A Latent Source Model for Online Time Series Classfication

Zhang Zhang
zsquared@math.umd.edu
University of Maryland, College Park

Advisor
Prof. Aravind Srinivasan
srin@cs.umd.edu
Department of Computer Science
University of Maryland, College Park

December 20, 2013

Abstract

We study a binary classification problem with infinite time series having more than two labels ("event" and "nonevent" or "trending" and "non-trending"). We want to predict the label of the time series given some training data set. Intuitively, the longer we wait, the longer the time series we can observe so that the prediction is more accurate. However, in many applications, such as predicting which topic will go popular in a social network or revealing an imminent market crash, making a prediction as early as possible is highly valuable.Motivated by these applications, we look into a latent source model which is a nonparametric model to predict the binary status of a time series. Our main assumption is that these time series only have a few ways to reach the binary status such as Twitter topic going trending online. The latent source model naturally leads to a weighted majority voting as the classification rule without knowing the latent source structure. In the project: 1. We will investigate the theoretical performance guarantees of the latent source model; 2. We will implement the model by python; 3. We will investigate the strategy to estimate the values of different parameters; 4. We will test our implementation and use the model to predict which news topics on Twitter will go viral to become trends and analyze the results.

1 Introduction

1.1 Motivation

Detection, classification, and prediction of events in the streams of information are important and interesting problems in science and engineering. From detecting a "fail" in the operation system, to predicting a a huge stock index spike, to revealing whether topics in a social network will go popular, extracting useful information from time the time dependent data is fundamental for making decisions.

Different from the 20th century, in the big data era, we have the access to large amount of data related to every human endeavor. Therefore, we are eager to find good ways to analyze the data and make decisions given the data. In social network, there are massive streams of user generated date which will reveal interesting facts, such as blogs and tweets, as well as data from portable electronic devices. These data provides us an amazing opportunity to learn the dynamics of human behavior in the communication online. "How do people make decisions? Who are they influenced by? How do ideas and behaviors spread and evolve?" These are questions that have been impossible to study empirically at scale until recent times.

Big data presents both opportunities and challenges. Big data can reveal the hidden underlying structure in a process of interest. On the other hand, given current computing and storage technology, making computations over so much data at scale is challenge. Fortunately, advances in distributed computing have made it easier than ever to exploit the structure in large amounts of data to do inference at scale. In this project, we do use Mapreduce method to preprocess the big data generated by Twitter. It is about 500G per day.

All of these examples we mentioned above share some common features. There exist some underlying process with specific properties or feature generating these time series such as stock index time series. Looking into time series, we may infer many information, such as detecting anomalous events, predicting the trends of the time series at some future point.

This is difficult to do in general. Many real-world processes defy description by simple models. Here is an paragraph I found in Dr. Nikolov's paper

> *Eugene Wigner's article 'The Unreasonable Effectiveness of Mathematics in the Natural Sciences' examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1700 pages. Perhaps when it comes to natural language processing and related fields, we are doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.*

In this project, we study the problem of prediction in a complex system using large amounts of data. Specifically, we focus on applying a latent source model to do a binary classification of time series given the big enough training data set (sufficient historical examples). We apply this to the problem of trending topic detection on Twitter and show that we can reliably detect trending topics before they are detected as such by Twitter. At the same time, we aim to introduce a more general setting for doing inference in time series based on a large amount of historical data.

2 Classification Method

2.1 Motivation

Suppose we have a space of objects Ω, a set of class labels Z, and a probability distribution μ on $\Omega \times Z$. For simplicity, we take the classes to be + and −. Based on objects X and labels Y drawn from μ, we would like to learn a classification function that each object to its correct class label. This is the standard supervised learning problem. The movivation to choose a "'good"'model is well described by Dr.Nikolov:

> *Typically, when one wants to learn a model of how objects map to labels one selects some model space and chooses the best model from that model space, preferring a model that fits the data but is not too complex.*

In this project, we are applying a setting for model specification and selection in supervised learning based on *a latent source model*. In this setting, the model is specified by a small collection of unknown *latent sources*. We assume that the data were generated by the latent sources.

Since we are entirely unaware of the structure of the latent sources, we assume that there are a relatively small number of distinct latent source objects in each class that account for all observed objects in that class. We denote them $t_1, \ldots, t_n$ for + and $q_1, \ldots, q_m$ for −. Each observation labeled + is assumed to be a noisy version of one of the latent sources $t_1, \ldots, t_n$. Similarly, each observation labeled − is assumed to be a noisy version of one of the latent sources $q_1, \ldots, q_m$.

