Predicting the Shape and Peak Time of News Article Views

Yaser Keneshloo*, Shuguang Wang†, Eui-Hong (Sam) Han†, Naren Ramakrishnan*
*Discovery Analytics Center, Virginia Tech,
yaserkl@vt.edu,naren@cs.vt.edu
†The Washington Post, Washington D.C.
shuguang.wang@washpost.com,sam.han@washpost.com

***Abstract*—Predicting the popularity of news articles—whether measured via retweets, clicks, or views—is an important problem for editors, journalists, and readers alike. In this paper, we introduce a new model to predict the shape of news article views, and use this model to determine when an article will likely reach its maximum number of views. Although volume prediction for news articles has been extensively studied predicting when a burst of views will happen, in what shape, and by how much, remains an open problem. We engineer several classes of features (metadata, contextual or content-based, temporal, and social), develop models for view shape classification, with particular attention paid to performing online, time-updated, prediction, i.e., using data before and during the early stages of article prediction to predict its eventual peak views and update earlier predictions. The system presented here is an emerging application being developed at The Washington Post and can be used to support article placement, updating, and promotion strategies.**

***Keywords*-Peak Prediction, Shape Prediction, Time Series.**

I. Introduction

Most everyday content that we experience in our online lives exhibit a bursty lifecycle, e.g., stock quotes [1], posts on social media such as Twitter and Facebook [2], and viral videos on YouTube [3]. For our purposes here, a burst can be characterized as a brief period of intensive activity followed by a long period of nothingness [4]. Explanations of bursty behavior revolve around a threshold behavior [5] wherein, for instance, people share or read about an article once a handful of his or her friends share or read that article. This reinforcing behavior around an object leads to a burst of activity.

Our focus here is to determine when an object of interest will likely reach its maximum views (or shares or likes). This is different from prior studies which focused primarily on either aggregate prediction or burst detection. Aggregate prediction is the task of estimating the total (but not peak) number of views or the total size of a cascade, e.g., [6], [7], [8], [9], [10], [11]. Burst detection aims to detect (but not predict) the burst time as it happens, e.g., see [12], [13].

Shape and peak time prediction in the context of news articles is very challenging for at least two reasons. First, traditional regression models cannot be directly applied due to the quick rise-and-fall pattern of bursts [14]. Secondly, different articles exhibit significantly distinct lifespans. For instance, we later explain in Section III-A2 that sports articles have much shorter lifespans than political articles at The Washington Post.

In this paper, we propose two approaches to predict peak time for news articles. First, we adopted a classification method similar to [18]. We extract many features and then train several classifiers to do the prediction with them. However, results of this approach are not good enough. We present a better alternative online approach. The online method that we propose combines a time-series clustering method called KSC [15] and a time-series detection method called $SpikeM$ [14] for the purpose of peak and shape prediction for news articles. Evaluation shows our online approach significantly outperforms classification approach in all settings.

Given the promising results of online approach, we are deploying the model at The Washington Post to support prediction for all published news articles. Everyday millions of visitors read thousands of articles at The Washington Post and generate hundreds of millions of page views. In order to process this tremendous amount of data in real time, we had to adopt the Spark Streaming infrastructure. Section IV presents more details on the deployment.

II. Related Work

Related work can be grouped into two categories: burst time detection and burst time prediction.

In burst time detection, we are given the entire time series and our goal is to spot bursts in its. This area of research is very prominent in the literature as there are significant applications, e.g., in telecommunications wherein we aim to detect network anomalies using bursts in the number of packages sent/received, in financial time series analysis wherein we aim to spot sharp price fluctuations in the stocks [1], or in detecting web trends [16]. Bursts in such time series are typically detected by dividing the time series into equal or variable windows and detecting the one that exhibits an anomaly [1].

In burst time prediction, the problem is, given the current time series, to predict whether we will witness a burst in the near future or not. Traditionally, classical linear methods such as auto regression (AR) and moving average (MA) [17] have been utilized in this purpose. However, as stated earlier, the quick rise-and-fall property of practical time

series makes such approaches inadequate and necessitates the development of better models. One of the well known approaches to modeling bursts is the work of Kleinberg [13], wherein he suggested to model the stream using an infinite state automaton so that bursts correspond to state transitions. Bursts associated with state transitions constitute a nested structure where a long burst of low intensity potentially contains several bursts of higher intensity inside it, in a recursive way. Using this method, Parikh et al. [12] proposed a method to rank the bursts in an ecommerce query setting and classify them based on their shape. In similar work [18], the burst detection problem is modeled as a binary classification problem wherein if a time series has a burst after a specific time μ, its respective class would be positive (and negative otherwise). To predict the exact burst time of the time series, the author suggests an iterative solution and classification for every desired point in time. This is a serious drawback since every potential burst point in a time series requires a new classifier to be trained. We show that the traditional way of predicting burst by classifying to binary classes does not necessarily provide the best result and we demonstrate how our proposed method overcomes these limitations.

III. Proposed Frameworks

In this section, we describe two approaches to model the shape and detecting the peak time of a news article. First is the classification approach that is inspired by a previous work [18]. Then we will present a better online alternative to this task.

A. Prediction as Classification

We predict shape and peak time of a news article in a two step classification process. We first transform the shape prediction task into a multi-class classification task. We identify four clusters of possible shapes of news articles published by The Washington Post, then we predict which cluster a news article belong to. This step provides basis of burst behavior of a news article. Using the collected features, we then transform the peak prediction task into a binary classification task for each of four clusters of articles, i.e., to predict if it is going to be a peak for a news article in next 30 mins, 1 hour, 2 hours, or 3 hours. This approach is very similar to [18], and we explore more features than the original approach. We use this method as our baseline and compare our proposed method with this classification.

