Segmenting, Summarizing and Predicting Data Sequences

Liangzhe Chen

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Application

March 28, 2018
Blacksburg, Virginia

Keywords: Sequence Mining, Segmentation, Topic Modeling

Segmenting, Summarizing and Predicting Data Sequences

Liangzhe Chen

ABSTRACT (academic)

Temporal data is ubiquitous nowadays and can be easily found in many applications. Consider the extensively studied social media website Twitter. All the information can be associated with time stamps, and thus form different types of data sequences: a sequence of feature values of users who retweet a message, a sequence of tweets from a user, or a sequence of the evolving friendship networks. Mining these sequences is an important task, which reveals patterns in the sequences, and it is a very challenging task as it usually requires different techniques for different sequences. The problem becomes even more complicated when the sequences are correlated.

In this dissertation, we study the following two types of data sequences. We show how to carefully exploit within-sequence and across-sequence correlations to develop more effective and scalable algorithms.

1. **Multi-dimensional value sequences:** We study sequences of multi-dimensional values, where each value is associated with a time stamp. Such sequences arise in many domains such as epidemiology (medical records), social media (keyword trends), and Critical Infrastructure (power outage data). Our goals are: for individual sequences, to find a segmentation of the sequence to capture where the pattern changes; for multiple correlated sequences, to use the correlations between sequences to further improve our segmentation; and to automatically find explanations of the segmentation results.

2. **Social media post sequences:** Driven by applications from popular social media websites such as Twitter and Weibo, we study the modeling of social media post sequences. Our goal is to understand how the posts (like tweets) are generated and how we can gain understanding of the users behind these posts. For individual social media post sequences, we study a prediction problem to find the users' latent state changes over the sequence. For dependent post sequences, we analyze the social influence among users, and how it affects users in generating posts and links.

Our models and algorithms lead to useful discoveries and solve real problems in Epidemiology, Social Media and Critical Infrastructure Systems. Further, most algorithms and frameworks we propose can be extended to solve sequence mining problems in other domains as well.

Segmenting, Summarizing and Predicting Data Sequences

Liangzhe Chen

ABSTRACT (general audience)

Temporal data is ubiquitous nowadays and can be easily found in many applications. Consider the extensively studied social media website Twitter. All the information can be associated with time stamps, and thus form different types of data sequences: a sequence of feature values of users who retweet a message, a sequence of tweets from a certain user, or a sequence of the evolving friendship networks. Mining these data sequences is an important task, which reveals patterns in the sequences, and helps downstream tasks like data compression and visualization. At the same time, it is a very challenging task as it usually requires different techniques for different sequences. The problem becomes even more complicated when the sequences are correlated.

In this dissertation, we first study value sequences, where objects in the sequence are multi-dimensional data values, and move to text sequences, where each object in the sequence is a text document (like a tweet). For each of these data sequences, we study them either as independent individual sequences, or as a group of dependent sequences. We then show how to carefully exploit different types of correlations behind the sequences to develop more effective and scalable algorithms.

Our models and algorithms lead to useful discoveries, and they solve real problems in Epidemiology, Social Media and Critical Infrastructure Systems. Further, most of the algorithms and frameworks we propose can be extended to solve sequence mining problems in other domains as well.

Acknowledgments

I would like to thank my advisor Prof. B. Aditya Prakash for all the guidance during this work; all the committee members (Prof. Naren Ramakrishnan, Prof. Edward A. Fox, Prof. Chang-Tien Lu and Prof. Yan Liu) for their time and valuable suggestions; and my collaborators and co-workers in Virginia Tech and Oak Ridge National Lab for all their help.

This dissertation is based on work partially supported by the National Science Foundation (IIS-1353346), the National Endowment for the Humanities (HG-229283-15), ORNL (Task Order 4000143330), the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the respective funding agencies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation involves the study of different types of data sequences, which arise in many domains and applications. We observe these sequences in sensor data, motion detection, climate monitoring, online social media and critical infrastructure, as well as in applications such as trend prediction [109, 40], event sequence analysis [82, 157], social influence extraction [77, 102], etc. Hence, analyzing these different types of data sequences, and further understanding their patterns in an automatic way, are crucial tasks that would benefit many areas.

In this dissertation, our focus is to exploit the within-sequence correlations and across-sequence correlations for a variety of segmentation and prediction problems, inspired mainly by their applications in three domains: Epidemiology, Social Media, and Critical Infrastructure Systems. Take multi-dimensional value sequences for an example. The medical records from hospitals form such sequences, where each data value contains the age, gender, and income of a patient infected with a particular disease. Can we segment the sequence to capture the dynamic of the disease's spreading pattern, identifying how and when such a pattern changes? Maybe the disease infects younger people first and then the elders, or maybe people with higher income get affected first? Can we identify these patterns automatically without user input? Answering these questions helps us understand how the disease spreads through the population, and thus improves our chance for a good intervention and immunization decision [27, 144]. For social media post sequences, we observe them extensively in popular social media websites such as Twitter, Weibo, Facebook, etc. The tweets generated by Twitter users naturally form sequences of short text, and they may show predictable temporal patterns. Can we model how each post is generated? Can we find patterns in users' posting behaviors and further predict them? Solving these problems helps us: understand the behavior of online social media users, reduce the processing and storage cost of the sequences, and benefit down-stream tasks like visualization, prediction, etc. [108, 170].

All of the above sequences can be correlated as well, which makes them even more difficult to analyze. Hence, for each type of sequence, we look at the sequences either as independent individual sequences or as multiple correlated sequences. Take social media post sequences for an example: as independent sequences, we consider and analyze the posts/tweets from different users separately, while as a set of correlated sequences, we consider the social influence between users, and model how that affects the generation of users' posts/tweets and connections to others. Similarly for multi-dimensional value sequences, as independent sequences, we treat each value sequence as independent, while as correlated sequences, we assume there is an underlying relation among sequences (which can potentially be represented by a network behind these sequences). We analyze these correlated sequences and their underlying relations together.

Shortcoming of existing methods. Most of the existing methods are inadequate to solve the above problems. For multi-dimensional value sequences, sequence mining methods, such as time series algorithms or event sequence algorithms, pose various restrictions to the format of the data sequences and hence have limited usage for our problems. For analyzing social media post sequences, existing Markovian or non-Markovian temporal topic models can find the state transitions within the document or the evolution of the topic, but they cannot find the author's state transitions behind the social media post sequences. Further, extracting the social influence among social media users is a difficult task that only gained popularity in recent years, and the limited related work focuses on different types of social influence compared to the problem we study.

Applications. We emphasize that all problems studied in this dissertation are motivated by real applications, and we use a wide range of real datasets for our analysis. These datasets can be categorized into the following types: 1) Epidemiology. Examples include Ebola disease reports released by the government, flu case counts from the Pan American Health Organization (PAHO), and simulated population networks released by the Network Dynamics and Simulation Science Laboratory (NDSSL) for national public health study. 2) Social Media. Examples include social networks like Twitter, movie review websites like Flixster, online shopping websites like Amazon, etc. Most of them provide APIs for data collection. In addition, there are multiple datasets available online that we can use. 3) Critical Infrastructure (CI). There exist many CI networks compiled by federal agencies or commercial vendors. We use the HSIP Gold data released by the US National Geospatial-Intelligence Agency (NGA) and the Department of Homeland Security (DHS).

1.1 Thesis Overview

In this section, we present our dissertation statement, and briefly summarize our main work.

1.1.1 Thesis Statement

Modeling/optimization approaches that take into consideration the within-sequence correlations (temporal and latent states relations) and across-sequence correlations (latent influence and spacial relations) can improve segmentation, summarization and prediction in data sequences from multiple domains, in comparison to approaches that ignore these correlations.

More specifically, for hospital medical/patient records in Epidemiology, and for failure records in Critical Infrastructure Systems, we develop automatic segmentation algorithms to capture the pattern change of the patients/disease and the number of outages respectively over time. Our segmentation algorithms work for value sequences with real or categorical, multi-dimensional values, and arbitrary time stamps (each data value can have arbitrary time stamps with no restriction). When the value sequences are correlated by an underlying network, we summarize and segment such networked sequences to detect pattern changes, and provide an algorithm to automatically give an explanation of the segmentation result for better interpretation and understanding. For social media post sequences, we propose two generative models to model the pattern changes in individual text sequences, and another generative model to capture the social influence among multiple correlated text sequences. Our models help in understanding users' online behavior, and they lead to good performance in trend prediction, link prediction, and retweet prediction.

1.1.2 Thesis Structure

We organize the dissertation into two parts: individual independent sequences, and groups of correlated sequences. The outline is shown in Table 1.1. We first give a survey of the related work in chapter 2, then we present our work from chapters 3 to 7, and finally we give our conclusions in chapter 8. In the following, we briefly introduce the research questions, contributions, and publications of our work.

1.2 Summary of Work

1.2.1 Part I: Individual Independent Sequences

In this section, for each type of sequence, we assume that the sequences are independent, and we study them separately without considering how they may affect each other. We start from value sequences, where we solve the segmentation problem to capture the change of the disease pattern in health reports. Then we move to text sequences, where we model tweet data for better public surveillance.

Table 1.1: Structure of the dissertation with references to the chapters.

	Pattern Shift Analysis	Occurrence Prediction
Independent Sequences	**Q1: How to segment independent data value sequences? (Chapter 3)**	**Q2: How to predict the latent health states of a sequence of social media posts? (Chapter 4)**
Multiple Sequences	**Q3: How to segment sequences of correlated values?(Chapter 5)** **Q4: How to explain the segmentation result? (Chapter 6)**	**Q5: How to predict the joint activity of multiple sequences of social media posts? (Chapter 7)**

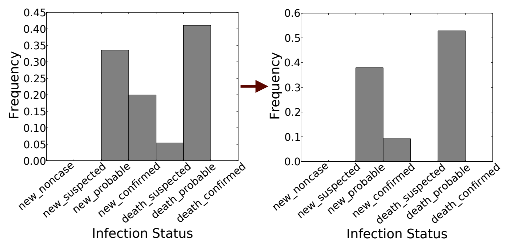
Chapter 3 (Segmenting Multi-Dimensional Value Sequences) [32] The main question we answer in this chapter is: given a multi-dimensional value sequence, can we find a time segmentation of the sequence, such that the patterns in adjacent time segments are different? Most existing methods, such as time series algorithms [109, 98, 145] and event sequence algorithms [82, 157], only work for specific types of value sequences, and hence they cannot be trivially applied to our problems, where each data value is a multidimensional, real/categorical value, and it has arbitrary time stamps (certain time periods may have more data values than others). We present an algorithm DASSA (DAta Sequence Segmentation Automatically), which does not make any prior assumption on either the data type or data distribution, and it automatically finds the number and identity of cut points efficiently. Through extensive experiments on a variety of real data, we show that DASSA outperforms all the state-of-the-art competitors, and it detects meaningful, intuitive patterns in real disease infection sequences. As shown in Figure 1.1(a), DASSA automatically finds segments where the flu infection pattern changes from elder people with higher income to younger people with lower income, and in Figure 1.1(b), the detected segments capture the increasing caution and prevention of the Ebola disease (decreasing number of deaths from the first segment to the second).

Chapter 4 (Modeling, Predicting Pattern Changes behind Tweet Sequences) [33, 34] In this chapter, we look at the data from popular online social media (Twitter). We propose generative models for Twitter users that capture latent health state transitions behind their tweet sequences for better flu trend prediction. The existing temporal/dynamic topic models either only capture transition of topics within a document/message [59, 11, 21], or capture the change of the topics themselves [23, 68]. Hence they cannot be applied for modeling the evolution of tweet sequences. In contrast, we propose temporal topic models HFSTM and HFSTM-A, that integrate a Hidden Markov Model to infer the latent state transition in the tweet sequence. Via our experiments on multiple flu-related tweet

	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:1	4.0	4.0	4.0	10.0	0.0	3.0	5.0
	4.0	3.0	4.0	10.0	0.0	3.0	2.0
	4.0	4.0	2.0	10.0	2.0	5.0	5.0
	4.0	3.0	4.0	10.0	2.0	3.0	2.0

	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:2	1.0	5.0	7.0	6.0	5.0	1.0	2.0
	4.0	5.0	7.0	3.0	0.0	1.0	1.0
	4.0	6.0	6.0	7.0	1.0	5.0	4.0
	2.0	5.0	5.0	7.0	0.0	3.0	2.0

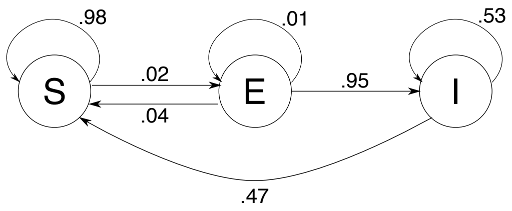
(a) Segmenting Portland simulated sequence



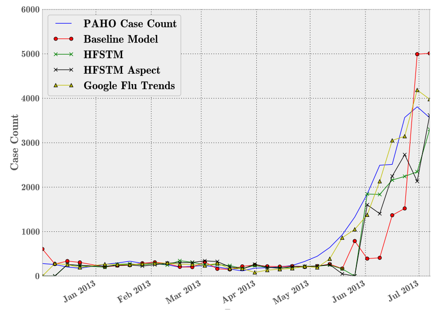
(b) Segmenting a sequence of Ebola-cases

Figure 1.1: Segmentation results from DASSA. (a) We show the most frequent feature values (discretized) in the two segments detected. The disease infects elder people with higher income, more vehicles first, and then it spreads to a younger population. (b) We show the distribution of infection status in the two segments detected. The number of newly confirmed cases dramatically drops, showing a sign of increasing caution and prevention of the disease.

datasets, we demonstrate that our model can capture meaningful state transitions in the tweet sequences, and the state transition leads to better flu trend prediction (Figure 1.2).



(a) State transition learned



(b) Predicting the flu trends

Figure 1.2: Results for a tweet dataset collected in South America from 2012 to 2014. (a) The state transition diagram with transition probabilities learned by HFSTM. S, E, I represent three different health states 'susceptible', 'exposed', and 'infected', respectively. (b) Using the models learned from HFSTM and HFSTM-A to predict the flu trend.

1.2.2 Part II: Multiple Dependent Sequences

This part of the dissertation is devoted to algorithms that adapt to and utilize the correlations between sequences. For example, in critical infrastructure systems, the number of people losing power in different counties can be correlated due to the interdependency among the underlying infrastructures. In social media, whether a user will adopt a meme/product may depend on his/her friends' behavior. Further, the tweet content published by users are also affected by the social influence from his/her friends. Hence, considering these correlations and interdependencies is important for analyzing the real dynamic in the sequences. We study how these correlations/interdependencies can be modeled and detected, and how we can utilize them in our analysis.

