Trend or No Trend: A Novel Nonparametric Method for Classifying Time Series

Zhang Zhang

Project Advisor: Prof. Aravind Srinivasan

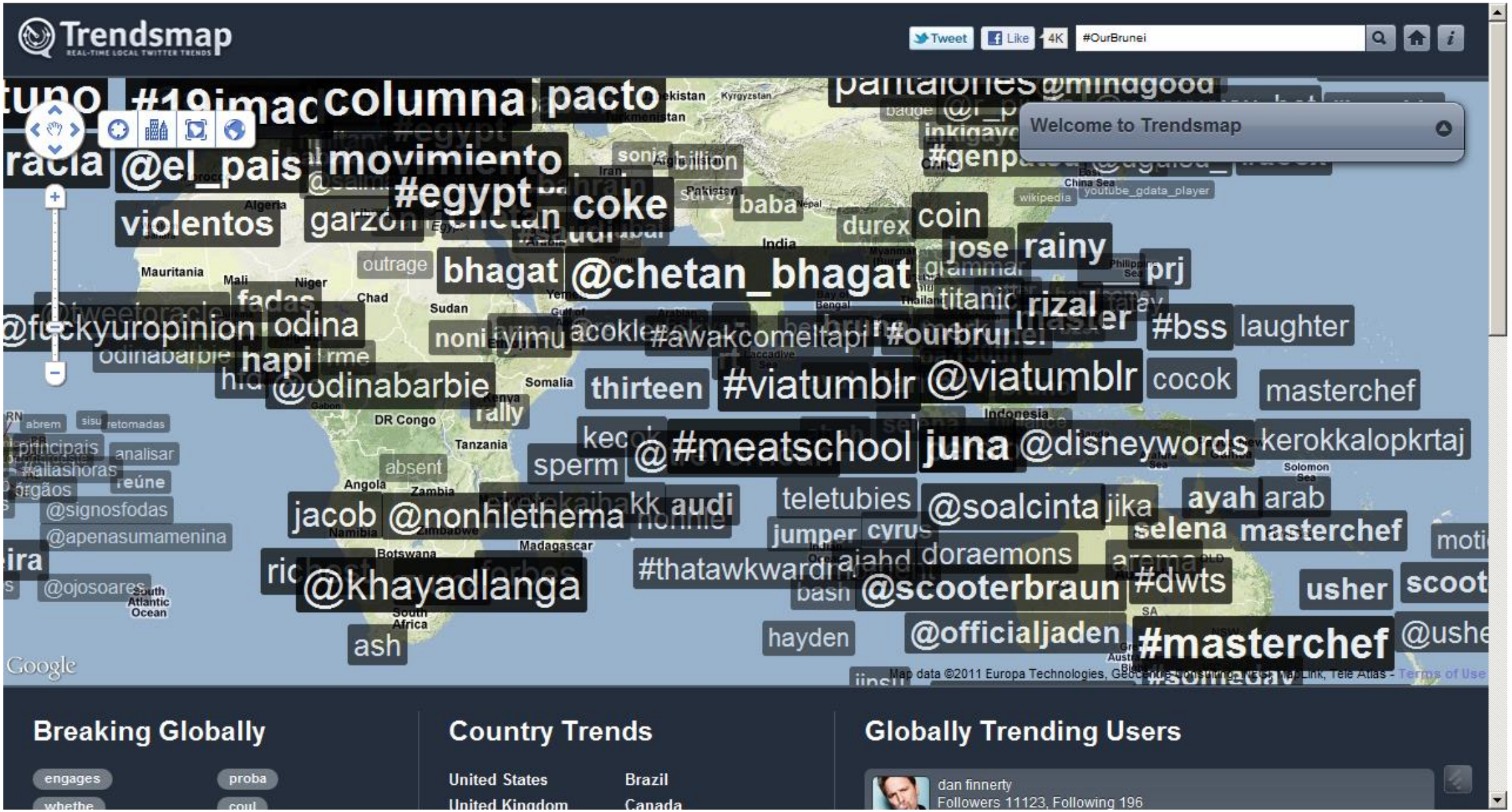
Course Advisor: Prof. Kayo Ide, Prof. Radu Balan

Outline

- Motivation
- Theoretical Latent Source Model
- Application on Twitter Classification
- Implementation
- Validation
- Results Discussion

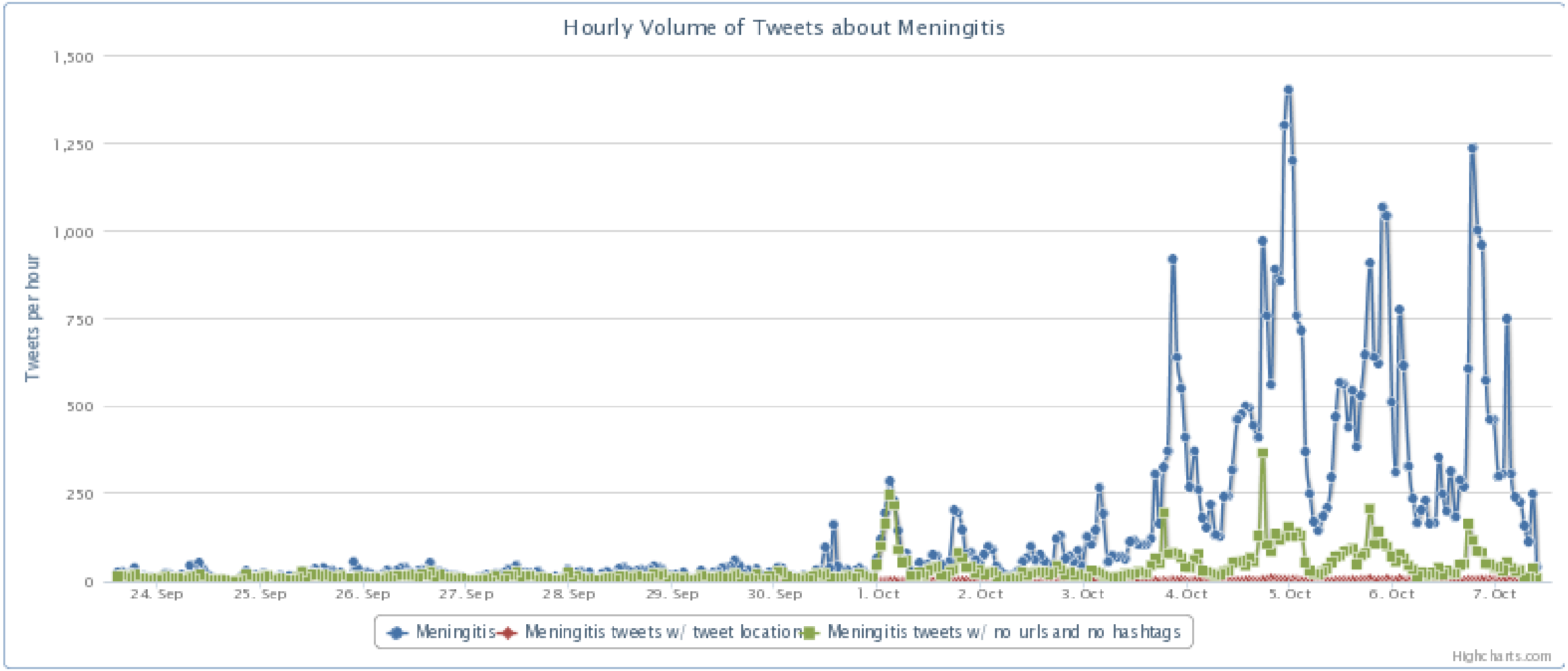
Motivation

Webpage Screenshot

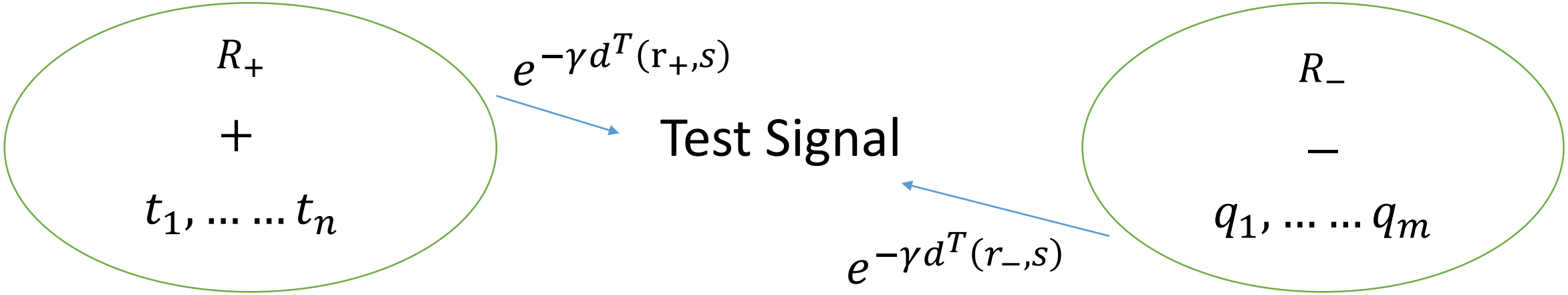


http://trendsmap.com/

Motivation


Hourly Volume of Tweets about Meningitis
Tweets per hour
1,500
1,250
1,000
750
500
250
0
24. Sep
25. Sep
26. Sep
27. Sep
28. Sep
29. Sep
30. Sep
1. Oct
2. Oct
3. Oct
4. Oct
5. Oct
6. Oct
7. Oct
Meningitis
Meningitis tweets w/ tweet location
Meningitis tweets w/ no urls and no hashtags
Highcharts.com