1) Feature Extraction: We extract four sets of features as enumerated in Table I. Many of these features have been proven to be very useful in a similar task of predicting click counts for news articles in [6], we believe they are useful in this task as well. We also introduce a few new features for peak prediction task such as using burst time information of historical articles which is explained in Table I. We explored temporal features proposed by [18] as well, but through our experiments, they do not contribute to the performance of the model.

We have collected features from all the articles that are accessed from September 2014 until October 2014 at The Washington Post. To estimate the freshness of a news article, we are comparing recent articles to archived historical articles. In our data, articles published before September 2014 are considered as historical articles.

2) Time Series Shape Identification: It is easier to predict the burst time of the article once we have an initial understanding of the pattern of views for a news article. In order to predict the pattern of click counts for a news article, we use a combination of clustering and classification. We first cluster all the time series in our training dataset. Then, we consider these clusters as class labels and use meta data and contextual features that are extracted from a news article before its publication to classify them. We adopted a recent clustering method KSC [15] to cluster time-series of training news articles into four cluster. Other clustering methods such as K-Shape [19] could be used as well. The choice of number of clusters is reached using Silhouette testing [**?**], in which a clustering with four clusters reached the highest score.

Fig 1 represents the cluster center for these four clusters along with a pie chart representing the distribution of burst time for articles in each cluster. The time series represented in Fig 1 shows an exponential rise and power low fall which is inline with the result of previous works [14], [15], [20]. The exponential rise in the time series is the reason why the burst time is leaned toward the left side of the plot.

The clusters are ordered so that C1 contains the most number of articles and C4 contains the least number of articles. Each cluster has its own characteristics and represents a set of specific articles:

- **Cluster 1**: C1 represents articles that have an extreme bursty behavior. More than 42% of articles in our dataset fall into this cluster, which shows that most of the news article have a very short lifespan and are only popular in a very short period of time. These news article are mostly Sports and National related topics. It is surprising that 70% of these articles have their peak four hours after their publication and only 17% have their peak before 30 minutes. This means that although there is a bursty behavior in the viewing pattern of these articles, despite the expected behavior, this bursty behavior will not happen right after the publication of news.
- **Cluster 2**: C2 represents articles that have a sudden burst and follow-up views by people. More than 74% of the articles in this cluster have their peak within 2 hours after publication. These are stories that are likely to go viral very rapidly and people talk about it even days after their burst time. These articles are mostly blog articles and usually talk about Politics and Opinions.
- **Cluster 3**: C3 represents articles that become popular

Table I: Features extracted from historical dataset

Metadata Feature	
Article Type	Each news is either a blog or article
Article Category	BookWorld, Editorial-Opinion, Food, etc
Article Section	Business, Lifestyle, Local, etc
Publication Date	We divide this feature to the time of the day and day of the week that an article is published
Author Name	There are more than 12k authors in our dataset where authors such as Valerie Strauss and Dan Steinberg has the most published article
Contextual Feature	
Sentiment	We extract both sentiment of the title and main article. The sentiment is defined as probabilities into four categories of negative, positive, compound, and neutral
Name Entities	Number of persons, location, and organization in an article
Readability of Text	Five measures that captures different aspects of readability of a document
Freshness of Article	1- Topic Intersection 2- Click count of 10 most similar articles and its average 3- Content similarity of 10 most similar articles and its average 4- Number of similar articles in the historical dataset 5- Maximum similarity and maximum click counts 6- Number of viral articles in the top 10 most similar articles 7- Is the article with maximum similarity a viral article? 8- Is the article with maximum count a viral article? 9- Burst time and burst time slot of the article with maximum similarity 10- Burst time and burst time slot of the article with maximum count
Temporal Feature	
FirstViewTimeDiff	Time difference between the publishing time and first page view
First 30 Minute View	Number of page views after 30 minutes of publication
Page View Acceleration	How fast an article is being read within the first 30 minutes
Page view Time-series *Normalized Page view Time-series*	The first 30 minutes time-series of views in a 5-minute time interval
Social Features	
FirstTweetTimeDiff	Time difference between the publishing time and first tweet
First 30 Minute tweet	Number of tweets after 30 minutes of publication
First 30 Minute Followers' Number	Number of followers of users who post the news within 30 minutes
Tweet/Follower Acceleration	How fast an article is being tweeted within the first 30 minutes
Tweet/Follower Time-series *Normalized Tweet/Follower Time-series*	The first 30 minutes time-series of tweets in a 5-minute time interval

gradually. No article in this cluster has its peak within one hour after its publication. Almost 78% of these articles have their peak after four hours of their publication. This behavior is completely aligned with the shape of the cluster. These are the stories that go popular slowly and lose their popularity at a similar pace. These news content are almost equally divided to blogs and articles on almost all topics. People will gain interest in blog articles after a good amount of people talk about it and that is the reason for the slow increase of number of views in the beginning.

- **Cluster 4**: C4 has a similar behavior to C1, with a small difference that there is a small follow-up after the peak. These articles covers Sports, National, and Foreign topics and usually are more trending than the news article in C1. More percentage of articles have their peaks in the first 30 minutes than those in C1 cluster. At the same time, there are fewer articles to peak 4 hours after their publication.