Chapter 5 (Segmenting Network Sequences with Node Labels) [**8**] In this chapter, we study sequences of values where the sequences themselves are connected by an underlying network. Such a sequence can be equivalently thought of as a sequence of networks with node labels. The main question we answer here is: given such a sequence of networks where each node has a binary label, and the node label may change over time, can we find a time segmentation, such that adjacent networks have different characteristics of nodes with the same label? Such a question naturally arises in social/contact networks where influence/opinion propagates. Existing dynamic graph algorithms [52, 153, 4, 174] detect abrupt changes based on the changes in communities/partitions, while we propose a more general algorithm SnapNETS, which does not require nodes to form communities/partitions. Further, SnapNETS is more comprehensive, because it considers nodes with all different labels to detect pattern changes (rather than only focusing on nodes with a specific label). Our experiments on several diverse real datasets show that SnapNETS finds cut points matching ground-truth or meaningful external signals, outperforming non-trivial baselines.

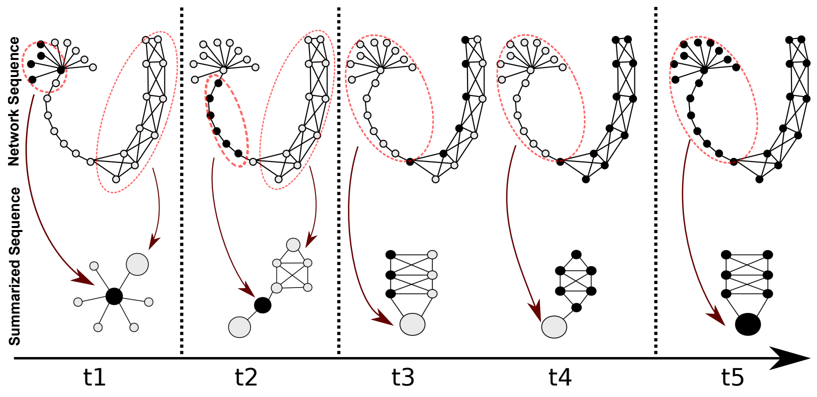
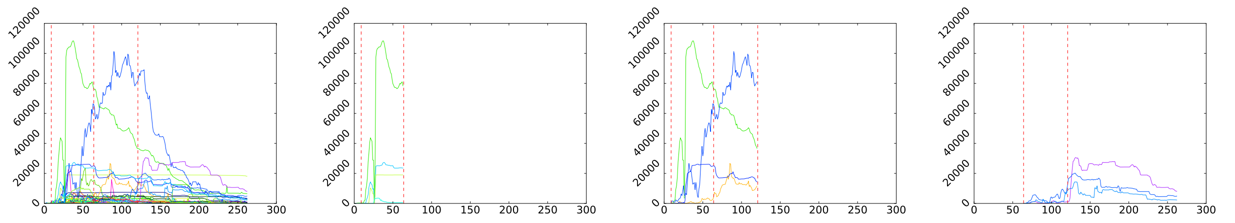


Figure 1.3: Results for a toy example. SnapNETS first finds summaries of networks to highlight the pattern (the summary networks are shown in the bottom line), and then detects the segmentation that captures the dramatic pattern change in the sequence (black dash lines represent the time cut points detected by SnapNETS).

Chapter 6 (Segmentation with Explanation for Multi-Dimensional Time Series) [**36**] In this chapter, we study the segmentation problem on the correlated outage data from recent hurricanes. Each sequence in the data captures the number of power outages for a county during the hurricane event. Further, we also design an optimization problem to automatically find an appropriate explanation for the segmentation results. Such a framework identifies the change point in the sequence and points out the culprit for the changes. We show in Fig. 1.4 an example for the county-level outages data for Hurricane Harvey. Fig. 1.4(a) shows the original data and the detected segmentation, and Fig. 1.4(b)(c)(d) shows the important time series detected that best explains the corresponding cut point. We see that our explanation highlights the time series with major pattern changes across the cut point.



(a) Harvey-segmentation (b) Explanation for cut0 (c) Explanation for cut1 (d) Explanation for cut2

Figure 1.4: Harvey segmentation and explanation results

Chapter 7 (Modeling and Predicting Tweets and Links) [**35**] Instead of considering each tweet sequence as independent, we now study correlated tweet sequences from different users, where one user's tweets may be affected by another through social influence. In this chapter, we study the problem of jointly modeling all three aspects in social media: user connections, their textual posts, and the social influence among them. We propose a generative model, that explicitly models how each post and link is generated by the user, either from self interest or social influence. Such a model helps us better model the role of social influence in affecting users' behavior in social media, and further helps us understand how information diffuses on such social networks. In Figure 1.5(a)(b), we show one of the communities we learned from Twitter data collected in 2009. It shows a clear interest focus on technology, and the celebrities with high influence (also identified by our model) are consistent with this interest. Further, we find interesting influence patterns in Figure 1.5(c): communities which are frequently influenced by the members inside the same communities (dark color in the diagonal), and the existence of an influential community that affects many of the other communities (a column of dark color).

1.3 Contributions

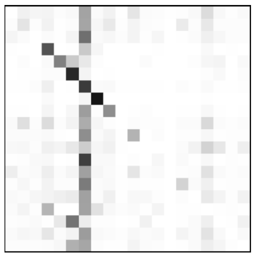
In this dissertation, we study a variety of segmentation and prediction problems in multiple domains. We show that our algorithms and models achieve outstanding performance in these problems by making use of various within-sequence and across-sequence correlations. Our work has been adopted in real applications. The URBANNET toolkit that we developed for monitoring, simulating, and analyzing Critical Infrastructure Systems is being used and licensed by Oak Ridge National Laboratory experts. We are also planning on merging

iphon
appl
app
googl
twitter
phone
window
video
mac
microsoft
mobil
ipod
use
game
store
facebook
download
blackberri
updat
youtub
web
play
file
post
softwar
instal
blog
launch
android
search
itun
code
touch
sale
free
san
pc
comput
nokia
at t
user
version
upgrad
guitar
releas
avail
develop
firefox
camera
browser
upload
new
devic
xbox
palm
soni
data

(a) Word cloud for c_{10}

User Screenname	A_{10} Value
engadget	0.0662
journalismnews	0.0432
shanselman	0.0350
hatebu	0.0343
twfeed	0.0326
pvponline	0.0317
crackberry	0.0237
theiphoneblog	0.0214
whiteafrican	0.0160
watch_akiba	0.0156
shauninman	0.0142

(b) Celebrities in c_{10}



(c) Influence matrix

Figure 1.5: Results for a tweet dataset collected for six months in 2009. (a) The word cloud for community 10 (c_{10}), showing an interest in technology ('iphone', 'app', etc.) (b) The corresponding celebrities in c_{10}. (c) The learned community-to-community influence strength: darker the color, higher the strength.

the CUT-n-REVEAL framework for analyzing power outages with the current URBANNET toolkit. Our HFSTM and HFSTM-A models for flu case surveillance also contribute to the EMBERS [137] project in the Discovery Analytics Center at Virginia Tech, which aims to forecast significant events from open source surrogates. We summarize our contributions in each chapter in the following.

- **Chapter 3: Novel, Efficient Multi-Dimensional Value Sequence Segmentation Algorithm.** We propose a novel algorithm DASSA for segmenting multi-dimensional value sequences, where the data value is multidimensional, real/categorical, and can have arbitrary time stamps. DASSA uses the temporal closeness between data values to achieve good segmentation performance. DASSA is efficient, and applicable to a much wider range of data sequences and applications than traditional time series algorithms or event sequence algorithms, which are designed only for certain types of data sequences.

- **Chapter 4: Novel Hidden State Modeling in Tweet Sequences for Better Surveillance.** We propose HFSTM and HFSTM-A to detect the latent state transition of text sequences by integrating the topic model with the Hidden Markov Model. When we apply our models on a flu-related tweet dataset, we show that the results detected by our algorithms help fill the gap between phenomenological methods for disease surveillance and epidemiological models, reconciling some contrasting behaviors between epidemiological and social models.

- **Chapter 5: Novel Efficient Segmentation Algorithm for Labeled Network Sequences.** We propose SnapNETS to introduce the characteristics of node labels into the segmentation problem. We consider nodes with all possible label values rather than focusing on only one value (as with most community-based dynamic graph algorithms).

Further, our algorithm works in a parameter-free manner that automatically detects the number of segments in the optimal segmentation.

- **Chapter 6: Novel Segmentation with Explanation Framework.** To the best of our knowledge, we are the first to design a framework (CUT-n-REVEAL) which not only gives the segmentation of the sequences, but also an explanation of the segmentation itself. Such explanations help us better understand the detected changes by pin-pointing the culprit sequences. It also facilitates quick utilization and deployment of the segmentation results in real applications.

- **Chapter 7: Novel Post and Link Level Social Influence Modeling:** To the best of our knowledge, we are the first to directly model how social influence affects the generation of each post and link. Our proposed algorithm PoLIM models the latent social influence via the community structure, and it can estimate whether a post or a link is generated by influence or not. Such fine-grained modeling introduces big challenges to inferring the model parameters; at the same time, it leads to more personalized and more precise prediction for both posts and links.

1.4 Publications

Related publications include:

1. **Liangzhe Chen**, K. S. M. Tozammel Hossain, Patrick Butler, Naren Ramakrishnan and B. Aditya Prakash. Flu Gone Viral: Modeling Flu on Twitter using Temporal Topic Models, *in the 14th IEEE International Conference on Data Mining (ICDM)*, 2014, Shenzhen, China

2. **Liangzhe Chen**, K. S. M. Tozammel Hossain, Patrick Butler, Naren Ramakrishnan and B. Aditya Prakash. Syndromic Surveillance of Flu on Twitter Using Weakly Supervised Temporal Topic Models, *in Data Mining and Knowledge Discovery Journal (DAMI), Volume 30, Number 3*, 2016

3. Sorour E. Amiri, **Liangzhe Chen** and B. Aditya Prakash. Automatic Segmentation of Network Sequences with Node Labels, *in the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2017, San Francisco.

4. **Liangzhe Chen**, Sorour E. Amiri and B. Aditya Prakash. Automatic Segmentation of Data Sequences, *in the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018, New Orleans

5. **Liangzhe Chen** and B. Aditya Prakash. Modeling Influence using Weak Supervision: A joint Link and Post-level Analysis, *in VT Computer Science Technical Reports, TR-18-02*, http://hdl.handle.net/10919/82747, 2018

6. **Liangzhe Chen**, Nikhil Muralidhar, Supriya Chinthavali, Naren Ramakrishnan and B. Aditya Prakash. Where and How: Segmentations with Explanations, *in VT Computer Science Techical Reports, TR-18-03*, http://hdl.handle.net/10919/82748, 2018

Chapter 2

Survey

In this chapter, we survey the related work in multiple areas, including sequence mining, topic modeling, critical infrastructure, and graph analysis.

2.1 Sequence Mining

The segmentation problem we study in this dissertation (mainly in chapters 3 and 6) is closely related to the fields of time series analysis and event sequence analysis. Although the algorithms in these field cannot be used to solve our problems directly, we introduce the major work in these categories below.

2.1.1 Time Series Analysis

There has been much work on time series, such as modeling co-evolving time series using multi-level HMMs [109], discovering patterns in data streams [162, 123], developing online algorithms for frequent sequence mining [113], time series segmentation [145, 98, 81, 43], change detection algorithms [117, 40, 165], temporal clustering [118, 180], etc. Batal et al. [16] convert time series into time-interval sequences of temporal abstractions. They find predictive patterns in complex multivariate time series data. Matsubara et al. [109] find the patterns in co-evolving time series. In [138], the authors propose a framework to reconstruct temporal models of cellular processes from time-course gene expression data. Li et al. [98] propose extended Kalman Filters to summarize and compress the multiple time sequences with missing values. Mueen et al. [113] develop an online algorithm to discover the frequent sequence in real time over the most recent history. Rakthanmanon et al. [136] show a way to search and mine truly massive time series efficiently. Their method can handle real-time monitoring of data streams with much faster arrival rates. Wang et al. [168] propose pHMM to reveal a global picture of the system that generates the time series data. Gensler et al. [54] present an approach to use update techniques for approximation to speed up some time series segmentation techniques. Same et al. [145] use Bayesian Information Criteria to build a threshold-free segmentation approach. Liu et al. [105] propose a segmentation criterion that improves computing efficiency, and propose two online piece-wise linear segmentation methods based on the criterion. Tseng et al. [163] combine a clustering technique, discrete wavelet transformation, and a genetic algorithm to automatically find segments in time series. Chiappa et al. [41] propose a Bayesian approach to a segmentation model based on the switching linear Gaussian state-space model, for the unsupervised time series segmentation task. Chen et al. [40] find a method for the contextual time series change (CTC) detection problem. All these methods, while very valuable, work on single or multiple time series, but we focus on more general data sequences with multi-dimensional values. In addition, the data points can have arbitrary time stamps, and certain time periods may have many more data points than others.

2.1.2 Event Sequence Mining

Many problems have been studied on event sequences. Related work includes finding summaries of event sequences [82], predicting future events [171], and segmenting element sequences [157]. Patnaik et al. [126] propose a streaming algorithm for mining frequent episodes over a window of recent events in the stream. Tatti et al. [158] employ a pattern set mining approach to consider a set of patterns as a whole, and find such pattern sets in event sequences. Wu et al. [173] incorporate the concept of utility into episode mining and address a new problem of mining high utility episodes from complex event sequences. Yang et al. [176] study the progression stages of a sequence of events/texts. Their datasets can be understood as one-dimensional categorical data sequences. In contrast, we study a more general case in DASSA, where the data can be multi-dimensional, and both real and categorical.

2.2 Social Media Analysis

As social media has gained in popularity in recent years, there has been much work on utilizing the social media data for various applications and purposes. In the following, we discuss social media's usage for epidemiology and for social influence analysis, which will prove to be related and useful in chapters 4 and 7.

2.2.1 Social Media for Epidemiology

In the epidemiological domain, various compartmental models (which explicitly model states of each user) are employed to study the characteristics of flu diffusion [64]. Some of the best-known examples of such models are SI [79], SIR [18], and SEIS [99], which are regularly used to model true flu case counts. Recently, several papers [110, 175] show that the social activity profiles do not exactly follow these models, and propose several other variants. Note that different epidemiological models are used for different diseases. We focus our work on flu since it is a very common disease.

