of some users participating in hashtag campaigns who set their connections to be private [9].

For the reasons above, it becomes important to design algorithms that do not require any type of features or information about the underlying network, but still are able to effectively predict cascade virality in the very early stages of the spread. Some initial but also strong attempts towards exploring this *network-agnostic* approach have already demonstrated the potential for effective and timely prediction based only on information that could be learned from the cascades themselves without requiring any other additional information [10]. However, most of the works available in the literature are mainly adopting either *"network-aware"* or at best *"quasi-network-agnostic"* approaches [1], relying on "less expensive" structural information, such as node degrees.

In this paper we propose a novel *network-agnostic* algorithm that predicts cascade virality based only on information explicitly available in the cascade itself (i.e., *the time between share events*). Our main premise is that the reaction time between the sequence of events encoded in a cascade should be a sufficient indicator to whether it will become viral or not in the near future. The reaction times in the early sequence of events can be used to model the cascade initial speed (i.e., the speed by which a cascade starts its spread), as well as its momentum. Analyzing the distribution of reaction times for viral and non-viral cascades on multiple datasets, and based on corroborating observation supporting our premise, we have modeled cascades as a time series, where each element of the series is the reaction time measured from the source signal. Furthermore, our work is partly inspired by iSAX [11], that is used for indexing time series data. Particularly, we apply a similar technique as iSAX on cascades to transform them into instances of one-dimensional point processes in time space, such that each time series of a cascade is transformed into discrete values by using equally-sized periods of times.

For the actual prediction task, we adopt a technique from the Natural Language Processing (NLP) community that has been used for sentence classification [12]. The algorithm exploits a deep convolutional neural network (CNN) model to effectively predict sentences; instead of sentences, we feed the neural network with the transformed time series. The choice of this model is inspired by an empirical observation that shows a similarity between the distribution of words appearance in sentences and the distribution of discretized values appearance in cascades (time series). Furthermore, CNNs achieve performances as good as RNNs, which are a natural fit in such settings, but they can be trained more easily.

Summary of contributions: CAS2VEC provides a novel network-agnostic approach that models information cascades as time series and discretizes them using time slices. Furthermore, CAS2VEC can predict whether a cascade is going to become viral or not based only on the timestamps of the events encoded in the cascade itself. To show the effectiveness of our algorithm in cascade prediction, we have performed extensive experiments and compared it against strong baselines. Our results show that CAS2VEC outperforms them by an increase between 10% and 20% in all the tasks.

The rest of the paper is organized as the following. Section III briefly describes some basic cascade definitions and assumptions that are related to our model. Section IV illustrates the design of our algorithm. We discuss the experimental evaluation of CAS2VEC against strong baselines in Section V. Then, Section II briefly describes the state-of-the-art methods used for cascade virality prediction. Finally, Section VI concludes the paper.

II. Related work

Many research works have dedicated effort to the prediction of web content popularity with the focus on achieving (i) good predictions (ii) in the shortest possible time windows and (iii) using the least possible information. Related research predicts cascades development either in terms of the potential size they can grow to (i.e., regression approach [1], [13]–[15]), or in terms of classifying them as viral or not-viral (i.e., classification approach [6], [14], [16], [17]). In both approaches, most works are based on either topological information or on features such as temporal properties, structure of the cascade at its first stage, the content in question, the early adopters, etc.

Other studies, however, have utilized little or no network information [1], [10], [18]–[20]. Recent studies predict content popularity based on *point process models* and node degree [1], [20], where as another study uses survival analysis technique and follows a network-agnostic approach [10]. Under the

[2]The dataset for these two hashtags is collected based on information available via https://github.com/datacamp/datacamp-metoo-analysis and https://github.com/awesomedata/awesome-public-datasets, respectively

[3]https://en.wikipedia.org/wiki/Me_Too_movement

regression approach, some works have taken the direction of predicting the optimum future size of a cascade (e.g., [1]), whereas others have provided time-based predictions of the cascade growth function (e.g., [15]). Regardless of the approach taken, most works have been based on either topological information or on features such as temporal properties, structure of the cascade at its first stage, the content in question, the source or key early adopters, etc.

On one hand, some works have based on generative models of these factors as distributions or stochastic processes that interpret the event series in the cascade [15], [21]. On the other hand, other works based on representative models through handcrafted and heuristic based features that are mainly extracted from knowledge about the domain and the content in question. These features are integrated using discriminative machine learning algorithms that can be used to achieve either the classification or the regression tasks [6], [14], [16].

In our study, we adopt a fully network-agnostic and domain insensitive approach, where only information available in the cascade is being deployed. We investigate a complementary approach using deep CNNs. A related approach has been taken by a very recent work [22], however with different cascade modeling schemes and classification algorithms.