From Fig 1, the life-span of a news article could be captured by these four clusters. We can predict the shape of viewing pattern of a news article by classifying it into one of these four classes. Besides metadata and contextual features, we introduce some additional effective features based on an interesting observation. Fig 2 enumerates 8 out of 10 most similar time-series to the given query news article. As you can see, the plot of most of these time series has a similar shape to the time series of the query article. We can use this resemblance between the time series of query article and its top ten similar articles to find the shape of the query article. The cluster number of these top ten most similar time-series are also used as additional features in the classification task.

Once we collect all features, we train a multi-class Random Forest classifier and use 10-fold cross validation to verify the performance on our data. The result is shown in Table II. Table III reports the detailed Precision, Recall, and Fmeasure value for each cluster. As you can see in this table, we have a good classification result for the first

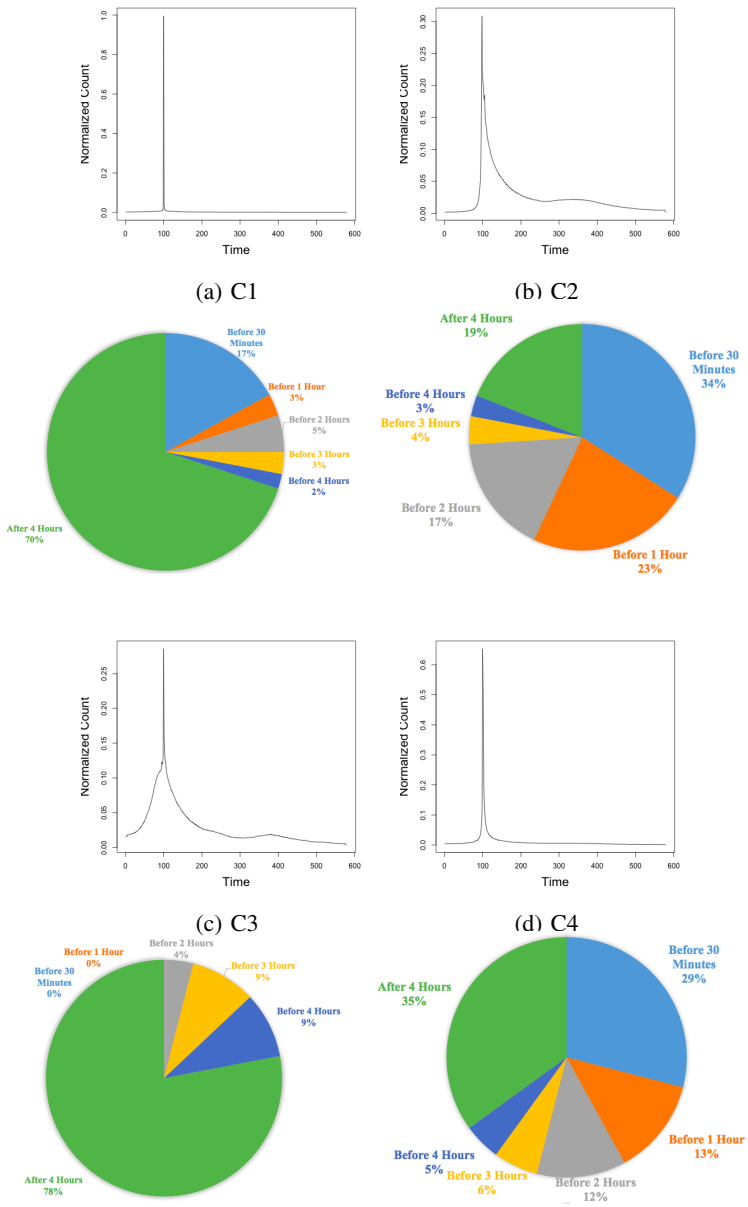

Figure 1: Different cascading pattern of news articles along with the pie chart representing the portion of news articles that have their peak before a specific time for each cluster.

three clusters, while a poor performance on cluster C4. As explained earlier, the reason for this poor performance is the abundance of all types of article in this category, which makes it hard for the classifier to distinguish the articles in this cluster from other clusters.

3) Peak Time Prediction: We have identifyed four possible viewing patterns of a new article, and we need to estimate the approximate peak time of the news article. We treat it as a binary classification task. In [18], the author proposed CPB (Classifiers to Predict the Bursts) which is a method that by using multiple classifiers in each time frame tries to predict whether we will have a peak in the μ future time windows. Therefore, the main prediction problem changes to a binary classification. The major problem with this method is that it generates a new classifier for each time frame and each value of μ. On the other hand, it collects temporal features and update them incrementally as they move along the time series of the news article. In our proposed method, we only use the features that are collected within the first 30 minutes after publication of an article.

Table II: Result of shape prediction by classifying the news article using features that are extracted before its publication.

Precision	Recall	Fmeasure	Weighted Precision	Weighted Recall	Weighted Fmeasure
0.7	0.7	0.7	0.7	0.7	0.68

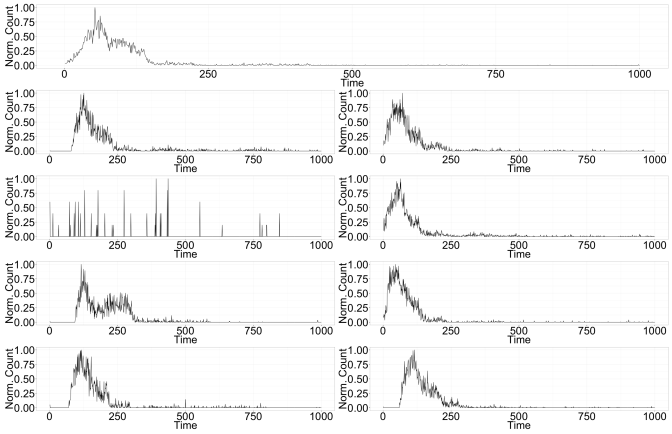

Figure 2: An example of the queried article (the most top plot) along with the plot of time series for its top eight most similar articles to it.