Weighted Majority Voting



$$d^T(r,s)$$

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

$$\hat{L}^T(s;\gamma) = \begin{cases} +1 & if \sum_{r \in R_+} e^{-\gamma d^T(r,s)} \geq \sum_{r \in \backslash R_-} e^{-\gamma d^T(r,s)} \\ -1 & \text{otherwise} \end{cases}$$

A latent Source Model and Theoretical Guarantees

Assumptions:

1. There are m unknown latent sources that generate observed time series. Let $\mathcal{L}$ denote the set of all latent sources; Let $\mathcal{L}_+ \subset \mathcal{L}$ be the set of latent sources with label +1, and $\mathcal{L}_- \subset \mathcal{L}$ be the set of those with label -1.
2. Sample latent source V from $\mathcal{L}$ uniformly at random.
3. Sample integer time shift Δ uniformly from $\{0,1,2,\ldots,\Delta_{\max}\}$.
4. Output time series $S: \mathbb{Z} \to \mathbb{R}$ to be latent source V advanced by Δ time steps, followed by adding noise signal $E: \mathbb{Z} \to \mathbb{R}$, i.e., $S(t) = V(t+\Delta) + E(t)$. The label associated with the generated time series S is the same as that of V. Entries of noise E are i.i.d. zero-mean sub-Gaussian with parameter σ, which means that for any time index t,

$$\mathbb{E}\left[\exp\left(\lambda E(t)\right)\right] \leq \exp\left(\frac{1}{2}\lambda^2\sigma^2\right) \qquad for\ all\ \lambda \in \mathbb{R}$$

Classification

If we knew the latent sources and if noise entries E(t) were i.i.d. $N\left(0, \frac{1}{2\gamma}\right)$ across t, then the maximum a posteriori estimate for label L given an observed time series S = s is

$$\hat{L}_{MAP}^{T}(s;\gamma) = \begin{cases} +1, & if\ \Lambda_{\mathrm{MAP}}^{T}(s;\gamma) \geq 1 \\ -1, & otherwise \end{cases}$$

where

$$\Lambda_{\mathrm{MAP}}^{T}(s;\gamma) \triangleq \frac{\sum_{v_+ \in V_+} \sum_{\Delta_+ \in D_+} \exp(-\gamma \parallel v_+ * \Delta_+ - s \parallel_T^2)}{\sum_{v_- \in V_-} \sum_{\Delta_- \in D_+} \exp(-\gamma \parallel v_- * \Delta_- - s \parallel_T^2)}$$

Classification

However, we do not know the latent sources, nor do we know if the noise is i.i.d. Gaussian. We assume we have the access to the training as given in Weighted Majority Voting. Then we approximate the MAP classifier by using training data as a proxy for the latent sources.

$$\Lambda^T(s;\gamma) \triangleq \frac{\sum_{r_+ \in R_+} \exp\left(-\gamma\left(\min_{\Delta_+ \in D} \| r_+ * \Delta_+ - s \|_T^2\right)\right)}{\sum_{r_- \in R_-} \exp\left(-\gamma\left(\min_{\Delta_- \in D} \| r_- * \Delta_- - s \|_T^2\right)\right)}$$

Applications may call for trading off true and false positive rates. We can do this by generalizing the decision rule:

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

Theoretical Guarantee

Let $m_+ = |\mathcal{L}_+|$, and $m_- = |\mathcal{L}_-| = m - m_+$. For any $\beta > 1$, under the latent source model with $n > \beta m log m$ time series in the training data, the probability of misclassifying time series S with label L using $\widehat{L}_\theta^T$ satisfies the bound

$$\mathbb{P}\left(\widehat{L}_\theta^T(S;\gamma) \neq L\right)$$

$$\leq \left(\frac{\theta m_+}{m} + \frac{m_-}{\theta m}\right)(2\Delta_{\max} \quad + 1)n \exp\left(-\left(\gamma - 4\sigma^2\gamma^2\right)G^T\right) + m^{-\beta+1}$$

Given error tolerance $\delta \in (0,1)$ and with choice $\gamma \in (1, \frac{1}{4\sigma^2})$, the upper bound is at most δ if $n > m log \frac{2m}{\delta}$ (i. e. $\beta = 1 + \frac{log\frac{2}{\delta}}{\log m}$), and

$$G^T(R_+, R_-, \Delta_{\max} \quad)$$

$$\geq \frac{\log\left(\frac{\theta m_+}{m} + \frac{m_-}{\theta m}\right) + \log(2\Delta_{\max} \quad + 1) + \log n + \log\frac{2}{\delta}}{\gamma - 4\sigma^2\gamma^2}$$

Theoretical Guarantee

Generalized weighted majority voting correctly classifies a new time series S with probability at least $1 - \delta$ if

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

$$G^T(R_+, R_-, \Delta_{\max}) = \Omega(\sigma^2 \left(\log \left(\frac{\theta m_+}{m} + \frac{m_-}{\theta m} \right) + \log(2\Delta_{\max} + 1) + \log \frac{m}{\delta} \right))$$

Thus, the gap between sets R_+ and R_- needs to grow logarithmic in the number of latent sources m in order for weighted majority voting to classify correctly with high probability.

Implementation

- Data Preprocessing
- Normalization
- Parameter Optimization
- Predictor
- Utility Functions

Data Preprocessing

- Data Collection
- Constructing Topic Rate Signal

Data Collection

- We obtain one month (Nov.2013) twitter data directly from Prof. Ramakrishnan at Virginia Tech.
- We collected 200 examples of topics that trended at least once and 200 examples of topics that never trended in Nov. 2013. The authors picked 500 instead.
- The type as well as the amount of data we have used is all publicly available via the Twitter API.

Topics

For Trending Topics:

- We filtered out the topics whose rank was never better than 5.
- We filtered out the topics that did not trend for long enough time (less than 30 mins).

For Non-Trending Topics:

- Random Selection

Signal Construction

The raw Twitter data is formatted in JSON and is around 600MB per day. Because of the I/O bound, it is very time consuming.

1. Apply distributed computing system and MapReduce programming model. (Coding by Python)

2. Since the twitter raw data is enriched(well organized), we inspect the format manually and consider each piece of twitter data as a string without robust JSON parser which is slow. (Coding by C)

Construct Topic Rate Signal $\rho(t)$

Method 1 Distributed System and MapReduce

- Map() uses robust JSON parser to read all JSON data and generate and sorted "hashtag: Time: Count"
- Reduce() generate the frequency of hashtag and time
- Apply Dictionary Data Structure in Python

Package: map.py, reduce.py, post_process.py, get_time_array.py

Construct Topic Rate Signal $\rho(t)$

Distributed System and MapReduce

- Advantage: Multiprocessing big data set to conquer I/O bound
- Disadvantage: Need More Resources

Construct Topic Rate Signal $\rho(t)$

Method 2 Consider Json as a String

- Map() parse Json file as a String and generate “hashtag: Time”.
- Reduce() will reduce the sorted mapped data by granulating the tweet times into < interval > (in minutes) sized bins between < start > and < end > and outputting the results into < hashtag > named files in < dir_name > directory where each line contains a bin count.