In the social media domain, related research has observed many strides in the last decade. Extensive data generated by these social networking sites (SNS) are being used to predict and forecast various societal events [181], finding user interests [151], or finding trending topics [177]. In particular, the study of topic and word trends has become an important predictor of real world events and news. These trends are much easier and faster to get from social media than from traditional methods (e.g., reliable CDC case counts typically have lags of more than a month) [57]. For disease prediction and forecasting, especially for flu, various methods have been proposed for large-scale [56] and small-scale predictions [42]. Furthermore, there are prediction methods that are solely based on Twitter [94, 45, 93]. Sadilek et al. [144] and Brennan et al. [27] studied the impact of different kinds of interactions to personal health–they calculate several features and predict the infection cases by classifications. In contrast, we directly model the overall state transitions for all users. Lamb et al. [87] discriminate tweets that express awareness of the flu from those about actual infections, and train a classifier by which a user can tell if the author of a tweet is really infected. Aramaki et al. [12] also trained classifiers for similar purposes. While their work is single-tweet-based, ours takes the tweet history into account. A tweet completely non-flu related is possible to be labeled as infected by our method if the tweets before and after both show signs of infection. Achrekar et al. [1], Culotta et al. [45], and Lampos et al. [89] fit a flu trend by analyzing tweets via various methods including keyword analysis, and compare their flu trend fitting with CDC results. Lampos et al. [89] present an automated tool using keywords to track the prevalence of Influenza-like Illness (ILI). These methods are very coarse-grained—they do not provide understanding of how the health state of a user changes over time, while we link the change of tweet pattern with standard epidemiological models. The unpublished recent work by [97] builds a Markov network to capture the spatial and temporal relations between different locations. Their definition of states is based on the number of infections in a location (such as rising state, declining state), but states in our work (HFSTM and HFSTM-A) are *epidemiological* states and they are learned directly from the tweet corpus.

2.2.2 Social Influence Analysis

The influence maximization problem aims to identify global influencers that would maximize the spread of the information based on a Linear Threshold model or a Independent Cascade model [80]. Mehmood et al. [112] generalize the IC model to deal with groups of nodes instead of single node influence. There are also many interests in further identifying influencers based on topics. Weng et al. [172] propose TwitterRank to find topic-level influencers on Twitter. Tang et al. [155] model topic-level social influence on large networks. Pal et al. [122] propose a set of carefully designed features to characterize social media authors, and use probabilistic clustering and with-in cluster ranking to identify topical authorities. While these works either focus on global or topic-level influencers, we model social influence in a finer-grained level between all the users. The combination of our community-level influence, with-in community popularity, and other parameters in our model PoLIM, can be used to compute the influence probability between any two users.

2.3 Topic Model

In both chapters 4 and 7, we propose variations of the topic model for different applications. In this section, we give a brief introduction of topic models and their previous usage for epidemiology and social influence analysis.

2.3.1 Topic Model for Epidemiology

The earliest topic modeling using LDA (Latent Dirichlet Allocation) [24] gained popularity for modeling different types of text documents (see [22] for review). Many variations of LDA have been proposed to model various problems. For modeling health related topics Paul et al. proposed the Ailment Topic Aspect Model (ATAM+) [127] to capture various ailments from a corpus of tweets. This model is based on a topic aspect model [128], author-topic model [152], and it does not consider the temporal information of the text messages (as we do in this dissertation). Another variant of LDA is temporal topic models which can be categorized into two groups: Markovian and non-Markov. [169] propose a non-Markov continuous time model for topic trends which can not be used to predict the user states. Gruber et al. propose a hidden topic Markov model (HTMM) [59], which assumes that all the words in a sentence have the same topic and there may be a topic transition between two consecutive sentences. In the paper [11], Andrews et al. propose a hidden Markov topic model (HMTM) that assumes that there is a topic transition between two consecutive words within a document. In the paper [21], Blasiak et al. use a hidden Markov model to capture topic transition within documents which are subsequently used to classify new messages. These methods only capture transition of topics within a document or a message, and they do not capture state transition of users *across* tweets. There are two other variants of LDA [23, 68] studying the evolution of topic distributions over time, while our model studies the transition between a set of topic distributions which does not evolve over time. Moreover, their models do not capture the topic changes between consecutive messages of a user. Another recent related work is by [176] which combines keyword distributions with a shortest path algorithm to find out a monotonically increasing stage progression of an event sequence. In our problem (HFSTM and HFSTM-A), flu states are not monotonic, and have transition probabilities, which their method does not learn.

2.3.2 Topic Model for Social Influence

Most of the topic models mentioned in the previous section only model the generation of the text without considering the relation, or the social influence among the text authors. In contrast, Nallapati et al. [116], Zhu et al. [183] and Bi et al. [19] jointly model the word generation and the link formation in Twitter. Within this line of work, the most closely related works include COLD by Hu et al. [77], a generative model that considers text content, temporal information, link structure and community level influence; and a topic-level influence model for heterogeneous networks by Liu et al [102]. The former models social influence's affect on link generation, while the latter models social influence for text generation. To the best of our knowledge, our model PoLIM is the first that uses weak supervision to integrate all three aspects: text content, social links, and user influence, where the user influence controls *both* the text content and the social links.

2.4 Critical Infrastructure Systems

One of the major applications we study in this dissertation (chapter 6) is analyzing CIS. We introduce two important CIS problems in the following for a better understanding of the field.

2.4.1 Infrastructure Vulnerability Analysis.

A number of existing works attempt to model and analyze the vulnerability of critical infrastructure systems [53, 92, 156]. Most of them focus on a single network. For instance, landslide hazard can be analyzed based on real-time data such as SPOT images [92] or remote sensor data [156]. However, modeling and analyzing the vulnerability of interdependent CI networks are more challenging, because the various interdependencies of critical infrastructure networks are challenging. This is mainly due to the fact that interdependencies between the infrastructures involve multiple dimensions such as types of coupling and interdependencies [141, 130]. For example, there exists a physical interdependency between power substations and oil refineries for energy generation. Also, during a natural disaster, such as a hurricane, there exist geographical interdependencies across all the critical infrastructures. Previous works on vulnerability analysis and simulation of interdependencies between critical infrastructure systems can be categorized into: empirical approach, agent based, system dynamics based, economic theory based, and network based approaches (see [120] for a review). Some of the simulation methods might be slow and need to be accelerated in modern parallel platforms [71, 179, 167, 72, 76, 74]. In addition, modern parallel accelerators, such as GPUs, could be used to optimize the computation [70, 73, 69, 75]. A few mathematical frameworks [28, 78, 100] and interdependency models [124, 48, 61] have been proposed for vulnerability analysis. Most of these works focus on only two critical infrastructures at a time. For example, Parandehgheibi et al. [125] study the interdependency between a communication network and a power grid network, and analyze the vulnerability of them as the minimum number of nodes that should be removed to cause the failures of D nodes in the other network. Dueñas-Osorio et al. [49] study the fragilities and interdependency between power and water network. They use a naïve cascade model (random number based) and Monte Carlo simulation to study the fragility of the system under certain types of attacks. Albert et al. [7] study the power grid network and determine its ability to transfer power between generators and consumers (using the connectivity loss metric) when certain nodes are disrupted.

2.4.2 Influence Maximization and Cascade Analysis.

The influence maximization problem aims to find the best seed nodes which maximize influence. This has been extensively studied for the Independent Cascade and the Linear Threshold models [80], where they gave a (1-1/e) approximation algorithm based on submodularity. Much work has focused on designing more efficient algorithms for the original problem [25, 39], or extending to continuous time models [46], or under uncertainty [38]. All these algorithms assume that the influence cascades locally through edges. A very recent work Opera [31] finds critical nodes in a CI network that would maximally decrease the connectivity in target networks. However, they still use a 'local' failure model, which does not capture the dynamics of CIS (they optimize on a different objective function based on the number of triangles in the network).

2.5 Graph Analysis

As we construct a correlation graph for multiple sequences in chapter 5, various types of graph analysis algorithms can be used for our problems. In the following, we give a brief introduction of related algorithms and explain how our methods are different.

2.5.1 Graph Summary and Sparsification

Graph summary and sparsification aim to find compact representations of graphs which maintain desired properties. The properties can be defined based on specific user queries [51], action logs [107], the encoding cost [104], weights of nodes and edges [135], and the drop of the leading eigenvalue [132]. These algorithms help to reduce the processing cost of large graphs, and maintain (sometimes amplify) the patterns in the graphs. Unlike these methods, our summarization algorithm SnapNETS maintains the structural as well as label dependent properties of a graph.

2.5.2 Dynamic Graph Analysis

Dynamic graph analysis is important due to the evolutionary nature of many networks we see nowadays (see [2] for an overview). Many traditional machine learning tasks on static graphs have been extended to dynamic ones [63, 3, 146] such as clustering and classification, link prediction, anomaly detection, and trend mining. There has been work in finding time cut-points according to change of patterns in dynamic graphs. Ferlez et al. [52] and Sun et al. [153] use the Minimum Description Length principle to detect the cut points when communities/partitions in the evolving network change abruptly, while [13] uses tensor decomposition to discover temporal communities in dynamic graphs. These only work on plain graphs (not labeled graphs) and are community-based. In contrast, our proposed algorithm SnapNETS studies the patterns in a more general way not restricted to communities or clusters.

Part I

Individual Independent Sequences

Chapter 3

Segmenting Multi-Dimensional Value Sequences

egmenting temporal data sequences is an important problem which helps with data dynamics in multiple applications such as epidemic surveillance, motion capture sequences, etc. In this chapter, we give DASSA, the first self-guided and efficient algorithm to *automatically* find a segmentation that best detects the change of pattern in data sequences. To avoid introducing tuning parameters, we design DASSA to be a multi-level method which examines segments at *each level* of granularity via a compact data structure called the *segment-graph*. We build this data structure by carefully leveraging the information bottleneck method with the MDL principle to effectively represent each segment. Next, DASSA efficiently finds the optimal segmentation via a novel average-longest-path optimization on the segment-graph. Finally we show how the outputs from DASSA can be naturally interpreted to reveal meaningful patterns.

We ran DASSA on multiple real datasets of varying sizes and it is very effective in finding the time-cut points of the segmentations (in some cases recovering the cut points perfectly) as well as in finding the corresponding changing patterns.

3.1 Introduction

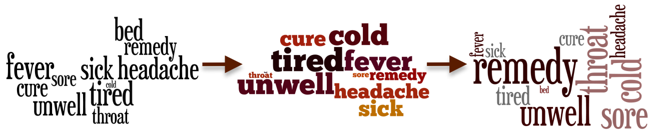
Given a data-sequence of Ebola infections, can we quickly tell when the characteristics of infected people change, possibly due to a mutation? In this chapter, we study the problem of *automatically* segmenting sequences of multi-dimensional data point (with categorical and/or real-valued features like age, gender, speed, etc.) so as to capture relevant trends and changes. The data observations can be unevenly distributed temporally and repeated multiple times in the sequence, naturally generalizing multi-variate time-series.

Such segmentations can be helpful in many real applications, as they may shed light on the underlying dynamics and patterns, thereby helping in modeling, anomaly detection, and also visualization. Consider epidemiological surveillance, where tracking disease propagation [159] can enhance the chance of a successful intervention and increase the situation awareness. For example, automatically finding changes in patient characteristics in a sequence of infected cases can help us point to changes in the disease itself. In Fig. 3.1(a), in a series of flu-infections, DASSA finds that the disease infects elder, richer people first, and then spreads to younger people with lower income. Similarly, figuring out the changes in how words are used together in user tweets (due to changes in users' health status) can help in estimating disease incidence [33]. See Fig. 3.1(b): in a series of flu-related tweets, we observe a transition between word usage in each segment from infection to recovery. Our motivation in this chapter is to design a general-purpose scalable segmentation algorithm for data sequences.

	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:1	4.0	4.0	4.0	10.0	0.0	3.0	5.0
	4.0	3.0	4.0	10.0	0.0	3.0	2.0
	4.0	4.0	2.0	10.0	2.0	5.0	5.0
	4.0	3.0	4.0	10.0	2.0	3.0	2.0

	age	Y	X	Income	Size	#Workers	#Vehicles
Segment:2	1.0	5.0	7.0	6.0	5.0	1.0	2.0
	4.0	5.0	7.0	3.0	0.0	1.0	1.0
	4.0	6.0	6.0	7.0	1.0	5.0	4.0
	2.0	5.0	5.0	7.0	0.0	3.0	2.0

(a) Segmenting a sequence of flu-cases



(b) Segmenting a sequence of words appearing in tweets

Figure 3.1: Our method DASSA gives meaningful cut-points: (a) Most frequent data values (discretized) in segments detected in a sequence of flu infections (*Portland*). (b) Word clouds for each detected segment in a Twitter keyword trend data (*Peru*). Size of a word is proportional to its frequency in the corresponding segment. More discussion in Sec. 3.5.

Informal Problem: Given a multi-dimensional data sequence, *automatically* find a *time segmentation* s.t. consecutive segments are not *similarly* informative.

Surprisingly, despite its importance, this general problem has not been studied widely. Such a problem cannot be trivially converted to one in time series, and has many challenges. The main properties we want a good solution to satisfy are:

- **P1** **(Generality):** No prior assumption on either data types or data distributions. The algorithm should work well regardless even if the data is skewed, or not forming clusters or not drawn from a known distribution.
- **P2** **(Self-Guided):** Automatically find the appropriate number and identity of cut-points without user input.
- **P3** **(Efficiency):** Finish within reasonable time for real datasets.

In this chapter, we present an algorithm DASSA (DAta Sequence Segmentation Automatically), which satisfies all of the three properties. To this end, we introduce three main ideas which may be useful for other segmentation problems as well: (a) looking at all possible segmentations efficiently using the so-called segment-graph; (b) compressing data in each segment based on temporal data distributions using Information Bottleneck and Minimum Description Length; and (c) using a novel path optimization to find the best segmentation, which automatically regularizes the number of segments and total segment difference. Via extensive experiments ranging from epidemiological to social, to motion capture datasets, we show how DASSA can recover high-quality segmentations, and meaningful patterns in practice. To the best of our knowledge, we are the *first* to present an efficient, self-guided method for the purpose of segmenting general data sequences.

3.2 Preliminaries

Information Bottleneck (IB): The IB method [161] compresses one signal X to the 'bottleneck' $\tilde{X}$ (much smaller than X in size) without much loss of its information related to another signal Y. The optimization problem it solves is:

$$\min I(\tilde{X};X) - \beta I(\tilde{X};Y) \tag{3.1}$$

where $I(\cdot;\cdot)$ represents the mutual information between two variables, β is the Lagrange multiplier. Given $|\tilde{X}|$ and $p(X,Y)$, this optimization problem can be solved iteratively [161] to provide these distributions: $p(\tilde{x}|x)$, $p(\tilde{x})$ and $p(y|\tilde{x})$; where x, $\tilde{x}$ and y are possible values of X, $\tilde{X}$, and Y.

Slonim et. al. [150] develop a variation of IB for word-document clustering, where they use words as the X signal, documents as the Y signal, and $\tilde{X}$ as the labels for words. In this formulation, they use hard-clustering (each x is mapped to exactly one $\tilde{x}$) to cluster words so that the information in the documents are maximally kept. They initialize the problem with no compression ($\tilde{X} = X$), and greedily choose the label pairs with smallest marginal loss $\delta I(\tilde{X},Y)$ (mutual information between $\tilde{X}$ and Y) to merge. The loss by merging $\tilde{x_i}$ and $\tilde{x_j}$ is defined as follows:

$$\delta I(\tilde{x_i},\tilde{x_j}) = (p(\tilde{x_i}) + p(\tilde{x_j})) * D_{JS}[p(y|\tilde{x_i}), p(y|\tilde{x_j})] \tag{3.2}$$

where D_{JS} is the Jensen-Shannon divergence.