2.2 Weighted Majority Voting

Weighted Majority Voting model is an important model in machine learning and we apply this model on time series in this project. Suppose we have a time series s: $\mathbb{T} \to \mathbb{R}$ that we want to classify as having either label +1 or -1. Here, each time series is represented as a function mapping $\mathbb{T}$ to $\mathbb{R}$, where we index time using $\mathbb{T}$ for convenience. We denote the sets of all time series with labels +1 and -1 as $\mathcal{R}_+$ and $\mathcal{R}_-$.

In this project, each $r \in \mathcal{R}_+$ gives a weighted vote $e^{-\gamma d^{(T)}(r,s)}$ on whether time series s has label +1, where $d^{(T)}(r, s)$ is some measure of similarity between the two time series r and s. $d^{(T)}(r, s)$ is chosen as the Euclidean distance between s and r in this project. (Similarity measure: the larger the more similar the signals; whereas for a distance: the smaller the more similar). (T) denotes the first T time steps of s, and the constant $\gamma > 0$ is a scaling parameter that determines the influence of each r. Similarly, each negatively-labeled example in $\mathcal{R}_-$ also casts a weighted vote on whether time series s has label -1. Essentially, each example time series in the training data says "'The time series looks like me"' with certain confidence.

In this project, we use squared Euclidean distance between time series as the similarity measure: $d^{(T)}(r,s) = \sum_{t=1}^{t}(r(t) - s(t))^2 = \|r - s\|_T^2$. Notice that this similarity measure only uses first T time steps of example time series r. Since time series in our training data are known, we would not restrict only the first T time steps but instead use the following similarity measure:

$$d^{(T)}(r,s) = \min_{\Delta \in \{-\Delta_{max},...,0,...,\Delta_{max}\}} \sum_{t=1}^{T}(r(t+\Delta) - s(t))^2 = \min_{\Delta \in \{-\Delta_{max},...,0,...,\Delta_{max}\}} \|r * \Delta - s\|_T^2 \quad (1)$$

where we minimize over integer time shifts and the maximum time shift $\Delta_{max} \geq 0$.

Finally, we sum up all of the weighted +1 votes and then all of the weighted -1 votes. The label with the majority of overall weighted votes is declared as the label for s:

$$\hat{L}^{(T)}(s,\gamma) = \begin{cases} + & \text{if} \quad \sum_{r \in \mathcal{R}_+} e^{-\gamma d^{(T)}(r,s)} \geq \sum_{r \in \mathcal{R}_-} e^{-\gamma d^{(T)}(r,s)} \\ - & \text{otherwise} \end{cases} \quad (2)$$

Using a larger time window size T corresponds to waiting longer before we make a prediction. We need to trade off how long we wait and how accurate we want our prediction.

2.3 A Latent Source Model

We assume there are m unknown latent sources(time series) that generate observed time series. We denote the set of all such latent sources by $\mathcal{V}$ and each latent source in $\mathcal{V}$ has a true label +1 or -1, which corresponding trending topic or non trending topic. Let $\mathcal{V}_+ \in \mathcal{V}$ be the set of latent sources with label +1, and $\mathcal{V}_-$ be the set of those with label -1. Let $m_+ \triangleq |\mathcal{V}_+|$ and $m_- \triangleq |\mathcal{V}_-| = m - m_+$. The observed time series are generated from latent sources as follows:
1. Sample a latent source V from $\mathcal{V}$ uniformly at random. Let $L \in \{+1, -1\}$ be the label of V.
2. Sample integer time shift Δ uniformly from $\{0, 1, ..., \Delta_{max}\}$.
3. Output time series S to be the latent source V advanced by Δ time steps, followed by adding noise signal E, i.e., $S(t) = V(t + \Delta) + E(t)$ for $t \geq 1$.

Importantly, we do not know the latent sources, nor do we know the noise distribution. We assume that we have access to training data. We make a further assumption that the training data were sampled out of the latent source model and that we have n different training time series. Denote $n_+ \triangleq |\mathcal{R}_+|, n_- \triangleq |\mathcal{R}_-|, \mathcal{R} \triangleq \mathcal{R}_+ \cup \mathcal{R}_-$, and $\mathcal{D} \triangleq \{-\Delta_{max}, ..., 0, ..., \Delta_{max}\}$. Then, we could generate the weighted majority voting rule:

$$\Lambda^{(T)}(s;\gamma) \triangleq \frac{\sum_{r_+ \in \mathcal{R}_+} exp(-\gamma(\min_{\Delta_+ \in \mathcal{D}} \|r_+ * \Delta_+ - s\|_T^2))}{\sum_{r_- \in \mathcal{R}_-} exp(-\gamma(\min_{\Delta_- \in \mathcal{D}} \|r_- * \Delta_- - s\|_T^2))} \quad (3)$$

By applying ROC curve, we may call for trading off true and false positive rates. One way to do so is to generalize the decision rule to declare the label to be +1 if $\Lambda^{(T)}(s,\gamma) \geq \theta$ and then sweep

over different values of parameter $\theta > 0$. The resulting decision rule, which we refer to as generalized weighted majority voting is thus:

$$\hat{L}_{\theta}^{(T)}(s,\gamma) = \begin{cases} +1 & \text{if} \quad \Lambda^{(T)}(s,\gamma) \geq \theta, \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

where setting $\theta = 1$ recovers the usual weighted majority voting.