III. Model and definitions

Cascades naturally capture a series of share events associated with the infection of users. Given that we are adopting a network-agnostic approach, we shall have no assumptions regarding the underlying connectivity of users, and we will simply consider a cascade as a sequence of events.

A *sequence of events* S is represented by the ordered list of *timestamps* at which the events occur:

$$S = [t_1, ..., t_s]$$

Timestamps are measured relatively to the first event of the sequence, so $t_1 = 0$. Since events are ordered, we have that $i < j \Rightarrow t_i < t_j$.

We use $S[i] = t_i$ to denote the timestamp of the i^{th} event. We write $t \in S$ to denote the fact that t corresponds to an event that has been included in S, i.e. $S[i] = t$ for some i between 1 and $|S|$, the length of the sequence.

We use $S(t_b, t_e)$ to denote the sub-sequence of events whose timestamps are between the beginning time t_b (included) and the end time t_e (excluded):

$$S(t_b, t_e) = [t : t \in S \wedge t_b \leq t < t_e]$$

For the sake of brevity, we use $S(t_e)$ to denote the prefix of the subsequence including the events occurring before t_e since the initial event $t_1 = 0$, i.e.

$$S(t_e) = S(t_1, t_e)$$

A *cascade* is an actual sequence of share events recorded from an online social network; the set of cascades $\mathcal{C} = \{S_1, S_2, S_3, \cdots, S_m\}$ are available as input of our problem.

There are several prediction tasks that can be computed over cascades; in this paper, we will focus on *virality prediction*, i.e. the task of deciding whether a cascade, after an observation period, is going viral or not before a given amount of time. From a practical perspective, this is one of the most important issue [10].

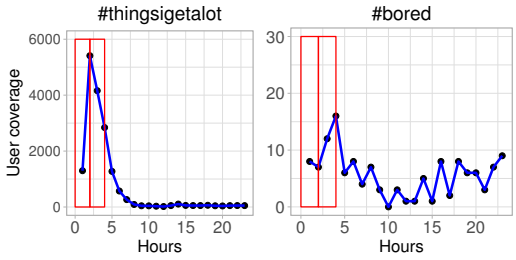

Fig. 2. Two slices of size 2 hours, applied to the user coverage distribution of a viral hashtag (`#thingsigetalot`) and non-viral hashtag (`#bored`), which have reached 13711 and 43 users in an observation window size of 4 hours.

To formally define the virality prediction problem, we consider the subsequence $O = C(t_o)$ of events occurring from the beginning of the cascade C up to an *observation time* t_o. We call such subsequence an *observation* of C; the period of time between 0 and t_o is called the *observation window*.

Given an observation $O = C(t_o)$, a *prediction window* is period starting from time t_o and lasting Δ time units, after which we want to establish whether a specific cascade is going viral or not. For this purpose, we consider the number of events $|C(t_o + \Delta)|$ that have occurred in C by the *prediction time* $t_p = t_o + \Delta$.

Similar to existing studies [3], [10], we consider two ways of determining whether a cascade is viral or not:

- through an absolute threshold $\theta_a \in \mathbb{R}^+$, the cascade C is viral if $|C(t_o + \Delta)| \geq \theta_a$;
- through a relative threshold $\theta_r \in (0, 1)$, the cascade C is viral if $|C(t_o + \Delta)| \geq |perc(C, \theta_r)|$, where $perc(C, \theta_r)$ is the θ_r-percentile among the cascades in $\mathcal{C}$.

Our problem is thus the following: we seek to predict whether a cascade C is going to be viral by the prediction time $t_p = t_o + \Delta$, by inspecting its observation $C(t_o)$.

IV. cas2vec

The design of our algorithm is inspired by the observation that most viral cascades spread like a wildfire within the very first few hours. In contrast, non-viral cascades require several hours just to reach merely a handful of users. For instance, Fig. 2 shows the user coverage distribution of two hashtags in a 24-hour period, one viral (`#thingsigetalot`) and one not (`#bored`).

Some state-of-the-art studies [1], [20], [23] start from the above assumption and develop elegant solutions based on *point processes*. Such techniques rely on the frequency (density) estimation of the rate of cascade growth during its observation period to predict its ultimate size after a certain period Δ.

Our approach is partially related, in the sense that it implicitly utilizes the rate of growth of the number of events within an observation period. However, it is completely network-agnostic. Based on our main premise, intuitively we seek to

model the initial speed of a cascade (that is, the speed by which a cascade starts its spread) or the user reaction times at the early stage of the cascade, as well as its momentum. As we shall empirically demonstrate in Section IV-A, this is a strong signal for potential virality.

From a high-level point of view, our solution is organized as follows. For each cascade C in our training data set $\mathcal{C}$, we perform three operations:

- we extract the observation $C(t_o)$, where t_o is the observation time at which the observation period ends and the prediction starts;
- we preprocess the observation $C(t_o)$ by transforming it into a format that can be fed to our classification task;
- we label the cascade C as viral or not viral, based on the threshold θ according to the number of events observed at time $t_o + \Delta$, as discussed in the previous section.