Minimum Description Length (MDL): The MDL principle [60] suggests that the best hypothesis for a given set of data, which captures the most regularity in the data, is the one that leads to the best compression of the data [166, 33]. MDL finds the best model which minimizes

$$Cost_T = Cost(M) + Cost(X|M) \tag{3.3}$$

where $Cost(M)$ is the cost to describe the model, $Cost(X|M)$ is for describing the data using the model.

3.3 Overview

We present the main principles of DASSA (Alg. 1 shows the basic steps) next.

Algorithm 1 Pseudo-code for DASSA

```
Require: $\mathbb{D}$
Ensure: The best segmentation $S^*$
1: $[\tilde{X}, p(\tilde{x}|x)]$=Cluster ($\mathbb{D}$).//Finding data clusters using IB and MDL (Sec. 3.4.2)
2: Build a node for every possible time segment $y$.//Constructing $\mathcal{G}$ (Sec. 3.4.1)
3: Add node $s$ and $t$ to represent the start and end time of $\mathbb{D}$.
4: Create edges for adjacent $y$'s.
5: Calculate the edge weights as the Euclidean distance between the corresponding conditional
   cluster distribution $p(\tilde{x}|y)$.
6: $S^*$ = DAG-ALP ($\mathcal{G}$, $h$, $s$, $t$).//Finding the average longest path as $S^*$ (Sec. 3.4.3)
```

Definition 3.1 (Data sequence) *A data sequence $\mathbb{D}$ is a list of tuples $(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)$, where $(\mathbf{x}_i, t_i)$ is an observation of d-dimensional vector $\mathbf{x}_i$ at time t_i.*

Let $X = \bigcup_i \{\mathbf{x}_i\}$. W.L.O.G. we assume time stamps are sorted, i.e., $t_1 \leq t_2 \leq \ldots \leq t_N$. Note that both $\mathbf{x}_i$'s and t_i's are not necessarily unique. We can have observations with the same data value, and there can be multiple data observations having the same time stamp. This general definition of data sequences covers special cases like time series, where the number of data observations at each time is the same; and event sequences, where $x_i's$ are one-dimensional categorical values. We want to design an algorithm that automatically finds segmentations for such data sequences, and that satisfies all the desired properties (**P1**-**P3**).

Main Ideas: To avoid introducing parameters like the desired number of segments and to find the segmentation in an automatic manner (**P2**), our search space would inevitably be the set of all possible segmentations, which is exponential in size. Our first main idea is to use a graph data structure (called the *segment-graph* $\mathcal{G}$) to efficiently represent and search among all possible segmentations of the data sequence. See Fig 3.2(c) for an example $\mathcal{G}$. The node set of $\mathcal{G}$ mainly represents all possible time segments $Y = \{y_{i,j}\}$ ($y_{i,j}$ is the segment from time i to j), s and t represent the start and end time, and the edge weights are distances (i.e. the 'difference') between adjacent time segments. With this data structure, segmentations of the data sequence are now mapped to paths from start time s to end time t in $\mathcal{G}$. Hence, finding the best segmentation is now converted to the problem of finding the 'best' start-to-end path in the segment-graph $\mathcal{G}$. To solve the segmentation problem using $\mathcal{G}$, two important questions remain unsolved: **Q1**: how do we define the difference metric $w(\cdot)$ between two time segments, and **Q2**: what is the best start-to-end path in the segment-graph, and how do we find it efficiently?

Q1: Segment difference. Due to **P1**, we cannot use model-based methods (which typically assume certain data distributions like Gaussians in each segment or overall in $\mathbb{D}$) for our problem. Our second main idea is to cluster data values based on their 'temporal closeness', and then represent each segment using their conditional cluster distributions ($p(\tilde{x}|y)$, the probability of cluster $\tilde{x}$ given a segment y). We can then measure the segment difference simply as the difference between their $p(\tilde{x}|y)$'s. Intuitively, clusters based on how data values are temporally distributed *over all possible segments* Y naturally captures the 'similarity' between data values, which is well-suited for segmentation problems: if two data values always occur close in time at multiple granularities, they contain similar information as to defining the best segmentation. A major advantage is that clustering 'temporally close' data values is not data-type specific and it does not need any prior assumptions on the data distributions. It is also more general than the traditional clustering of data with *similar* values, as data values with similar temporal occurrence *may not* have similar values.

Due to **P2**, we want to find these temporally similar data clusters in a principled, unsupervised fashion. The **I**nformation **B**ottleneck (IB) formulation is very well-suited for this task—thinking of segments Y as 'documents' and data values X as 'words' allows us to leverage IB to cluster data values with similar segment distributions $p(y|x)$ *without* specifying an explicit distance metric. As IB is non-parametric, to automatically find the appropriate number of clusters, we further design and optimize a novel **M**inimum **D**escription **L**ength code. Both IB and MDL are based on sound information theory principles. Note that in contrast to other methods (topic modeling, biclustering, etc), IB has exact formal solutions and other advantages.

Q2: Best path. The main challenges are (a) how to define this best path; and (b) how to find it efficiently in the potential exponential search space. In the optimal segmentation, we require the adjacent time segments to be different which may naïvely suggest choosing a path with the maximum sum of weight. At the same time, we want to avoid over-segmenting (having more segments than needed). Due to these considerations, instead, we propose to define the best path in $\mathcal{G}$ as the one that has the maximum *average* edge weight. This definition intrinsically balances the difference of segments and the number of segments, and finds the segmentation automatically without setting the number of segments as an input parameter (**P2**). We further propose a novel efficient DAG-ALP algorithm for finding the average longest path for DAG.

3.4 Details of DASSA

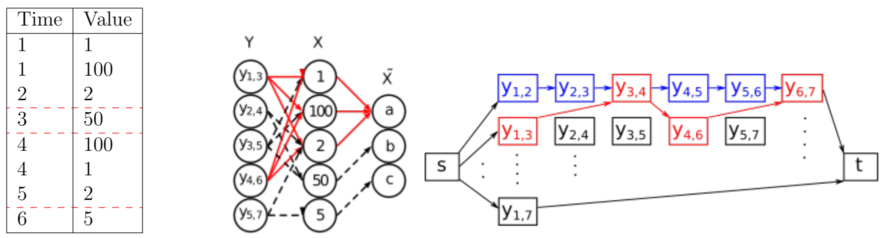
We now give details about DASSA. First, we introduce s_{min} as the unit time length, and divide the time period into these small time units. Hence, a time cut point c_i can be defined as $c_i = t_{min} + i \cdot s_{min}$, $i \in \mathbb{N}$, and $t_{min} \leq c_i \leq t_{max}$, where $t_{min} = \min(t_i)$, and $t_{max} = \max(t_i)$. Now we define a time segment.

Time segments and MTS: A time segment $y_{i,j}$ is a time interval between any two cut points $[c_i, c_j)$. A Minimum Time Segment (MTS) is a time segment $y_{i,j}$ between two adjacent cut points, i.e., $j = i + 1$.

Naturally following, all MTS's have length s_{min}, and they are the smallest time segments of our interest. We further define the set of all possible segments $Y = \{y_{i,j} | c_j - c_i \leq s_{max}\}$, where we assume s_{max} is the maximum segment size we allow in a segmentation (like a year in a Twitter dataset). In experiments, when a natural upper bound is available, we set the s_{max} accordingly, otherwise we set it trivially as $t_{max} - t_{min}$. Note that, we introduce s_{min} and s_{max} mainly to incorporate domain knowledge if there's any. Our algorithm still looks at segments at all granularities of all sizes (in multiples of s_{min}) as we explain later. In principle, we can set these parameters via cross validation, but our results are robust when there are slight changes of them.

Segmentation: A segmentation S is a set of consecutive segments $S = \{y_{a_1,a_2}, \ldots, y_{a_m,a_{m+1}}\}$ where each $y_{a_i,a_j} \subset Y$ and $c_{a_1} = t_{min}, c_{a_{m+1}} = t_{max}$.

We show a running example data sequence in Fig. 3.2(a). The optimal segmentation is shown with the red dash line, which captures the fact that 1, 100, 2 occur together in the sequence.



Time	Value
1	1
1	100
2	2
3	50
4	100
4	1
5	2
6	5

(a) (b) (c)

Figure 3.2: (a) shows an example data sequence, (b) results from our Cluster algorithm, X and Y are connected if the data value x appears in the corresponding y. Values 1, 100, 2 are merged to cluster a because they occur together in the sequence, (c) the segment-graph $\mathcal{G}$, the path/segmentation found by DASSA is marked as red.

3.4.1 Segment-Graph

We construct a Directed Acyclic Graph (DAG) segment-graph $\mathcal{G}$ (V, E, W) to efficiently represent and search among all possible segmentations. We show $\mathcal{G}$'s structure below.

Nodes (V): For each segment $y_{i,j}$ in Y, we construct a corresponding node in a graph $\mathcal{G}$. We also add two extra nodes to the graph: a source node s and a target node t (i.e., $V = \{y_{1,2}, y_{1,3}, \ldots, y_{2,3}, \ldots\} \cup \{s, t\}$).

Edges (E): We create a directed edge from node $y_{i,j}$ to node $y_{k,l}$ iff $j = k$, i.e., they are adjacent segments. Source node s links to all nodes with start time t_{min}; target node t, absorbs links from all nodes with end time t_{max}.

$\mathcal{G}$ is clearly a DAG (as we cannot go back in time), and every path from s to t is one-to-one mapped to a segmentation of the sequence. Hence the segmentation problem is now converted to the problem of finding the best path in $\mathcal{G}$.

3.4.2 Q1: Defining Edge Weights

The edge weight $w(e(y_{i,j}, y_{j,k}))$ measures the difference between adjacent segments $y_{i,j}$ and $y_{j,k}$. We now propose our algorithm Cluster, which combines IB and MDL to automatically cluster data values based on their segment distributions $p(y|x)$ to capture their 'temporal similarity', and define the edge weight as the difference between $p(\tilde{x}|y)$. To facilitate calculating the occurrence of the same value, for features with real values, we discretize them to a constant number of bins of equal size/width as in past literature [149]. In the following, we assume all $\mathbf{x_i}'s$ are discretized.

Finding clusters using IB We define the set of clusters we want to find as $\tilde{X} = \{\tilde{x_1}, \tilde{x_2}, \cdots, \tilde{x_l}\}$, where each $\tilde{x_i}$ contains a set of $x's$ in the data space X, and l is the number of clusters (we discuss how to automatically find l in the next section). We assume each x belongs to only one $\tilde{x}$.

We want to cluster X to $\tilde{X}$ so that $x's$ with similar occurrence in Y are merged in the same $\tilde{x}$. For this task, we re-purpose the word/document formulation of IB [150], designed to cluster words based on the word-document structure. We interpret the set of data values X in our setting as the 'words' and the set of all possible time segments Y as the 'documents'. Since such IB formulation would cluster data values with similar segment distributions $p(y|x)$ (in an information theoretic way *without* specifying a distance metric), we essentially cluster data values with similar temporal occurrence over all time segments. We initialize $\tilde{X} = X$ (each x_i in its own cluster $\tilde{x_i}$), then iteratively merge $\tilde{x_i}, \tilde{x_j}$ pairs which minimizes the loss of temporal information specified as $\delta I(\tilde{x_i}, \tilde{x_j}) = (p(\tilde{x_i}) + p(\tilde{x_j})) * D_{JS}[p(y|\tilde{x_i}), p(y|\tilde{x_j})]$, where D_{JS} is the Jensen-Shannon divergence. Such an iteration process continues until we reach the desired number of clusters l^*. In implementation, we use a priority queue to efficiently find the best data values to merge in each iteration, and reduce the time complexity from $O((|X| - l)|X|^2)$ to $O((|X| - l)|X| \log |X|)$.

Number of clusters using MDL To *automatically* find the appropriate number of clusters l^* in $\mathbb{D}$, we propose to use the MDL principle: the best model for the data is the one that expresses the data losslessly with the smallest code length. To apply MDL, we construct a model class for any cluster number l which combines the corresponding cluster information and some other information (which is needed to express the data losslessly), and then select the best model (and the corresponding l^*) based on the data and the model cost.

Model description. As IB is a lossy compression method, we cannot express the data losslessly using just the cluster information. Hence, we augment the IB results with the following additional information: (a) $p(\mathbf{x}_j|\tilde{x}_i)$, data value distribution in each cluster; (b) $p(y|\tilde{x}_i)$, the probability of a cluster being in a time segment; and (c) $p(\tilde{x}_i)$, the prior for $\tilde{x}_i$. Formally our model is $\theta = \{l, N, |Y|, p(\tilde{x}_i|\mathbf{x}_j), p(y|\tilde{x}_i), p(\tilde{x}_i), p(\mathbf{x}_j|\tilde{x}_i)\}$, where $l = |\tilde{X}|$, and $N = |\mathbb{D}|$. To describe the model, we need to encode the set $\theta \in \mathcal{M}$. So our model description cost is:

$$\begin{aligned} C(M) = &\log^* l + \log^* N + \log^* |Y| + N \log l \\ &- (l|Y| + |\tilde{X}| + l|X|) \log \epsilon \end{aligned} \tag{3.4}$$

where ϵ is the precision for the probability values ($\epsilon = 10^{-5}$ indicates a precision of 0.00001), and $\log^*(n) = \log n + \log\log n + \ldots$ (it is roughly the number of bits to encode an integer $n \geq 1$).

Data description. To describe the data, naïvely one can describe $(\mathbf{x}_j, \{y|t_j \in y\})$ for all y covering t_j. We observe that all time segments $\{y|t_j \in y\}$ containing $\mathbf{x}_j$ must also contain the MTS that covers $\mathbf{x}_j$ (followed from our segment definition). Hence, the likelihood of observing $\mathbf{x}_j$ is equivalent to the likelihood of observing it in the MTS that contains it. Using this observation, we can reduce the number of $(\mathbf{x}, y)$ pairs we need to describe from $|X||Y|$ to $|X|$. We then derive the final data description cost as:

$$\begin{aligned} Cost(X|M) &= -\log_2 L(X, Y|\theta) = -\sum_{(\mathbf{x}_j, y)} \log_2 p(\mathbf{x}_j, y|\theta) \\ &= -\sum_{(\mathbf{x}_j, y)} \log_2 p(\mathbf{x}_j|\tilde{x_*}, \theta) p(y|\tilde{x_*}, \theta) p(\tilde{x_*}|\theta) \end{aligned} \tag{3.5}$$

where $\tilde{x_*}$ is the corresponding cluster for x.