2.4 Theoretical Guarantee of Misclassification

Given the most recent proof from [1], we are able to investigate the theoretical guarantee of the latent model. We denote S as the time series we want to classify. For each time series, there exist a latent source V, shift $\Delta' \in \mathcal{D}$, and noise signal E' such that

$$S = V * \Delta' + E' \tag{5}$$

With a training set of size $n \geq mlogm$, for each latent source $V \in \mathcal{V}$, there exist at least one signal in $\mathcal{R}$ that is generated from V.

Then, we note that R is generated from V as

$$R = V * \Delta'' + E'' \tag{6}$$

where $\Delta = \Delta' - \Delta'' \in D$ and $E = E' - E'' * \Delta$.

Now, we want to bound the probability of error of classifier. The probability of error or misclassification using the first T samples of S is given by

$$\begin{aligned} &\mathbb{P}(\text{missclassification of S using its first T samples}) \\ &= \mathbb{P}(\hat{L}_{\theta} = -1 | L = +1)\mathbb{P}(L = +1) + \mathbb{P}(\hat{L}_{\theta} = +1 | L = -1)\mathbb{P}(L = -1) \end{aligned} \tag{7}$$

where we know that $\mathbb{P}(L = +1) = m_+/m$. Then, we will focus on bounding $\mathbb{P}(\hat{L}_{\theta} = -1 | L = +1)$. By the Markov's inequality,

$$\mathbb{P}(\hat{L}_{\theta} = -1 | L = +1) = \mathbb{P}(\frac{1}{\Lambda^{(T)}} \geq \frac{1}{\theta} | L = +1) \leqslant \theta \mathbb{E}[\frac{1}{\Lambda^{(T)}} | L = +1] \tag{8}$$

Now,

$$\mathbb{E}[\frac{1}{\Lambda^{(T)}} | L = +1] \leqslant \max_{r_+ \in \mathcal{R}_+, \Delta_+ \in \mathcal{D}} \mathbb{E}[\frac{1}{\Lambda^{(T)}(r_+ * \Delta_+ + E; \gamma)}] \tag{9}$$

Let $r_+ \in \mathcal{R}$ and $\Delta_+ \in \mathcal{D}$. Then for any time series s,

$$\Lambda^{(T)}(s;\gamma) \geq \frac{exp(-\gamma ||r_+ * \Delta_+ - s||_T^2)}{\max_{r_- \in \mathcal{R}_-, \Delta_- \in \mathcal{D}} exp(-\gamma ||r_- * \Delta_- - s||_T^2} \tag{10}$$

After evaluating the above for $s = r_+ * \Delta_+ + E$, we could see that

$$\begin{aligned}&\frac{1}{\Lambda^{(T)}(r_+ * \Delta_+ + E; \gamma)} \\ &\leq \max_{r_- \in \mathcal{R}, \Delta_- \in \mathcal{D}} \{exp(-\gamma ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) exp(-2\gamma \langle r_+ * \Delta_+ - r_- * \Delta_-, E\rangle_T)\}\end{aligned} \tag{11}$$

where we denote $\langle q, q' \rangle_T \triangleq \sum_{t=1}^{T} q(t)q'(t)$, and $||q||_T^2 \triangleq \langle q, q \rangle_T$. Then, we could get the following bound:

$$\begin{aligned}&\mathbb{E}\left[\frac{1}{\Lambda^{(T)}(r_+ * \Delta_+ + E; \gamma)}\right] \\ &\leq \mathbb{E}\left[\max_{r_- \in \mathcal{R}, \Delta_- \in \mathcal{D}} \{exp(-\gamma ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) exp(-2\gamma \langle r_+ * \Delta_+ - r_- * \Delta_-, E\rangle_T)\}\right] \\ &\leq \mathbb{E}\left[\sum_{r_- \in \mathcal{R}_-, \Delta_- \in \mathcal{D}} \{exp(-\gamma ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) exp(-2\gamma \langle r_+ * \Delta_+ - r_- * \Delta_-, E\rangle_T)\}\right] \\ &= \sum_{r_- \in \mathcal{R}_-, \Delta_- \in \mathcal{D}} exp(-\gamma ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) \prod_{t=1}^{T} \mathbb{E}\left[exp(-2\gamma(r_+(t + \Delta_+) - r_-(t + \Delta_-))E(t))\right] \\ &\leq \sum_{r_- \in \mathcal{R}_-, \Delta_- \in \mathcal{D}} exp(-\gamma ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) \prod_{t=1}^{T} exp(4\sigma^2\gamma^2(r_+(t + \Delta_+) - r_-(t + \Delta_-))^2) \\ &= \sum_{r_- \in \mathcal{R}_-, \Delta_- \in \mathcal{D}} exp(-(\gamma - 4\sigma^2\gamma^2)||r_+ * \Delta_+ - r_- * \Delta_-||_T^2) \\ &\leq \Delta_{max} n_- exp(-(\gamma - 4\sigma^2\gamma^2)G^{(T)})\end{aligned} \tag{12}$$

where we use the independence of entries of E and we assume that E is zero mean Gaussian distribution. We also denote $G^{(T)}$ as the gap between two training data set as below

$$G^{(T)}(\mathcal{R}_+, \mathcal{R}_-, \Delta_{max}) \triangleq \min_{r_+ \in \mathcal{R}_+, r_- \in \mathcal{R}_-, \Delta \in \mathcal{D}} ||r_+ * \Delta_+ - r_- * \Delta_-||_T^2 \tag{13}$$

This quantity measures how far apart the two different classes are if we only look at length T chunks of each time series and allow all shifts of at most Δ_{max} time steps in either direction.

Now, we obtain that

$$\mathbb{P}(\hat{L}_\theta(s; \gamma) = -1 | L = +1) \leq \theta \Delta_{max} n_- exp(-(\gamma - 4\sigma^2\gamma^2)G^{(T)}) \tag{14}$$

Repeating a similar argument yields

$$\mathbb{P}(\hat{L}_\theta(s; \gamma) = +1 | L = -1) \leq \frac{1}{\theta} \Delta_{max} n_+ exp(-(\gamma - 4\sigma^2\gamma^2)G^{(T)}) \tag{15}$$

Finally, we obtain the bound as

$$\begin{aligned}&\mathbb{P}(\text{misclassification of S using its first T samples}) \\ &\leq \left(\theta + \frac{1}{\theta}\right) \Delta_{max} n exp(-(\gamma - 4\sigma^2\gamma^2)G^{(T)})\end{aligned} \tag{16}$$

2.5 Detection

2.5.1 Class Estimator

Given the latent source model, we assume that reference signals have length $N_{ref} \geq N_{obs}$. We will first consider the case $N_{ref} = N_{obs}$ and then generalize in the following section.

Our class estimation rule is simple: assign to the observation the class with the highest probability:

$$R(\mathbf{s}) = \frac{\mathbb{P}(+|\mathbf{s})}{\mathbb{P}(-|\mathbf{s})} = \frac{\sum_{\mathbf{r}\in\mathcal{R}_+} \exp(-\gamma d(\mathbf{s},\mathbf{r}))}{\sum_{\mathbf{r}\in\mathcal{R}_-} \exp(-\gamma d(\mathbf{s},\mathbf{r}))} \tag{17}$$

and check if it exceeds a threshold of $\theta = 1$. The choice $\theta = 1$ corresponds to the Weighted Majority Rule in Section 2.2. For a quadratic distance function, this becomes

$$R(\mathbf{s}) = \frac{\sum_{\mathbf{r}\in\mathcal{R}_+} \exp(-\gamma \sum_{i=1}^{N_{obs}} (s_i - r_i)^2)}{\sum_{\mathbf{r}\in\mathcal{R}_-} \exp(-\gamma \sum_{i=1}^{N_{obs}} (s_i - r_i)^2)} \tag{18}$$

The estimator for the class label L is therefore

$$\hat{L}(\mathbf{s}) = \begin{cases} + & \text{if} \quad R(\mathbf{s}) > \theta \\ - & \text{if} \quad R(\mathbf{s}) \leq \theta \end{cases} \tag{19}$$

In practice, values other than 1 may also be used for the threshold θ. We will explore it further in the parameter setting part.

2.5.2 Safty Guard

Since our signal has noises, we could use some more time steps before determining which class the observation belongs to. We require that the observation signal is judged to belong to a particular class for several consecutive time steps. We shall call this number of required time steps D_{req}. We explore the effect of D_{req} further.

