**The Pennsylvania State University**
**The Graduate School**

# EVENT DETECTION IN TWITTER DATA: A HIDDEN MARKOV MODEL-BASED CHANGE POINT ALGORITHM

A Thesis in
Statistics
by
Ame Osotsi



Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2016

The thesis of Ame Osotsi was reviewed and approved* by the following:

Qunhua Li
Assistant Professor, Penn State Department of Statistics
Thesis Advisor

John Fricks
Associate professor, Penn State Department of Statistics

David Hunter
Professor
Department Head, Penn State Department of Statistics

*Signatures are on file in the Graduate School.

# Abstract

Twitter is a popular microblogging platform that displays real-time status updates from over 140 million users a day. The users post about anything, from daily life events to important global events. We attempt to analyze this rich source of user-generated data using hidden Markov models, which have been very successful in describing time series observations. The aim of this project is to quantify how conversation in Twitter evolve in response to two major events: an unexpected school shooting, and the Super Bowl. We use a hidden Markov model-based change point algorithm. This thesis first introduces the data and the hidden Markov models underlying the change point algorithm. We then describe the change point algorithm and related problems such as finding confidence intervals, model selection, and computing summaries. Finally, we show results on the two datasets and propose future avenues of research.

# Table of Contents

# List of Figures

# List of Tables

# List of Symbols

| | |
|---|---|
| $T$ | Number of observations. |
| $m$ | Number of hidden states. |
| $K$ | The number of unique words in the dataset. |
| $C_{1:T}$ | Latent (unobserved) Markov chain. $C_i \in \{1, 2\}$ |
| $\delta$ | Markov chain initial distribution, Section 2.2. |
| $\gamma$ | Markov chain transition probability matrix. |
| $X_{1:T}$ | The observations. |
| $X_i$ | The observation at time $i$. A sparse vector. |
| $n_i$ | Observed number of words in the tweet at time $i$. |
| $p_{c_i}$ | The vector of $K$ multinomial probabilities associated with state $c_i$. |
| $p$ | The matrix of probabilities, combining $p_{c=1}$ and $p_{c=2}$. |

# Acknowledgments

I am indebted to my parents for living, and to my teachers for living well. Thank you to Dr. Qunhua Li, Dr. David Hunter, and Dr. John Fricks. I am thankful to Dr. Aleksandra Slavkovic, Dr. Naomi Altman, Dr. Michael Coco, and Dr. Owen Byer.

Chapter 1

# Introduction

Twitter is a popular microblogging platform. Previously, creating a public blog was a complicated task, but now anyone can quickly sign up for a Twitter account and instantly publish their thoughts to the world. Every day, 140 million users post over 400 million status updates. The tweets are most often text, but they increasingly include links, videos and images. The company provides an API where people can download tweets. Due to their real time nature and easy public availability, tweets have become a focus for exploring general topics and emerging events which matter most to a broad audience. This thesis attempts to quantify how conversations in Twitter evolve in response to two major events: a school shooting and the Superbowl. The school shooting was sudden and unexpected, while the Super Bowl was anticipated months in advance.

Our approach was inspired by Chakrabarti and Punera [1]. In this paper, the authors introduced a modified Hidden Markov Model to identify and produce a summary for sub-events in a sports dataset. Summarizing sport events from tweet streams also drew the attention of Kubo et al [2]. The authors detect bursts in the Twitter stream, then identify good reporters who provide event summaries. We also found interesting the work of Sakaki et al.[3], who used a particle filter to estimate the location of earthquakes by treating the tweets as sensors.

This thesis continues previous work carried out by my supervisor Dr. Qunhua Li and her student Tong Wu. Given a time series of Twitter data, they looked at frequencies of special words to characterize the evolution of conversations about events. For example the frequency of words such as “lockdown”, “police” and

"shot" increased immediately after the shooting then steadily decreased over time. This approach gave an idea of the social impact of the shooting on students.

For the next step, we thought to use a hidden Markov model (HMM) on the data. By fitting a 2-state model, we thought to try separating tweets about the shooting from tweets not about the shooting. Perhaps a model with more states could describe the evolution of the conversations in more detail. As we will later see, the results were so noisy that we had to switch to a HMM-based change point model, which then gave reasonable results.

This thesis will be organized as follows. In Section 1 we introduce the research objectives and describe the data. In Section 2 we introduce hidden Markov models and describe the fitting process. In Section 3 we describe the specific model and change point algorithms we used on this dataset. In Section 4 we give results and discuss caveats. Section 5 gives the conclusion and future work.

## 1.1 The Data

The dataset we use was collected by Dr. Jie Shan's group at Purdue University. It consists of tweets known to come from posters near Purdue University. The first tweet is on January 2, 2014, and the last is on June 1, 2014. There are in total 71,711 tweets in the dataset. Each tweet is a maximum of 140 characters long. We also have information on the date the tweet was posted, longitude and latitude of the poster, and details of the user account including profile information and number of friends. All this information is publicly available through the website or the API. For this project we only worked with the text of the tweets and the precise time they were posted.

We focus on two events. On January 21, 2014 at approximately noon, one student walked into class and shot a fellow student then surrendered to police. On February 2, 2014 the Seahawks and the Broncos faced off in the Super Bowl. This game broke the then-record for the most discussed game on Twitter, with over 24.9 million tweets [4]. We compare the two events in terms of social impact by looking at the number of different discussions we can detect, the duration of discussions, and the kinds of topics discussed in the detected discussions.

### 1.1.1 Exploratory Data Analysis

We began by extracting 2 datasets, one for each event. For the January 21st shooting, we took tweets from January 20th to January 22nd. For the February 2nd Super Bowl, we took tweets from January 29th to February 3rd. This gave us 3878 tweets for the shooting, and 2755 tweets for the Super Bowl. Immediately we can see (and the Exploratory Data Analysis supports) that the shooting generated many more tweets despite covering a shorter time period.

Note that for the Super Bowl, we initially tried working with the Purdue tweets from February 1st to February 3rd. However, in my first attempt we found we were not getting good results with the methods described in Section 3. But when we extended the dataset a day earlier, i.e., from January 29th to February 3rd, we started getting good results. We think this is because (unlike the shooting) the beginning of the Super Bowl is not as easily detected through the tweets. People talk about the Super Bowl for days before kickoff, so we need more baseline data in our algorithms.

Our initial analysis shows that in principle events could be readily detected. Below in Figure 1.1 we plot log-interarrival times (i.e., the logarithm of the times between tweets) from the Shooting dataset against the index of the tweets. The shooting event (represented by the vertical line) is clearly seen from the time data.

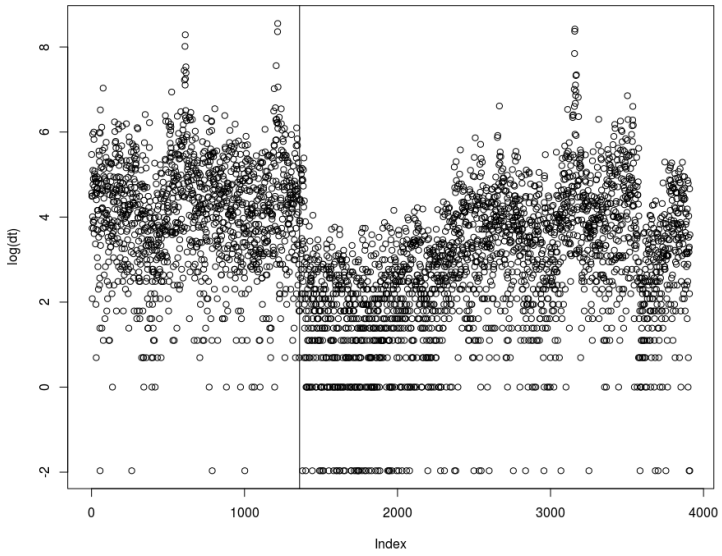


Figure 1.1: Log-interrarrival times drop significantly after the vertical line that marks the position of the first shooting tweet.

