Analyzing the Temporal Dynamics of the News Cycle

Jaewon Yang
Stanford University
crucis@stanford.edu

ABSTRACT

We explore the dynamics of the evolutionary patterns in the popularity of topics. To measure the popularity quantitatively, we regard the quoted phrase as a carrier of a topic, and track the frequency of its quotation over time. Our motivation starts from finding distinctive patterns that arise in the temporal behavior of the popularity. To find these patterns, we formulate a time series clustering problem based on their shape. Since our goal is to find a set of time series that have common shape, we propose a new similarity measure for shape, which is invariant from scaling and shifting, and develop a new method of clustering for our measure. Our algorithm proves to give better results to k-means both in theoretically and in experimentally. Based on these clusters, we investigate the role of big web site in the evolution of the popularity of topics. We observe that the interaction of web site and topics is very distinct over clusters, and that the overall influence of web sites show a quick decay over time.

1. INTRODUCTION

Quantitative analysis of how topics gain and lose popularity across the Web has drawn much attention from various fields such as computer science, business, and social science. Despite the intensity of attention, the questions about the temporal dynamics of the popularity of topics remained as an open question. The central reason is the difficulty in measurement of popularity. It is very challenging to observe the evolution of every topic around the web since any topic can polymorph as it propagates. Tracking this change, either manually or by a sophisticated algorithm, cannot be feasible because of huge amount of information. Recently, Leskovec et al. [3] suggested a algorithm of tracking the transition of topics that is robust and simple enough to scale up to massive data. [3] regarded quoted phrases as the unit for propagation of topics. [3] succeeded to find a group of related quoted phrase from the data during the last U.S. election. For each clustered phrases, [3] made a detailed record about which web site mentioned which phrase at what time. Using the result of [3], we perform in-depth analysis of the dynamics behind the evolution of topics.

2. FINDING DISTINCTIVE EVOLUTIONARY PATTERNS : ALGORITHM

Our first question is what are the evolutionary pattern of topics in terms of popularity. Some phrases will be quoted for a lone time while others will not. In order to measure the temporal evolution of phrases' popularity, we generated a time series for each phrases, by measuring the frequency of quotation, which we will call "volume", during a certain interval of time. Then, it is obvious that the shape of this time series represents the evolutionary pattern. If we can find a group of phrases that have similar shape in their time series, then we can consider the centroid of the cluster as a common pattern observed in some phrases. Therefore, Our question becomes how to cluster time series based on their shape.

2.1 Measure of Similarity in Shape

Measuring the similarity in shape is very difficult, especially with time series we have. They are so diverse in their absolute value, and the timing of reaching peak, as in the figure 1. We need to resolve the both issues about scale and shift. One might normalize each time series by a certain criteria and use ordinary distance measures such as euclidian distance. However, there can be numerous criteria about which value to normalize, such as sum, peak, or norm, and it is not clear which one is the best for the problem. To avoid this ambiguity, we propose a new measure of similarity which is invariant to scale and shift. Given two time series x and y, we propose $\hat{d}(x,y)$ as follows:

$$\hat{d}(x,y) = \min_{\alpha,k} \frac{||x - \alpha y_k||}{||x||}$$

where y_k is the result of shifting y by k time units. This metric finds an optimal shift and scaling coefficient for matching two time series. Computational complexity is reasonable since we can find α as a closed form when the shift k is fixed. Simple algebra shows that $\alpha = \frac{x^T y}{||y||^2}$. There is no straightforward method for searching k, but we can try a few number of shift after aligning the peak of two time series. This heuristic finds the shift that is very close to the optimal since the aligning peak time is very close to the optimal shift, given that most time series have a very sharp peak. This metric also has an interesting geometrical interpretation. If we regard time series as vectors, $\hat{d}(x,y)$ is the