    1. Parallel processing: Job Queue and Struct Job

    2. Streaming in parse the stream without reading it all into memory, we only have to allocate enough memory to store the values of one hashtag

Package: map.c, reduce.c, timebin.c, timebin.h, queue.c, queue.h

{"interaction": {"author": {"username": "sofiaacevedot", "link": "http://twitter.com/sofiaacevedot", "id": 450077293, "name": "Sofia Acevedo", "avatar": "http://a0.twimg.com/profile_images/3348150786/98eba0b38cb3e60502b50f577b29d3b1_normal.jpeg"}, "created_at": "Tue, 23 Apr 2013 06:41:38 +0000", "content": "RT @SussyVargasM: #ElSueñoDeTodoAlumno —Good morning guys I'm your new English Teacher, my name is Niall Horan Gallagher, I hope you guys enjoy the class.", "source": "Twitter for iPad", "link": "http://twitter.com/sofiaacevedot/statuses/326586646977978368", "type": "twitter", "id": "1e2abe0d939bad00e074f904d55f8654", "schema": {"version": 3}}, "language": {"confidence": 99, "tag": "en"}, "twitter": {"retweeted": {"source": "web", "created_at": "Tue, 23 Apr 2013 04:59:01 +0000", "id": "326560820051992576", "user": {"lang": "en", "statuses_count": 6074, "name": "Its a BritishThings♥", "friends_count": 1870, "created_at": "Mon, 30 Apr 2012 22:11:59 +0000", "description": "If I had british accent, I'd never shut up.", "time_zone": "London", "profile_image_url": "http://a0.twimg.com/profile_images/3551175633/564b33dffc8b1f00273efdba03b09e58_normal.png", "followers_count": 719, "id_str": "567683289", "location": "Santa Cruz - Bolivia", "favourites_count": 2665, "listed_count": 1, "id": 567683289, "screen_name": "SussyVargasM"}}, "id": "326586646977978368", "retweet": {"count": 4, "lang": "en", "text": "#ElSueñoDeTodoAlumno —Good morning guys I'm your new English Teacher, my name is Niall Horan Gallagher, I hope you guys enjoy the class.", **"created_at": "Tue, 23 Apr 2013 06:41:38 +0000", "hashtags":** ["ElSueñoDeTodoAlumno"], "source": "Twitter for iPad", "user": {"lang": "es", "statuses_count": 716, "screen_name": "sofiaacevedot", "friends_count": 221, "created_at": "Thu, 29 Dec 2011 21:06:33 +0000", "profile_image_url": "http://a0.twimg.com/profile_images/3348150786/98eba0b38cb3e60502b50f577b29d3b1_normal.jpeg", "name": "Sofia Acevedo", "followers_count": 54, "id_str": "450077293", "favourites_count": 223, "id": 450077293, "description": " HI! I AM DIRECTIONER! :3 "}, "id": "326586646977978368"}}, "demographic": {"gender": "female"}, "embers_islive": 1, "salience": {"content": {"sentiment": 4}}, "date": "2013-04-23T06:41:38", "embers_stream": "mainv2", "embersId": "datasift:1e2abe0d939bad00e074f904d55f8654"}

Construct Topic Rate Signal $\rho(t)$

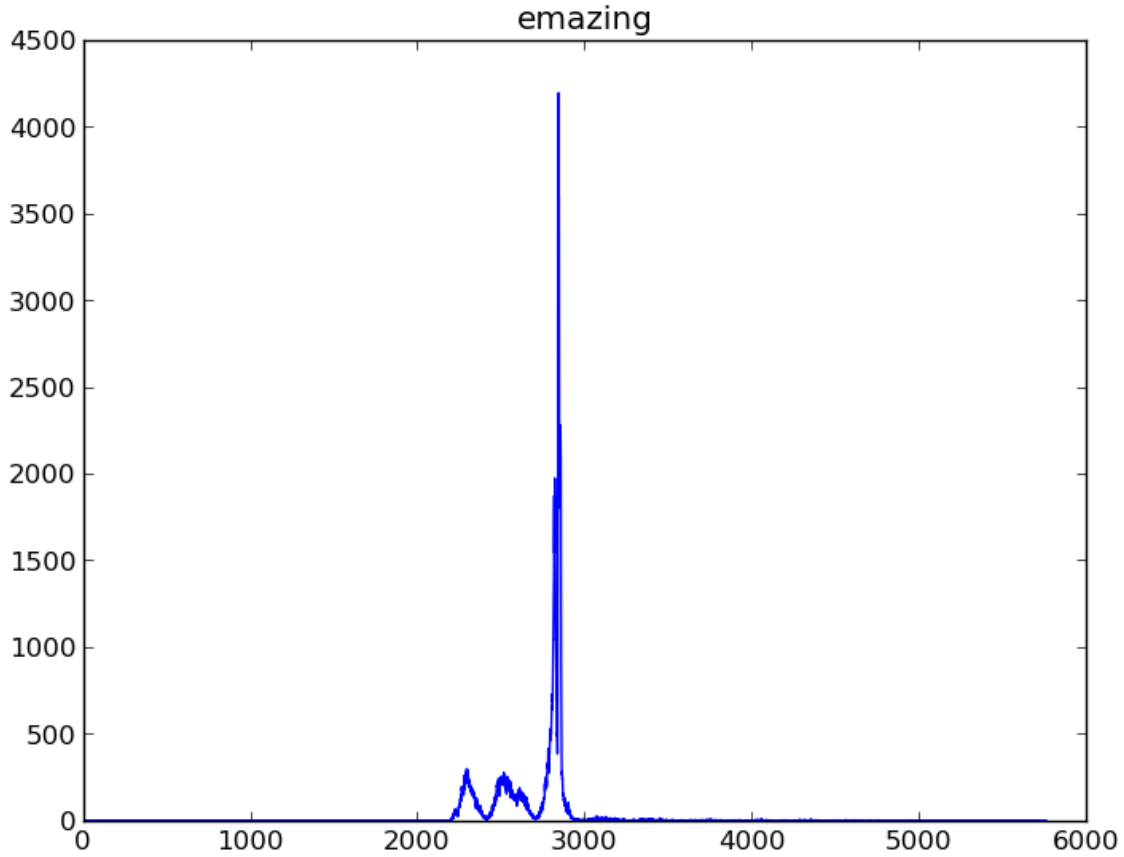
Pseudo Map-Reduce

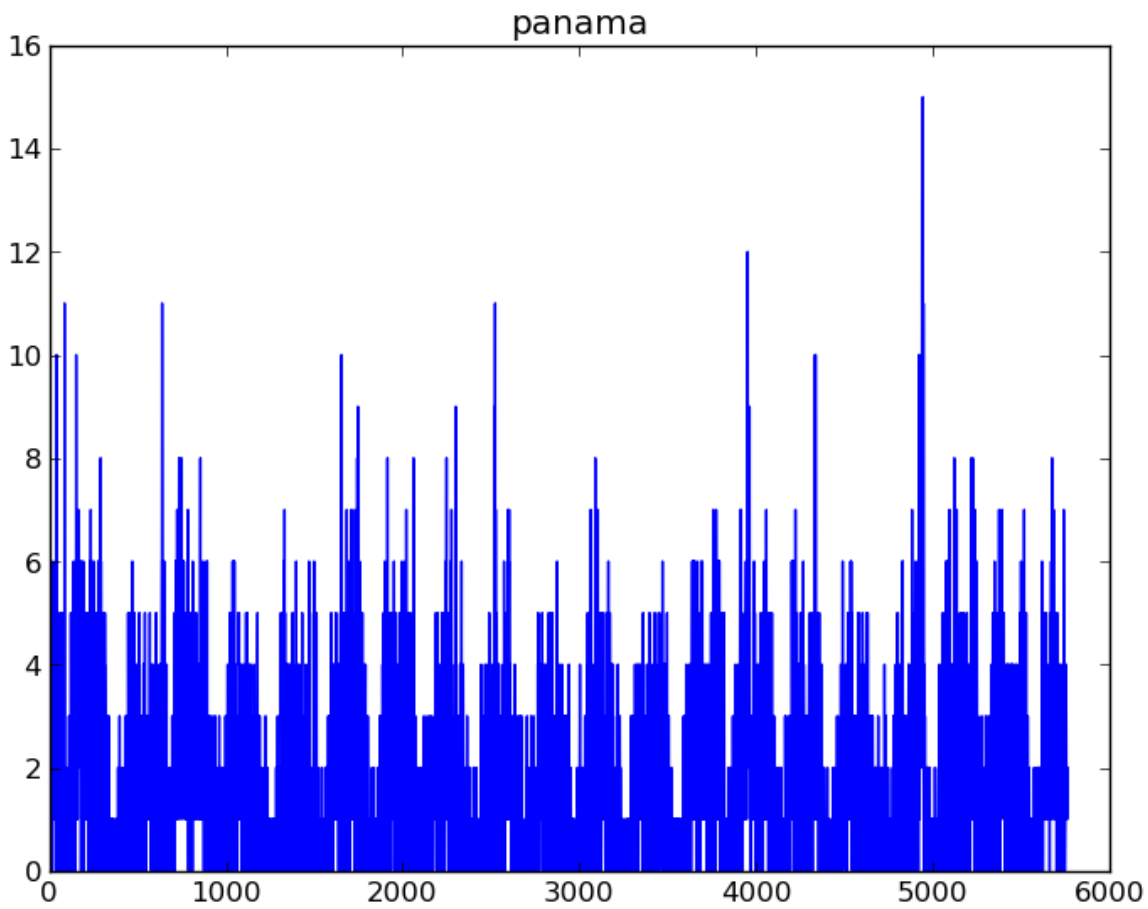
- Advantage: Use Limited Resource to Generate Signal very fast
- Disadvantage: Json file must be organized the same