2.5.3 Align Issues between Observation Signal and Reference Signal

We have assumed that the reference signals and the observations have the same length. However, it is more convenient to extend this to reference signals of arbitrary length. For reference signals and observations of the same size, we just easily use the distance function d previously described. In practice, however, there are some complications. First, observations will generally be short and reference signals will be much longer. Second, reference signals will often have unknown phase. That is, the events underlying the reference signals may have occurred at arbitrary time shifts with respect to one another(e.g. trending time set by Twitter is different). Furthermore, when comparing the observation to each reference signal, there is no temporal point of reference between the two. A

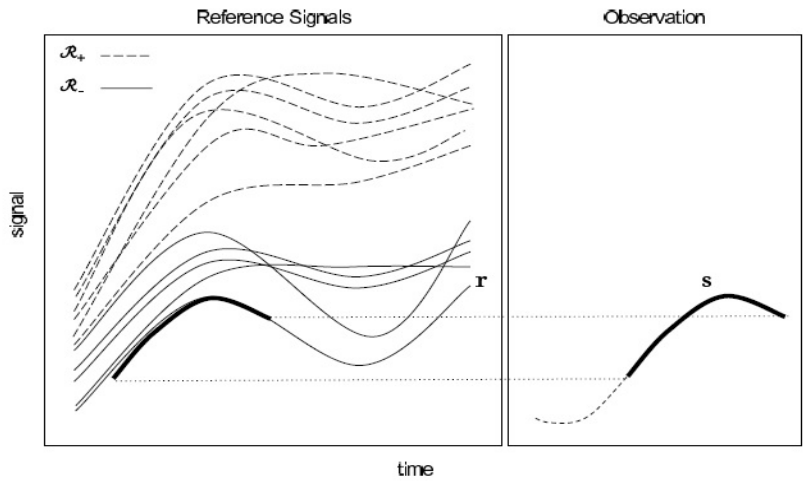

Figure 1: To compare a long reference signal to a short observation, we compute the distance between the observation and the closest *piece* of a reference signal

natural solution to both of these complications is to check whether the observation resembles any *piece* of the reference signal of the same size as the observation. Figure 1 illustrates this.

We modify the diatance function. We assume that all observations are of length N_{obs} and all reference signals are of length $N_{ref} \geq N_{obs}$. We define the distance between a reference signal $\mathbf{r}$ and an observation $\mathbf{s}$ as the minimum distance between $\mathbf{s}$ and all contiguous subsignals of $\mathbf{r}$ of length N_{obs}.

$$d(\mathbf{r}, \mathbf{s}) = \min_{k=1,\ldots,N_{ref}-Nobs+1} d(\mathbf{r}_{k:k+N_{obs}-1}, \mathbf{s}) \tag{20}$$

Conveniently, for $N_{ref} = N_{obs}$, this new distance function reduces to the distance function previously defined. Finally, using the generalized version of d, the ratio of class probabilities $R(\mathbf{s})$ from the previous section becomes

$$R(\mathbf{s}) = \frac{\sum_{\mathbf{r}\in\mathcal{R}_+} \exp(-\gamma \min_{k=1,\ldots,N_{ref}-Nobs+1}(\sum_{i=1}^{N_{obs}} (s_i - r_{i+k-1})^2)}{\sum_{\mathbf{r}\in\mathcal{R}_-} \exp(-\gamma \min_{k=1,\ldots,N_{ref}-Nobs+1}(\sum_{i=1}^{N_{obs}} (s_i - r_{i+k-1})^2)} \tag{21}$$

3 Implementation on Twitter Topics Prediction

In this section, we consider the application of the method in section 2 toward detection of trending topics on Twitter. We discuss the Twitter service, the collection and preprocessing of data, and the experimental setup for the detection task.

3.1 Overview

3.1.1 Overview of Twitter

Overview of the twitter service is from Twitter official website, papers, and wikipedia. Twitter is a real-time messaging service and information network. Users of Twitter can post short (up to 140 characters) messages called *Tweets*, which are then broadcast to the users *followers*. Users can also engage in conversation with one another. By default, Tweets are public, which means that anyone can see them and potentially join a conversation on a variety of topics being discussed. Inevitably, some topics gain relatively sudden popularity on Twitter. For example, a popular topic might reflect an external event such as a breaking news story or an internally generated inside joke or game. Twitter surfaces such topics in the service as a list of top ten trending topics.

3.1.2 Twitter-Related Definitions

Talking about Tweets, topics, trends and trending topics can be ambiguous, so here we make precise our usage of these and related terms.