Using the transformed sequences and their associated labels, we train our classifier based on an 1D convolutional neural network.

A. Preprocessing Cascades

The observation period is divided into a collection of *slices*, i.e. equally-sized time windows. For example, Fig. 2 illustrates an observation window of 4 hours, divided in two slices of 2 hours each, visualized through red boxes. Slices are identified by the *slice size* t_s; the size of the observation window t_o should be an integer multiple of t_s, such that the *number of slices* N_s is equal to t_o/t_s.

Based on the slices, we generate the following two kinds of preprocessed sequences:

a) Counter sequence: the sequence of integers representing the number of events included in each slice:

$$C^c = [\ |C(i \cdot t_s, (i+1) \cdot t_s)| : 0 \leq i < N_s\ ]$$

b) Discrete sequence: the sequence generated by discretizing all the events within each slices, *i.e.* by assigning each event within a slice the index of the slice itself.

$$C^d = [\ \lceil C[i]/t_s \rceil : 1 \leq i \leq |C|\ ]$$

For example, look again at Fig. 2 with cascades C_1 (`#thingsigetalot`) and C_2 (`#bored`). By considering an observation window size of 4 hours and a slice size of 2 hours, the counter sequences are equal to $C_1^c = [6\,709, 7\,002]$ and $C_2^c = [15, 28]$; in the former, there are 6 709 events in the first 2 hours, and 7 002 in the second 2 hours. In the later, the numbers are just 15 and 28. The discrete sequences are equal to:

$$C_1^d = [\overbrace{1, \ldots, 1}^{6\,709}, \overbrace{2, \ldots, 2}^{7\,002}]$$
$$C_2^d = [\overbrace{1, \ldots, 1}^{15}, \overbrace{2, \ldots, 2}^{28}]$$

Counter sequences and discrete sequences have different predicting power. Besides, counter sequences are much faster to train as a result of a fixed length of training sequences, *i.e.* N_s, while discrete sequences gives us the flexibility of choosing larger values for the length of sequences at the expense of slower training time.

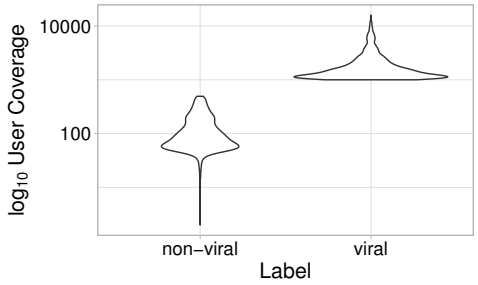

Fig. 3. The distribution of the user coverages for the viral and non-viral classes. The user coverage distribution is computed at observation time t_o as $|C(t_o)|$ and virality is computed at prediction time $t_o + \Delta$. A cascade is viral if $|C(t_o + \Delta)| \geq 1,000$ and not-viral if $|C(t_o + \Delta)| \leq 500$

Based on our assumption regarding the dynamics of viral and non-viral cascades, we base our algorithm on the following conjecture:

Conjecture 1: Consider two cascades C_1 and C_2 and an absolute threshold θ. Given an observation t_o and a prediction windows size Δ. If the cascade sizes of C_1 and C_2 at time $t_o + \Delta$ are such that $|C_1(t_o + \Delta)| \geq \theta$ and $|C_2(t_o + \Delta)| \ll \theta$, then $|C_1(t_o)| \gg |C_2(t_o)|$.

According to the conjecture, within the observation window, we expect a significant number of events for viral cascades and very few of them for the non-viral ones. For example, looking again at Fig. 2, we have 13 711 events for the viral hashtag `#thingsigetalot` and just 43 events for the non-viral hashtag `#bored`. More generally, the user coverage distribution for the two classes, shown in Fig. 3, further establishes an empirical case for the conjecture.

B. CNN model for cascade prediction

Once cascades are preprocessed using slices, we adopt the CNN model [12] to predict whether they will go viral or not.

The architecture of the model [12] adopted for cascade prediction is shown in Fig. 4. Originally this model was proposed for sentence classification in natural language documents, and it has been shown to be effective for this classification task. In addition, our choice is motivated by recent studies that have shown the CNN-based models outperform existing state-of-the-art techniques in time-series classification tasks [24], [25].