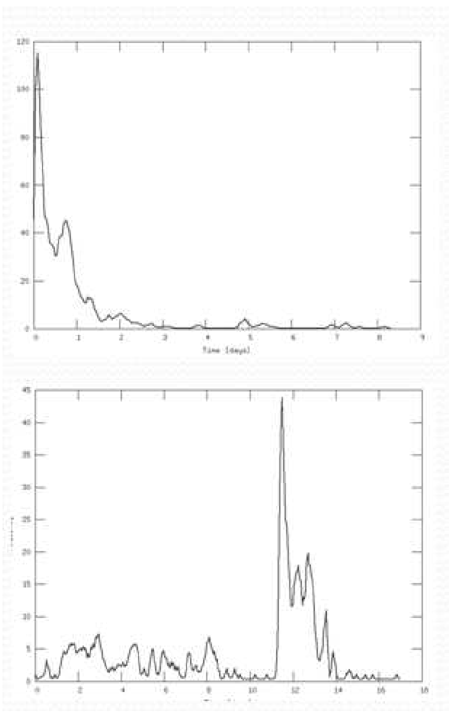

Figure 1: Two time series for phrase volume over time, from the MemeTracker web site[1]. The peak value of top and bottom series are 123 and 45 respectively.

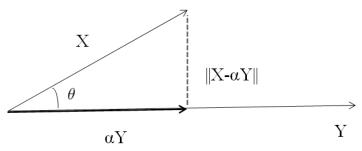

Figure 2: The geometrical interpretation of $\hat{d}(x,y)$.

distance from x to its projection on y, as in the figure 2. We can see that $\hat{d}(x,y)$ is just $\sin(\theta)$, where θ is the angle between two vectors.

2.2 K-Spectral Centroid Clustering

We introduce a novel algorithm, K-Spectral Centroid(K-SC), an iterative algorithm that is suitable for $\hat{d}(x,y)$, our new metric. Similarly to k-means, we iterative between two stages, an assignment step and a refinement step. Our goal is to find an assignment and the center of the K clusters that minimize the sum of squared distance within clusters, namely F,

$$F = \sum_k \sum_{x_i \in C_k} \hat{d}(x_i, \mu_k)^2, \tag{1}$$

where μ_k, C_k is the center and the assignment varialbe of the k-th cluster. More formally, we are given x_i's, a set of time series, and the number of cluster K, and start clustering with random guess of K clusters. In an assignment step, we assign each x_i to the closest cluster, based on $d(\cdot)$. This process is identical to the assignment step of k-means except that a different distance function.

After finding the membership for every cluster, both K-SC and k-means update the new center for each cluster. k-means updates the new center to be the average of all members of the cluster, because the mean is the minimizer of sum of squared Euclidean distance within cluster. However, we are using a different distance function, $\hat{d}(x,y)$. The new center for our algorithm, μ^*, should be the minimizer of the sum of $\hat{d}(x_i,\mu)^2$ within the cluster:

$$\mu^* = \arg\min_{\mu} \sum \hat{d}(x_i, \mu)^2. \tag{2}$$

Since an iterative algorithm like K-SC should update numerous times until convergence, it is crucial how fast the above minimization problem can be solved. The simplicity in update step of k-means made it prominent clustering algorithm for large data. Fortunately, the sum of $\hat{d}(x_i,\mu)^2$ also has unique minimizer in a closed form. It turns out that μ^* is the eigenvector of a matrix K corresponding its smallest eigenvalue, where $K = (\sum_x \frac{x_i x_i^T}{||x_i||^2} - I)^2$. Here is the derivation :

$$\begin{aligned}
\mu^* &= \arg\min_\mu \textstyle\sum_x \frac{||\alpha x_i - \mu||^2}{||\mu||^2} \\
&= \arg\min_\mu \frac{1}{||\mu||^2} \textstyle\sum_x ||\frac{x_i^T \mu}{||x_i||^2} x_i - \mu||^2 \\
&= \arg\min_\mu \frac{1}{||\mu||^2} \textstyle\sum_x ||\frac{x_i x_i^T \mu}{||x_i||^2} - \mu||^2 \\
&= \arg\min_\mu \frac{1}{||\mu||^2} \textstyle\sum_x ||(\frac{x_i x_i^T}{||x_i||^2} - I)\mu||^2 \\
&= \arg\min_\mu \frac{1}{||\mu||^2} \textstyle\sum_x \mu^T (\frac{x_i x_i^T}{||x_i||^2} - I)^2 \mu \\
&= \arg\min_\mu \frac{1}{||\mu||^2} \mu^T \textstyle\sum_x (\frac{x_i x_i^T}{||x_i||^2} - I)^2 \mu
\end{aligned}$$

$$\mu^* = \arg\min_\mu \frac{\mu^T K \mu}{||\mu||^2}, \quad K = (\textstyle\sum_x \frac{x_i x_i^T}{||x_i||^2} - I)^2$$

Instead of finding a mean of members, we find a smallest eigenvector related to the sum of spectral expansions of members, namely "Spectral Centroid".