The vertical line at an x-axis value of approximately 1408 indicates the position of the first tweet mentioning the shooting. It occurs soon after 12:00 pm. The y-axis shows log-interarrival times. Right after the shooting, the tweets started coming in much faster. Then slowly the inter-arrival times started going back up to previous levels.

Similarly, as we see in Figure 1.2 below, log-interarrival times also drop during the Super Bowl. The vertical lines indicate the true start and finish times of the game. Given the clear signal in the time data, we wanted to see what patterns we could uncover in the Twitter conversations.

### 1.1.2 Data Cleaning

After picking out the two datasets, we cleaned the data. The raw data had many instances of spam, misspelled words, hyperlinks, and non-English sentences. We began by deleting all links and special (non-numeric or alphabetical) characters. We then filtered the words through the Snowball stemmer [5], a popular stemming algorithm. In information retrieval, stemming is the process of reducing words to

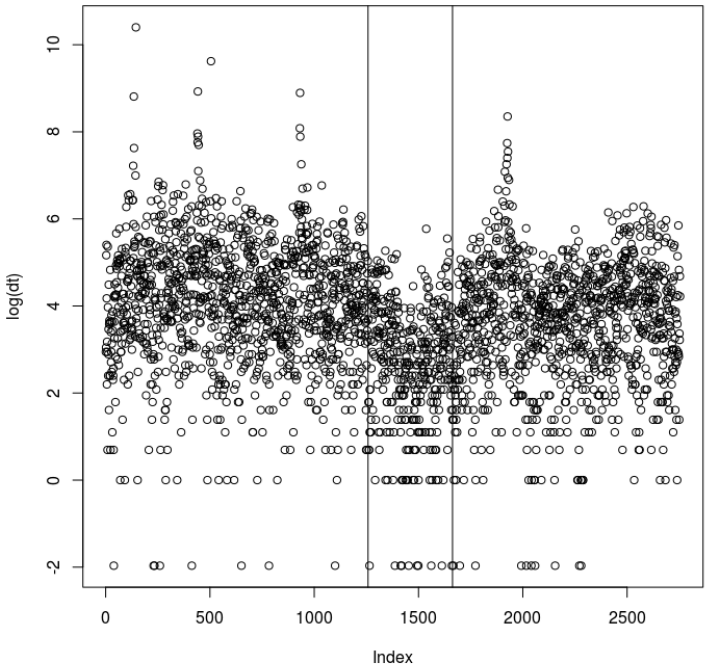


Figure 1.2: Log-interrarrival times drop during the Super Bowl. Vertical lines show true start and finish times, at dataset indices 1258 and 1663.

their word stem, base or root form. For example "doing" and "did" can both be reduced to "do". This reduces the dimensionality of the data, making it much easier for algorithms to recognize patterns. To further reduce the dimensionality, we removed all but the top 4000 most common words in the Super Bowl dataset. The shooting dataset was processed in a similar fashion - we ran it through the stemmer, then removed all but the top 5000 most common words.

In order to use the tweets in mathematical models, we transformed them into a vector of words using the bag-of-words representation. This is a very common model in natural language processing and information retrieval. In this model we only care about the frequency of occurrence of each word we ignore all punctuation and sentence structure. Each tweet is represented as a vector of length $K$, where there are $K$ unique words in the dataset. In the Super Bowl dataset $K = 4000$, and in the shooting dataset $K = 5000$. For each of the $K$ unique positions in the vector, we keep track of the number of times the word associated with that

position occurs in the tweet. As an example, Table 1.1 and 1.2 below show the first five tweets in the shooting dataset and their partially encoded form.

Table 1.1: The first 5 shooting Tweets after running through the Snowball stemmer

| |
|---|
| @saeXXXX what time is the concert |
| i liter don't think it's real life that i'm about to see @ambeXXXX and her mom... die. |
| @ploXXXX i did some search onlin about current use and all i could find was garden advic |
| what time is your perform @pataXXXX |
| "i guess you need beverag to keep their attent now, huh? or has it alway been like that...." |

Table 1.2: Partially encoded text

| |
|---|
| 0, 1, 2, 3, 4, 5 |
| 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 |
| 23, 6, 24, 25, 26, 27, 15, 28, 29, 19, 30, 6, 31, 32, 33, 34, 35 |
| 1, 2, 3, 36, 37, 38 |
| 39, 40, 41, 42, 43, 16, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55 |

The partially encoded representation in Table 1.2 means the first observation is a vector of length 5000, with 1's in the first 6 positions and 0's everywhere else. The second observation has 17 instances of 1 at the indicated positions, with 0 elsewhere. So this is usually a very sparse vector. We used a Hidden Markov Model to describe the resulting time series.

Chapter 2

# Introduction to Hidden Markov Models

Hidden Markov models are a very successful latent variable model, with applications in diverse areas such as speech recognition, handwriting recognition, and bioinformatics. They are used on time-series data, usually for inference but sometimes also prediction. Hidden Markov models assume that the observations we have are noisy versions of an underlying unobserved Markov chain. For example in speech recognition we might interpret the hidden states as words that make up a sentence, where the observations are sound frequencies with variation based on accents, gender, etc. In this application, we attempt to infer and interpret the states of the Markov chain, given some simplifying assumptions. We initially thought the states of the HMM might separate shooting vs non-shooting tweets.

## 2.1 The Hidden Markov Model

In Table 2.1 below we write out the model summary, for ease of interpretation. We denote by $X_{1:T}$ the time series of observations up to time T, and $C_{1:T}$ the latent (unobserved) Markov chain. For this thesis $C$ takes on $m$ discrete values[1], and the Markov chain transition probabilities are independent of time so we have a discrete-space 2-state Markov chain. $X_i$ are the words of the specific tweet at time

[1] In this thesis $m = 2$, with reasons given in the beginning of Chapter 3.

Table 2.1: Notation

| | | | Dimensions |
|---|---|---|---|
| $T$ | = | number of observations. | 1 |
| $m$ | = | number of hidden states. | 1 |
| $K$ | = | The number of unique words in our dataset. | 1 |
| $C_{1:T}$ | = | The hidden states. $C_i \in \{1, 2\}$ | $T$x1 |
| $\boldsymbol{\delta}$ | = | Initial probability distribution for the Markov chain. | $m$x1 |
| $\boldsymbol{\gamma}$ | = | Transition probability matrix. | $m$x$m$ |
| $X_{1:T}$ | = | The observations. | $K$x$T$ |
| $X_i$ | = | Observation at time $i$. A sparse vector. | $K$x1 |
| $n_i$ | = | Observed number of words in the tweet at time $i$. | 1 |
| $X_i \vert C_i$ | $\sim$ | Multinomial$(n_i, p_{c_i})$ | |
| $\boldsymbol{p}$ | = | The matrix of probabilities, combining $p_{c=1}$ and $p_{c=2}$. | $K$x$m$ |

The model summary. In bold, $(\delta, \gamma, p)$ are the parameters to be learned.