For the sake of being self-contained, we give an overview of the model; however, because of space limitations, we will be restricted to a brief description sufficient enough to replicate our results. Interested users are referred to the original paper [12]. Instead of words in sentence classification, we have the discretized values (numbers) obtained by transforming the sequences as shown in Section IV-A. The input of the model is a preprocessed sequence C_i (e.g., a sequence of counter, labeled as *preprocessed sequence input* in Fig. 4). Each input c_i in the sequence is represented by an embedding vector $\mathbf{c_i} \in \mathbf{R}^d$. The entire sequence is denoted by a matrix, referred

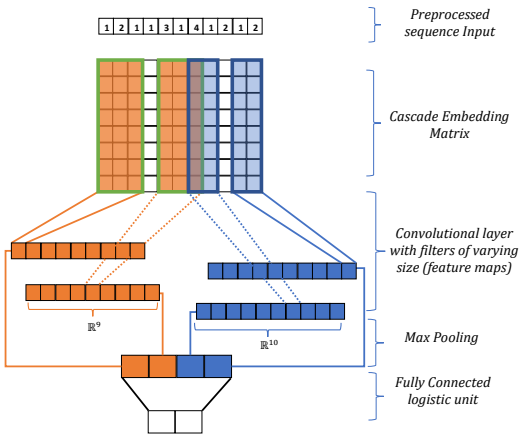

Fig. 4. The CNN model adopted for cascade prediction

to as *cascade embedding matrix* in Fig. 4, and it is denoted by $\mathbf{M} = [\mathbf{c_1}, \ldots, \mathbf{c_s}]$ where s is the length of the sequence.

Assume the model that we are going to describe is trained and all its parameters are tuned to their optimal values. Then, the prediction task starts by applying a set of p filters on the cascade embedding matrix in the convolutional layer. That is, we apply p filters (denoted by different colors) of different sizes on every possible slice of the input (the cascade embedding matrix). More formally,

$$\phi_l = f(\mathbf{w_i} \cdot \mathbf{m_k} + b)$$

where the vector $\mathbf{w_i} \in \mathbb{R}^{kd}$ is the i^{th} filter, $b \in \mathbb{R}$ is the bias, f is an activation function, such as *relu*, k is the size of the i^{th} filter, and ϕ_l is the l^{th} feature value. $\mathbf{m_k} = \mathbf{M}[j] \oplus \ldots \oplus \mathbf{M}[j+k] \in \mathbb{R}^{kd}$ is a concatenation of the k–columns of the matrix $\mathbf{M}$. Generally, the i^{th} filter of size k is applied $s-k+1$ times, to give a feature map $\boldsymbol{\phi_i} = [\phi_{i,1}, \ldots, \phi_{i,s-k+1}]$. $\boldsymbol{\phi_i}$ captures patterns in high-level features, such as *n–grams* in language documents. In our setting this corresponds to patterns within small sub-sequences depending on the filter size.

Next, a max-pooling (or a max-overtime-pooling) operation is applied over each feature map, which is simply a $\max(\boldsymbol{\phi_i}) = \widehat{\phi_i} \in \mathbb{R}$ of each feature map $\boldsymbol{\phi_i}$. Intuitively, this corresponds to selecting the best feature that is activated when a certain pattern in the input space is detected. The max-pooling output, more formally $\mathbf{z} = [\widehat{\phi_1}, \ldots, \widehat{\phi_p}]$, is followed by a fully connected logistic classification layer. The vector $\mathbf{z}$ can be viewed as the final set of features extracted for the current cascade, and it will be used to predict the cascade into one of the two classes $y = \{1 = \text{viral}, 0 = \text{non–viral}\}$.

c) Training the Model: The above description assumes that the model is trained; to perform the training, the optimization objective of the model is specified as the minimization of the misclassification error of the preprocessed sequences. More formally, we adopt the standard binary cross-entropy objective function:

$$\min \sum_i y_i \log(h(S_i)) + (1 - y_i) \log(1 - h(S_i))$$

Here, S_i and $y_i \in \{0, 1\}$ are the i^{th} preprocessed sequence and class label, respectively. h is the proposed model that produces a probability distribution (prediction) $\mathbf{y}$ for the given input sequences S over the classes (viral and non-viral):

$$\mathbf{y} = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{v}) + b$$

where $\mathbf{v}$ is a Bernoulli distribution used for dropout regularization as proposed in [12].

Ultimately, the model parameters $[\mathbf{M}, b, \mathbf{w_i}, \mathbf{w}]$ are trained using the back-propagation algorithm.

V. Experiments and Results

In this section, we report on the experiments we performed to evaluate our approach. Before discussing the actual results, we introduce the datasets that have been used as input; we discuss the competing approaches against which we compare our results; and finally, we describe the experiment settings.

A. Datasets

We have evaluated our approach over two well-known datasets:

- *Twitter*: This dataset has been widely used for cascade prediction [1], [10]. It contains a full month of Twitter data from October, 7^{th} to November 7^{th}, 2011. There are a total of 166,076 tweets that have been retweeted at least 50 times.
- *Weibo*: This dataset contains 225,126 tweets recorded on the Chinese micro-blogging site Weibo [4], [5].

B. Baselines

We have compared our algorithm against three competing approaches; nevertheless, some well-known baselines have not been included, because their source code is not available [10].