The total cost. Combining the above, the total cost of this description based on the model we described is $C_T = C(M) + C(X|M) = Eq.\ 3.4 + Eq.\ 3.5$. The best model minimizes C_T, i.e. $\theta^* = \arg\min_\theta C_T$, and θ^*'s corresponding l value is the optimal number of clusters. This cost function is hard to optimize: hence we leverage a greedy approach that naturally fits the iteration process we introduced before. We keep greedily merging $\tilde{x}_i, \tilde{x}_j$, and for each s_{mdl} merges, we calculate the corresponding C_T. This iteration process stops (reaching optimal l^*) when C_T begins to increase.

Final edge weights Once we find $\tilde{X}$ and $p(\tilde{x}|x)$, we can calculate the cluster distribution $p(\tilde{x}|y)$ in each segment y by counting the number of times members of each cluster occur in the segment. And the edge weight between segments y_a and y_b in G can be defined as $w(y_a, y_b) = \text{Dist}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$. We want that any distance metric $\text{Dist}(\cdot, \cdot)$ we use should satisfy the following property:

Property 1 *For any three consecutive segments u, v, t, and if v can be further divided into segments v_1 and v_2 (i.e., if $v = [c_i, c_j)$, $v_1 = [c_i, c_k)$, $v_2 = [c_k, c_j)$), then $w(e(u, v)) + w(e(v, t)) \leq w(e(u, v_1)) + w(e(v_1, v_2)) + w(e(v_2, t))$.*

Intuitively, this property makes the segmentation problem well defined in the sense that adding more cut-points always gives us more difference/pattern changes (measured by the sum of edge weights)—hence 'zooming-out', i.e., aggregation by looking at larger time-segments should only *decrease* the difference. Note that capturing more pattern changes does not always lead to a *better* segmentation: having a segmentation with many small changes may be less desirable than one which captures only a few *globally significant* changes at the right segment sizes. Hence how to find the best segmentation is a separate problem.

We use the popular Euclidean distance between distributions (like used in [103]), i.e., $w(e(y_a, y_b)) = D_{EU}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$, which satisfies property 2. (See supp. Appendix A). The proof follows from the subadditivity (triangle inequality) of D_{EU}. In contrast, the well-known KL divergence does not satisfy this property in general.

3.4.3 Q2: Finding the Best Path

In the weighted $\mathcal{G}$, the problem of finding the optimal segmentation is now reduced to finding the 'best' path from the set of all valid paths $\mathcal{P}$ in $\mathcal{G}$.

We argue that a good segmentation should regularize the total segment difference with the number of segments: having many small changes is less desirable than capturing just the significant ones. Hence, we propose to solve the Average Longest Path problem (ALP) to find the best path.

Given: Segment-graph (DAG) $\mathcal{G}$ (V, E, W) with a start node s and end node t.

Find: Path S^* from s to t with maximum average weight: $S^* = \arg\max_{S \in \mathcal{P}} \frac{\sum_{\text{e in } S} w(e)}{|S|}$.

We present a novel ALP algorithm DAG-ALP with $O(h \cdot |E|)$ on general DAGs (h is the maximum path length in the DAG). Our idea is that the ALP from s to t must also be the *longest (most heavily weighted)* path among all paths with the same number of nodes. Hence, we calculate all the longest paths with *different* lengths (number of nodes) from s to t, and find the one giving the maximum average edge weight. More concretely, DAG-ALP uses a multi-layer structure, where the first layer L_0 contains only the beginning node s, and layer L_i contains the nodes which can be reached from s by i steps. When we iterate through layers, we maintain the weight ($lp_i(v)$) of the longest path from s to $v \in L_i$, and the parent node of v in L_i ($\pi(v, i)$) in the longest path. After all iterations, we get longest paths from s to t with different lengths, and we output the one with the largest average weight. Alg. 2 shows the brief pseudo-code. Due to the structure of our segment-graph, DAG-ALP finds the ALP in $\mathcal{G}$ in $O(E)$ time (as h is bounded by the length of the data sequence).

Time and space complexity The pseudo-code of DASSA is shown in Alg. 1. With priority queue, reduction of unnecessary data description, and DAG-ALP, our final time complexity is $O((|X| - l^*)|X| \log |X| + |E|)$. To find the ALP we only need to store the previous layer in DAG-ALP, hence the overall space complexity of DASSA is $O(|\mathbb{D}|)$. In practice, for all

Algorithm 2 Pseudo-code of DAG-ALP

```
Require: a weighted DAG $\mathcal{G}$ (V, E, W), $h$, $s$, $t$
Ensure: Average longest path
1: $Layer_0 = \{s\}$ // initialize the first layer
2: $lp_0(s) = 0$ // the longest path form $s$ to $s$ with length 0 is initialized as 0
3: for $i = 1$ to $h$ do
4:     $Layer_i$ ={nodes directly connected to any nodes in $Layer_{i-1}$}
5:     Calculate $lp_i(\cdot)$ for nodes in $Layer_i$ using $lp_{i-1}(\cdot)$
6: $ALP = \arg\max(\frac{lp_i(t)}{i})$
```

datasets used in our experiments, DAG-ALP finishes within $40s$, and the complete algorithm takes $30m$ to run on average (including one with 2 million data observations), satisfying **P3**.

3.5 Experiments

Setup. Our experiments are conducted on a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory and DASSA takes 30m to run on average for our datasets. For all the datasets, we set a discretization level $k = 10$ as it leads to a reasonable running time, and the performance is stable around 10 ($k = 5, 15$ gives similar results). When constructing the segment-graph in practice, we ignore segments with less than 5% of $|D|$ data values (which is a small fraction of all segments), as they have too few observations, and are not interesting for the final segmentation.

Datasets. DASSA works for general data sequences, hence we collected real world datasets from different domains to test. Tab. 3.1 shows the content of each data sequence. These sequences contain different data types like age, town Id (categorical), sensor observations (real), etc., different time-units and some of them (like *Portland*, *Ebola*) have arbitrary time stamps (a data point can have any time stamp value, and as a result there may be different number of data points at each time stamp).

Dataset	**Domain**	s_{min}	s_{max}	**Data sequence** $\mathbb{D}$	**Ground truth**
Portland	Epidemiology	0.2	1.0	$\{[age, y, x, income, size, \#workers, \#cars]_i, t_i\}_{i=1}^{N}$	✓
ChickenDance	Motion Seq.	10s	300s	$\{[Sensor_1, Sensor_2, Sensor_3, Sensor_4]_i, t_i\}_{i=1}^{N}$	✓
Twitter	Social Media	10d	100d	$\{[\#(Word_1), \#(Word_2), \ldots, \#(Word_{12})]_i, t_i\}_{i=1}^{N}$	-
Ebola	Epidemiology	4d	48d	$\{[Infection\ Status, Town\ ID]_i, t_i\}_{i=1}^{N}$	-
PUC-Rio	Motion Seq.	150s	600s	$\{[\text{6 demographical, 12 sensor features}]_i, t_i\}_{i=1}^{N}$	✓

Table 3.1: Summary of Datasets

Baselines. To the best of our knowledge, there is no algorithm that finds segmentations for general data sequences as we do. Hence, we first adapt a time series algorithm *Dynammo* [98] (also used in [109]) as our baseline. Additionally, we compare with three variations of DASSA (*EMP*, *TopicM*, *LP* in Tab. 3.2). Note that unlike DASSA which detects the no. of cut points

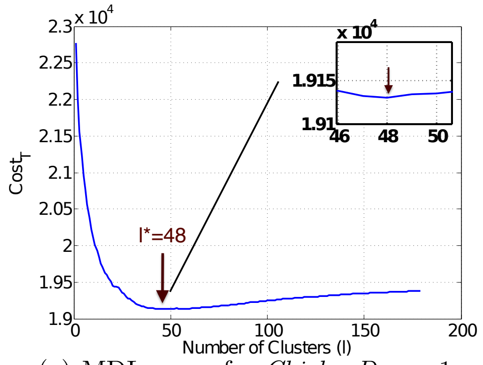
Baseline	**Description**
EMP	Defines the distance between segments based on the empirical data distribution $p(\mathbf{x}_j\|y)$ instead of $p(\tilde{x}_i\|y)$.
TopicM	Finds clusters of values using topic modeling instead of our IB-based data clustering.
LP	Finds the longest path instead of the ALP as the optimal segmentation.
Dynammo	Averaging data points in a sliding window to construct multi-dimensional time series, then feed the time series and the no. of cut points to *Dynammo*.

Table 3.2: Baselines description.

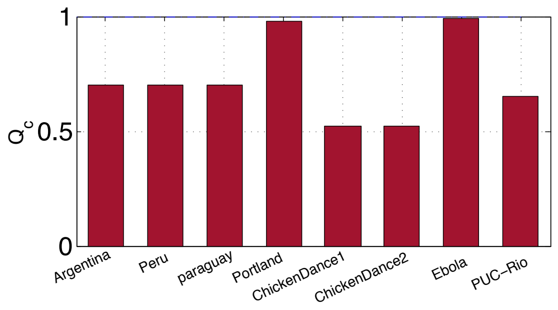
automatically, *Dynammo* needs this as an input. We set this value from the ground truth when one is available, otherwise we set it as the number detected by DASSA.

3.5.1 Results

Testing each component of DASSA: We check the number of clusters found by MDL. Fig. 3.3(a) shows that the MDL-curve is indeed near-convex, and it suggests an optimal number of clusters. We examine the quality of the detected clusters by designing a Silhouette score Q_c to measure the 'temporal similarity' of data values in the clusters. The Silhouette score (Fig. 3.3(b)) shows that the data values in the clusters we found truly appear close in time (all $Q_c > 0.5$). We also compare our ALP path optimization with the longest path (LP) optimization, which finds the path with the maximum sum of edge weights. Our ALP path optimization outperforms the LP optimization in all of the datasets with ground truth segmentations (see Tab 3.3).



(a) MDL curve for *ChickenDance* 1



(b) Q_c scores

Figure 3.3: (a) MDL curves of *ChickenDance* 1: C_T vs. number of clusters l. (b) Q_c scores. Note $Q_c > 0.5$ for all datasets—indicates high quality clusters.

Quality of segmentations: We measure our final segmentation output here. We show the $\mathbf{F_1}$ **score** for datasets with ground truth segmentation (*Portland*, *ChickenDance*, *PUC-Rio*),

Dataset	**DASSA**	*TopicM*	*EMP*	*LP*	*Dynammo*
ChickenDance 1	**1**	0.85	0.76	0.63	0.57
ChickenDance 2	**1**	0.6	0.90	0.54	0.71
Portland	**1**	1	0.66	0	**1**
PUC-Rio	**0.66**	0.46	0.25	0.44	0.25

Table 3.3: F_1 score of DASSA, *TopicM*, *EMP*, *LP* and *Dynammo* on different datasets with ground-truth segmentation: DASSA gets perfect cuts in most of the datasets.

and our case study results for *Twitter* and *Ebola*.

Quantitative evaluation In short, DASSA gives much better F_1 scores.

Portland: DASSA finds the exact ground truth ($F_1 = 1$ in Tab. 3.3), and *EMP* has a much lower score (∼0.6). *TopicM* and *Dynammo* also gets $F_1 = 1$ in this dataset, but in all other datasets, DASSA outperforms both of them. We show the most frequent values in the two segments of the segmentation found by DASSA in Fig. 3.1(a). It shows that elderly people, with higher incomes, larger number of workers in family, and more vehicles are infected first. Then younger people with lower incomes, fewer vehicles get infected. It illustrates that DASSA is capable of detecting the pattern of disease propagation. And the results are easily interpretable.

ChickenDance: We find the exact ground truth ($F_1 = 1$). As shown in Fig. 3.4, DASSA discovers all the distinct chicken dance motions precisely. In contrast, the cut points detected by *EMP*, *TopicM* and *Dynammo* do not correctly find the time when a different motion takes place: they either miss the correct cut points, or have unnecessary additional ones.

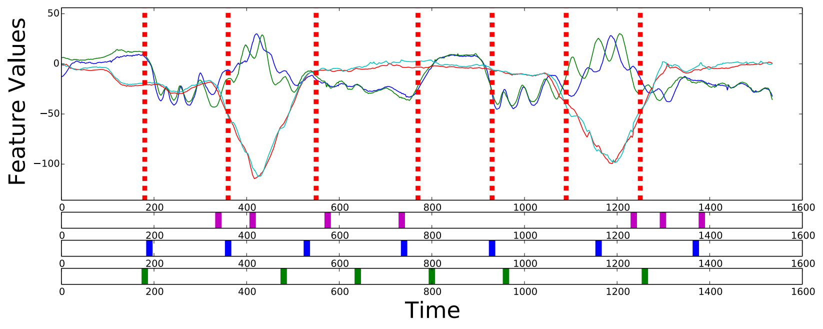


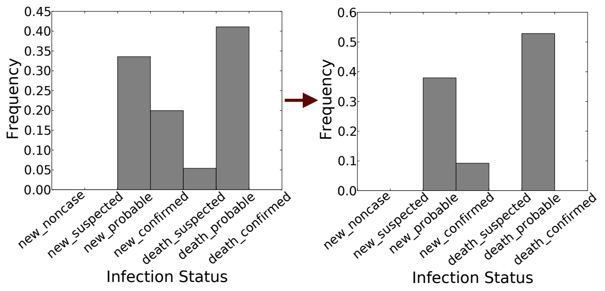
Figure 3.4: DASSA segmentation results for *ChickenDance*. The cut points of *Dynammo* (purple in the 1st row), *TopicM* (blue in 2nd row), and *EMP* (green in 3rd row) are shown below the DASSA.

PUC-Rio: This dataset was originally collected for classification tasks. Finding the difference between actions is itself a non-trivial task. Interestingly, DASSA is powerful enough to capture some meaningful segments. We see that in Tab. 3.3, DASSA reaches a F_1 score of around 0.66, which again outperforms both *EMP* and *TopicM*.

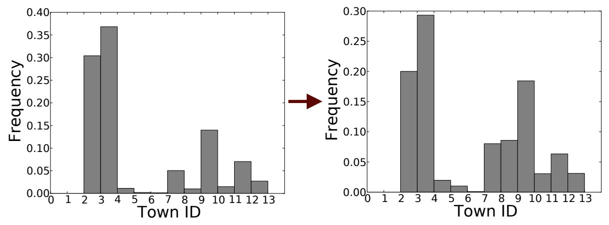
Case studies DASSA gives meaningful segments, compared to baselines.

Twitter (Peru, Paraguay, Argentina): To explore the segmentation found by DASSA, we look at users' tweets in each segment and count the number of each word to draw word clouds. The size of a word in the word cloud is proportional to the frequency of its usage in the segment. As shown in Fig. 3.1(b), DASSA finds three segments. We observe that the sizes/frequencies of infection-related words like 'headache', 'tired' and 'fever' are decreasing from segment to segment. On the other hand, the frequency of the word 'remedy' gradually increases. This matches what we expect from a typical infection cycle: from getting exposed, to getting sick, and finally to be cured. Recent work [33] also matches what we found in the word cloud (unlike us they use complex temporal graphical models to figure out similar word clouds). In contrast, we find that *Dynammo* and *TopicM* fail to capture meaningful word transitions.