$i$, represented as a vector of frequencies.

Finally as this is a hidden Markov model (HMM), we assume that given $C_i$, $X_i$ is conditionally independent of all other $X_j$ and $C_j$. In this case it seemed most natural to model $X_i|C_i$ as multinomial distribution. We have $K$ possible words in the dataset, and a finite number of occurrences (successes) in each tweet. If the model fits as expected, the multinomial word probabilities should change depending on which of the $m$ states we are in.

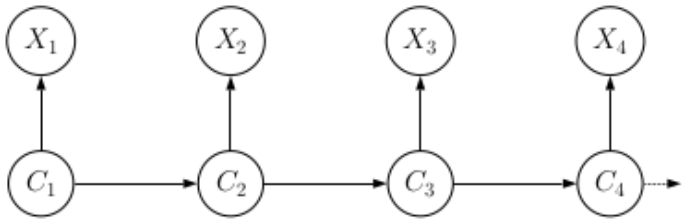


Figure 2.1: Illustrating the HMM dependency structure.

These are significant simplifying assumptions. For a HMM we are assuming $X_i|C_i$ are independent of each other for all $i$, but the true dependence structure of tweets is likely much more complex because we have many participants with many conversations going on at the same time. It is unlikely that the Markovian

assumption holds exactly. However HMMs work with speech recognition and the dependence structure is complex there too, so we hoped this approximation would work on Twitter data.

We use a maximum likelihood framework to fit the model. The log-likelihood of the HMM is

$$\begin{aligned}\log\Big(Pr(X_{1:T}, C_{1:T})\Big) &= \log\Big(\delta_{c_1}\prod_{t=2}^{T}\gamma_{c_{t-1},c_t}\prod_{t=1}^{T}p_{c_t}(x_t)\Big)\\ &= \log\delta_{c_1} + \sum_{t=2}^{T}\log\gamma_{c_{t-1},c_t} + \sum_{t=1}^{T}\log p_{c_t}(x_t).\end{aligned} \tag{2.1}$$

As earlier, $X_{1:T}$ are the observations up to time $T$ and $C_{1:T}$ are the hidden Markov chain states. We have 3 main parameters of interest: $\theta = (\delta, \gamma, p)$. By definition $\delta$ is the initial Markov chain distribution (at $c_1$), $\gamma_{jk}$ are the transition probabilities from Markov chain states $j$ to $k$, and $p_{c_t}$ are the multinomial word probabilities in state $c_t$.

We calculate $p_{c_t}(x_t)$ in Equation 2.1 using the multinomial probability mass function. Given a tweet at time $i$ of observed length $n_i$, we are implicitly modeling it as a multinomial process of extracting $n_i$ words from $K$ unique categories with associated probabilities $p_{c_i}$.

We use the EM algorithm [6] to maximize the likelihood with respect to $\delta, \gamma$ and $p$, then use the Viterbi algorithm [7] get the most likely sequence of hidden states $C$.

## 2.2 Fitting the Model Using the EM Algorithm

To set up for EM, we follow [8]. We first rewrite the log-likelihood as if we knew the hidden states C.

$$\begin{aligned}\log\Big(Pr(X_{1:T}, C_{1:T})\Big) = \sum_{j=1}^{m}u_j(1)\log\delta_j + \sum_{j=1}^{m}\sum_{k=1}^{m}\Big(\sum_{t=2}^{T}v_{jk}(t)\Big)\log\gamma_{c_{t-1},c_t}\\ + \sum_{j=1}^{m}\sum_{t=1}^{T}u_j(t)\log p_{c_t}(x_t).\end{aligned} \tag{2.2}$$

This is just a rewrite of Equation (2.1). We have $u_j(t) = 1_{c_t=j}$, and we sum over all possible states. Similarly $v_jk(t) = 1_{c_{t-1}=j \text{ and } c_t=k}$. In the E step of EM, we use the backwards and forwards algorithms to calculate $\hat{u}$ and $\hat{v}$ as their conditional expectations of $u$ and $v$ given the observations. Then in the M step we maximize the log-likelihood with $\hat{u}$ and $\hat{v}$ replaced for $u$ and $v$.

### 2.2.1 E step

More precisely, in the E step we calculate $\hat{u}$ and $\hat{v}$ as

$$\begin{aligned} \hat{u}_j(t) &= E(u_j(t)|x_{1:T}) = P(c_t = j|x_{1:T}) = \alpha_j(t)\beta_j(t)/L_T \\ \hat{v}_{jk}(t) &= P(c_{t-1} = j \text{ and } c_t = k|x_{1:T}) = \alpha_j(t-1)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T. \end{aligned} \tag{2.3}$$

In the above equation $L_T$ is the likelihood of the observations, and $\alpha$ and $\beta$ are from the forward and backward algorithms detailed below.

#### 2.2.1.1 Forward procedure

We define $\alpha_i(t) = P(X_1 = x_1, ..., X_t = x_t, C_t = i)$ as the probability of seeing the $x_1, x_2, ..., x_t$ observations and ending up in (hidden) state $i$ at time $t$. Using the recursive forward algorithm, we can calculate this conveniently.

$$\begin{aligned} \alpha_i(1) &= \delta_i p_i(y_1) \\ \alpha_j(t+1) &= p_j(x_{t+1}) \sum_{i=1}^{m} \alpha_i(t)\gamma_{ij}. \end{aligned} \tag{2.4}$$

#### 2.2.1.2 Backward procedure

Let $\beta_t(i) = P(X_{t+1} = x_{t+1}, ..., X_T = x_T|C_t = i)$. Here $\beta_t(i)$ represents the probability of seeing the ending observation sequence $x_{t+1}, ..., x_T$ given starting (hidden) state $i$ at time $t$. We calculate $\beta_i(t)$ as

$$\begin{aligned} \beta_i(T) &= 1 \\ \beta_i(t) &= \sum_{j=1}^{m} \beta_j(t+1)\gamma_{ij}p_j(y_{t+1}). \end{aligned} \tag{2.5}$$

Using the definitions, we can show $\alpha_i(t)\beta_i(t) = P(X_{1:T} = x_{1:T}, C_t = i)$. We then calculate the likelihood $L_T = P(X_{1:T} = x_{1:T}) = \sum_i \alpha_i(t)\beta_i(t)$ by summing over all Markov chain states, for any t. In practice it is best to use $t = T$, since $\beta_i(T) = 1$ for all $i$.

### 2.2.2 M-step

In the M-step of the EM algorithm, we first replace $u$ and $v$ with $\hat{u}$ and $\hat{v}$ in Equation 2.3. We then maximize the log-likelihood in Equation 2.2 with respect to $\delta, \gamma$, and the $K * m$ probabilities in $p$, We set the derivative equal to zero and get closed form expressions for all parameters as

$$\begin{aligned} \delta_j &= \frac{\hat{u}_j(1)}{\sum_j \hat{u}_j(1)} \\ \gamma_{jk} &= \frac{f_{jk}(t)}{\sum_k f_{jk}}, \\ &\text{where } f_{jk}(t) = \sum_{t=2}^{T} \hat{v}_{jk}(t). \end{aligned} \tag{2.6}$$

Clearly $\delta_j$ is the expected proportion of time spent in state $j$ at $t = 1$, and $\gamma_{jk}$ is the expected number of transitions from state $j$ to state $k$ normalized by the number of transitions away from state $j$ (to any state including $j$).