- SEISMIC: This is a recent, state-of-the-art study that predicts the popularity of tweets using a *self-exciting point process* model [1]. It estimates the infectiousness of a tweet at time t, based on the number of reshares R_t at time t, then the estimated infectiousness is used to predict the ultimate size R_∞ of the tweet. We follow a similar strategy as [10] to label tweets based on R_∞, that is viral if and only if $R_\infty \geq \theta$. We have used the source code provided by the authors [4].
- Logistic Regression (LOR): This baseline has been used in previous studies [6], [10]. We use a set of features $X = [x(1), \ldots, x(N_s)]$ computed based on the notion of slices in Section IV-A, where $x(i)$ is the number of users in slice i and N_s is the number of slices.
- Linear Regression (LR): This is also a baseline similar to the one used in [1], [10]. It is specified as:

$$\log R_\infty = \log(\alpha \cdot R_t) + b + \epsilon,$$

where ϵ is a noise term with Gaussian distribution. We apply a similar thresholding as we did with SEISMIC to label R_∞ as viral and non-viral, taking into account the log transformation.

[4]http://snap.stanford.edu/seismic/

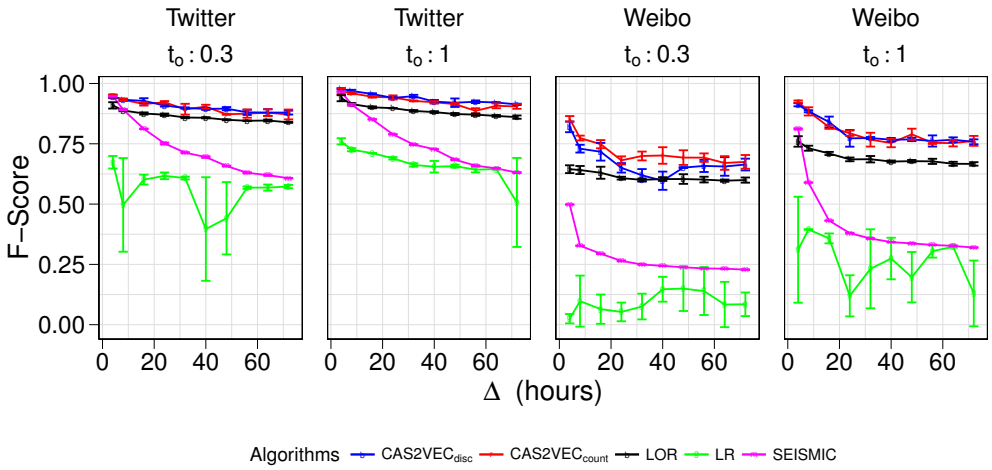

Fig. 5. Virality prediction results for both of our datasets. For Twitter, *filter sizes* = 3, 5, 7 and for each filter we have 16 of them. For Weibo, *filter sizes* = 2, 4, 5, 7 and for each filter we have 64 of them. For both datasets, the embedding size d is 128, the number of units in the fully connected layer is 32, and the *number of slices* is 40.

C. Evaluation Settings

To evaluate the performance of our algorithm against the baselines, we have used the following settings. Recall that the prediction problem is based on an observation time t_o and a prediction window Δ. So, in all the reported results for all the classification algorithms, we have trained a single classifier for every given value of Δ. Furthermore, since the class distribution is highly skewed and the viral class is very rare, we use down-sampling in all the experiments.

We tune the hyper-parameters, e.g. the number and size of filters, using a *development set* (*dev-set*) during the training process. Once the hyper-parameters are tuned, then for all the experiments we fix the parameters at these values and evaluate the performance of algorithms. Towards this end, we have used a 3-fold cross validation that does not include the dev-set and reported the average result along with the error margins.

Similar to previous studies [10], we have used are the F-score with $\beta = 3$ (since it is a rare class prediction) and recall as the evaluation metric . In all the experiments, the threshold for labeling cascades is $\theta_a = 700$ that is equivalent to $\theta_r \approx 98\%$.

D. Results

d) Predicting Virality: In the first set of experiments, we evaluate the performance of our algorithm and the baselines in predicting the virality of cascades based on a given observation t_o and prediction window Δ expressed in hours. Here, our goal is to evaluate the performance of algorithms in effectively classifying both classes. Fig. 5 reports the evaluation results. All the variants of our algorithm (CAS2VEC$_{count}$, CAS2VEC$_{disc}$, CAS2VEC$_{fusion}$) outperform the baselines, and provide very similar results. The strongest baselines are SEISMIC and LOR; in the Twitter dataset, SEISMIC achieves F-scores between 94% and 60% for $t_o = 0.3$ hours and between 96% and 63% for $t_o = 1$ hour. LOR is more robust than SEISMIC and it achieves F-scores between 90% and 83% for $t_o = 0.3$ and between 93% and 86% for $t_o = 1$ hour. Whereas, CAS2VEC variants are very robust in predicting far in the future than all the baselines and achieves F-scores between 97% and 88% and between 97% and 91% for $t_o = 0.3$ and $t_o = 1$ hour respectively.