Table III: The detailed performance of the shape prediction method representing how the model works for each cluster

Metric/Classes	1	2	3	4
Precision	0.67	0.67	0.73	0.66
Recall	0.73	0.56	0.94	0.28
Fmeasure	0.70	0.61	0.82	0.39

Our method could be considered as a simpler version of the CPB, however it is as effective as this method. We first explain CPB method and then explain our model in terms of this model. Each time-series in CPB is divided into K time windows and it extracts various temporal features using these K time windows. A spike is the $f(k_{max})$ that satisfies $\forall 1 \leq k \leq K, f(k_{max}) \geq f(k)$, and k_{max} is the time window of the spike and $f(.)$ is the number of page views at any given time interval. Given two constants K and μ and the observed cascade spreading in time interval $[t_0, t_{current}]$, where t_0 is the starting time of the cascade and $t_{current}$ is the current time, the μ_{th} future time interval is defined as $[t_{current} + \frac{\mu-1}{K} \times (t_{current} - t_0), \frac{\mu}{K} \times (t_{current} - t_0)]$. Given this definition, CPB tries to find whether we see a peak in the μ future time window. On the other hand, by solving the above problem incrementally, it is able to predict in which time window the burst will appear.

We create 4 different classifiers to predict whether we will have a burst within 30 minutes, within an hour, within two hours, and within three hours after publication of an article. In order to make the prediction for the first 30 minutes, we only use the contextual and metadata feature, while for the rest of the classifiers we use the whole feature set. We consider all the articles that have their peak before a specific time, as the positive class and all the other articles as negative class. Therefore, we are trying to have a high true positive while reducing the false positive. Table IV presents the result of our SVM classifiers for the peak prediction.

The recall value represents how well we can classify articles in the positive class. As you can see in this table, in all the experiment the recall value is close to maximum, meaning that if an article have a peak before that specified time, we will be able to report that. However, the low precision value shows that we also report lots of articles that have their peak after that specified time, as the ones that have their peak before it. Therefore, we have to try to reduce the amount of false positive rate in our classifiers. As you can see in Table IV, the precision value increases as we try to make prediction for a farther future. This is due to the fact as we try to predict for farther future, each dataset turns to look like a balanced dataset and number of positive and negative samples tend to get closer. This means that rather dealing with a rare-class classification problem in the beginning, we are dealing with a balanced dataset in the future.

Table IV: Result of predicting approximate time-frame where a peak would occur for an article using classification framework

	Precision	Recall	Fscore	#Pos
First 30 Minutes	0.19	0.99	0.31	3747
First One Hour	0.28	0.99	0.43	5512
First Two Hours	0.36	0.99	0.52	7201
First Three Hours	0.41	0.99	0.57	8151

B. Online Peak Prediction

In Section III-A, we described a classification method to predict the peak of news articles. This method is an offline method since the prediction is done once and model is not updated with changes of article behaviors. After exploring a rather comprehensive set of features with latest classification models, it is difficult to achieve a significantly better prediction results with the this classification framework. Alternatively, we propose an online method. We track real time click time-series of each news articles after their publication and identify similar historical click time-series of previous articles and use them to predict shape and peak times for newly published news articles. In this method, we use the *SpikeM* algorithm [14] to predict whether we will have a peak in the future time intervals. Although *Spikem* is originally created for time-series detection, we are the first to use it for predicting peak time of news articles. In the following section, we shortly explain the *SpikeM* model and

how we adapt this method to predict the shape and peak time of a news article.

1) SpikeM *Algorithm:* The *SpikeM* algorithm is designed to model the rise and fall in the time-series pattern of different events. This model can take into account time-series characteristics such as exponential rise and power-law fall pattern, periodicities, and avoiding divergence to infinity. *SpikeM* model uses a handful of parameters that each of them captures one of these characteristics. The main equation for detecting the pattern of the time-series is as follows:

$$\Delta B(n+1) = p(n+1).\Big(U(n).\sum_{t=n_b}^{n}(\Delta B(t)+ S(t)).f(n+t-1)+\epsilon\Big) \quad (1)$$

Where $\Delta B(n)$ is the number of people who just clicked on the article, $U(n)$ is the number of un-informed people[1], $S(t)$ represents when the time-series peaks, ϵ is the noise parameter of the effect of external activities around the article which is usually $\epsilon \approx 0$, $f(t)$ is the amount of infectiveness of an article at time t which is defined as $f(t) = \beta \times t^{-1.5}$. As you can see it is a decaying function over time which means that as we move along time the infectiveness of the article also fades away. β represents the interestingness of the article and a low β value means that no one cares about this news. $p(n)$ captures the periodicity of the event and has the following equation:

$$p(n) = 1 - \frac{1}{2}P_a(\sin(\frac{2\pi}{P_p}(n+P_s)+1) \quad (2)$$

The periodicity of the time-series is controlled using three parameters which are the strength of periodicity, P_a, the actual period, P_p, and phase shift of the periodicity P_s. The periodicity captures mainly the fact that readers tone down their activity during the night.