The closed form update for the multinomial probabilities $p_j$ are found using Lagrange multipliers. If we assume there are $k$ event probabilities $p_j = (p_j^{(1)} \cdots p_j^{(k)})$ such that $\sum_{i=1}^{k} p_j^{(i)} = 1$, the update equations for each $p_j^{(i)}$ are

$$p_j^{(i)} = \frac{\sum_t \hat{u}_j(t) x_t^{(i)} / n_t}{\sum_t \hat{u}_j(t)}, \tag{2.7}$$

where $x_t^{(i)}$ is the number of successes for event $i$ at time $t$, and $n_t$ is the total number of successes (number of words) at time $t$.

As usual in EM, the E step is computed with the previous iteration's parameter estimates. Then in the M step we get the parameter estimates for our current iteration. In our dataset EM converged in less than 10 iterations. Since EM only gives a local maximum, we performed 3 repetitions (randomly changing the initial

values each time) and chose the largest likelihood.

## 2.3 Finding the most likely hidden states using the Viterbi algorithm

Once we have the parameter estimates $\hat{\theta}$ at the (local) maximum, we use the Viterbi algorithm [7] to find the most likely sequence of hidden state sequence $C_{1:T}$ that could have generated the observations. This can be quite important, for example in speech recognition where the sequence might indicate the actual words in a sentence, given the observations as frequency and amplitude sound data. Or in handwriting recognition we are given image data and we want to infer the intended text.

We want to maximize $P(C_{1:T}|X_{1:T};\hat{\theta})$. Or equivalently and more conveniently, we maximize the joint distribution $P(X_{1:T}, C_{1:T};\hat{\theta})$ given in the log-likelihood, Equation (2.1). Doing an exhaustive search with $2^T$ possible hidden path sequences is computationally infeasible for large $T$, so instead we use the Viterbi algorithm. We first run the recursive algorithm forward to find the final state of the optimal state sequence. After we know the final state, we then work backwards to find the 2nd to last state, etc.

Following [8], the Viterbi algorithm begins by defining

$$\begin{aligned} \xi_i(1) &= P(C_1 = i, X_1 = x_1) = \delta_i p_i(x_1) \\ \xi_i(t) &= \max_{c_1, c_2 \cdots c_{t-1}} P(C_{1:t-1} = c_{1:t-1}, C_t = i, X_{1:t} = x_{1:t}), \end{aligned} \tag{2.8}$$

where $\xi_j(t)$ is calculated recursively as

$$\xi_j(t) = (\max_i(\xi_i(t-1)\gamma_{ij})p_j(x_t).$$

We then find the optimal sequence of states by backwards recursion.

$$\begin{aligned} \hat{c}_T &= \arg\max_i \xi_i(T) \\ \hat{c}_t &= \arg\max_i(\xi_i(t)\gamma_{i,i_{t+1}}) \end{aligned} \tag{2.9}$$

Chapter 3

# Model and Methods

As earlier, we started with the shooting data comprising all tweets from January 20th to January 22nd. After cleaning it according to Section 1.1.2, we fit a 2-state HMM with results shown in Figure 3.1 below[1].

The graph at the top of Figure 3.1 shows the most likely sequence of states after fitting the HMM and running the Viterbi algorithm. It is difficult to see the pattern until we take a (non-overlapping) average of every 50 values, giving the 2nd graph. The vertical line on the 2nd graph shows the position of the first shooting tweet. Clearly the model is picking up on the signal; state 1 seems to be associated with the shooting. But given the difficulty in fitting just 2 states, we found it impossible to fit 3 or more states in an interpretable way. The result was too noisy. We decided instead to use a change point algorithm.

[1]See Appendix A for more details

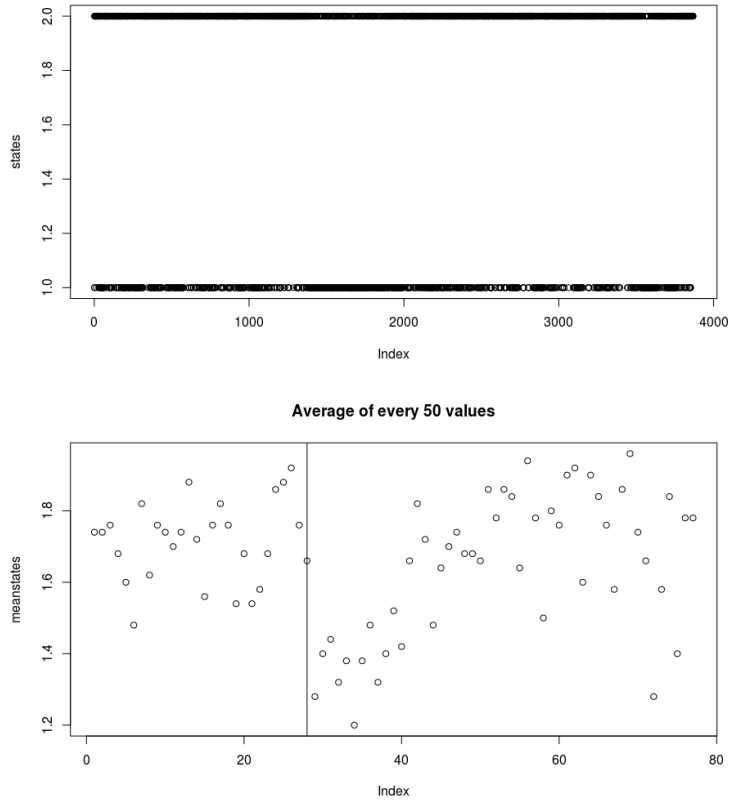


Figure 3.1: Above: The most likely sequence of states after Viterbi. Below: A non-overlapping average of every 50 values in the first graph. The vertical line at t=28 shows the location of the first shooting tweet.

## 3.1 Estimating change points

For a change point algorithm, we choose a point on the timeline on which to split the timeline into two segments, then fit a hidden Markov Model on each of the two segments. We then add the two individual log-likelihoods and compare that sum with the sum from a different candidate change point. The maximum log-likelihood over the whole timeline identifies the optimal change point.

More precisely, we are performing an exhaustive search to get

$$\hat{\tau}_1 = \arg\max_{\tau} ML(\tau) \qquad\qquad \tau \in [1:T]$$

given

$$\begin{aligned} ML(\tau) &= l(x_{1:\tau}|\hat{\theta}_1) + l(x_{(\tau+1):T}|\hat{\theta}_2) \\ &= \log\Big(\delta_{c_1}\prod_{t=2}^{\tau}\gamma_{c_{t-1},c_t}\prod_{t=1}^{\tau}p_{c_t}(x_t)\Big) + \log\Big(\lambda_{c_1}\prod_{t=\tau+2}^{T}\phi_{c_{t-1},c_t}\prod_{t=\tau+1}^{T}q_{c_t}(x_t)\Big). \end{aligned} \tag{3.1}$$

As we saw in Section 2.2, $l(x_{a:b}|\hat{\theta})$ is the log-likelihood after fitting a 2-state HMM on the Twitter data $x_{a:b}$. Then $\hat{\theta}_1 = (\delta, \gamma, p)$ and $\hat{\theta}_2 = (\lambda, \phi, q)$ are the parameter estimates at the maximum, as found using EM.

Figure 3.2 below shows an example of the exhaustive search process. We calculate $ML(\tau)$ for $\tau \in [1:T]$ and find the first change point is around $\hat{\tau}_1 = 1300$.