For the Weibo dataset, LOR achieves F-scores between 64% and 59% for $t_o = 0.3$ hour and between 75% and 66% for $t_o = 1$ hour. SEISMIC's performance on Weibo is poor and it achieves F-scores between 49% and 22% and between 81% and 31% for $t_o = 0.3$ and $t_o = 1$ hour respectively. CAS2VEC on the other hand achieves a significantly high performance, which is more than the performance of other baselines by at least 10%, *i.e.* F-scores between 85% and 67% for $t_o = 0.3$ hour and between 92% and 76% for $t_o = 1$ hour.

In the following, unless stated otherwise, we focus on CAS2VEC$_{count}$, as it is faster to train.

The above experiments give us a perspective on how far an algorithm can effectively predict in the future stages of a cascade life. As we can see from the plots, performance decreases as Δ increases, as it is difficult to predict far in the future. This, however, does not tell us how early can an algorithm predicts a virality.

e) Early Prediction: The next step is to analyze how early in time virality can be predicted. Subbian et al. have observed that most of the events occur within twice the median virality time measured over all the cascades [10]. In our datasets, the median time to virality is 8 hours for Twitter and 17 hours for Weibo. Based on that, we select a distinct (but fixed) prediction time $t_p = t_o + \Delta$ for each of the dataset, i.e. 16 hours for Twitter and 34 hours for Weibo.

We then vary the size of the prediction window size Δ, from 1 hour to $t_p - 1$ hours and evaluate how early the algorithms can predict virality. Parameter Δ is similar to the time-to-virality parameter defined in [10]. Note that having fixed the prediction time, these means that the observation time t_o varies inversely w.r.t. the prediction windows size, from $t_p - 1$ hours to 1 hour. In both cases, the variation step is 1 hour.

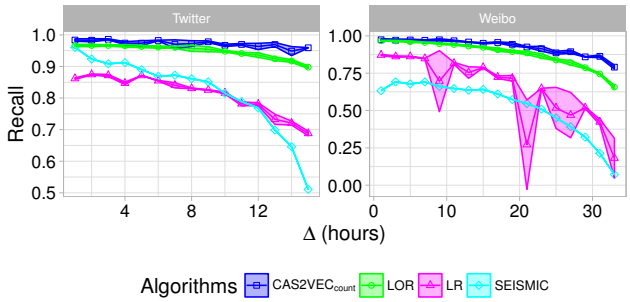

Fig. 6. Evaluation results of early prediction experiments for the Twitter and Weibo datasets. The same hyper-parameter values as Fig. 5 is used

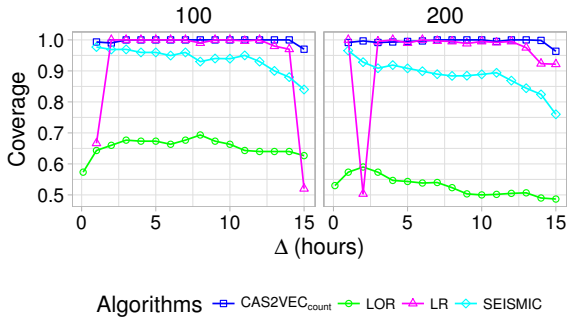

Fig. 7. Break-out coverage for $k = 100$ and $k = 200$ for the Twitter dataset.

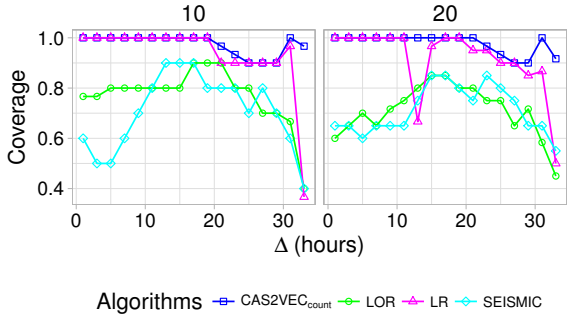

Fig. 8. Break-out coverage for $k = 10$ and $k = 20$ for the Weibo dataset.

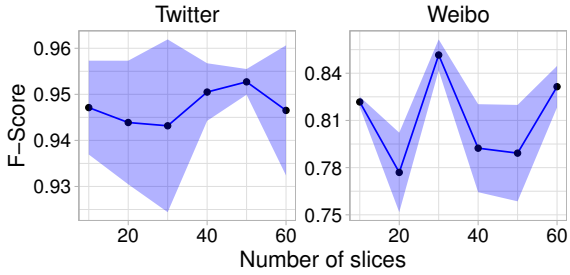

Fig. 9. Effect of the number of slices on virality prediction at $t_o = 1$ hour and $\Delta = 12$ hours.