Definition 3.1. *We define a* ***topic*** *to be a phrase consisting of one or more words delimited by spacing or punctuation. A word may be any sequence of characters and need not be an actual dictionary word.*

Definition 3.2. *A Tweet is* ***about*** *a topic if it contains the topic as a substring. Tweets can be about many topics.*

Example 3.1. *The following tweet by the author (handle @Zsquared) contains the string "AMSC663" Hence, it is considered to be about the topic "AMSC663".*

"AMSC663 Project drives me crazy."

Definition 3.3. *A* ***trending topic*** *is a topic that is currently on the list of trending topics on Twitter. If a topic was ever a trending topic during a period of time, we say that the topic* ***trended*** *during that time period.*

Definition 3.4. *A trending topic will also occasionally be referred to as a* ***trend*** *for short.*

Definition 3.5. *The* ***trend onset*** *is the time that a topic first trended during a period of time.*

Example 3.2. *If the topic "government shutdown" is currently in the trending topics list on Twitter, we say that "government shutdown" is trending, and that is a trend. The topic "government shutdown" has a trend onset, which is the first time it was trending in a given period of time. This could, for example, correspond to when the "shutdown" happened. After "government" is no longer trending, we say that "government shutdown" trended.*

3.1.3 Problem Statement

At any given time there are many topics being talked about on Twitter. Of these, some will trend at some point in the future and others will not. We wish to predict which topics will trend. The earlier

we can predict that a topic will trend, the better. Ideally, we would like to do this while maintaining a low rate of error.

3.1.4 Proposed Solution

Our approach to detecting trending topics is as follows. First, we gather examples of topics that trended and topics that did not trend during some period of time. Then, for each topic, we collect Tweets about that topic and generate a time series of the activity of that topic over time. We then use those time series as reference signals and apply the classification method we discussed in previous sections.

3.2 Data

3.2.1 How to Collect Data

The classification model requires a set of reference signals corresponding to topics that trended and a set of reference signals corresponding to topics that did not trend during a time window of interest. These reference signals represent historical data against which we can compare our most recent observations to do classification.

The data collection process can be summarized as follows. First, we collected 500 examples of topics that trended and 500 examples of topics that never trended during the sample window. We then sampled Tweets from the sample window and labeled each Tweet according to the topics mentioned therein. Finally, we constructed a reference signal for each topic based on the Tweet activity corresponding to that topic.

How to Choose appropriate Topics

We collected a list of all trending topics on Twitter as well as the trending times and their rank in the trending topics list on Twitter. We filtered out topics whose rank was never better than or equal to 3. In addition, we filtered out topics that did not trend for long enough as well as topics that reappear multiple times during the sample window (the time of the first appearance to the time of the last appearance is greater than 24 hours). One topic cannot be trending multiple times in one day. The former eliminates many topics only trend for a very short time. The latter eliminates topics that correspond to multiple events.

For topics that were not trending, we first sampled a list of n-grams (phrases consisting of n "words") occurring on Twitter during the sample window for n up to 5. We filtered out n-grams that contain any topic that trended during the sample window, using the original, unfiltered list of all topics that trended during the sample window. For example, if "Kobe Bryant" trended during the sample window, then "Kobe" would be filtered out of the list of topics that did not trend. We also

removed n-grams shorter than three characters, as most of these did not appear to be meaningful topics. Lastly, we sampled 500 n-grams uniformly from the filtered list of n-grams to produce the final list.

Construction of Topic Signal from Tweets

3.2.2 From Tweets to Signals

Tweet Rate

We bin the Tweets into time bins of a certain length. We use time bins of length two minutes. Let $\rho[n]$ be the number of Tweets corresponding to a topic in the n^{th} time bin. To better understand the "'rate"', we could let the cumulative volume of of Tweets up to time n be

$$v[n] = \sum_{m \leq n} \rho[m] \tag{22}$$

Baseline Normalization

Many non-trending topics have a relatively high rate and volume of Tweets, and many trending topics have a relatively low rate and volume of Tweets. One important difference is that many non-trending topics have a high baseline rate of activity while most trending topics are preceded by little. For example, a non-trending topic such as "New York" is likely to have a consistent baseline of activity because it is a common word. To emphasize the parts of the rate signal above the baseline and de-emphasize the parts below the baseline, we define a baseline b as

$$b = \sum_{n} = 1^n \rho[n] \tag{23}$$

and a baseline normalized signal ρ_b as

$$\rho_b[n] = (\frac{\rho[n]}{b})^\beta \tag{24}$$

The exponent β controls how much we reward and penalize rates above and below the baseline rate. We use $\beta = 1$ in this project.

Spike Normalization

Another difference between the rates of Tweets for trending topics and that of nontrending topics is the number and magnitude of spikes. We emphsize such spikes, while de-emphasizing smaller spikes.

$$\rho_{b,s}[n] = |\rho_b[n] - \rho_b[n-1]|^\alpha \tag{25}$$

in terms of the already baseline-normalized rate ρ_b. The parameter $\alpha > 1$ controls how much spikes are rewarded. We use $\alpha = 1.2$.