Ebola: We explore the feature values in the two segments detected by DASSA. In Fig. 3.5(a) we see that the death and newly confirmed cases reduce significantly from segment 1 to segment 2, which shows a sign of increased caution for the disease. We also notice from the change of distribution of towns (Fig. 3.5(b)) that at first the infection mostly occurs in town 2 and 3 which are 'Kono' and 'Kambia' in Sierra-Leone. Then it spreads to other towns (e.g., town 9 which is 'Bo' in Sierra-Leone). DASSA *automatically* finds a segmentation that captures this disease propagation pattern; giving a better understanding of the situation.



(a) Change of infection status



(b) Change of infection towns

Figure 3.5: DASSA results for *Ebola*. (a) Distribution of infection status for the two segments detected. (b) Distribution of infection towns for the two segments detected.

3.6 Discussion

A segmentation algorithm implicitly contains its own distance measurements between time-segments. In this chapter, we define the distance as a carefully designed metric between the associated 'co-occurrence' cluster distributions in the segments ($p(\tilde{\mathbf{x}}|y)$). One might naturally think to define the segment distance as the distance between the clusters in segments themselves; but doing so has multiple issues. As an example, we ran one classic subspace clustering algorithm (*Fires* [85]) on our datasets. We observe that *Fires* simply does not output any clusters for many segments (for example the last two segments in the optimal segmentation of *Argentina*, *Paraguay*, *Peru*), and it cannot detect the same good segmentations as DASSA does. We believe similar problems would happen to other traditional clustering algorithms as well. The cluster-based distance measurements intrinsically do not handle well datasets where there is no clustering. In addition, using the clusters themselves to represent the dataset will lose information as many data points are not in any of the clusters.

3.7 Conclusions

We introduce DASSA, a novel, general, self-guided and efficient algorithm, to automatically segment data sequences. We construct a segment-graph to efficiently represent and search among all possible segmentations. Then we propose an IB-MDL-based clustering algorithm to capture temporal similarities between data values. Finally, a novel DAG-ALP algorithm is presented to automatically find the segmentation. DASSA has good performance on all datasets we collect: discovering ground truth, finding high quality segmentations, and providing interpretable real patterns.

Our framework is general for segmentation problems and extending it for more complex sequences (such as image sequences) can be interesting future work. Future work can also look into a parallelized or online version of DASSA.

Chapter 4

Modeling and Predicting Pattern Changes behind Tweets

Surveillance of epidemic outbreaks and spread from social media is an important tool for governments and public health authorities. Machine learning techniques for nowcasting the flu have made significant inroads into correlating social media trends to case counts and prevalence of epidemics in a population. There is a disconnect between data-driven methods for forecasting flu incidence and epidemiological models that adopt a state based understanding of transitions, that can lead to sub-optimal predictions. Furthermore, models for epidemiological activity and social activity (like on Twitter) predict different shapes of volume changes and have important differences.

In this chapter, we propose two temporal topic models (one unsupervised model as well as one improved weakly-supervised model) to capture hidden states of a user from his/her tweets, and aggregate states in a geographical region for better estimation of trends. We show that our approaches help fill the gap between phenomenological methods for disease surveillance and epidemiological models. We validate our approaches by modeling the flu using Twitter in multiple countries of South America. We demonstrate that our models can consistently outperform plain vocabulary assessment in flu case-count predictions, and at the same time get better flu-peak predictions than competitors. We also show that our fine-grained modeling can reconcile some contrasting behaviors between epidemiological and social models.

4.1 Introduction

Web searches and social media sources, such as Twitter and Facebook, have emerged as surrogate data sources for monitoring and forecasting the rise of public health epidemics. The celebrated example of such surrogate sources is arguably Google Flu Trends where user query volume for a handcrafted vocabulary of keywords is harnessed to yield estimates of flu case counts. Such surrogates thus provide an easy-to-observe, indirect, approach to understanding population-level health events.

Recent research has brought intense scrutiny of Google Flu Trends, often negative. [91] provide explanations for Google Flu Trend's lackluster performance. Some of the reasons are institutional (e.g., a cloud of secrecy about which keywords are used in the model, affecting reproducibility and verification), some are operational (e.g., lack of periodic re-training), while others could be indicative of more systemic problems, e.g., that the vocabulary for tracking might evolve over time, or that greater care is needed to distinguish which aspects of search query volume should be used in modeling. These problems are not unique to Google Flu Trends; they would resurface with any syndromic surveillance strategy, e.g., developing a flu count modeler using Twitter.

Motivated by such considerations, we aim to better bridge the gap between syndromic surveillance strategies and contagion-based epidemiological modeling such as SI, SIR, and SEIS [10]. In particular, while models of social activity have been inspired by epidemiological research, recent work [111, 175, 142] has shown that there are key aspects along which they differ from biological contagions. Specifically, evidence from [111, 44] shows that the activity profile (or the number of new people using a hashtag/keyword) shows a power-law drop—in contrast standard epidemiological models exhibit an exponential drop [64]. Also, there is some evidence that hashtags of different topics show an exposure curve which is not monotonic, resembling a complex contagion [142].

In this chapter, we show that we can reconcile the apparently contrasting behaviors with a finer-grained modeling of biological phases as inferred from tweets. For example, sample tweets "Down with flu. Not going to school." and "Recovered from flu after 5 day, now going to the beach" denote different states of the users (also see Figure 4.1). We argue that correcting for which epidemiological state a user belongs to, the social and biological activity time-series are actually similar. Hashtags and keywords merge users belonging to different epidemiological phases. We separate these states by using a temporal topic model in this chapter. In addition, thanks to the finer-grained modeling, our approach gets better predictions of the incidence of flu-cases than direct keyword counting and also sometimes gets better predictions of flu-peaks than sophisticated methods like Google Flu Trends.

Our contributions are:

1. We propose temporal topic models (HFSTM and HFSTM-A) for inferring hidden biological states for users, and an EM-based learning algorithm for modeling the hidden epidemiological state of a user. The HFSTM-A model is robust to noisy and large vocabularies.

2. We show via extensive experiments using tweets from South America that our learners indeed learn meaningful word distributions and state transitions. Further, our methods

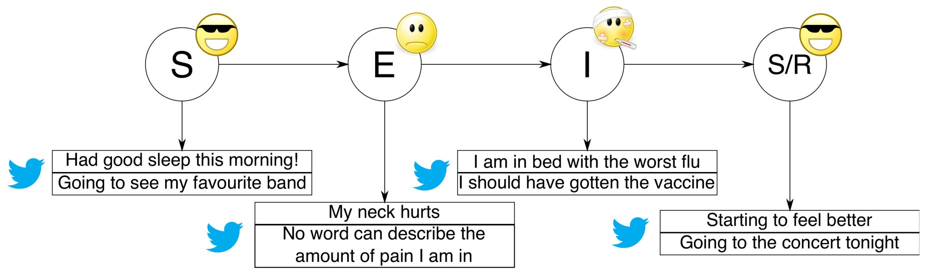

S
E
I
S/R
Had good sleep this morning!
Going to see my favourite band
My neck hurts
No word can describe the amount of pain I am in
I am in bed with the worst flu
I should have gotten the vaccine
Starting to feel better
Going to the concert tonight

Figure 4.1: A toy example showing possible user states and a tweet pattern associated with each state when a user is infected with flu for a time period

can better forecast the flu-trend as well as flu-peaks by aggregating user states in a region over a time period.

3. Finally, we show that the state information learned by our models reconciles the social contagion activity profile with standard epidemiological models.

Our work can be seen as a stepping stone to better understanding of contagions that occur in both biological and social spheres.

4.2 Formulation of Models

We formulate our models in this section. The hypothesis is that a tweet stream generated by a user can be used to capture the underlying health condition of that particular user, and that the health state (e.g., flu state) of a user remains the same within a tweet. Then we use our models to capture the flu states of a user–which are S (healthy), E (exposed), or I (infected)–based on the classic flu-like Susceptible-Exposed-Infected-Susceptible SEIS epidemiological model. These states model the different health conditions of a person throughout the lifecycle of the infection. In this study, we first introduce the HFSTM model. Then we show the limitation of HFSTM, and propose an improved model HFSTM-A (HFSTM with aspects) to address the issue.

4.2.1 Hidden Flu-State from Tweet Model (HFSTM)

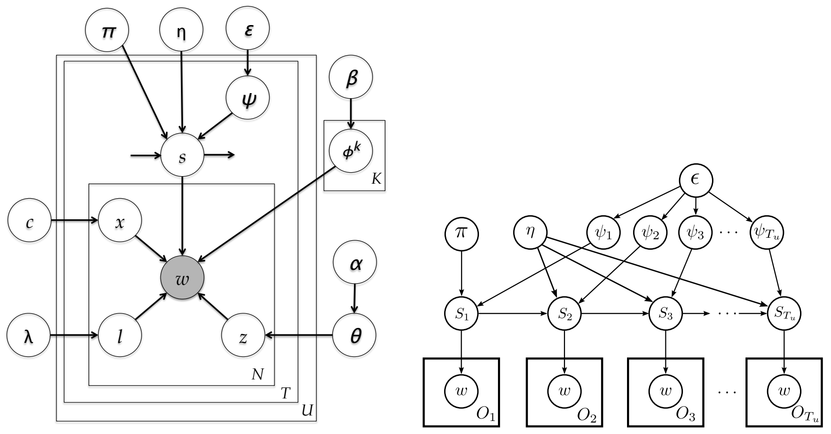
A tweet is a collection of words and a tweet stream is a collection of tweets. The number of tweets varies across users and the number of words in a tweet varies within and across users. We denote the t-th tweet of a user by $O_t = \langle w_{t1}, w_{t2}, \ldots, w_{tN_t}\rangle$ where w_{tn} denotes the n-th word in the tweet and N_t denotes the total number of words in the tweet. Let $\mathcal{O}_u = \langle O_1, O_2, \ldots, O_{T_u}\rangle$ be the tweet stream generated by a user u and $\mathcal{S}_u = \langle S_1, S_2, \ldots, S_{T_u}\rangle$ be the underlying state of the stream $\mathcal{O}_u$. Here T_u denotes the length of the stream of a

Table 4.1: Symbols used for HFSTM and HFSTM-A

Symbol	Meaning
S	Flu state
S_t	Flu state for the t-th tweet
ϵ	State switching parameter
π	Initial state distribution
η	Transition probability matrix
l	Binary background switching variable
x	Binary switch between flu and non-flu words
y	Aspect of word
λ	Parameter for the Bernoulli distribution for l
c	Parameter for the Bernoulli distribution for x
ϕ	Topic distribution
σ	Prior for l when aspect is introduced
γ	Prior for x when aspect is introduced
T_u	Total number of tweets for the u-th user
N_t	Total number of words in t-th tweet
w	Word variable in the template model
w_{tn}	The n-th word in the t-th tweet
$TopicM$	Non-flu related topic
θ	Prior for non-flu related topics
α	Hyper parameter for topic distributions
ψ	State switching variable
K	Total number of states
β	Dirichlet parameter for word distributions
U	Number of users

user u and $S_t \in \{S, E, I\}$. Let $\mathcal{O} = \langle \mathcal{O}_1, \mathcal{O}2, \ldots, \mathcal{O}_U \rangle$ be the collection of tweets for U users, from which we aim to learn the parameters of our model. We use K to denote the number of states that S_t can take (see Tab. 4.1 for notation).

Our initial model—Hidden Flu-State from Tweet model HFSTM—is a probabilistic graphical model which captures the tweet structure of a flu-related tweet. It is a temporal topic model for predicting the state sequence of a user given $\mathcal{O}_u$ and is illustrated in Fig. 4.2(a). An expansion of the plate notation for the same is illustrated in Fig. 4.2(b). In this model each word w for $O_t \in \mathcal{O}_u$ is generated when the user is in a particular flu state (S_t) or the user talks about a non-flu related topic ($TopicM_i$). For example, in the message "I have caught the flu. Feeling feverish. Not going to school" the words 'flu', 'feverish', 'caught' are generated because the user is in the "infected" state and the words 'going' and 'school' are generated by non-flu related topics. Sometimes a word can be generated due to noise which is also accounted for in our model.



(a) HFSTM (b) State transition

Figure 4.2: (a) Plate notation for HFSTM: The variable S captures the hidden state of the user in which the user generated this tweet. The LDA-like topic variable *TopicM* captures non-flu related words. (b) HFSTM state variables expanded: Each message O_t is associated with a state S_t, which remains the same for flu-related words in O_t. Switching from one state to another is controlled by a binary switching variable ψ and the next state S_{t+1} from the current state S_t is drawn using transition probabilities η.

The generative process for the model is shown in Alg. 3. A binary variable l determines whether or not a word is generated from a background distribution. The binary variable x determines whether the current word is generated from non-flu related topics or flu-state distributions. The value of l and x are generated from Bernoulli distributions parameterized by λ and c. The non-flu related topics follow the LDA like mechanism [24]. The state for the first tweet is drawn from the initial distribution denoted by π. We assume that the states of the subsequent tweets are generated due to a state transition or by copying from the previous state which is determined by a binary switching variable ψ with prior parameter ϵ. The state S_t (for $2 \leq t \leq T_u$) of the subsequent tweets are drawn from transition matrix η and previous state S_{t-1} with probability ϵ or copied from the previous state S_{t-1} with probability $1-\epsilon$. Once the state of a tweet is determined, a word is generated from a word distribution defined by that state.

Let $O_t = (w_1, \ldots, w_N)$ be the words that are generated when a user is in a particular state. The likelihood of the words generated by a user in that state is given below.

$$\begin{aligned} p(\mathcal{O}_u) &= \sum_{S_t} p(\mathcal{O}_u, S_t) = \sum_{S_t} p(O_1 \ldots, O_T, S_t) \\ &= \sum_{S_t} \sum_{S_{t-1}} p(O_t|S_t) p(S_t|S_{t-1)} p(O_{t-1}, S_{t-1}) \qquad (4.1) \end{aligned}$$

[11] show that such kind of likelihood function is intractable. In HFSTM the unknown parameters that we want to learn are $H = \{\epsilon, \pi, \eta, \phi, \lambda, c\}$. The posterior distributions over

Algorithm 3 Generator($\lambda, c, \eta, \pi, \alpha, \beta, \epsilon$) for HFSTM

```
Require: A set of parameters.
Ensure: Topics and flu state of each user.
 1: Set the background switching binomial $\lambda$
 2: Choose $\phi \sim \mathrm{Dir}(\beta)$ for the non-flu topics, flu states, and background distribution
 3: Choose initial state $s_1 \sim \mathrm{Mult}(\pi)$
 4: Draw each row of $\eta$ using $\mathrm{Dir}(\alpha)$ {Trans. matrix}
 5: Draw $\theta \sim \mathrm{Dir}(\alpha)$
 6: for each tweet $1 \leq t \leq T_u$ do
 7:   if not the 1st tweet in the corpus then
 8:     Draw $\psi_t \sim \mathrm{Ber}(\epsilon)$
 9:     if $\psi_t = 0$ then
10:       $S_t \leftarrow S_{t-1}$
11:     else
12:       $S_t \leftarrow \mathrm{Mult}(\eta_{S_{t-1}})$
13:   for Each word $w_i$, $1 \leq i \leq N_t$ do
14:     Draw $l_i \in \{0, 1\} \sim \mathrm{Ber}(\lambda)$ {Background switcher.}
15:     if $l_i = 0$ then
16:       Draw $w_i \sim \mathrm{Mult}(\phi^B)$ {Draw from background distribution.}
17:     else
18:       Draw $x_i \in \{0, 1\} \sim \mathrm{Ber}(c)$
19:       if $x_i = 0$ then
20:         Draw $z_i \sim \mathrm{Mult}(\theta)$
21:         Draw $w_i \sim \mathrm{Mult}(\phi^{z_i})$ {Draw from non-flu related distribution.}
22:       else
23:         Draw $w_i \sim \mathrm{Mult}(\phi^{s_t})$ {Draw from flu related distribution.}
```

these unknown variables are also intractable since the posterior distributuions depend on the likelihood function. We hence developed an EM-based algorithm HFSTM-FIT to estimate the parameters H of the model (we omit the details for this algorithm as it is very similar to the inference algorithm for the extended HFSTM-A model, and we would elaborate on the latter in Sec. 4.2.4).