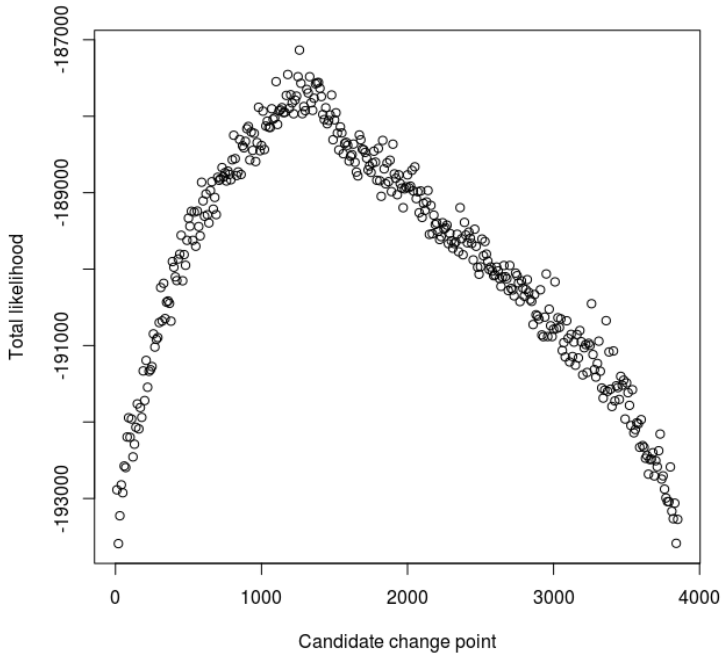


Figure 3.2: Log-likelihood values $ML(\tau)$ for various candidate change points $\tau$.

After finding the first change point, we repeated this same process on the

smaller dataset ($x_{\hat{\tau}_1:T}$) starting from the first change point to the end. The algorithm Searchforchangepoint is formally described below.

**Data**: Tweets ($x_{\hat{\tau}_n:T}$)
**Result**: Next change point $\hat{\tau}_{n+1}$
initialization;
**for** $i \in \hat{\tau}_n : T$ **do**
  Using EM with 3 random initializations, calculate the two likelihoods $ML(\tau) = l(x_{\hat{\tau}_1:\tau}|\hat{\theta}_1) + l(x_{(\tau+1):T}|\hat{\theta}_2)$.
**end**
$\hat{\tau}_{n+1} = \arg\max_\tau ML(\tau)$ $\tau \in [\hat{\tau}_n : T]$

**Algorithm 1:** Searchforchangepoint

We continued this process until no further change points could be found, i.e., the graph increased monotonically, indicating that the change point is at the boundary. Note that we identify the change points sequentially, with new change point locations conditional on previously identified change points. The main advantage is this procedure is computationally fast. After we find the change points, Section 3.2 deals with confidence intervals and Section 3.3 deals with model selection.

## 3.2 Confidence intervals

We used a parametric bootstrap technique to find confidence intervals indicating uncertainty on the location of the change points. We first fit a HMM on both sides of our known change point to get two sets of parameters $\hat{\theta}_1$ and $\hat{\theta}_2$. We then used the fitted parameter values to generate data on each side of the change point, i.e., we simulated data $X_i$ according to $\hat{\theta}_1$ for all $i$ before the change point, then simulated $X_i$ according to $\hat{\theta}_2$ for all $i$ after the change point.

We then used our algorithm Searchforchangepoint to find a change point. We would repeat this process - generate more data based on the same fitted values and find yet another change point. The confidence interval is then given by the quantiles of the sampling distribution. I chose a sample size of 100. The entire

algorithm is formally described below.

**Data**: Tweets ($x_{\hat{\tau}_{n-1}:T}$), computed change point $\hat{\tau}_n$.
**Result**: W = 95% confidence interval for the location of change point $\hat{\tau}_n$.
initialization;
Obtain $\hat{\theta}_1$ by fitting a 2-state HMM on ($x_{\hat{\tau}_{n-1}:\hat{\tau}_n}$) ;
Obtain discrete probability distribution $f_1$ from the observed frequencies of tweet lengths on ($x_{\hat{\tau}_{n-1}:\hat{\tau}_n}$);
Obtain $\hat{\theta}_2$ by fitting a 2-state HMM on ($x_{\hat{\tau}_n:T}$) ;
Obtain distribution $f_2$ from the frequencies of tweet lengths on ($x_{\hat{\tau}_{n-1}:\hat{\tau}_n}$);
**for** $i \in [1:100]$ **do**
    generate data ($\hat{x}^{(i)}_{\hat{\tau}_{n-1}:T}$) in 2 steps as follows:
    **for** $j \in [\hat{\tau}_{n-1} : \hat{\tau}_n]$ **do**
        Get tweetlength(j) by sampling from $f_1$ ;
        Generate tweet(j) by sampling from Multin( tweetlength(j), $\hat{\theta}_1(p)$ );
    **end**
    **for** $j \in [\hat{\tau}_n + 1 : T]$ **do**
        Get tweetlength(j) by sampling from $f_2$ ;
        Generate tweet(j) by sampling from Multin( tweetlength(j), $\hat{\theta}_2(p)$ );
    **end**
    Samples(i) = Searchforchangepoint($\hat{x}^{(i)}_{\hat{\tau}_{n-1}:T}$) #as seen in Section 3.1
**end**
W = .025 and .975 quantiles of Samples ;

**Algorithm 2:** Computing confidence intervals

As we see in Figure 3.3, the output sampling distribution is symmetrically distributed fairly close to the input change point $\hat{\tau}_n = 1360$.

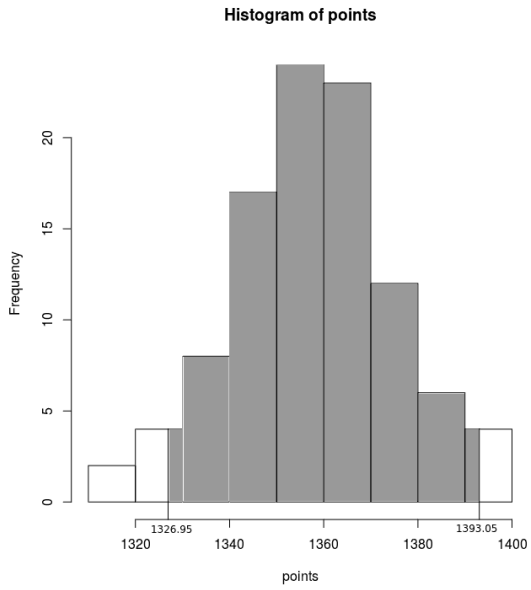


Figure 3.3: Histogram of the bootstrap sample generated on the first change point in the shooting dataset. The shaded areas mark everything inside the 95% confidence interval

## 3.3 Model Selection using AIC

The above procedure with confidence intervals gives an idea about the variability associated with our estimate, but it does not help with model selection. We want to know whether we should include the identified change point or not. We initially tried to use bootstrapping and a likelihood ratio test, but the null distribution was too complicated. From the literature we found people typically use information criteria to decide between the null and alternative hypothesis [9]. We used the Akaike Information Criterion (AIC) [10] because it worked reasonably well in simulation. The preferred model by this criterion is the one with the lowest AIC value.

One advantage of AIC is its simplicity. Let $L$ be the maximum likelihood after fitting the model, and let $k$ be the number of estimated parameters. Assuming we have $m = 2$ states and we are fitting a model without change points, $\delta$ has 1 free parameter. $\gamma$ has 2, and $p$ has $4000 * 2 - 2$ free parameters. This gives us $k = 8001$.