SpikeM uses the *Levenberg-Marquardt (LM)* to minimize the sum of the errors between the actual number of viewers at time n, $X(n)$ and the estimated $\Delta B(n)$:

$$D(X,\theta) = \sum_{n=1}^{n_d}(X(n) - \Delta B(n))^2 \quad (3)$$

where n_d is the duration of the time-series. We consider the *SpikeM* model as a smoothing algorithm that can be used to provide a better resolution of the time-series in our historical dataset. Through our experiments, we find out that this model provide a much better result than other smoothing algorithms such as LOESS smoother [21].

2) SpikeM *for Shape Prediction:* In [14], the author explained how to use *SpikeM* algorithm in order to use it for forecasting the volume and pattern of the next time-intervals. According to this method, we have to wait until we see the first spike in a time-series to be able to predict the rest of the time-series. This comes from the fact that once an event like "Harry Potter" movies, shows a pattern in the past, it pretty much shows the same pattern of user activities in the future. Therefore, using the spike patterns in the past, we would be able to infer *SpikeM* parameters and then use them on the new time-series to predict the future. Although, this hypothesis highly relies on the fact that similar events would entice the same behavior in user activities, it would not have any solution for events that does not have a history.

Therefore, in order to use the *SpikeM* model, we have to use the model in a way that it uses the pattern of similar articles in the past and use them for prediction. To solve this problem, we suggest two approaches to find the similar time-series to an article. As mentioned in Section III-A2, for each news article in our dataset, we extract top ten similar articles to it. As we argued in Section III-A2 and according to Fig 2, the viewing pattern of the queried time-series is quite similar to the viewing pattern of these top ten articles. Therefore, these are the strongest candidates for finding the similar pattern.

In the second approach, we find the similarity on all our historical dataset. This could be viewed as the method that is used by Google Correlate[2], which finds the most similar time-series to the input time-series in their dataset.

In both of these methods, we try to find the most similar article to the queried article. We later explain in this section different measures to identify the most similar time-series to the queried article. Having similar events with similar viewing pattern is all we need to use *SpikeM* algorithm to predict the future time-intervals of the queried article. Our solution on using the top ten articles and *SpikeM* consists of the following steps:

- For each queried article, we extract top ten similar articles to it and extract their time-series.
- For each of these ten time-series, we fit the *SpikeM* model and store their respective parameters.
- We use the parameters of these ten models to create ten time-series. As we move along the time, we compare the time-series of the current article with these ten time-series. We use four different measures to estimate the closeness of the each of these ten time-series to the queried article.
- Please note that as we move along the time, for different time intervals, we find the most similar time-series. This is due to the fact that in first few hours after the publication of article, we see a lot of fluctuation in the article that is chosen as the most similar articles.

The above-mentioned approach could also be used for finding the most similar article in the whole historical dataset. In this method, rather finding the most similar article among only the top ten articles, we find the similarity of the queried article with the time-series in the whole dataset.

Using the above-mentioned procedure, we are able to have an online prediction for all articles that have at least one similar article in our historical dataset. Although it is possible that trend of the current article behave completely

[1] N represents total number of people who viewed the article, and $B(n)+U(n) = N$

[2] correlate.google.com

different from the historical ones, there is still a large number of articles that imitate the same behavior.

3) Similarity Measure: Time-series similarity metric is a very important component in our framework. Besides two methods we used to measure time-series similarity in III-B2, we compare the performance of three different similarity measures in this paper. We use a normalized version of time-series in which we normalize the time-series by the maximum value that is been recorded up to the current time. We use Euclidean distance, the KSC similarity measure [15], and our proposed method which is a modified version of KSC method which we call Lagged KSC (LKSC) to compare time-series.

Normalized Time-Series

Given the current time t and a time-series $\mathbf{x}_t$, the normalized time-series is calculated by $\hat{\mathbf{x}}_t = \frac{\mathbf{x}_t}{max(\mathbf{x}_t)}$. Therefore, given two normalized time-series $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{y}}_t$, the Euclidean distance between these time-series is calculated using $Euclid(\hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t) = \sqrt{\sum_{i=1}^{t}(\hat{\mathbf{x}}_t^i - \hat{\mathbf{y}}_t^i)^2}$, where $\hat{\mathbf{x}}_t^i$ is the number of clicks at time point i.

KSC

As mentioned in [15], there are some problems in using the Euclidean distance to find the similarity between two time-series. We need a distance measure that is invariant to translation and scaling. Therefore, given two normalized time-series $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{y}}_t$, KSC defines the distance between them as:

$$d(\hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t) = argmin \frac{||\hat{\mathbf{x}}_t - \alpha\hat{\mathbf{y}}_t^{(q)}||}{||\hat{\mathbf{x}}_t||} \tag{4}$$

where $\hat{\mathbf{y}}_t^{(q)}$ is the result of shifting the $\hat{\mathbf{y}}_t$ time-series by q time units and $||.||$ is the l_2 norm. With a fixed value for q, α is considered as a convex function and we could find the optimal value of α by setting the gradient to zero:

$$\alpha = \frac{\hat{\mathbf{x}}_t^T \hat{\mathbf{x}}_t^{(q)}}{||\hat{\mathbf{x}}_t^{(q)}||^2} \tag{5}$$

Whereas given a fixed q, we can simply find the optimal value for α, there is no trivial solution to find the best value for q. Therefore, KSC shifts all the time-series to peak at a specific time unit and then use this distance measure to find the similarities between time-series. However, in our problem, we are not aware of the peak time of the current article and therefore, we consider the maximum value that is seen so far as the peak value and use this value for normalization.