2.3 Speed-up by Incremental Approach

Although the update step of K-SC has solution in a closed form. The complexity of finding an eigenvector follows $O(d^3)$, where d is the dimensionality of data. Since we deal with Time-series that span for a long period of time, $O(d^3)$ is not trivial cost. Similar to k-means, in addition, the performance and speed of K-SC is very sensitive to initialization, especially in high-dimensional case[4]. We address these two problem by using the approach of Ik-means. Using Haar wavelet, we can represent time series with different resolution. Ik-means starts k-means from coarser resolution, after k-means converges for that level, it uses the result as a initial point for the next level. In low dimensional space, k-means is both fast and robust. In high resolution, Ik-means converges faster than normal k-means because they start from more plausible point than random initialization. Following this idea, we can reduce the running time of K-SC algorithm to comparable level to that of k-means, as will be discussed in the next section.

3. EXPERIMENTAL RESULT OF K-SC CLUSTERING

Among total 71,000 phrases in the data set, we choose 1000 phrases with biggest total volume. Then, we exclude noisy quotes such as "I love you" or "I don't know" by picking the

Table 1: Statistics of the results of clustering

Method	F	$\sum \hat{d}(\mu_i, \mu_j)^2$
KM with NO	52.1	0.3
KM with Peak	51.1	3.0
K-SC	**41.2**	**3.1**

Table 2: The running time of the methods

Method	Running time
KM	11m 54s
K-SC	18m 01s
Incremental KM	07m 50s
Incremental K-SC	10m 02s

phrases that keep high volume anytime.[1] After eliminating spurious phrases, we perform K-SC clustering for 6 clusters. As a baseline, we compare k-means clustering without any normalization, and with normalizing peak volume to be 100. Figure3 shows the result of three way of clustering. While the centers of the clusters from k-means with no normalization look very close each other, the centers from other two cases are quite distinct. Three of the 6 clusters stays in high volume for a longer time than 1 day. Other 3 clusters with short duration also have different shape in their rising and decay. In order to measure the performance of K-SC in quantitative manner, we measured F in the equation 1. To measure the distinctiveness among the centers of clusters, we compute the sum of squared distance between the centers of clusters, $\sum \hat{d}(\mu_i, \mu_j)^2$; its big value means that it is hard to match clusters with scaling and shifting. In the table 1, K-SC algorithm shows the best similarity inside clusters, and the biggest dissimilarity between the centers of clusters. We tried sum normalization and standardization as well but peak normalization turns out the best normalization for our data. This results show that K-SC algorithm successfully find a distinct clusters that share a common shape inside them, even from a data that are very diverse in scale and shift.

3.1 Speed-up by Incremental Method

We show that K-SC maintain competitiveness in its speed in table 2. Although there exists a considerable gap between a simple k-means and simple K-SC, we can reduce this gap significantly by using incremental approach. While the one-shot K-SC consumes about 60% more time than simple k-means, incremental K-SC takes only 25% more time than incremental K-means, and it is faster than simple k-means. This results suggest that K-SC can be used in large data set and would show reasonable speed.

3.2 Additional Issues about Clustering

Length of Sequence: In the original data set, most of phrases continued over 6 months. Considering that most of them have sharp peak, the quotes made a long time before or after the peak is likely to be a noise, a random quote. For better precision of calculation, the dimensionality reduction is crucial as well. Also, we are interested in the dynamics in fine scale, which happens hourly during the peak of certain phrase. We conclude that we have to truncate a time-series around the peak. In order to find where to truncate, we measured the time when the volume at that time is over a certain fraction of the peak volume for the first time,("appearing time") and the time when the volume stays over the fraction of the peak volume for the last time("quit time"). When we measure the median of the duration between two times for the fraction of 20%, the value was around 100. The plot for various value of fraction is shown in the figure 6, where the median is the second line from the top.