Then the AIC of a model is given by

$$AIC = 2k - 2\log(L). \tag{3.2}$$

When comparing two models, the preferred model by this method is the one with the smaller AIC value. The first term is a penalty term that increases when a model has more estimated parameters. The second term rewards goodness of fit (a better fitting model has a smaller minus log-likelihood). By increasing AIC, the penalty in the first term discourages overfitting (overfitting occurs because increasing the number of parameters in the model usually improves the goodness of fit).

After finding the change points according to Section 3.1, we performed model selection by fitting HMMs on the segments, adding log likelihoods, then calculating AIC. We selected the model with the lowest AIC.

## 3.4 Computing summaries using cosine similarity

Once we have estimated the change points, we want to summarize the tweets in the intervals between change points. This gives an idea of what people were talking about between change points. For this, we use cosine similarity. Since we already have a vector representation for the tweets, we can define the distance between tweets as the cosine of the angle between vectors. This measure is very similar to correlation; if the vectors are centered, cosine similarity and correlation are identical.

$$\begin{aligned} \text{CosSim}(x, y) &= \frac{\mathbf{X} \cdot \mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \\ \text{Correlation}(x, y) &= \text{CosSim}(x - \bar{x}, y - \bar{y}). \end{aligned} \tag{3.3}$$

So, similar to Chakrabarti and Punera [1], we define the score of a tweet as the sum of all cosine distances between the tweet and all others (inside one of the

intervals between change points). Since the cosine value is larger if the angle is smaller and two vectors are more similar, we want as a summary measure the top $n$ tweets with highest scores. This procedure is similar to finding central points. Given a cloud of points, the ones in the center are closest to every other point in the cloud and can be used as a summary measure. The algorithm is formally described below.

**Data**: Tweets $(x_{\hat{\tau}_{n-1}:\hat{\tau}_n})$
**Result**: Set of n summary tweets S
initialization: let $Z = [\hat{\tau}_{n-1} : \hat{\tau}_n]$;
**for** $i \in Z$ **do**
  score$(x_i) = \sum_{j \in Z} CosSim(x_i, x_j)$
**end**
S = top n tweets with highest scores

**Algorithm 3:** Computing summary tweets

Chapter 4

# Results and Discussion

## 4.1 Shooting dataset

We used the methods of Section 3.1 on the shooting dataset and found 3 change points: at dataset indices 1360, 2520, and 3195. The first change point served as a check that this method works. This is because the first shooting tweet occurred at 1408, so I found there was reasonable agreement. All three change points were selected; the smallest AIC was achieved by the three change point model. Summary data including actual time and confidence intervals are given below.

Table 4.1: Shooting dataset change points

| Change Point Location | Actual time, 2014 | 95% Confidence Interval |
|---|---|---|
| $\hat{\tau}_1 = 1360$ | 21 Jan 11:12 AM EST | [1327.0, 1393.1] |
| $\hat{\tau}_2 = 2520$ | 21 Jan 4:45 PM EST | [2484.4, 2566.6] |
| $\hat{\tau}_3 = 3195$ | 22 Jan 3:19 AM EST | [3165.2, 3227.0] |

Below we show summaries of tweets in the four time periods between the three change points. We use the techniques in Section 3.4 to find and output the top 5 tweets closest to everything else by cosine distance.

| Summary of top 5 tweets in the Shooting dataset | |
|---|---|
| Time Index | Representative tweets |
| 0-1360 | To the girl I elbowed in the head while getting off of the Silver Loop this morning: I am so sorry! I hope you don't have a concussion.<br>the awesome thing about a January birthday is that the weather is always god awful and I have to go to school<br>I get asked for favors all the time but the second I ask someone to do the simplest thing it's like they don't even know who I am. #SAD<br>Extremely proud of the Denver Broncos. It's surreal to get to the Superbowl and you have to give the Patriots a ton of credit as well.<br>If you are a sports fan, there is no way in hell you should be rooting for the thugs in Seattle to beat Peyton and the Broncos |
| 1361-2520 | My next class was supposed to be in the building that the shooting happened... Thank god I wasn't in there.<br>I REFUSE to go to the rest of my classes the same afternoon as a shooting. I won't go.<br>I hope everyone on Purdue's campus stays safe. My thoughts and prayers go out to the family of the victims.<br>No Purdue. I will not continue regular operations and go to the physics building which is right next to the EE.<br>So I was in the building where there is a shooting this is crazy |
| 2521-3195 | @kaXXXX I miss you too! And I think of you very often myself. I look forward to when you can come join the toobahs!<br>I'm ready for my parents waking up soon and finding out the news and calling me with the frightening voice. But meow I hope you a good dream<br>I think the vigil tonight is Great but I really wish it was a few days later I'm still a little shook by today and would rather stay away<br>@JiXXXX I can and I will. First stage is denial of course so I understand that it's hard to see it at first.<br>@saXXXX I honestly wasnt able to go. Or I would. Saw it on tv. Lots of people standing in the 5 degree weather. Great to see |
| 3195-end | I really need to go work out....however I feel like I have to look nice since 99% of all the good looking guys on campus are there<br>Between HJR3 and all these Purdue shootings I can't decide whether I want to leave the state or stay and try to help<br>The minute and a half my hot pocket is cooking is the longest minute in a half known to man.<br>One of my good friends new the shooter. What a small world. You just never know who is going to snap. So tragic....<br>I uninstalled league of legends because now I have 3 extra hours per day to be a better person instead of wasting my time on a shitty game |

Given the summary tweets, interpretation of the regimes is straightforward. Before the first change point we find everyone is just talking about normal life events. Between the first and the second change point, the summary tweets are about shock and worry. Between the second and the third, the level of worry has gone down and people are talking about the vigil that night. After the third change point, only 2 of the 5 tweets reference the shooting, and the rest have gone back to normal life events.

## 4.2 Super Bowl dataset

We used the methods of Section 3.1 on the Super Bowl dataset and found 2 significant change points: at 910 and 1620. We found one more at 2175, but it was not selected. The model with 2 change points had the smallest AIC. Summary data including confidence intervals are given below.

Table 4.2: Super Bowl dataset change points

| Change Point Location | Actual time, 2014 | 95% Confidence Interval |
|---|---|---|
| $\hat{\tau}_1 = 910$ | 02 Feb 3:31 AM EST | [885.9, 953.6] |
| $\hat{\tau}_2 = 1620$ | 02 Feb 9:16 PM EST | [1587.8, 1642.1] |

Below we show summaries of tweets in the three time periods between the three change points.