Lagged KSC

KSC lacks in finding the best shift value q since we do not know the peak time of the current article. Therefore, we have to find a way to deal with finding the best value for q. As mentioned earlier, if we normalize the time-series of the current article according to the maximum value that is been recorded until now, we can use this method to shift the time-series and match peaks of the current article with its top ten time-series. In order to find the q value, we use the Cross Correlation between the normalized version of current time-series and its similar articles. The Cross Correlation method is a measure of similarity of two series as a function of lag

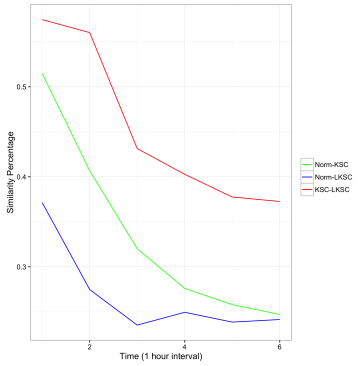

Figure 3: For each pair of similarity measures, this plot represents the percentage of article that have identical most similar article over each time interval

of one relative to the others. The Cross Correlation between two series $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{y}}_t$ is measured as follows:

$$(\hat{\mathbf{x}}_t \star \hat{\mathbf{y}}_t)[n] = \sum_{m=-\infty}^{\infty} \hat{\mathbf{x}}_t^*[m]\hat{\mathbf{y}}_t[m+n]. \tag{6}$$

where $\hat{\mathbf{x}}_t^*$ denotes the complex conjugate of $\hat{\mathbf{x}}_t$ and n is the lag. In our application, $\hat{\mathbf{x}}_t$ is the current time-series while $\hat{\mathbf{y}}_t$ represents the time-series of similar articles. We find the best lag value that could maximize the Cross Correlation between two time-series and consider it as the value for q:

$$q = \arg\max_n(|(\hat{\mathbf{x}}_t \star \hat{\mathbf{y}}_t)[n]|) \tag{7}$$

Although both the KSC and LKSC will not provide a correct measure of the distance between the two time-series (since we use a portion of time-series to compare them), it could be considered as an approximation for the distance between the two.

Since each of these measures capture a different aspects on similarity between two time-series, the result of the most similar article would also be different for these measures. To understand the amount of differences between these measures, for each pair of measures, we plot the percentage of articles that found an identical similar article. Fig 3 represents this percentage over each time interval[3]. As you can see in this figure, the similarity between the Normalized measure and LKSC is the least while KSC and LKSC has the largest similarity in the first 30 minutes. Moreover, once we go along the time, the similarity of all pairs decreases which magnifies on how different each of these measures behave when we compare enough number of data points from two time-series. On the other hand, Fig 4 compares the average distance of the queried article and its most similar article along the time. Fig 4 also plots the distance of the KSC and LKSC method when using the whole dataset to search for the most similar article. These plots are shown using KSC^*

[3]The first time interval represents the first 30 minutes, while all the other time intervals are separated with one hour. Therefore, time interval 6 represents the similarity for five hours after the publication

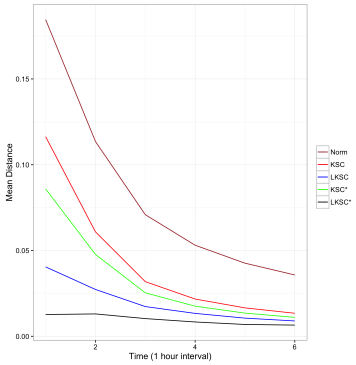

Figure 4: The average distance between the time-series of the queried article and its most similar article that is found by each measure over time

and $LKSC^*$, respectively. As you can see, the Euclidean distance gives the largest distance among other measures. As stated in [15], this confirms the inefficiency of the Euclidean distance for measuring the distance between two time-series. Moreover, our modification on the KSC method, Lagged KSC, gives a much smaller distance than KSC which shows how effective this approach is in finding the most similar article. Furthermore, LKSC provides a better distance than KSC^*. Please note that LKSC only searches among the top ten similar articles while KSC^* searches through the whole dataset to find the most similar article. Not surprisingly, $LKSC^*$ provides the least distance between the queried article and its most similar article which shows the effectiveness of this method among other measures.

To show the effectiveness of each of these measures in finding the most similar article, we plot the time-series of an article along with their most similar *SpikeM* generated time-series. Fig 5 represents the plots of time-series of an article over 24 hours. Each plot represents the time-series of the article along with the most similar *SpikeM* generated time-series to it according to a specific measure. We use the first 48 data points, representing the page views after four hours of publication of the article, to compare the two time-series. The left plots show the results using the Euclidean, KSC, and LKSC measures on the top ten similar articles, while the right plots shows the result of search the whole dataset using KSC^* and $LKSC^*$. As you can see in this figure, the KSC method completely failed to find the similar article, while the Euclidean method finds a more realistic time-series to the trend of the article. Although Euclidean method found a similar time-series, it still fails to capture the invariant that exists between the two time-series. This problem is completely resolved by LKSC and $LKSC^*$ which capture both the trend of the article and the invariant that exists in the time-series.