[1]we measured the ratio between volume around the peak and volume far outside the peak.

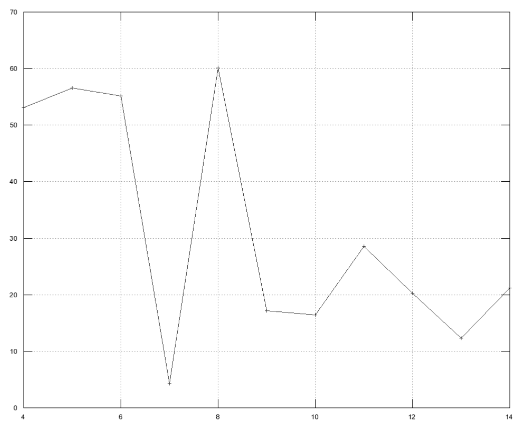

Figure 4: The value of Hartigan's index over the number of clusters

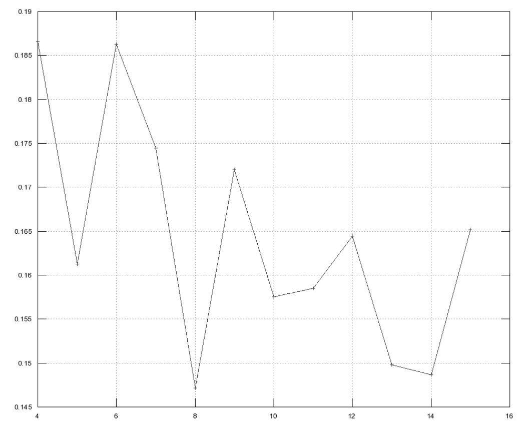

Figure 5: The value of Average Silhouette over the number of clusters

Number of Clusters: Choosing the number of clusters for k-means is still an open question. Since the objective function of k-means and K-SC, the sum of squared distance within cluster, is non-decreasing in the number of clusters, comparing the objective function does not make sense. Thus, we employed two major measures, Hartigan's Index and Average Silhouette, to measure the quality of clustering over different choice of the number of clusters. Both of them are independent of the number of clusters, and the higher value means the good quality. Figures 4 and 5 suggest that 6 clusters give relatively good value of index compared to other numbers.

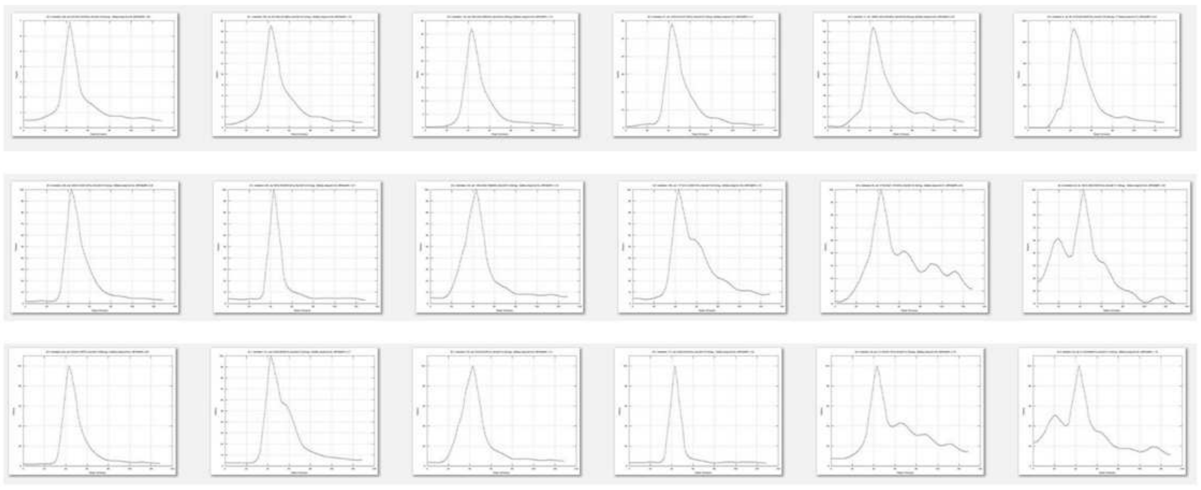

Figure 3: The first row: k-means with no normalization The second row: k-means with peak normalization The third row: K-SC clustering