| Summary of top 5 tweets in the Super Bowl dataset | |
|---|---|
| Time Index | Representative tweets |
| 0-910 | @SaXXXX thought lebron was shaq. #totalhockeyfan #notevenclose #NBA2k14<br>Longest day of my life but I finally made it through formal rush #onedownone-togo<br>@_MeXXXX lol .. You'll be fine my nigga<br>Everything was in my coat! Not a failure after all!<br>Great evening visiting with m_spXXXX!! I love this cousin of mine!!! #boilerUP #snow #cousins |
| 911-1620 | Rob and I only ones at the party that want the Seahawks to win<br>Torn between the excitement that it's the Super Bowl and the despair that it's the last day of a Football for 6 months ????????<br>I have a math tutoring session during the Super Bowl...so if that's not dedication to trig I don't know what is.<br>Can the month of February just not happen? I'm just stressed thinking about all the things I have to study for.<br>If the Seahawks win tonight @DaXXXX will be just the second Black QB to win a Superbowl. I hope he balls and gets the #SBMVP too! |
| 1621-2175 | I want to be a millionaire #EsuranceSave30<br>I really want to get back to my wrestling shape, but I know that's not possible unless I train with the team again.<br>@MoXXXX a year ago today you tweeted the tweet that made it in the yearbook. I was telling my friends and wanted to tell you.<br>I think that @fruXXXXX enjoyed himself a little bit too much last night. @BlXXXX and I are going to have to take him out. ????<br>I wish I knew how to talk to girls |
| 2176-end | You stole my happy, you made me cry took the lonely and took me for a ride and I wanna undoooooo it you had my heart now I want it back ????<br>I need to go clothes shopping but this weather makes me mad I always have to wear a oversized jacket<br>When I'm delirious cause I haven't really slept well 4 the past few days I do crazy things like go to the gym so I guess it has it's perks.<br>I think the person who made my coffee put in an extra 473759592947 expresso shots .. I feel like I can do a triathlon right now<br>I feel the need to brag about how musically talented my roomie is. Currently listening to her sing All Of Me and play the piano ???????????????? |

The $t = 910$ change point corresponds to a time of approximately 3am on Feb 2. The game started at 6:30pm, but the whole day was Super Bowl Sunday. People on Twitter were excited about the game long before it started. All the summary tweets between the first and second change points reference the Super Bowl. The $t = 1620$ change point corresponds to 9:16pm. I looked at the data and found the game had not quite ended by 9:30, but people on Twitter were already moving on because the result was clear. The Seahawks defeated the Broncos 43-8.

So we could detect the end but not the beginning of the game. It also makes sense that this method detected no sub-events during the game (e.g., interceptions or touchdowns). This dataset did not contain much commentary about in-game details.

## 4.3 Computation time

These algorithms were implemented in R, and ran on an Intel i7-4790K processor. Calculating the multinomial probability mass function in R with such large $K$ took too long, so instead we called the GNU Scientific Library [11] through C. Fitting EM takes less than 1 minute. When properly parallelized, searching for a change point takes at most 30 minutes. Calculating confidence intervals takes up to 20 hours.

Chapter 5

# Conclusion and Future Work

In this thesis, we have derived a HMM-based change point algorithm that proves to be quite capable of finding reasonable change points in noisy data. We first introduced hidden Markov models, then discussed the methods used in this thesis. Using our methods on 2 different datasets, we derive change point estimates, confidence intervals, and perform model selection. We also show that it is also possible to reasonably summarize the tweets between change points using cosine similarity.

We can extend this work in many directions but the most urgent seems to be to find a way to reduce the noise in the hidden Markov model. Perhaps if we change the model a little as resembling Chakrabarti and Punera [1], the noisy results seen in Figure 3.1 could be cleaned up. If that works, we might be able to interpret more than 2 hidden states, and the change point model might not be necessary. We are also looking into combining similar words to reduce the dimension of the data. Perhaps then combining adjacent tweets might also improve performance.

This thesis does not take into account the time data, e.g., as seen in Figure 1.1. We attempted using a continuous-time hidden Markov model, but there was no improvement over the discrete-time HMM. Nevertheless we think there might be a way to improve results by taking into account the time data.

Similarly as we see in Figure 5.1 below, tweet lengths $n_i$ also change before and after the shooting. The vertical line indicates the position of the first shooting tweet. It appears that people tend to write longer sentences after the shooting. It might help to model $n_i$ using a Poisson or Negative Binomial distribution, with the parameters being state dependent.

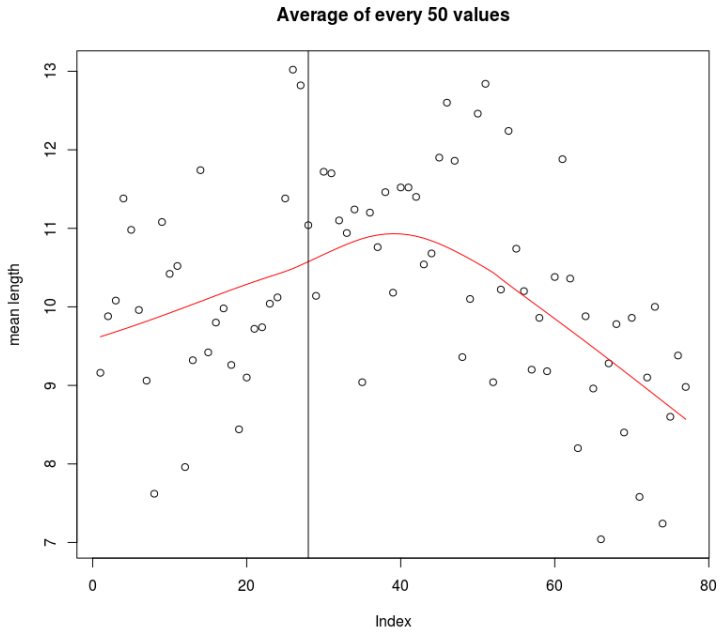


Figure 5.1: Plotting the average of every 50 tweet length values $n_i$ in the shooting dataset. The Lowess curve shows the trend. The vertical line shows the position of the first shooting tweet.

Finally, we are not entirely sure a hidden Markov model is necessary for getting good results in the change point algorithm. Perhaps there is a simpler model that works, e.g., without the Markovian assumption.

Appendix A

# Assessing the model fit on the shooting dataset

In Chapter 3, we fit a 2-state HMM on the shooting dataset and found very noisy results. This Appendix gives more details about the fitted model.

## A.1 Fitted parameter values

After fitting, we found the following values

$$\delta = (1, 0)$$
$$\gamma = \begin{pmatrix} 0.619 & 0.380 \\ 0.194 & 0.806 \end{pmatrix} \tag{A.1}$$

Displaying the multinomial probabilities $p$ is more difficult, as there are $K = 5000$ values in each of $m = 2$ states. We begin by graphing them below in Figure A.1. The index is arbitrary and just an artifact of the encoding process, as seen in Table 1.1 and 1.2. Note that (as seen in Figure 3.1) state 1 is associated with the shooting.

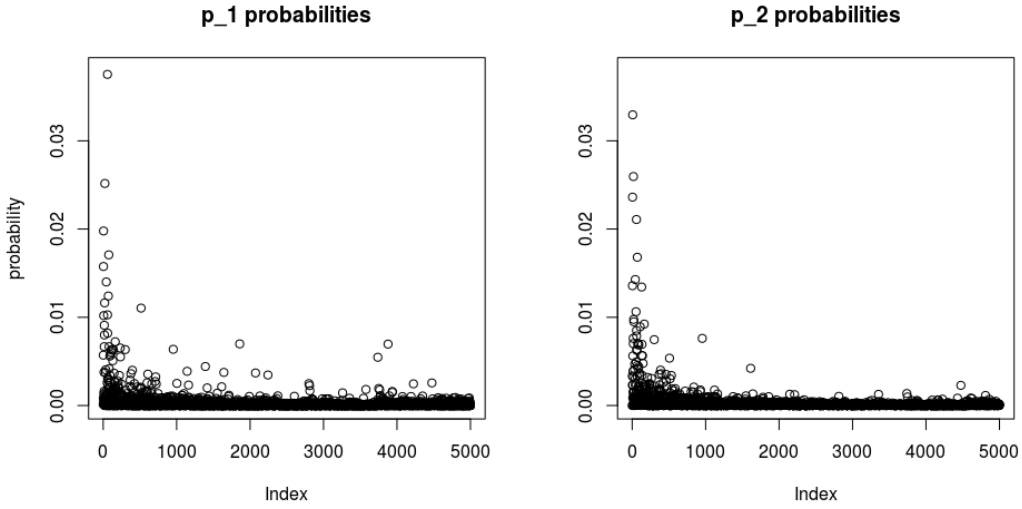


Figure A.1: Plotting the multinomial probabilities in each of the 2 states

By sorting the probabilities in descending order, we now output the words associated with the 70 largest probabilities in each state.