Natural Language Filtering

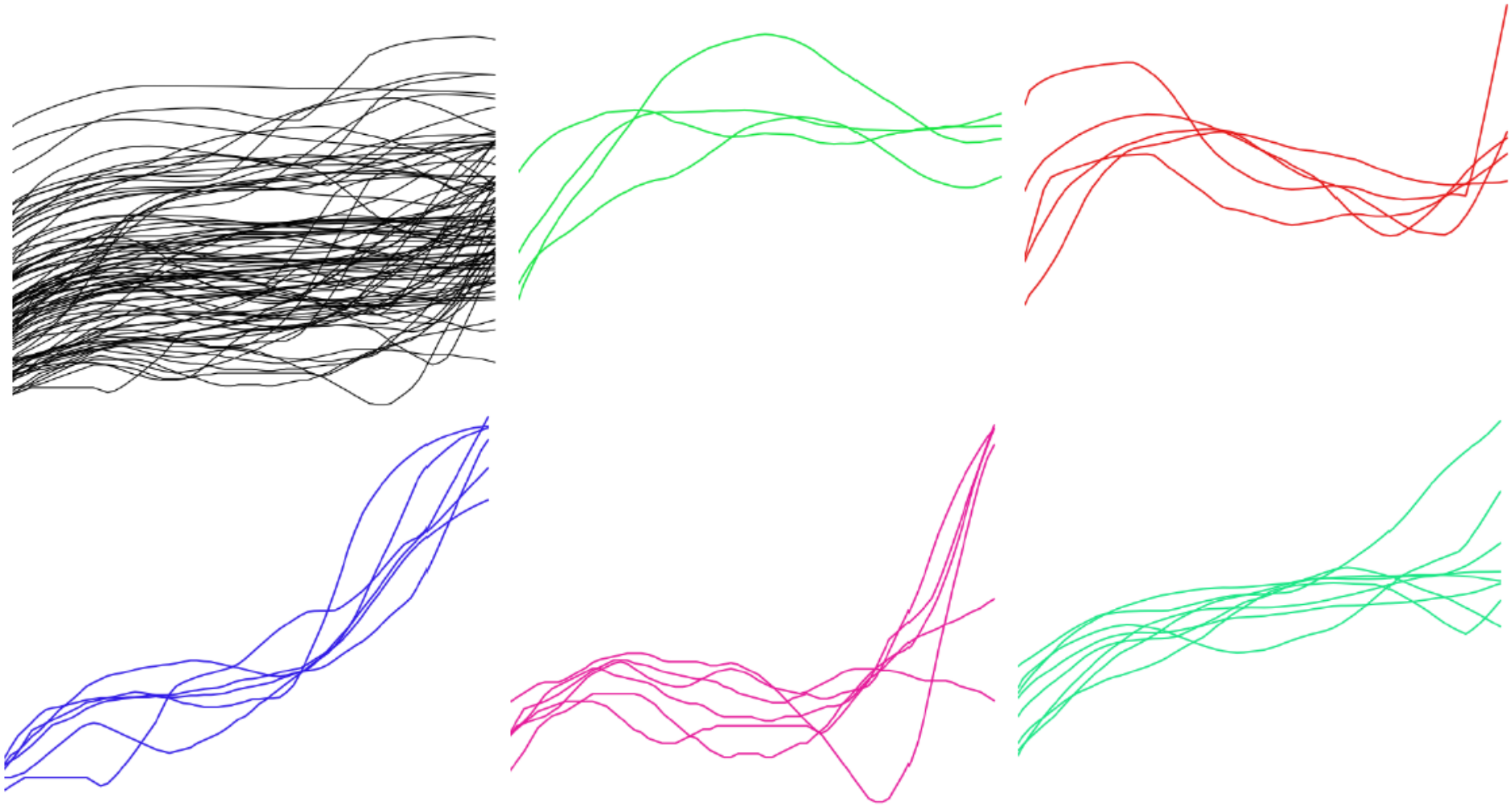
- Convert unicode encoded strings into ASCII equivalent
- The script use the unicode module to find ASCII equivalents of unicode characters.
- Convert Upper Case to Lower Cases