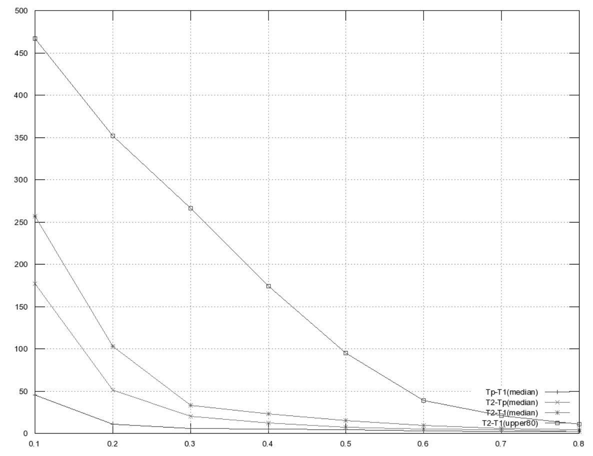

Figure 6: The time gap between "appearing time" and "quit time" versus a range of fraction.

4. FINDING THE INFLUENCE OF NEWS WEB SITE

Having found the distinctive clusters from our data set, we turn to the next question: what are the factor of these evolutionary patterns? and how we can estimate it? Among many candidates for factors, we focus on the influence of notable web sites. This hypothesis is based on our intuition that, a quotation from a massive news agency would cause more people to mention the phrase. The data about which web site mentioned which phrases can easily be found from our data set. We choose the 100 web sites based on their total quotes on the 1000 phrases we choose earlier. In order to estimate their influence, we make a hypothesis that the volume of the phrase at the next time is the sum of influences that big web sites that mentioned it during recent a few hours. Mathematically, it is expressed as the following linear equation:

$$V(t) = \sum_{k=1}^{k=T} M_i(t-k) I_i(k) + c$$

where $V(t)$ is a volume at time t, $M_i(t-k)$ is the indicator function of the i-th website mentioning the phrase, $I_i(k)$ is the influence of i-th website after k time unit, and c is a constant bias term. From a data set, we can find $V(t)$ and M_i, and our goal is to estimate I_i and c. This problem is an estimation problem which can be converted to a convex optimization problem by using a convex loss function, l

$$\begin{array}{ll} \text{minimize} & \sum_t l(V(t) - (\sum_{k=1}^{k=T} M_i(t-k) I_i(k) + c)) \\ \text{subject to} & \dots \end{array}$$

If we choose l as L2-norm, it becomes a least square problem. Instead, we can use a Huber norm[2], which acts like L1-norm for large values, and L2-norm for small values, for being robust to outliers. In addition, we could add the constraints about I_i's in ... part("subject to"), such as that I_i should not be negative, or that $I_i k$ should be smooth enough. Another flexibility comes from the choice of T, the total time duration of the influence.

4.1 Impact for Clusters

The distinct clusters that we find in the previous sections would open better opportunity to see how big web sites interact with different type of phrases. For each clusters, I plotted the average timing of quotation of top 30 websites. The further bottom means the more quotation the website made, and the further left means that the average timing is earlier. Figures 7 and 8 shows that the timing and the name of websites are considerably different. Observing this difference, we decide to set up a separate model for each clusters. Finding the reason for such difference would be an interesting open question.

5. EXPERIMENTAL RESULT

We are to estimate the influence of web sites from our data set. Since we have a number of choices, we have to select