Smooth

Tweet rates, and the aforementioned transformations thereof, tend to be noisy, especially for small time bins. We convolve the signal with a smoothing window of size N_{smooth}. Applied to the spike-and-baseline-normalized signal $\rho_{b,s}$, this yields the convolved version

$$\rho_{b,s,c}[n] = \sum_{m=n-N_{smooth}+1}^{n} \rho_{b,s}[m] \tag{26}$$

Branching Processes and Logarithmic Scale

It is reasonable to think of the spread of a topic from person to person as a branching process. It is also reasonable to measure the volume of tweets at a logarithmic scale to reveal these details. Here we may have problems if the signal vector have zero entries. If the entries are zero, we will set them as zero too in this Scaling step.

$$\rho_{b,s,c,l}[n] = \log \rho_{b,s,c}[n] \tag{27}$$

Constructing a Reference Signal

The signal $\rho_{b,s,c,l}[n]$ resulting from the steps so far is as long as the entire time window from which all Tweets were sampled. Such a long signal is not particularly useful as a reference signal. If the reference signal for a topic that trended spans too long of a time window, only a small portion of it will represent activity surrounding the onset of the trend. Hence, it is necessary to select a small slice of signal from the much longer rate signal.

In the case of topics that trended, we select a slice that terminates at the first onset of trend. That way, we capture the pattern of activity 1 ending up to the trend onset, which is crucial for recognizing similar pre-onset activity in the observed signal. We do not include activity after the true onset because once the topic is listed in the trending topics list on Twitter, we expect the predominant mode of spreading to change. For topics that did not trend, we assume that the rate signal is largely stationary and select slices with random start and end times. For simplicity, all slices are a fixed size of N_{ref}.

3.3 Map-Reduce Raw Data

We use Twitter API to get the raw data from twitter. The raw tweets we obtain is from the South America written by Spanish. The data format is called JSON. Each JSON file is corresponding to a

tweet, and we have millions of tweets every day so that the data set is very big. In order to construct the time series corresponding to each tweet topic, we apply Map-Reduce framework to handle the big data set.

People use hashtag to label the topic of his tweet when they post it, and actually not all the people will use hashtag. We only focus on the tweets with hashtag and ignore tweets without hashtag. Since each JSON file include the information of the tweet such as user information, time information, tweet content, hashtag information, size, and etc. We use Map-Reduce method to count the number of tweets corresponding to one hashtag every 2 mins.

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a $Map()$ procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a $Reduce()$ procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

In this part, by using Python, we developed the $Map()$ function, $Reduce()$ function, $Postprocess()$ function, $getarray()$ function to construct the hashtag signal. They are all deliverable.

We have a big problem here. The JSON file does not include the "Trending" information,so we actually do not know which hashtag was trending before. Since we need to construct two classes of training data, "Trending" information is necessary. We now decide to record the trending topics happened in South America by hands every day.

3.4 Experiment

We measure the performance on two fronts: **error rate and relative detection time**. We divide the set of topics into a training set and a test set using a 50/50 split. For each topic in the test set, we wish to predict if the topic will trend. If the topic really did trend, we wish to detect it as early as possible relative to the true trend onset while incurring minimal error.

3.4.1 Detection Setup

In principle, to test the detection algorithm, one would step through the signal in the entire sample window for each topic in the test set and report the time of the first detection. In practice, we take a shortcut to avoid looking through the entire signal based on the following observations about the activity of topics that trended and topics that did not.

First, for topics that trended, there is little activity aside from that surrounding the true onset of the trend. In the rare event that a detection is made very far from the true onset, it is reasonable to assume that this corresponds to a completely different event involving that topic and we can safely ignore it. Thus, the only part of the signal worth looking at is the signal within some time window from the true onset of the trend.

Second, topics that did not trend exhibit relatively stationary activity. Therefore, it is reasonable to perform detection only on a piece of the signal as an approximation to the true detection performance.

Definition 3.6. *Let h_{ref} be the number of hours corresponding to N_{ref} samples. At 2 minutes per sample, h_{ref} is given by $N_{ref}/30$.*

For TEST topic signals that have trended, we do detection on the window spanning $2h_{ref}$ hours centered at the true trend onset. For topics that did not trend, we randomly choose a window of the desired size.

3.4.2 Parameter Exploration and Trials

We explore all combinations of the following ranges of parameters, excluding param- eter settings that are incompatible (e.g. $N_{obs} > N_{ref}$). For each combination, we conducted 10 random trials.