Construct Topic Rate Signal $\rho(t)$

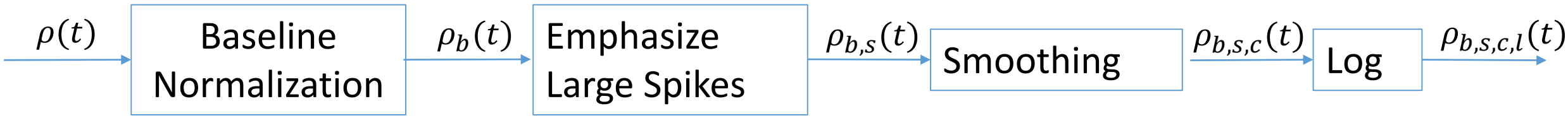




Construct Topic Rate Signal $\rho(t)$



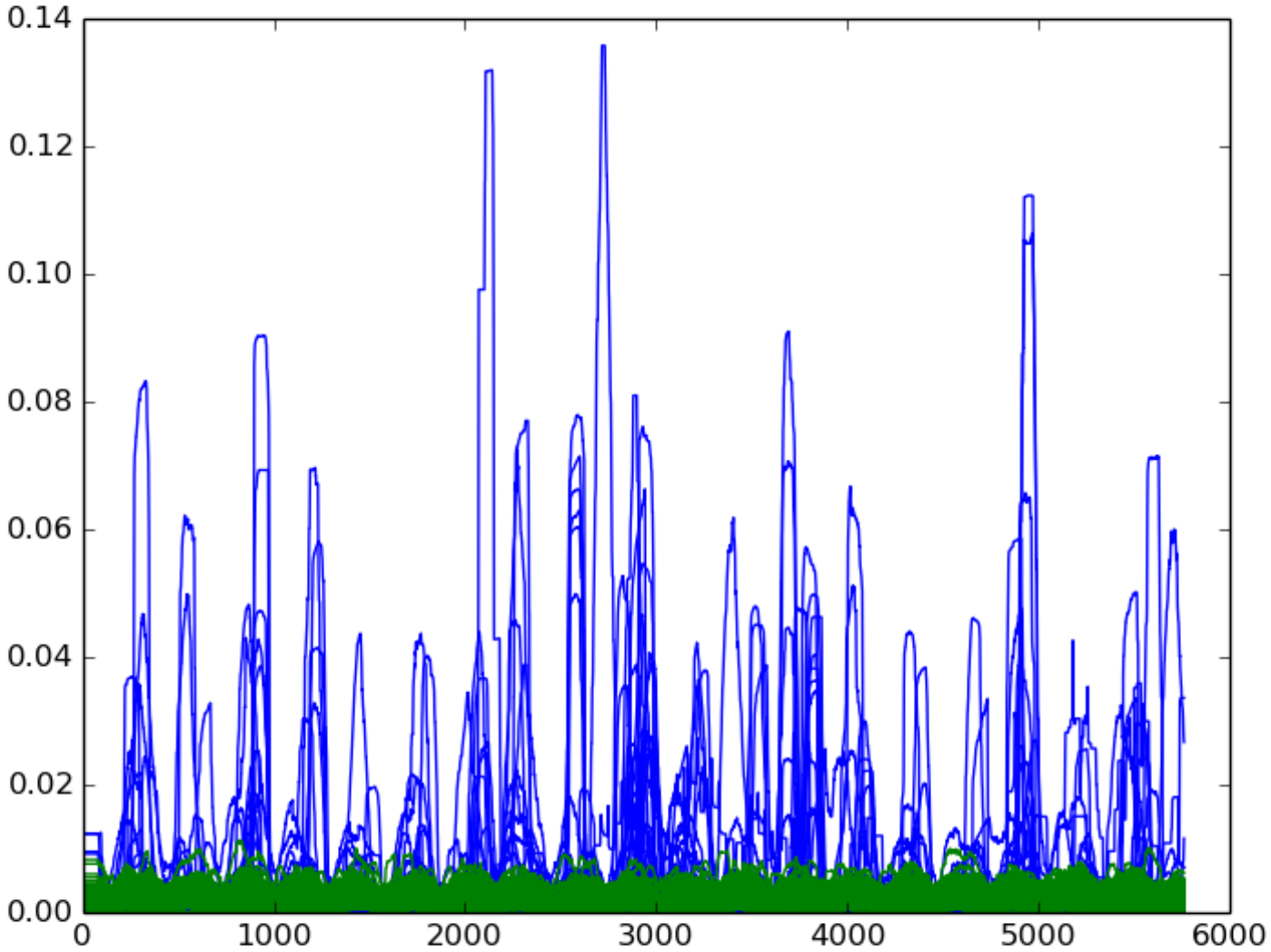
Normalization



Normalization

- Activity Signal for a topic: $\rho[n] \quad n = 1, \ldots \ldots, N_{obs}$
- Normalize away the baseline: $\rho_b[n] = (\rho[n]/b)^{\beta} \qquad b = \sum_n \rho[n]$
- Emphasize spikes: $\rho_{b,s}[n] = |\rho_b[n] - \rho_b[n-1]|^{\alpha}$
- Convolve the result: $\rho_{b,s,c}[n] = \sum_{m=n-N_{smooth}+1}^{m=n} \rho_{b,s}[m]$
- Measure the volume at logarithmic scale: $\rho_{b,s,c,l}[n] = log(1 + \rho_{b,s,c}[n])$

Normalization



Construct Reference Signal

- Trending Signal: Truncate at the trending time with length h_{ref}
- Non-trending Signal: Randomly select a piece of signal h_{ref}

Experiment and Empirical Parameter Optimization

Parameters:

- γ: 0.1, 0.5, 1.0, 5.0, 10.0
- N_{obs}: 10.0, 30.0, 50.0, 100.0, 120.0
- N_{smooth}: 20.0, 40.0, 80.0, 120.0, 160.0, 300.0
- h_{ref}: 4, 6, 10, 14, 17
- D_{ref}: 1, 2, 3, 4
- θ: 0.65, 1, 3

Experiment and Empirical Parameter Optimization

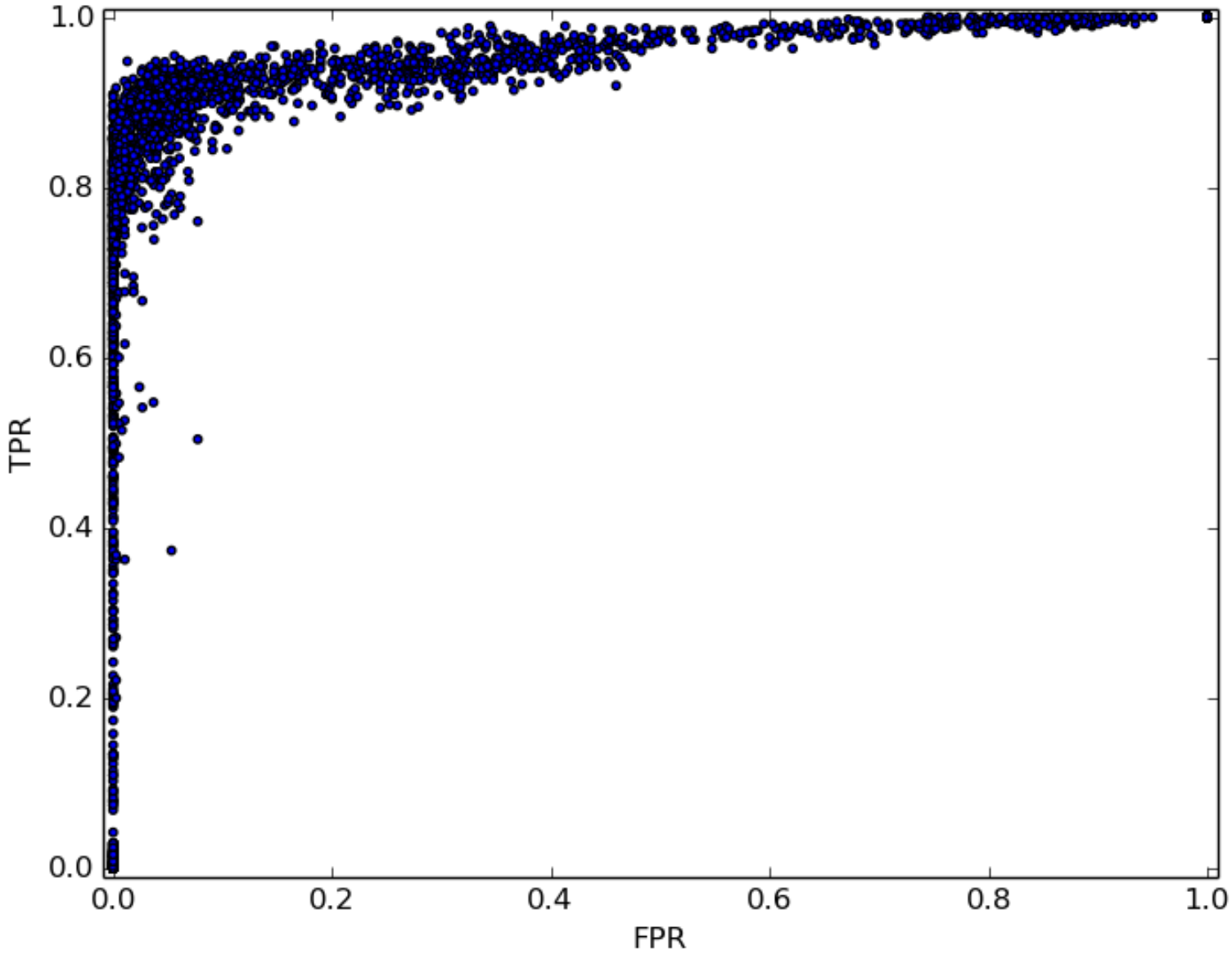
1. Run 5 random trials for each parameter sets. For each trial, we randomly 50/50 split to generate test and training signals.

2. The experiment will output a .txt file contains all the information about each parameter set. For each parameter set, we output:

True Positive Rate, False Positive Rate, Probability of early detection, Probability of late detection, Expected early detection time, Expected late detection time, Parameter set

3. Implemented by Multithreading. We chose to make each parameter set be multitreaded. Each thread gets a fair share of test signals to compute against ref signals.