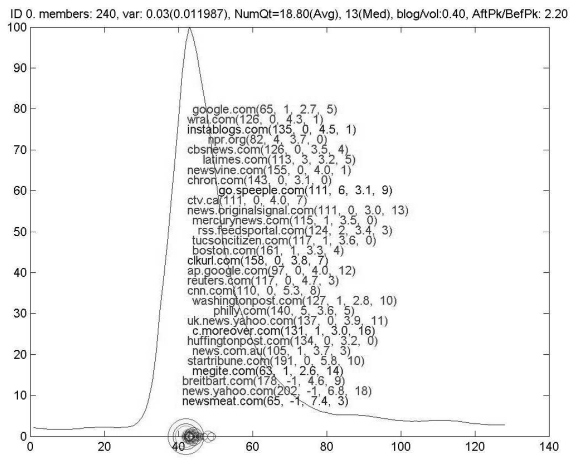

Figure 7: The timing and size of big websites for the first cluster

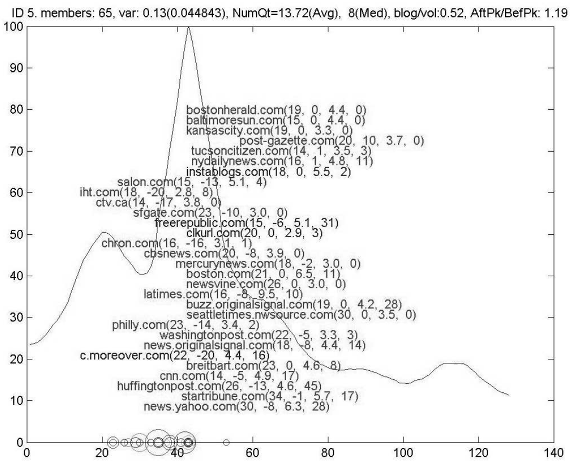

Figure 8: The timing and size of big websites for the last cluster

Table 3: The best model for each clusters

Cluster	T	Obj	NonNeg	$\|\|error\|\|/\|\|V\|\|$
0	8	H	Y	0.63
1	16	H	Y	0.68
2	16	H	Y	0.61
3	8	H	Y	0.92
4	16	H	Y	0.63
5	8	H	Y	0.81

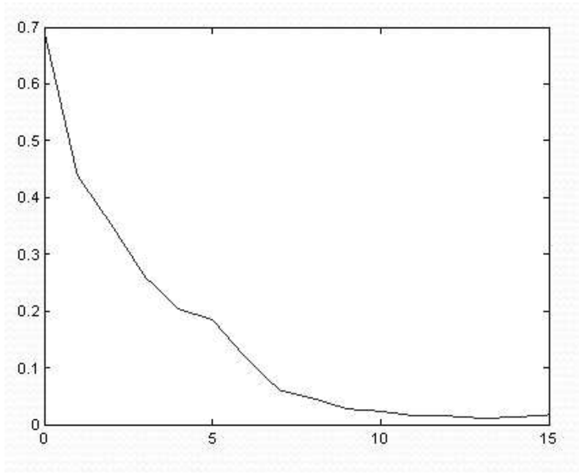

Figure 9: The average influence of 100 big web sites over time

a model before estimation. Different models have different objective functions and various degree of flexibility. For a fair comparison, we perform a 5-fold cross-validation test, where we estimate a model from 80% of data, and validate its error from the rest 20%, and repeat it 5 times so that every data is used for validation once. The model with the smallest validation error is chosen. Table 3 shows the best model chosen and their cross validation error, expressed in the fraction of L2-norm. In every cluster, using Huber norm as a loss function gives better accuracy than using L2-norm, and applying nonnegativity constraint($I_i \geq 0$) help better estimation, which is consistent on our intuition that a mention from big web sites would not cause a negative effect on the popularity of the phrase. We averaged all $I_i(k)$ we estimated over time index k. Figure 9 shows that the average influence shows a quick decay in general.

6. FUTURE WORK

We observe that K-SC algorithm can find a distinctive shape regardless of scaling and shifting issues. We can apply K-SC algorithm for other cases where we are interested in clustering by the best matching score between data, like what we had in "matching in shape".

For estimation of influence, much work remains to be done. First of all, the validation error and reconstruction error of our model is too high. Part of the reason comes from the noisy nature of the original data. However, we could find better solution by incorporating other factors. Recently, we observed that Auto Regressive and Moving Average(ARMA) model can perform comparably well to our model. We can combine ARMA model with our variable to one unified model which is called "ARMAX" model. I would evaluate the performance of ARMAX model and see what ARMAX model would estimate as the influence of web sites. Also, using some semantic information, such as "phrases about politics", could be an interesting possibility to explore.

7. REFERENCES

[1] Memetracker website. `http://memetracker.org`.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, New York, NY, USA, 2009. ACM.

[4] V. M. K. E. Lin, J. and D. Gunopulos. Iterative incremental clustering of time series. In *Proceedings of the IX Conference on Extending Database Technology*, 2004.