In the following experiment (Fig. 6), we evaluate the recall score only for the viral calls, that is measured by the fraction of viral cascades detected by an algorithm out of all the viral cascades.

CAS2VEC obtains the best result in all of them. As one might expect, the algorithms achieve good results for small values of Δ. For example, all algorithms except linear regression achieve more than 96% recall in the Twitter dataset, with SEISMIC achieving the highest of all, i.e. 99%. Such result is trivial, however, and we want algorithms to be robust in their prediction as we increase Δ.

As we get close to $\Delta = 15$ hours (Twitter) and $\Delta = 33$ hours (Weibo), which is equivalent to observing the cascade growth just for 1 hour, the performance of the baselines drop faster than CAS2VEC, which achieves the best recall. SEISMIC achieves the best results up to $\Delta = 8$, which is after observing for more than 7 and 25 hours for Twitter and Weibo respectively. However, as we go beyond, SEISMIC starts to decrease quickly; when $\Delta = 15$ (Twitter) and $\Delta = 34$ (Weibo), it recalls only 86% and 42% of the viral cascades, respectively. The other strong baseline, LOR, at the end points drops to 89% and 56% of recall, while CAS2VEC outperforms the baselines and gets to 95% and 62% for Twitter and Weibo respectively.

Besides the virality predictions shown previously, these experiments demonstrate that CAS2VEC is highly robust compared to the state-of-the-art method, SEISMIC, and the strong baseline, LOR, in predicting cascades virality as early as possible.

f) Break-out Coverage: One of the important tasks in cascade prediction is detecting break-out events. Towards this end, similar to [1], [10], we take the top-k viral cascades and evaluate the performance of algorithms in effectively covering such cascades in their prediction. That is, the fraction of correctly predicted cascades out of the top-k viral cascades.

The results of this experiment are reported in Fig. 7-8. Yet again, CAS2VEC consistently achieves a significant performance gain, specially as Δ increases. Note that even though LOR was a strong baseline in the earlier experiments, its performance degrades when it comes to detecting just the top-k break-out cascades. Similar to the previous experiment, it is important to achieve a high coverage as we increase Δ. In particular, the strongest baseline in this experiment achieves only 83% break-out coverage for $k = 100$, and 76% for $k = 200$. CAS2VEC, however, achieves a remarkable performance of 95% and 90% for $k = 100$ and $k = 200$, respectively. For the Weibo dataset all the baselines score below 50% and 60%, where as CAS2VEC achieves a more than 90% for $k = 10$ and 20, respectively.

g) Effect of hyper-parameters: In order to further validate our proposal, we conducted two brief experiments on the effect of its hyper-parameters. First, we analyzed how the performance varies with the number of slices. As shown in Fig. 9, the performance increases as we increase the number of slices – up to a certain value. For Twitter, as we go from 10 to 30 the performance drops and starts to improve until we get to $N_s = 50$, which is the best spot. Where as in Weibo the best F-score is achieved at $N_s = 30$. We have found out that values between 30 and 50 give the best results.

The other hyper-parameter of our algorithm is sequence

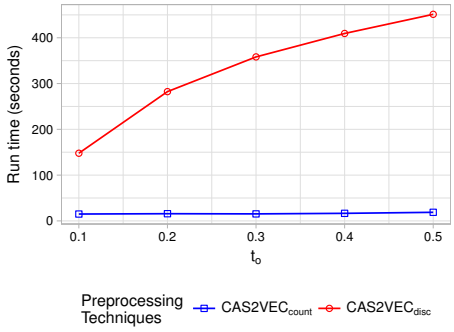

Fig. 10. Effect of sequence length on running time.

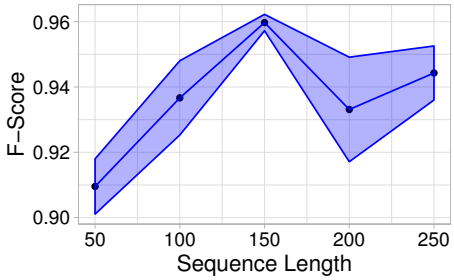

Fig. 11. Effect of seq. length on virality prediction.

length; in particular, it is the major factor in the run time of our algorithm. Fig. 10 show the effect of sequence length (determined by t_o) for the two variants of our algorithms. Particularly CAS2VEC$_{disc}$ requires more time to finish an epoch as we increase the sequence length. However, Fig. 11 shows that increasing the sequence length beyond a certain value (150 in the figure) does not give significant performance gain.