Experiment and Empirical Parameter Optimization



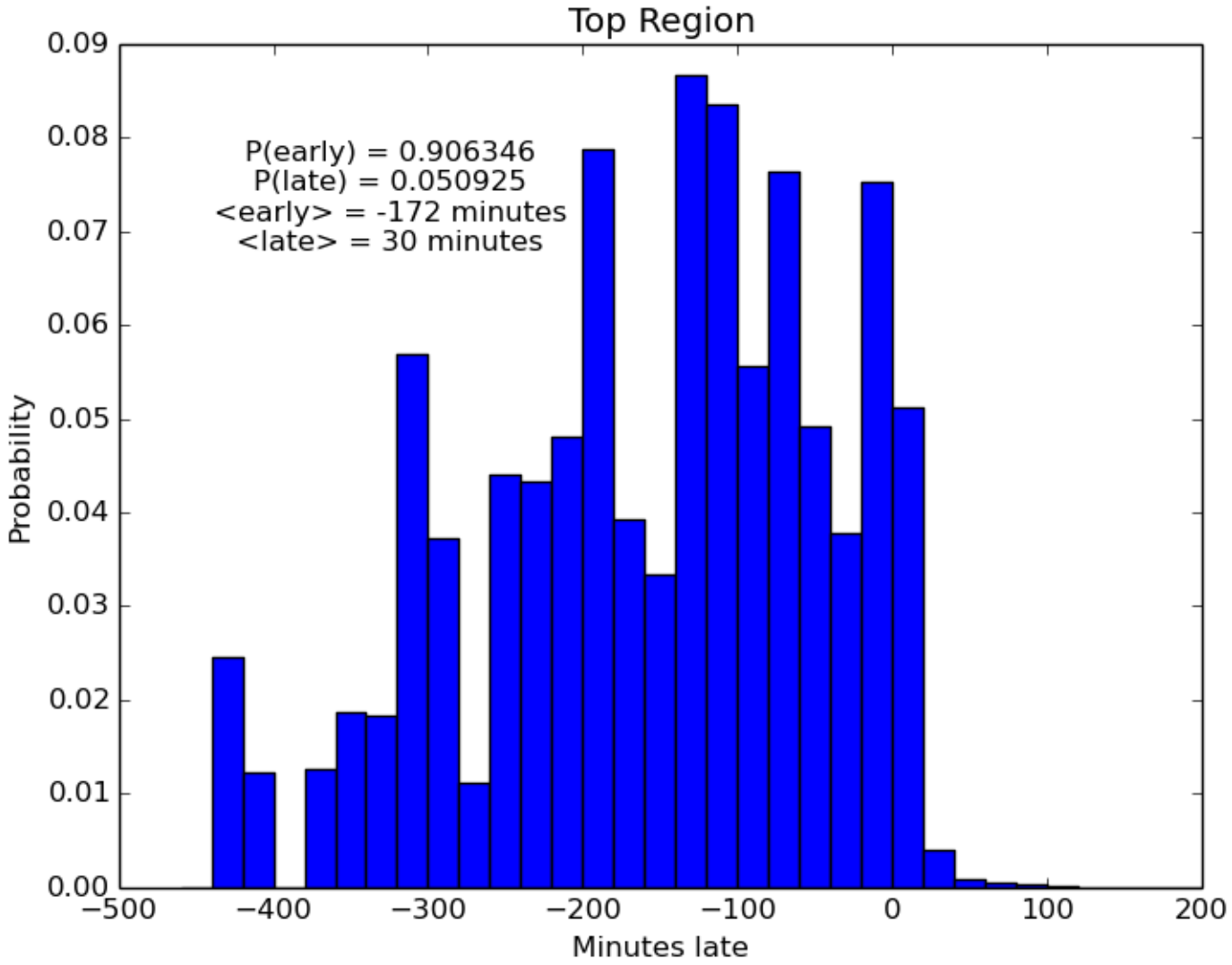
Experiment and Empirical Parameter Optimization

Definition (Top region). (FPR, TPR) is in the top region if FPR > 0.25 and TPR > 0.75.

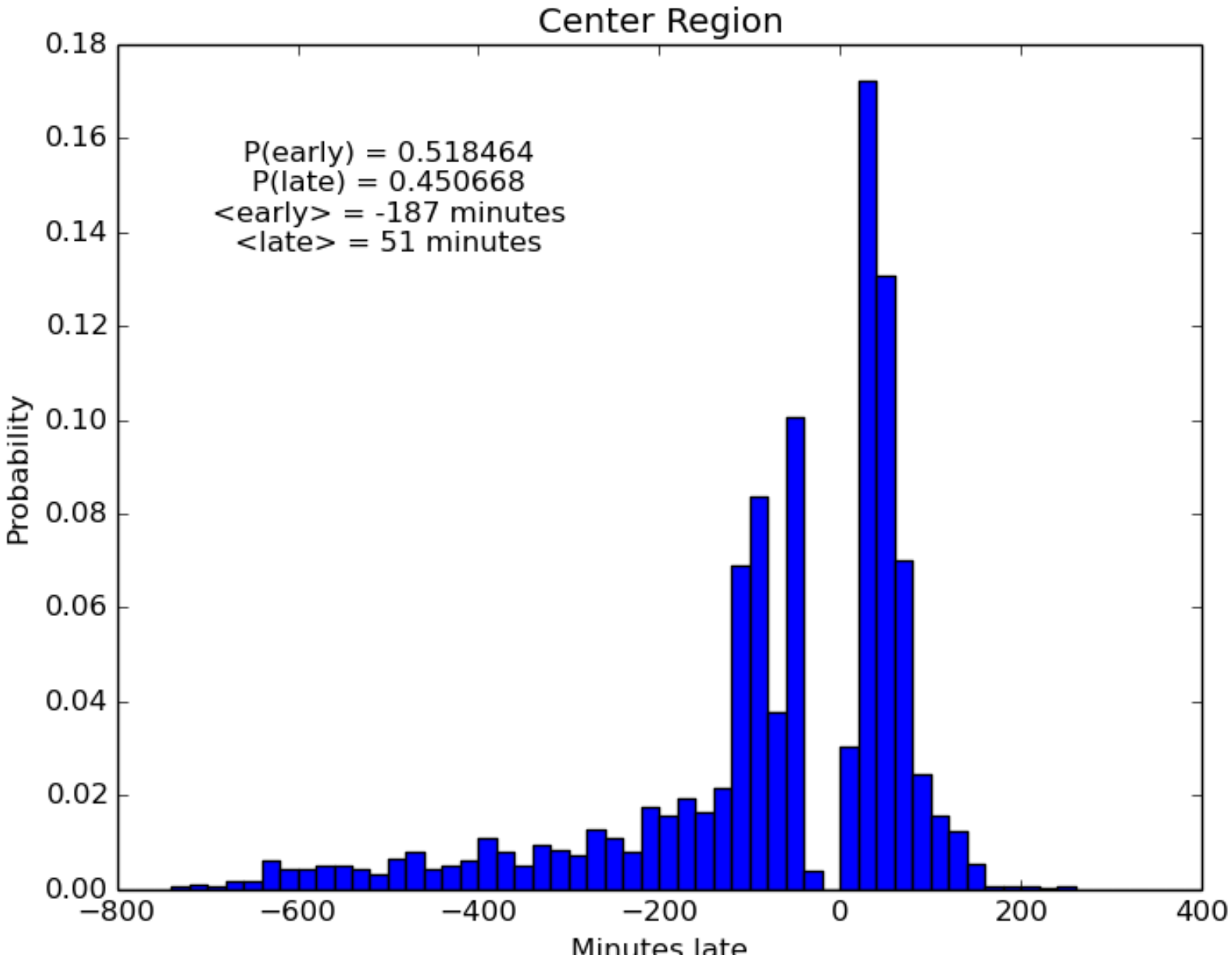
Definition (Center region). (FPR, TPR) is in the top region if FPR ≤ 0.25 and TPR > 0.75.

Definition (Bottom region). (FPR, TPR) is in the top region if FPR ≤ 0.25 and TPR ≤ 0.75.