State 1: ['?', '.', 'the', 'in', 'is', 'you', 'a', 'and', 'purdu', 'this', 'i', "i'm", 'on', 'to', 'of', 'safe', '#prayforpurdu', 'that', 'at', 'xe2x82xac?', 'my', 'thank', 'so', 'for', 'are', 'stay', 'now', 'what', 'be', 'right', 'shoot', 'not', 'everyon', 'just', 'was', 'we', 'serious', "it's", 'campus', 'all', 'atoc', 'with', 'peopl', 'happen', 'build', 'pleas', '', 'shot', 'me', 'lol', 'no', 'get']

State 2: ['i', 'to', 'the', '?', 'a', 'you', 'is', 'my', 'it', 'and', '.', 'of', 'for', 'this', 'me', 'thank', 'that', 'so', 'in', "i'm", 'go', 'be', 'just', 'on', 'lol', 'not', 'are', "don't", 'love', 'like', 'with', 'was', 'no', 'have', 'class', 'at', 'all', 'up', "it's", 'but', 'we', 'get', 'if', 'do', 'know', 'now', 'what', 'good', 'about', 'day', 'today', 'purdu']

### A.1.1 Discussion

From the list of 70 largest probabilities, we see a possible reason for the intermingling of states seen in Figure 3.1. Clearly many words are common to both shooting and non-shooting conversations. Nevertheless state 1 picks up many words such as "safe", "shoot", "shot",and "prayforpurdu", which are associated with the shooting.

Appendix B

# HMM output with change points

In this Appendix, we show Figure 3.1b with and without the three change points marked. As mentioned earlier, we obtained the points by fitting a hidden Markov model on the entire shooting dataset, then running Viterbi to get predicted states. The graph shows a non-overlapping average of every 50 values.

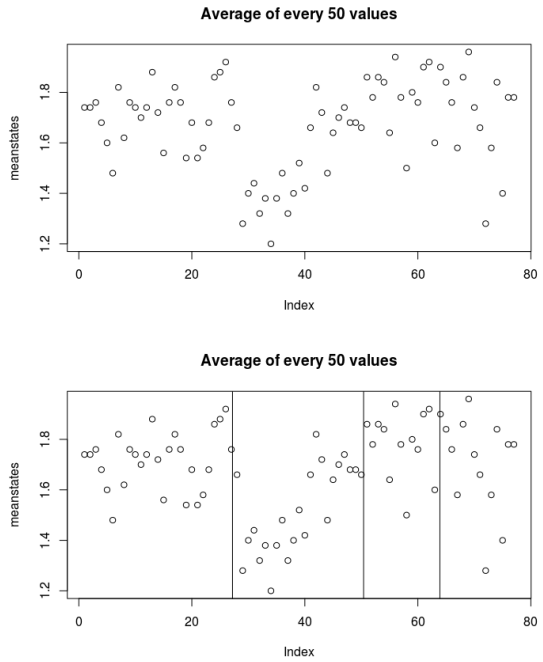


Figure B.1: Above: we reproduce Figure 3.1b. Below: we mark the 3 change points.

Figure B.1 shows that it's not always possible to find change points from looking at the hidden Markov model output. For example we might think there is a change point at Index = 41, but there isn't.

In an attempt to look for further patterns, we fit a 2-state HMM on each of the 4 windows between change points seen in Figure B.1b above. Figure B.2 below shows the output. We cannot conclude anything from this output. There might be a cause for worry if we saw a clear shift as in Figure B.1a above.

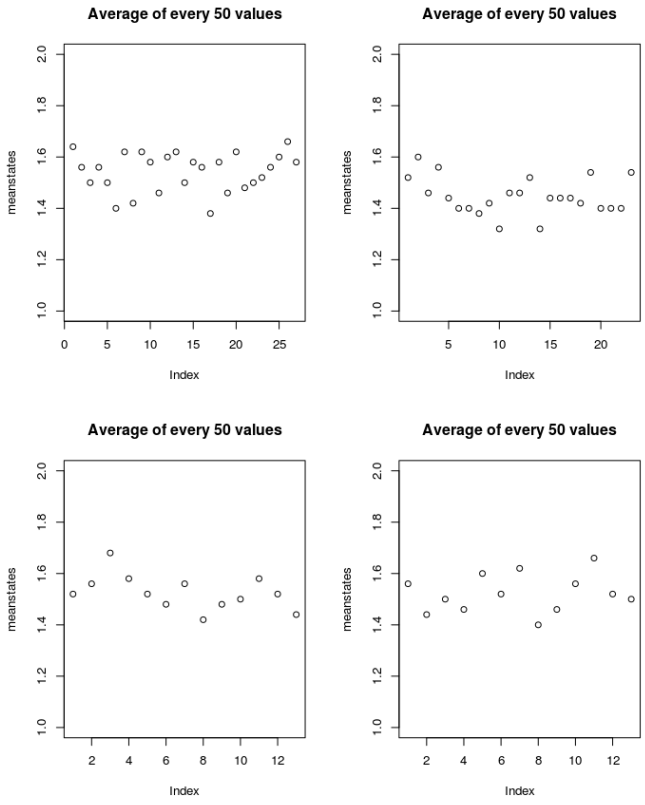


Figure B.2: We fit a HMM on each of the 4 sets of tweets between change points. No pattern was found.

# Bibliography


[1] Chakrabarti, D. and K. Punera (2011) "Event Summarization Using Tweets." *ICWSM*, **11**, pp. 66–73.

[2] Kubo, M., R. Sasano, H. Takamura, and M. Okumura (2013) "Generating live sports updates from twitter by finding good reporters," in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, IEEE Computer Society, pp. 527–534.

[3] Sakaki, T., M. Okazaki, and Y. Matsuo (2010) "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*, ACM, pp. 851–860.

[4] Gross, D. (2014), "Super Bowl sets Twitter records," . URL `http://www.cnn.com/2014/02/03/tech/social-media/super-bowl-social-twitter/`

[5] Porter, M. and R. Boulton (2001), "Snowball stemmer," .

[6] Dempster, A. P., N. M. Laird, and D. B. Rubin (1977) "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

[7] Forney, G. D. (1973) "The viterbi algorithm," *Proceedings of the IEEE*, **61**(3), pp. 268–278.

[8] Zucchini, W. and I. L. MacDonald (2009) *Hidden Markov models for time series: an introduction using R*, CRC press.

[9] Killick, R. and I. Eckley (2014) "changepoint: An R package for changepoint analysis," *Journal of Statistical Software*, **58**(3), pp. 1–19.

[10] Akaike, H. (1974) "A new look at the statistical model identification," *IEEE transactions on automatic control*, **19**(6), pp. 716–723.

[11] Team, G. et al. (2015), "Gnu scientific library–reference manual," .