Fig 5 represents the result of using different distance measures over 48 data points of the queried article and

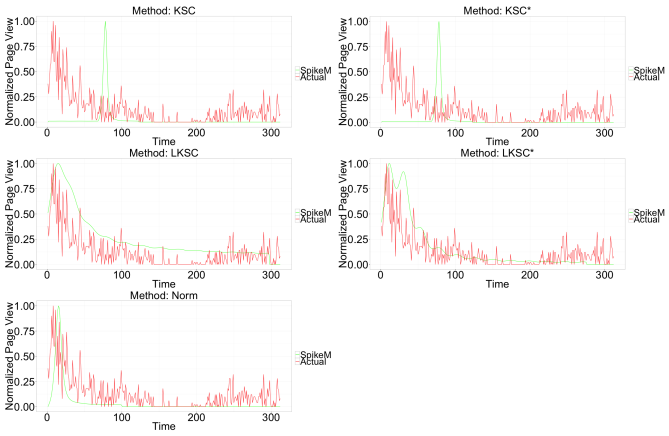

Figure 5: The result of finding the most similar article using KSC (top-left), LKSC (middle-left), Euclidean method (bottom-left), KSC^* (top-right), and $LKSC^*$ (middle-right) by accommodating the first 48 data points

the *SpikeM* generated time-series. However, as mentioned earlier, we start finding the most similar article to a queried article after 30 minutes of its publication. Fig 6 shows how the most similar article will change as we move along the time and use more data points to find it. Fig 6 shows the result of using LKSC measure to find the most similar article. As you can see, although just by using 6 data points

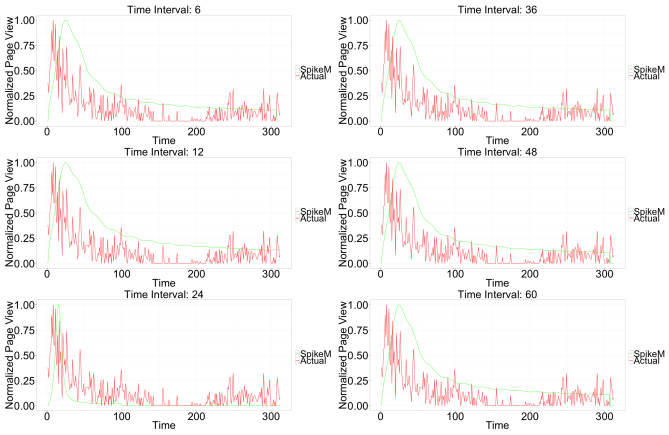

Figure 6: How most similar article will change over time, if we use LKSC measure to find it

(top-left plot) LKSC is able to find the right trend for the article, it lacks to shift the *SpikeM* generated time-series to match the data, correctly. As we move along the time, however we see that the *SpikeM* model that is similar to the time-series also changes to a narrower time-series that could better capture the trend and invariant in the time-series.

4) Classification vs. Online: To compare the performance of the *SpikeM* method with the classification approach explained in Section III-A3, we use *SpikeM* with LKSC distance to predict the peak time of each time-series and compare the peak time with the actual peak time of the

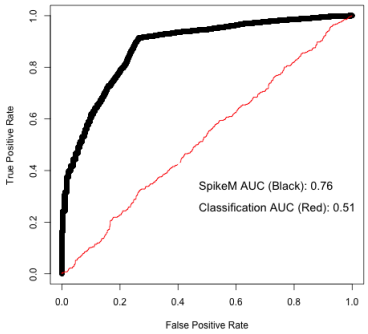

Figure 7: The comparison of the ROC curve for the classification and *SpikeM* method for the first 30 minutes.

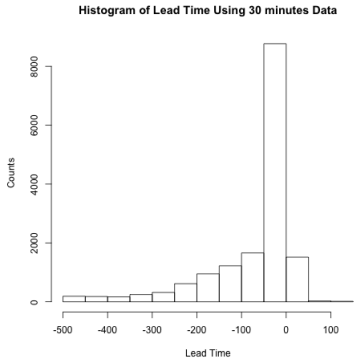

Figure 8: The lead time of online peak prediction for the first 30 minutes

time-series. For a given time interval t (like 30 minutes), let's consider the prediction as p and the actual peak time as a. We create a binary label for prediction and actual peak as follows:

$$l_p = 1 \text{ if } p <= t, \text{ otherwise } l_p = 0$$
$$l_a = 1 \text{ if } a <= t, \text{ otherwise } l_a = 0 \quad (8)$$

Using these two binary vectors for prediction and actual peak time, we are able to compare our result to the classification result. Table V represents the results for the *SpikeM* method. Online approach significantly outperformed the classification framework (Table IV) in all settings. This shows nonlinear methods such as *SpikeM* is a much better alternative in predicting shape and peak times of news articles.

Fig 7 plots the ROC curve for the two approaches and compares the AUC value for these two methods[4]. From the result, the classification method is no better than a random classifier, while *SpikeM* provides a better prediction.

Table V: Result of using *SpikeM* to predict the peak

	Precision	Recall	Fscore	AUC
First 30 Minutes	0.48	0.99	0.65	0.76
First One Hour	0.47	0.99	0.64	0.77
First Two Hours	0.65	1	0.79	0.76
First Three Hours	0.68	1	0.81	0.75
First Four Hours	0.93	1	0.96	0.75

5) Lead Time: As mentioned earlier, we use *SpikeM* not only to predict the shape of the time-series, but also to predict the peak time of the news article. Section III-B2 elaborates on how to predict the shape of the time-series using the *SpikeM* algorithm. One advantage of having a good method for shape prediction is that we can use the predicted pattern to find the peak time of the article. In this method, we consider the peak time of the most similar *SpikeM* generated time-series to a queried article, as our prediction for the peak time of the article. The *lead time* is defined as the time difference between the predicted peak time and the actual peak time. A positive lead time means that we predicted the peak time after the actual peak, while a negative lead time means that we predicted the peak before the actual peak. For all articles, we find the most similar *SpikeM* generated time-series to the pattern of the queried article according to the measures that are explained in Section III-B3.