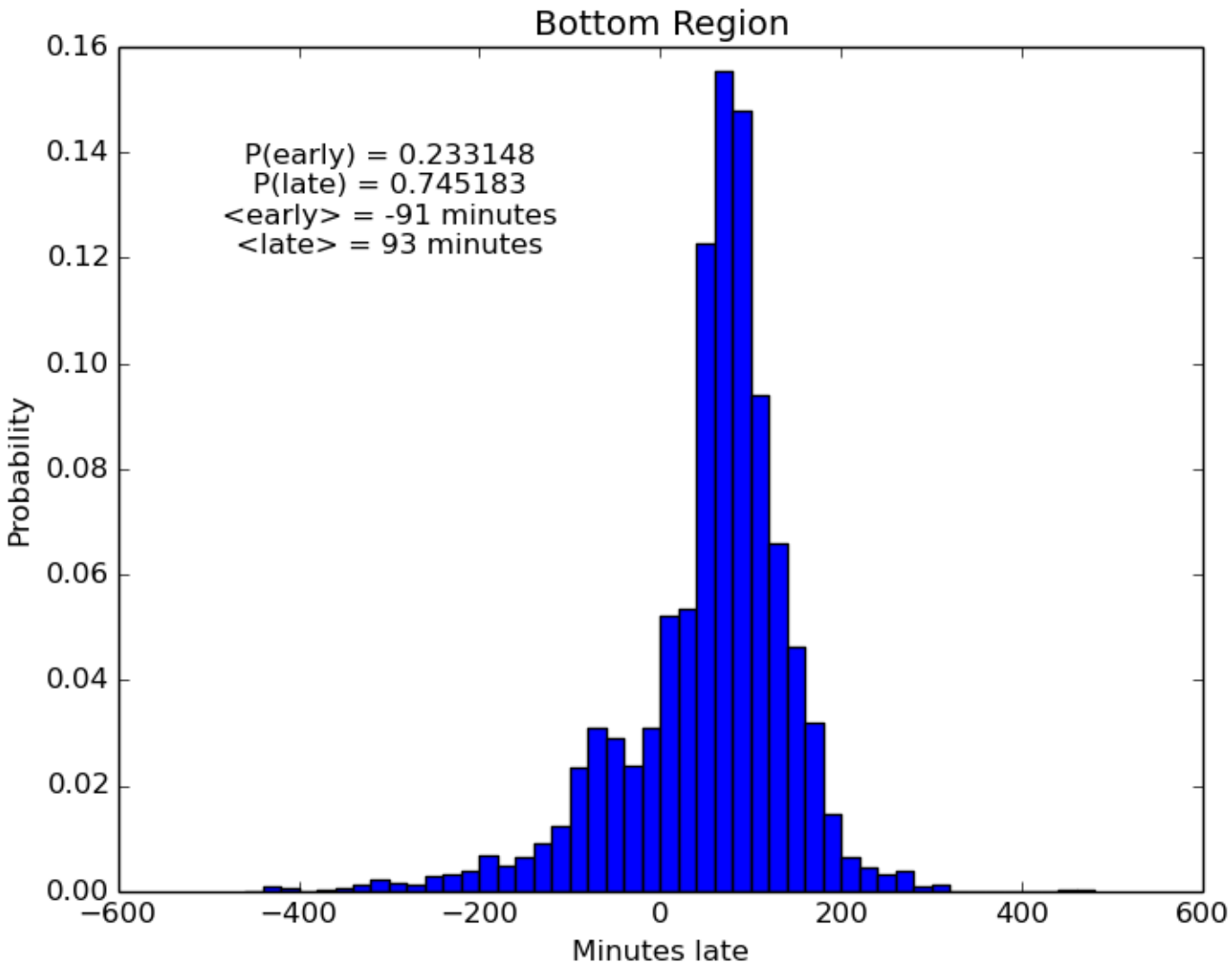
Experiment and Empirical Parameter Optimization



Experiment and Empirical Parameter Optimization



Experiment and Empirical Parameter Optimization



Effect on Position of ROC space

	$< M_{obs} >$	$< h_{ref} >$	$< \gamma >$	$< D_{req} >$	$< \theta >$	$< N_{somooth} >$
TOP	192.31	11.59	4.82	2.49	0.73	108
CENTER	220.73	10.45	3.38	2.52	1.00	108
BOTTOM	214.89	11.07	1.45	3.61	2.72	118

D_{req}, θ, γ appears to have significant influence on the position

Effect Along ROC curve

$$\Delta_{p,i}^{FPR} = \frac{FPR(p_i) - FPR(p_{i-1})}{p_i - p_{i-1}}$$

$$\Delta_{p,i}^{TPR} = \frac{TPR(p_i) - TPR(p_{i-1})}{p_i - p_{i-1}}$$

Effect Along ROC curve

Δ_{FPR}	M_{obs}	h_{ref}	γ	D_{req}	θ	$N_{somooth}$
TOP	-0.0013	0.0224	0.1096	-0.0436	-0.9400	0.0020
CENTER	-0.0045	0.0134	0.058	-0.0118	-0.0546	0.0001
BOTTOM	0.0002	0.0201	0.0002	-0.0021	-0.0002	0.0000

Δ_{TPR}	M_{obs}	h_{ref}	γ	D_{req}	θ	$N_{somooth}$
TOP	-0.0002	0.0017	0.0079	-0.0147	-0.19	0.0000
CENTER	0.0002	0.0001	0.0265	-0.0225	-0.0118	0.0002
BOTTOM	0.0011	0.0092	0.3558	-0.0531	-0.1050	0.0021

Parameter Recommendation

Cost(FP)<<Cost(TP)
We recommend the parameter settings in the third row of Table 1. For finer tuning, we can increase h_{ref} and D_{req}, decrease γ

Cost(FP)>>Cost(TP)
We recommend the parameter settings in the first row of Table 1. For finer tuning, we can decrease h_{ref} and γ

Cost(FP)≈Cost(TP)
We recommend the parameter in the second row of Table 1. For finer tuning, we can increase N_{obs}

Predictor : Two Versions

1. Given weight of type 1 and type 2 error, search the parameter file and find the minimum cost parameter set.

2. Normalize Reference signal and truncate reference signal.

3. Version 1 is for classification: Input the hashtag and predictor will output the decision, TP, FP, FN, TN, detection time

4. Version 2 is for prediction: divided the whole test signal into historical signal part and stream in signal part. Simulate the online stream in process.

5. I set another threshold to lower bound the number of tweets around the detection time.

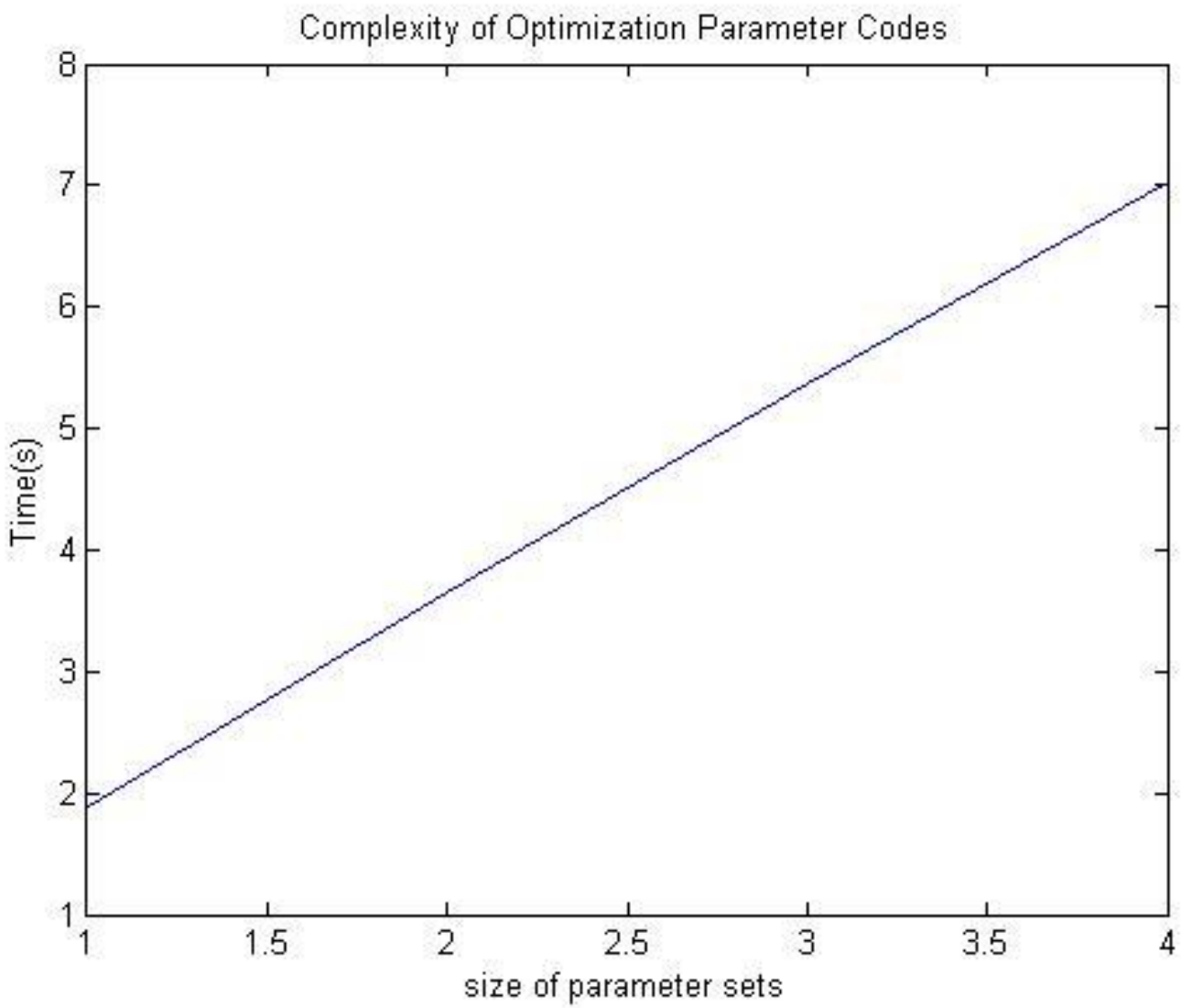
Utilities

arraylist.h

- X Macroing to easily create different arraylist types.
- All of these utility functions are #define which is basically forcing them as inline. Reduce function call overhead.
- Optimization Options