VI. Conclusions

This paper presents CAS2VEC, a novel algorithm for the prediction of the virality of social network cascades. Traditionally, network structure parameters and features extracted from the underlying domain have been used to perform the prediction either as a regression or a classification tasks. However, network information and predictive features are expensive to get either because of privacy constraints or need for domain knowledge and awareness on external affecting factors.

Unlike previous work, our approach is fully network-agnostic: it is solely based on timing information explicitly encoded in the cascade. We make use of state-of-the-art techniques for sentence classification using CNNs for the actual prediction over time series. Our experiments show that time sequences in cascades are sufficient to make timely and accurate virality predictions.

CAS2VEC achieves an increase in prediction accuracy between 10% and 20% in all the tasks w.r.t. F-score and recall compared to strong baselines in the field, while being fully network-agnostic. As future work, we plan to extend the model by incorporating extra features like content, early adopters, and analyze the interpretability of the learned features.

References

[1] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "SEISMIC: A self-exciting point process model for predicting tweet popularity," in *Proc. of KDD'15*, ACM, 2015.

[2] C. Li, J. Ma, X. Guo, and Q. Mei, "DeepCas: An end-to-end predictor of information cascades," in *Proc. of WWW'17*, Int. World Wide Web Conferences Steering Committee, 2017.

[3] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality prediction and community structure in social networks," *Sci. Rep.*, vol. 3, no. 2522, 2013.

[4] J. Zhang, J. Tang, J. Li, Y. Liu, and C. Xing, "Who influenced you? predicting retweet via social influence locality," *ACM Trans. Knowl. Discov. Data*, vol. 9, pp. 25:1–25:26, Apr. 2015.

[5] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li, "Social influence locality for modeling retweeting behaviors," in *Proc. of IJCAI'13*, pp. 2761–2767, AAAI Press, 2013.

[6] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?," in *Proc. of WWW'14*, (New York, NY, USA), pp. 925–936, ACM, 2014.

[7] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Commun. ACM*, vol. 53, pp. 80–88, Aug. 2010.

[8] A. Grover and J. Leskovec, "Node2Vec: Scalable feature learning for networks," in *Proc. of the KDD'16*, (New York, NY, USA), pp. 855–864, ACM, 2016.

[9] E. Sadikov, M. Medina, J. Leskovec, and H. Garcia-Molina, "Correcting for missing data in information cascades," in *Proc. of WSDM'11*, pp. 55–64, ACM, 2011.

[10] K. Subbian, B. A. Prakash, and L. Adamic, "Detecting large reshare cascades in social networks," in *Proc. of WWW'17*, pp. 597–605, Int. World Wide Web Conferences Steering Committee, 2017.

[11] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "iSAX 2.0: Indexing and mining one billion time series," in *Proc. of ICDM'10*, (Washington, DC, USA), pp. 58–67, IEEE Computer Society, 2010.

[12] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. of EMNLP'14*, pp. 1746–1751, 2014.

[13] O. Tsur and A. Rappoport, "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities," in *Proc. of WSDM'12*, pp. 643–652, ACM.

[14] L. Weng, F. Menczer, and Y. Ahn, "Predicting successful memes using network and community structure," in *Proc. of ICWSM'14*, The AAAI Press, 2014.

[15] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang, "From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics," in *Proc. of ICDM'15*, pp. 559–568, 2015.

[16] P. Cui, S. Jin, L. Yu, F. Wang, W. Zhu, and S. Yang, "Cascading outbreak prediction in networks: a data-driven approach," in *Proc. of the KDD'13*, pp. 901–909, ACM, 2013.

[17] M. Jenders, G. Kasneci, and F. Naumann, "Analyzing and predicting viral tweets," in *Proc. of WWW'13*, ACM, 2013.

[18] D. Agarwal, B.-C. Chen, and P. Elango, "Spatio-temporal models for estimating click-through rate," in *Proc. of WWW'09*, (New York, NY, USA), pp. 21–30, ACM, 2009.

[19] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system," *Proc. of the National Academy of Sciences*, vol. 105, no. 41, pp. 15649–15653, 2008.

[20] S. Gao, J. Ma, and Z. Chen, "Modeling and predicting retweeting dynamics on microblogging platforms," in *Proc. of WSDM'15*, (New York, NY, USA), pp. 107–116, ACM, 2015.

[21] C. Bauckhage, K. Kersting, and F. Hadiji, "Mathematical models of fads explain the temporal dynamics of internet memes.," in *Proc. of ICWSM'13*, 2013.

[22] C. Gou, H. Shen, P. Du, D. Wu, Y. Liu, and X. Cheng, "Learning sequential features for cascade outbreak prediction," *Knowledge and Information Systems*, pp. 1–19, 2018.

[23] S.-H. Yang and H. Zha, "Mixture of mutually exciting processes for viral diffusion," in *Proc. of ICML'13*, pp. II–1–II–9, JMLR.org, 2013.

[24] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, pp. 162–169, Feb 2017.

[25] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016.