4.2.2 Issues with HFSTM

HFSTM requires a 'clean' vocabulary, i.e., a vocabulary that does not contain many background words. In real datasets, there is a huge imbalance between background and flu-related words. For example, among 100 tweets from a user, only two or three may be related to his/her health state. As there is no supervision used in HFSTM, each word has the same probability of passing/failing the switches (see the parameters λ, c in 4.2.1), which biases our model towards background words. Hence it is likely for HFSTM to learn the complex state transitions among background words rather than among the flu-related words. If the dataset contains many tweets about some hot event such as a football game, the model would learn the state transition in the sport game rather than in the flu infection since the number of sport-related tweets overwhelms that of the flu-related tweets. For this reason, HFSTM needs a vocabulary that does not contain many background words, otherwise we observe that HFSTM learns the state transition behind the non-flu-related topics. As a consequence, it highly depends on the accuracy of the selection of words, which decreases its generality.

4.2.3 Improving the Model—HFSTM-A

Due to the issues with HFSTM, we propose a new model HFSTM-A (HFSTM with aspects) so that we can provide it with a larger noisier vocabulary. Our key idea is to explicitly include our belief of which words are likely to be useful for state transitions. Hence we add such weak supervision to HFSTM by introducing an 'aspect' value (y) for each word (a related approach has been used by [127]). We call this new model HFSTM-A. This aspect y takes two values $\{0, 1\}$ based on whether the word is flu related or not. It then biases the switching probabilities so that background words are less likely to be explained by the state topic distributions. Note that this supervision is weak because the aspect of a word does not directly decide if a word is flu-related; it only increases/decreases the probability of a word being regarded as flu-related or not. Those words which we do not mark as related are still possible to be analysed by state topic distributions. As a result of the changes, HFSTM-A can handle much noisier vocabularies, and have performance comparable with the HFSTM model.

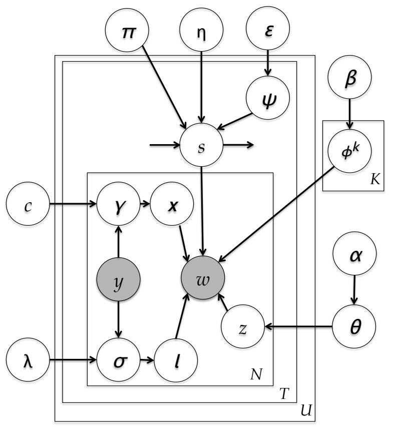


Figure 4.3: Plate notation for HFSTM-A: The aspect value y is an observed variable for each word, and this variable biases the probability of a word being generated by the various topics (see Sec. 4.2.3)

Algorithm 4 Generator($\lambda, c, \eta, \pi, \alpha, \beta, \epsilon$) for HFSTM-A

```
Require: A set of parameters.
Ensure: Topics and flu state of each user.
 1: Set the background switching binomial $\lambda$
 2: Choose $\phi \sim (\beta)$ for the non-flu topics, flu states, and background distribution
 3: Choose initial state $s_1 \sim \text{Mult}(\pi)$
 4: Draw each row of $\eta$ using Dir($\alpha$) {Trans. matrix}
 5: Draw $\theta \sim \text{Dir}(\alpha)$
 6: for each tweet $1 \leq t \leq T_u$ do
 7:   if not the 1st tweet in the corpus then
 8:     Draw $\psi_t \sim \text{Ber}(\epsilon)$
 9:     if $\psi_t = 0$ then
10:       $S_t \leftarrow S_{t-1}$
11:     else
12:       $S_t \leftarrow \text{Mult}(\eta_{S_{t-1}})$
13:   for Each word $w_i$, $1 \leq i \leq N_t$ do
14:     Draw $y_i \in \{0,1\}$ (observed)
15:     Draw $l_i \in \{0,1\} \sim \text{Ber}(\lambda_{y_i})$ {Background switcher.}
16:     if $l_i = 0$ then
17:       Draw $w_i \sim \text{Mult}(\phi^B)$ {Draw from background distribution.}
18:     else
19:       Draw $x_i \in \{0,1\} \sim \text{Ber}(c_{y_i})$
20:       if $x_i = 0$ then
21:         Draw $z_i \sim \text{Mult}(\theta)$
22:         Draw $w_i \sim \text{Mult}(\phi^{z_i})$ {Draw from non-flu related distribution.}
23:       else
24:         Draw $w_i \sim \text{Mult}(\phi^{s_t})$ {Draw from flu related distribution.}
```

More concretely, in the plate notation of this new model HFSTM-A (see Fig. 4.3), y is the observed aspect value for a word, where l and x are the binary values which decide whether the word is generated by background topic, non-flu topic, or state topic distribution. In HFSTM, these two values are generated by the Bernoulli distribution with parameters λ and c. Now in HFSTM-A, y biases these probabilities and may thus change the values of l and x.

The generative process for the HFSTM-A model is shown in Alg. 4. In contrast to Alg. 3, we see that in Alg. 4 the value of l and x are now generated from Bernoulli distributions parameterized by λ_{y_i} and c_{y_i}, which are biased by the observed aspect value y_i. The definition of λ_{y_i} and c_{y_i} are shown below.

$$\lambda_{y_i=0} = \lambda \tag{4.2}$$
$$\lambda_{y_i=1} = \lambda + b * (1 - \lambda) \tag{4.3}$$
$$c_{y_i=0} = c - a * c \tag{4.4}$$
$$c_{y_i=1} = c + a * (1 - c) \tag{4.5}$$

where a, b are the fixed biases we add to the switching probabilities. Basically, if a word is labeled as flu-related ($y_i = 1$), we increase its probability of passing the background switch ($\lambda_{y_i=1}$) and its probability of passing the non-flu topic switch ($c_{y_i=1}$), by pushing these probabilities closer to 1. In the equations above, we take a proportion (b and a respectively) of the residuals and add it to the probability; and if a word is not flu-related ($y_i = 0$), we decrease its probability of passing the non-flu topic switch ($c_{y_i=0}$), by pushing the probability towards 0 (we use a to shrink the value in the corresponding equation). Note that if a word is not flu-related, it can still be generated by non-flu topics. Hence its probability of passing the background switch ($\lambda_{y_i=0}$) is kept unbiased. In our experiments, we test different parameter settings, and find the performance good and stable around $a = 0.4$, $b = 0.4$.

4.2.4 HFSTM-A-FIT: Inference and Parameter Estimation

We next show an EM-based algorithm HFSTM-A-FIT to estimate the parameters $H = (\epsilon, \pi, \eta, \phi, \lambda, c)$ of our model.

At each time point t a user can be in any of the $2K$ states where the first K states denote that the user happens to be in the state due to a state transition from his state at time $t-1$ and the rest of states from $K+1 \ldots 2K$ denote that the state of the user is simply copied from the state of the user at time $t-1$.

E-Step

We use a forward-backward procedure for estimating parameters. We define the forward probability $A_t(i)$ and the backward probability $B_t(i)$ for a tweet stream as follows.

$$A_t(i) = P(t_1, t_2, \ldots, t_t, S_t = i)$$
$$B_t(i) = P(t_{t+1}, \ldots, t_{T_u} | S_t = i)$$

$A_t(i)$ is the joint probability of the partially observed sequence until time t and state S_t is i. $B_t(i)$ is the joint probability of the partially observed sequence from $t+1$ to T_u, given state S_t is i. Both $A_t(i)$ and $B_t(i)$ can be solved inductively. See the linked equations for more details on how $A_t(i)$ and $B_t(i)$ are calculated.

Let $\gamma_t(i)$ be the probability of being in state S_i for the t_{th} tweet given the observed tweet sequence $\mathcal{O}_u$.

$$\begin{aligned}\gamma_t(i) &= P(S_t = i|\mathcal{O}_u)\\ &= \frac{A_t(i)B_t(i)}{\sum_{i=1}^{2K} A_t(i)B_t(i)}\end{aligned}$$

To estimate the transition probability we define $\xi_t(i,j)$, the probability of being in state i at t, and in state j at $t-1$ given the $\mathcal{O}_u$.

$$\begin{aligned}\xi_t(i,j) &= P(S_t = i, S_{t+1} = j|\mathcal{O}_u)\\ &= \frac{P(S_t = i, S_{t+1} = j, \mathcal{O}_u)}{P(\mathcal{O}_u)}\end{aligned}$$

M-Step

In this step we re-estimate the parameters ϵ, π, η, ϕ, c, λ. We only show the estimations of π and η below. [1]

$$\pi_i = \frac{\sum_{u=1}^{U} \gamma_1(i)}{\sum_{u=1}^{U} \sum_{i=1}^{K} \gamma_1(i)}$$

$$\eta_{ij} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} (\xi_t(i,j) + \xi_t(i+K,j))}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{j=1}^{K} (\xi_t(i,j) + \xi_t(i+K,j))}$$

4.3 Experiments

We describe our experimental results next. All the experiments are designed to answer the following questions:

1. Can HFSTM and HFSTM-A robustly learn in presence of different noise levels in a dataset? (See Sec. 4.3.2)
2. What are the state-topic distributions learnt by our models? (See Sec. 4.3.3)
3. Is the state transition table learned reasonable? (See Sec. 4.3.4)
4. How do our models perform for flu case-count and peak predictions? (See Sec. 4.3.5)

[1] More information in Appendix B.

5. Finally, as mentioned before, several papers [111, 175] have shown that the rising and falling pattern of keywords count in social media does not match with that in epidemiological model. By including the extra state information, can we bridge this gap between social and epidemiological activity? (See Sec. 4.3.6)

4.3.1 Experimental Setup

First we describe our setup in more detail. Our algorithms were implemented in Python.[1]

Vocabularies

To ensure that the most important words (directly flu-related words like 'flu', 'cold', 'congestion', etc.) are included in our vocabulary, we first build a flu-related keyword list. [30] construct a flu-keyword list, by first manually setting a seed set, then using two methods (pseudo-query and correlation analysis, see their paper for more details) to expand this seed set, and then finally pruning it to a 114 words keyword list. A similar keyword-construction procedure (expanding by crawling websites) was also used by [88]. For our experiments, we include the same 114 keywords from [30] first. We then manually select and include 116 words, which are not directly related to flu, but may implicitly imply the state of a user, such as 'hopeless', 'bed', 'die', 'sad', etc. We use these (a total of 230) words as the vocabulary for HFSTM since it cannot deal well with a noisier vocabulary (see Sec. 4.2.2).

For HFSTM-A, the extension of HFSTM which is designed to handle larger vocabularies with much background noise, we enlarge the size of the vocabulary by simply adding the most frequent words in the corpus. After automatically extracting these top words, we get a final vocabulary of 2739 words.[1] All other words not in our vocabulary but occurring in the corpus are mapped to a single generic block-word. We label a word as 1 (flu-related) if it is in the previous 230 words list (note again this is only weak supervision; this label does not directly decide whether this word is generated by background topics, or state topics). Thanks to our model design, as we describe later, HFSTM-A is able to learn meaningful state transitions and topic distributions, in spite of having a more than $10X$ larger vocabulary.[1]

Datasets

We collected tweets generated from 15 countries in South America for the period from Dec. 2012 to Aug. 2014 using Datasift's Twitter collection service.[2] Briefly, what the company does is using the Basis technology[3] natural language processing facilities to lemmatize words, and using a custom set of geocoding algorithms to detect the location of a tweet since only

[2]`http://datasift.com/`

[3]`http://www.basistech.com`

5% of tweets are actually geotagged. We then improve the quality of our dataset by removing bots and spammers (by checking the tweeting frequencies, number of similar contents, etc.), and retweets.

We create a training dataset *TrainData*, using the tweets from Jun. 20, 2013 to Aug. 6, 2013, which contains a peak of infections. We created three evaluation sets using tweets from different time-periods: *TestPeriod-1* (Dec. 1, 2012 to Jul. 8, 2013), *TestPeriod-2* (Nov. 10, 2013 to Jan 26, 2014), and *TestPeriod-3* (Mar. 1, 2014 to Aug. 31, 2014). *TestPeriod-1* and *TestPeriod-2* are time periods before and after our training period in the same year (2013). We further test our models trained from 2013 on *TestPeriod-3*, which covers a complete flu season in the next year, 2014. The number of flu related tweets (containing at least one flu keyword) for these test periods are $\sim 1.8M$, $\sim 0.3M$, and $\sim 4M$ respectively. For the two individual countries used in Sec. 4.3.5 for *TestPeriod-1*, this number is 60k for Chile, and 112k for Argentina. We use tweets that occurred during the flu season in 2013 as the training set for maximizing the number of samples that are tagged as infected. We choose the three test sets as they either contain a complete flu season, or contain interesting rising patterns (detecting the rising part of the disease is one of the most challenging tasks in surveillance [29]). For creating training data we perform keyword and phrase checking (from our vocabulary) to identify a set of users who have potentially tweeted a flu-related tweet. We then fetch their tweet streams from the Twitter API for the training period. We then use the Datasift service to preprocess these tweets (stemming, lemmatization, etc.), and get our final training dataset of roughly 34,000 tweets. Under such a setting, our inference algorithm HFSTM-A-FIT takes around 2 hours to run on a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory.

We collected data from The Pan American Health Organization [121] for the ground-truth reference dataset for flu case counts (trends). PAHO is the ground-truth medical report source for South America and it plays the same role in South America as CDC does in the USA (CDC does not provide flu trend data for South America). Note that PAHO gives only per-week counts.