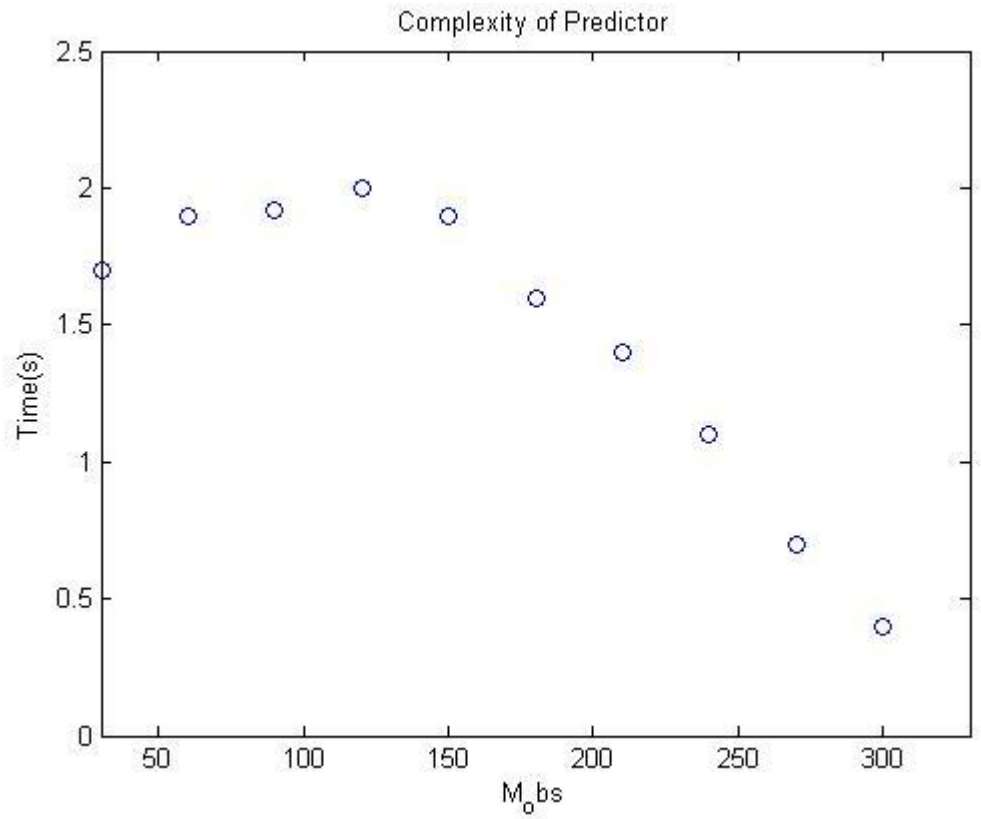
Complexity Analysis

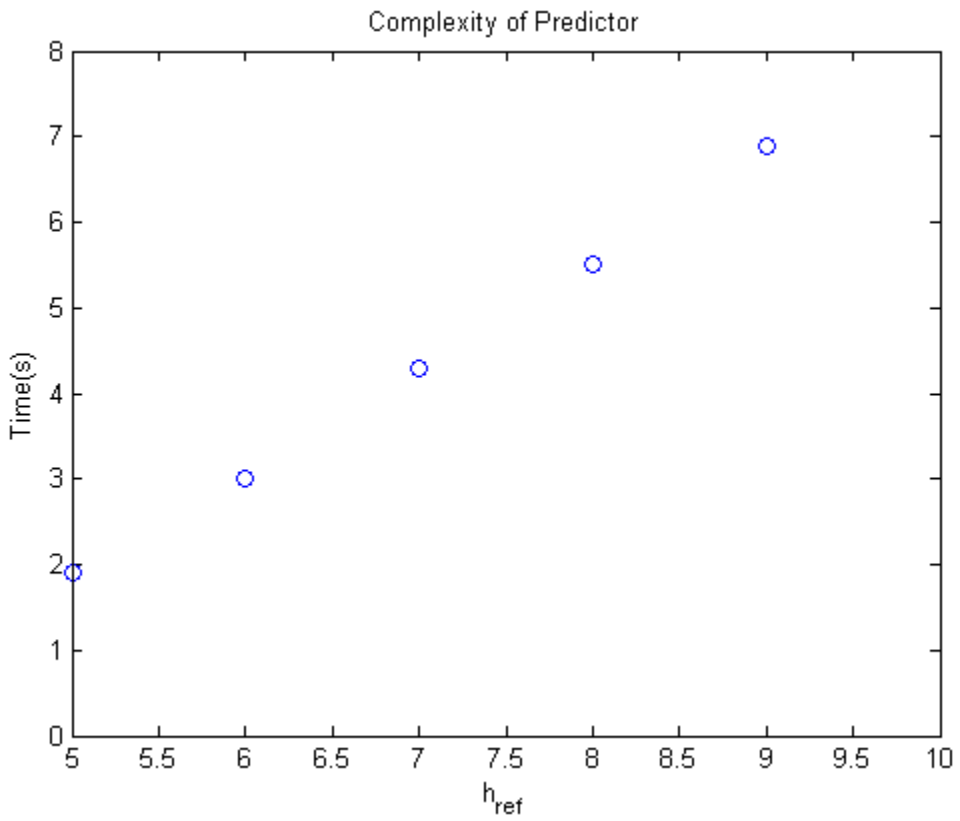
Parameter Optimization $O(size\ of\ parameter\ sets)$



Complexity Analysis

Predictor $O(M_{obs}(h_{ref} - M_{obs} + 1)|R_+ + R_-|)$





Validation

1. Preprocessing: Python codes Validation
2. Normalization: Python Codes Validation
3. Parameter Optimization: Python Codes Validation
4. Predictor: Python Codes Validation
5. Experimental Results Correspond to Author’s results

Further Work

- Predict Event such as Protest. Find a good way to estimate the value of the threshold I added in Predictor and to update the value over time.

- Computationally, we could use a more sophisticated algorithm instead of our core detection algorithm. For instance, Rakthanmanono have shown a way to efficiently search over trillions of time series subsequences.

- Our probability based metric involves exponential decay based on the distance between signals, most reference signals that are far away from the observation can safely be ignored. Thus, instead of computing the distance to all reference signals, which could become costly, we can operate on only a very small fraction of them.

Software Package Delivered

1. Preprocessing
   map.c reduce.c convert.py timebin.h timebin.c queue.h queue.c

2. Normalization
   norm.c norm_dir.c

3. Parameter Optimization
   param_optimize.c

4. Predictor
   predict.c

5. Utility
   arraylist.h makefile graph_multiple_dir.py graph_signal.py plot.rates.py
   graph_signals.py graph_signal_picking_set.py plot_regions_detection_times.py

Schedule

- 10/1 – 10/30 Learn Programming language: python and C
- 11/1 – 11/30 Write codes to classify data as different topics
- 1/1 – 1/30 Write normalization
- 1/1 – 1/30 Write Parameter Optimization
- 2/1 – 2/30 Write Predictor
- 3/1 – 3/30 Validation and Organize the Codes
- 4/1 – 4/30 Test
- 5/1 – 5/13 Write Documentation

Reference

[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

[8] Hui Ding, Goce Trajecvski *Querying and mining of time series data: experimental comparison of distance measure*, Proceedings of the VLDB Edowment, 2008

[9] Dan Feldman, Matthew Faulkner *Scalable training of mixture models via corests*, In advances in Neural Information Processing System 24, 2011

[10] Shiva Prasad, Huahua Wang *Online dictionary learning with application to novel document detection*, In Advances in Neutral Information Processing System 25, 2012

[11] Dnaiel Hsu and Sham Kakade *Learning Mixture of spherical gaussians:Moment methods and spectral decompositions*, 2013

[12] Michael Mathioudakis and Nick Koudas. *Trend Detection over the Twitter Stream*, In procedings of the 2010 ACM SIGMOD International Conference, 2010

[13] Alex Nanopoulos, Rob Alcock *Feature-based classification of time series data*, International Journal of Computer Research, 10, 2010

Thanks

Prof. Aravind Srinivasan

Prof. Naren Ramakrishnan

Dr. Rachel Ding

Dr. Amit Chavan