Fig 8 represents the histogram of the lead time using this method for the first 30 minutes. The ideal shape for this histogram is to have as few articles as possible on the right side of zero point, while having as many articles as possible close to the zero point on the left side. As you can see in this figure, a large portion of news article has their lead time close to zero, which shows the effectiveness of *SpikeM* in predicting the peak time.

IV. Deployment at The Washington Post

Knowing the peak time of articles allows newsroom team to promote more effectively these articles in different ways such as putting them on home page or social media networks. Given the promising results of the online prediction approach, we are deploying the model to predict the most significant peak time of articles published by The Washington Post.

To start simple, we only use click time series of news articles published at The Washington Post, we can incorporate time-series from other sources such as social media networks in the future. Millions of users are reading articles at The Washington Post and it is practically very challenging to collect time-series for news articles in real time. We have to adopt a big data infrastructure to support the prediction. The online prediction approach needs constant click updates for all news articles, thus we adopt the Spark Streaming framework for this purpose. Figure 9 demonstrates a high level architecture of our prediction system at The Washington Post.

Click updates for all Washington Post news articles are queued in Kafka in real time, and backend Spark Stream-

[4]Due to space limitation, we avoid plotting the ROC curve for all time intervals, but the improvement for all time-intervals stays the same

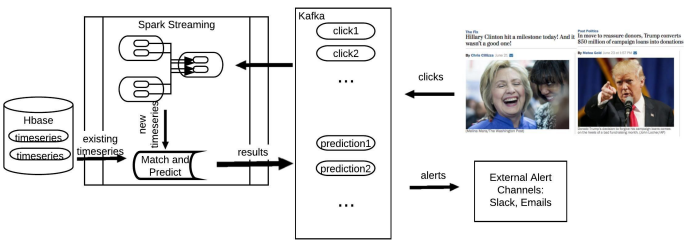

Figure 9: The architecture of predication system at The Washington Post

ing process constantly consuming these millions of click updates and transform them into time-series for these articles. *SpikeM* model matches these updated time-series with historical articles and makes prediction as necessary. The resulting predictions are then queue back to Kafka and then forward to external notification channels. Newsroom teams can then be notified about when certain articles are expect to peak.

V. Conclusion

In this paper, we introduced two problems on forecasting the pattern of click time-series and peak time of the news articles. We proposed offline and online methods to predict the shape of the click time-series before and after publication of an article. While we use a combination of clustering and classification method to provide a solution for peak time prediction problem. Through our analysis on a real news dataset collected from The Washington Post, we provided a framework that could be used to both forecast the pattern of click time-series and peak time of the news article.

References

[1] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *Proc. of KDD'03*. ACM, 2003, pp. 336–345.

[2] S. A. Myers and J. Leskovec, "The bursty dynamics of the twitter information network," in *Proc. of WWW'14*. ACM, 2014, pp. 913–924.

[3] S. Alcock and R. Nelson, "Application flow control in youtube video streams," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 24–30, 2011.

[4] A. Vázquez, J. G. Oliveira, Z. Dezsö, K.-I. Goh, I. Kondor, and A.-L. Barabási, "Modeling bursts and heavy tails in human dynamics," *Physical Review E*, vol. 73, no. 3, p. 036127, 2006.

[5] K. Lerman and R. Ghosh, "Information contagion: An empirical study of the spread of news on digg and twitter social networks." 2010.

[6] Y. Keneshloo, S. Wang, E.-H. S. Han, and N. Ramakrishnan, "Predicting the popularity of news articles," in *Proc. of SDM'16*, 2016.

[7] T. Zaman, E. B. Fox, E. T. Bradlow *et al.*, "A bayesian approach for predicting the popularity of tweets," *The Annals of Applied Statistics*, vol. 8, no. 3, pp. 1583–1611, 2014.

[8] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[9] L. Marujo, M. Bugalho, J. P. d. S. Neto, A. Gershman, and J. Carbonell, "Hourly traffic prediction of news stories," *arXiv preprint arXiv:1306.4608*, 2013.

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

[11] R. Bandari, S. Asur, and B. A. Huberman, "The pulse of news in social media: Forecasting popularity," *arXiv preprint arXiv:1202.0332*, 2012.

[12] N. Parikh and N. Sundaresan, "Scalable and near real-time burst detection from ecommerce queries," in *Proc. of KDD'08*. ACM, 2008, pp. 972–980.

[13] J. Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[14] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: model and implications," in *Proc. of KDD'12*. ACM, 2012, pp. 6–14.

[15] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proc. of WSDM'11*. ACM, 2011, pp. 177–186.

[16] X. Wang, C. Zhai, X. Hu, and R. Sproat, "Mining correlated bursty topic patterns from coordinated text streams," in *Proc. of KDD'07*. ACM, 2007, pp. 784–793.

[17] T. C. Mills, *Time series techniques for economists*. Cambridge: Cambridge University Press, 1990.

[18] S. Wang, Z. Yan, X. Hu, S. Y. Philip, Z. Li, and B. Wang, "Cpb: a classification-based approach for burst time prediction in cascades," *Knowledge and Information Systems*, pp. 1–29, 2015.

[19] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proc. SIGMOD ICMD'15*. ACM, 2015, pp. 1855–1870.

[20] S. Sundereisan, A. Bhadriraju, M. S. Khan, N. Ramakrishnan, and B. A. Prakash, "Sanstext: Classifying temporal topic dynamics of twitter cascades without tweet text," in *ASONAM'14*. IEEE, 2014, pp. 649–656.

[21] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," *Journal of the American statistical association*, vol. 83, no. 403, pp. 596–610, 1988.