- γ: 0.1, 1, 10
- N_{obs}: 10, 80, 115, 150
- N_{smooth}: 10, 80, 115, 150
- h_{ref}: 3, 5, 7, 9
- D_{ref}: 1, 3, 5
- θ: 0.65, 1, 3

3.4.3 Validation of Implementation

The whole software is divided into four parts. First part is the MapReduce which construct the original time series corresponding to each hashtag.To validate the first part, we could use a different small sample data to generate the results and then compare the generated results with the true results. We need 100% correct rate.

For the first part we have finished, we first apply a small sample of raw data to do the mapreduce part. The MapReduce framework Hadoop first will examine if my codes could be parallized, and then

will run and output the results. Given the sample data, we could check the output data easily because it is just the count of each hashtag at time n. After validating the MapReduce codes, we would validate the getarray() function to see if it will give us the correct time series of each hashtag. Since when we write the MapReduce codes, we specify the format of the output, we are able to construct a sample data very easily and input the data into the getarray() function to check if it gives us the correct time series.

My getarray() function is wrong at the first place. Basically, the thread 0 will split each line of the input file, read line by line and send the hastag information to other threads, and other threads start to construct the signal from the time when the hashtag was first found and does not output zero if no hashtag found at a specific time. Therefore, the output is not a time series. We modified the code to ask slave thread to search hashtag at each time step from the beginning and output zero if it does not found matching hashtag at a specific time step.

The second part is to normalize the original signal. To validate this part, we just use simple array data to test the codes. Since true results is known, we can easily test the codes. 100% correct rate is required in this part.

The third part is to generate optimized parameters. This part is done offline before we can launch the test software. As we discussed, we use brute force search to figure out a good parameter combination given the cost of TPR and FPR. Then, we use this parameter set in the test software. To validate this part, we just simply use artificial array to test if the distance calculation is correct. The true results are known, so we can easily test whether the code is correct. 100% correct rate is also required.

The fourth part is the test software. It will use the same function from the third part, so we don't need to validate it. We could just test the whole software when we get this point.

3.4.4 Evaluation

To evaluate the performance of our method, we compute the false positive rate and true positive rate for each experiment, averaged over all trials. In the case of true detections, we compute the detection time relative to the true onset of the trending topic. We analyze the effect of the algorithm parameters on the tradeoff between false positive rate, true positive rate, and relative detection time. We propose parameter regimes appropriate for three situations: 1) the cost of a false positive outweighs the cost of a false negative, 2) the cost of a false negative outweighs the cost of a false positive, and 3) the costs of a false positive and a false negative are comparable.

4 Project Schedule and Milestones

Oct 1,2013 - Oct 31,2013
Learn Programming language: Python

Accomplished

Nov 1,2013 - Nov 30,2013
Write codes to classify data as different topics

Accomplished

Dec 1,2013 - Dec 30,2013
Write codes of the model

Jan 1,2014 - Jan 31,2014
Write codes to simulate ROC curves

Feb 1,2014 - Feb 28,2014
Testing Implementation

Mar 1,2014 - Mar 31,2014
Write Documentations

5 Deliverables

The deliverables for this project are the Python codes that implement the Latent Source Model to dection twitter trending topics and any code used for testing. The code will be optimized for performance and effective memory management, as well as being fully documented. Report at various stages throughout the course will detail the approach, implementation, validation, testing, and extensions of the algorithm. With this information, a researcher will be able to reproduce any results present in our reports.

References

[1] George Chen, Stanislav Nikolov, Devavrat Shah *A Latent Source Model for Online Time Series Classification*, CIPS Conference, 2013

[2] Sitaram Asur, Bernardo A. Huberman, Gabor Szabo, and Chunyan Wang *Trends in social media: Persistence and decay*, In ICWSM, 2011.

[3] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella *Emerging topic detection on twitter based on temporal and social terms evaluation*, In Proceedings of the Fifth International Conference on Weblogs and Social Media, 2011

[4] Hila Becker, Mor Naaman, and Luis Gravano *Beyond trending topics: Real- world event identification on twitter*, In ICWSM, 2011.

[5] Alon Halevy, Peter Norvig, and Fernando Pereira *The unreasonable effectiveness of data*, IEEE Intelligent Systems, 2011

[6] Yen-Hsien Lee, Chih-Ping Wei, Tsang-Hsiang Cheng, and Ching-Ting Yang *Nearest neighbor based approach to time series classification*,Decision Support Systems, 2012

[7] Juan Rodriguez and Carlos Alonso *Interval and dynamic time warping based decision trees*, In proceedings of the 2004 ACM Symposium on Applied Computing