4.3.2 Robustness and Consistency (Q.1)

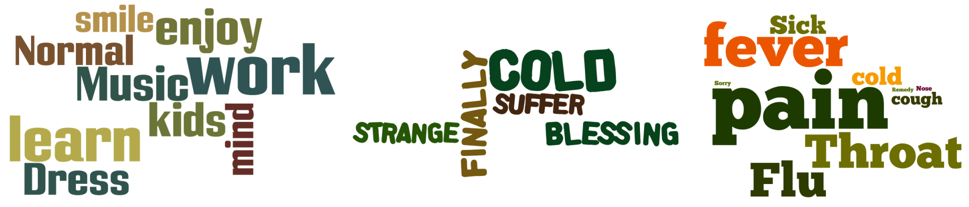
To first check the performance of our models under different conditions, we set up three kinds of simple synthetic datasets for the learners. We first choose a set of fixed parameters as base settings for generating a dataset. We then vary the background switching parameter (λ) for creating a set of datasets with different noise levels (to clarify, note that via λ we are only varying the number of background words in the dataset here, not in the vocabulary). For the third variant of datasets, we vary the number of users for generating a set of datasets. Firstly, in all the datasets, our learner was able to recover the true parameters, and show a good estimation of switching variables, transition probabilities and word distributions on these synthetic datasets. Tab. 4.2 shows the estimation error of π, η and the word distribution for each state, measured by the KL distance between the true parameter and the estimated value. Secondly we see that the performance of our models is pretty robust: it does not degrade with a substantial increase in noise level, and the learner is also stable when we increase the number of users. Finally, note that HFSTM-A learns similar quality results like HFSTM, in spite of a much enlarged vocabulary.

Expt	KL of π		KL of η		KL of ϕ_0		KL of ϕ_1		KL of ϕ_2		KL of ϕ_3	
	m1	m2	m1	m2	m1	m2	m1	m2	m1	m2	m1	m2
base	0.04	0.04	0.08	0.02	0.24	0.57	0.2	0.08	0.2	0.12	0.2	0.04
$\lambda = 0.1$	0.04	0.04	0.08	0.03	0.01	0.48	0.01	0.17	0.01	0.13	0.01	0.04
$\lambda = 0.3$	0.04	0.04	0.03	0.50	0.00	0.14	0.01	0.01	0.01	0.01	0.01	0.01
$\lambda = 0.5$	0.04	0.04	0.03	0.05	0.01	0.70	0.01	0.17	0.01	0.16	0.01	0.06
$\lambda = 0.9$	0.04	0.04	0.04	0.01	0.00	0.45	0.01	0.07	0.01	0.08	0.01	0.02
$U = 50$	0.04	0.04	0.29	0.26	0.04	0.27	0.07	0.01	0.06	0.01	0.09	0.01
$U = 70$	0.04	0.04	0.30	0.42	0.05	0.14	0.08	0.03	0.04	0.03	0.09	0.03
$U = 90$	0.04	0.04	0.08	0.01	0.02	0.52	0.03	0.07	0.03	0.02	0.03	0.01
$U = 110$	0.04	0.04	0.01	0.40	0.00	0.17	0.01	0.01	0.01	0.01	0.01	0.01
$U = 130$	0.04	0.04	0.00	0.01	0.00	0.71	0.01	0.04	0.01	0.04	0.01	0.01
$U = 150$	0.04	0.04	0.06	0.07	0.00	0.20	0.01	0.01	0.01	0.01	0.01	0.01

Table 4.2: Robustness and consistency of our models (m1 = HFSTM, and m2 = HFSTM-A) using synthetic datasets. In the 'base' setting, we use 100 users, and a vocabulary of size 92, where the number of background words, state words, and non-flu topic words are 20, 60, and 12 respectively. We vary the the number of background words (by varying λ) and the number of user from 50 to 150. It can be seen that the performance of both models do not suffer from increasing noise levels in the dataset (different from the noise in the vocabulary), and it is pretty stable when we increase the number of users in the experiments.

4.3.3 Word Distributions Learnt for Each State (Q.2)

In short, our model learns meaningful topic word distributions for the flu states from real data (*TrainData*). See Fig. 4.4—it shows a word cloud for each state-topic distribution (we renormalized each word distribution after removing the generic block-word) we learnt using HFSTM-A. Note that both HFSTM-A and HFSTM learn meaningful distributions; here we only show results from HFSTM-A since the result from HFSTM is similar. The most frequent words in each state matches well with the S(usceptible), E(xposed) and I(nfected) states in epidemiology. These word distributions in Fig. 4.4 correspond to the S, E, I states shown in Fig. 4.5. As shown in the figure, the S state has normal words, the E state starts to gather words which are indicating an exposure to the disease (and contains both 'S-like' and 'I-like' words), while the I state gets many typical flu-related words. The I state captures flu-related keywords like flu, fever, pain; the E state contains words like cold, suffer, strange; and the S state has words like enjoy, work, music, smile.



(a) S state (b) E state (c) I state

Figure 4.4: The translated word cloud for the most probable words in the S, E and I state-topic distributions as learnt by HFSTM-A on *TrainData*. Words are originally learned and inferred in Spanish; we then translate the result using Google Translate for ease of understanding. The size of the word is proportional to its probability in the corresponding topic distribution. Our model is able to tease out the differences in the word distributions between them.

4.3.4 Transition Probabilities Learned Between States (Q.3)

We show the state transition diagram *learned* from real data (*TrainData*) by HFSTM-A in Fig. 4.5. Again, HFSTM-A is as good as HFSTM, with a much larger vocabulary. The initial state probability learned is $[0.91, 0.02, 0.07]$, with high probability that a tweet starts at state S, and with much lower probabilities it starts at state E or I. We observe that for each state, it firstly has the tendency to stay in that state, which is reasonable because a twitter user is likely to post more than one tweet in any given state. When there is a transition occuring, we see that transition between S and E is larger than between E and I, showing the fact that the probability of truly getting infected is lower than the probability of just getting exposed. Interestingly, these transition probabilities match closely with the standard epidemiological SEIS model and intuition (a patient will eventually get to the I state once entering the E state).

We also investigate the most-likely state sequence for each user learned by HFSTM-A. Using the parameters learned by our model, we take a sequence of tweets from one user, and use MLE to estimate the state each tweet is in. Tab. 4.3 shows multiple examples of these transitions (we show the translated English version here using Google Translate and further refined by a native speaker) using HFSTM-A (the results are similar to that of HFSTM). As we can see, our model is powerful enough to learn the Exposed state, before the user is infected. This also shows the accuracy of our transition probabilities between the flu states.

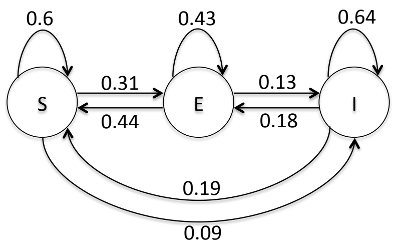


Figure 4.5: The transition diagram between flu-states automatically learned by HFSTM-A. The probabilities are rounded up for simplicity. Note that the structure of the state transitions is close to the standard epidemiological SEIS model.

4.3.5 Fitting Flu Trend using Additional State Information (Q.4)

Additionally, to test the predictive capability of our models, we design a flu-case count prediction task on our test datasets, after training on *TrainData*. We compare four models: (A) the baseline model, which uses classical linear regression techniques and word counts to predict case count numbers; our models (B) HFSTM and (C) HFSTM-A, where we improve the word counts by attaching to each word the state estimated by MLE; and (D) GFT (Google Flu Trend). In all four cases we use a LASSO based linear regression model to predict the number of cases of influenza like illnesses recorded by PAHO (the ground-truth). We predict per-weekly values as both PAHO and GFT give counts only on a weekly basis.

The baseline model uses a set of features created from the counts of 114 flu related words. For *TestPeriod-1*, we count these words over $1.8M$ tweets from $0.72M$ users that were filtered by containing at least one keyword from our vocabulary (similarly for *TestPeriod-2*, *TestPeriod-3*). These word counts were then collated into a single feature vector defined as the number of tweets containing a single word per week. We then regressed this set of counts to the PAHO case counts for each week.

Our models improve upon the baseline model by incorporating the state of the user when a word was tweeted. In this way we capture the context of a word/tweet as implied by our HFSTM and HFSTM-A models. For instance, if the word 'cold' is used in a normal conversation it probably means temperature but if it is used while a person is in the I state it is likely referring to flu related symptoms. For our models, we also use a LASSO regression to make predictions in a similar fashion. However the feature vector is created from a count of the top 20 words from each state, appended to the word of each state, such that $(cold, S)$ is counted differently from $(cold, I)$.

For GFT, we directly collect data from the Google Flu Trends website[4], and then apply the same regression as used in other methods to predict the number of infection cases. Note that as GFT is a state-of-the-art production system with highly optimized proprietary

[4]`http://www.google.org/flutrends`

User	Date	Tweet Message	State
1	4 Jul 2013	S: @ PauFigueroaentoces yes .... but by then I wouldn't like to feel like I feel now because I wouldn't be able to enjoy the vacations.	Healthy
	4 Jul 2013	I finished my first job, one more to go, and me feeling so bad... I want to rest.	Healthy
	4 Jul 2013	@ Kimy2Ramos My queen, I hope you're having a great time... because I feel terrible. I have a headache and fever =(... I love you a lot.	Exposed
	4 Jul 2013	@ PauFigueroaflu, with the flu, headache, body ache, and even my sight hurts... Couldn't ask for anything else.	Infected
	4 Jul 2013	time to studyyy...	Healthy
2	10 Jul 2013	I'm feeling like a boss for working on this by myself. I'm gonna pass, no doubt about it hahahaha.	Healthy
	10 Jul 2013	already Wednesday? Today to Aliados, how awesome.	Healthy
	11 Jul 2013	Any season is spring for me if I'm with you.	Exposed
	11 Jul 2013	It's just great how I got sick. Sad part is that I can't even miss school -.-	Exposed
	11 Jul 2013	It was so great to see a scene from Peter y Pablo, how much I missed those things.	Exposed
	11 Jul 2013	Oh, how much I hate you Tabcin. You're gross -. -	Healthy
	11 Jul 2013	Lately I've been missing those little things that made you so unique. I wonder where all those virtues went: S	Exposed
	11 Jul 2013	I'm feeling awful: fever, headache, dizziness, chest pain, snot, snot, snot and more snot and a sore throat. Am I missing something?	Infected

Table 4.3: Example user state sequences from real-world tweets (translated to English by a native Spanish speaker). We used HFSTM-A to classify tweets to different states. As we can see from the table, our model can capture the difference between different states and also the state transitions.

vocabulary lists, we do not expect to beat it consistently, yet as we describe later, we note some interesting results.

In all types of models the same LASSO regression is applied to the time series. For each time point a LASSO regression was fit to the last 10 weeks of data. The model was then used to predict either for zero, one, or two weeks in the future, depending on the lag; the best lag was chosen for each method. We evaluate all these methods for different countries (individually and aggregated) in South America on *TestPeriod-1*, *TestPeriod-2* and *TestPeriod-3*. We first discuss results on *TestPeriod-1* and *TestPeriod-2*, which are in the same year (2013) as the *TrainData*, then we show the qualitatively similar results on *TestPeriod-3*, which is in a different year (2014).

Fig. 4.6(a)–(d) show the comparison between the four models for different scenarios in

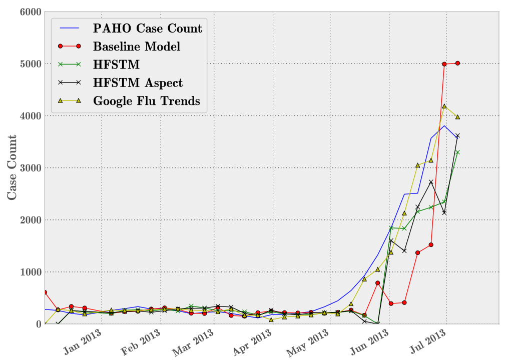
2013. Fig. 4.6(a) and (d) show the aggregated cases for all countries for *TestPeriod-1* and *TestPeriod-2*. We further expand the test cases by including two example countries: Argentina and Chile for *TestPeriod-1* in Fig. 4.6(b) and (c). We chose Argentina and Chile as they had the largest number of tweets in our dataset. We make several observations. Firstly, as expected from the previous results, the performance of HFSTM and HFSTM-A are close to each other in all cases, despite a large vocabulary difference. The RMSE values of HFSTM-A for the four plots are $501, 437, 108, 345$ respectively, and the values for HFSTM are $485, 453, 115, 350$, respectively. The difference for our methods was only about 12. Secondly, it is clear from the figures that both HFSTM and HFSTM-A outperform the baseline method (of keyword counting) in all cases—demonstrating that the state knowledge is important and our models are carefully learning that information correctly (as a contrast to the difference between HFSTM and HFSTM-A above, the RMSE value difference between HFSTM-A and the baseline for the 4 plots are about $[210, 112, 120, 80]$, respectively). Finally, we also see that the predictions from our methods are comparable qualitatively to the state-of-the-art GFT predictions, even though our methods were just implemented as a research prototype without sophisticated optimizations. In fact, although GFT performs better than HFSTM and HFSTM-A in Figures 4.6(a) and (b) in the RMSE scores, for Figures 4.6(c) and (d), our methods perform as well, and even *outperform* GFT (with an average RMSE difference of about 24). Significantly, in Figures 4.6(a), (c) and (d), GFT clearly overestimates the peak which our methods do not (this is an important issue with GFT which was also documented and observed in the context of another US flu season as well [29]).

For *TestPeriod-3* in 2014, we have similar observations. The performance of HFSTM and HFSTM-A are close to each other (with 544, 599 RMSE values). GFT, although having a better RMSE value (421), clearly *overestimates* the peak. The baseline method exhibits the worst performance with an RMSE value of 871. All of the results on our test datasets show that including the epidemiological state information of users via our models can potentially benefit the prediction of infection cases.

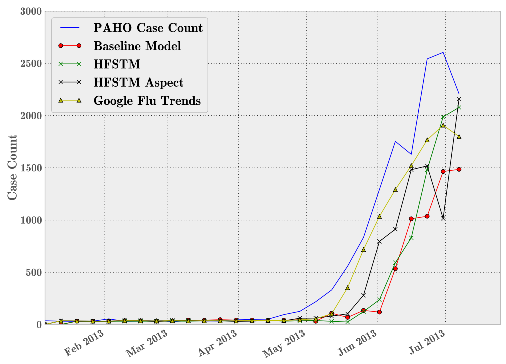
4.3.6 Bridging the Social and the Epidemiological (Q.5)

Finally, as mentioned before, another key contribution of our models is to bridge the gap between epidemiological models and social activity models. An important recent observation [111, 175] was that the fall-part of any social activity profile is power-law—in contrast to standard epidemiological models like SEIR/SIR which give an exponential drop-off. How can they be reconciled? In the following, we show that accounting for the differences in the epidemiological state as learnt by our models, the two different activity profiles look the same, i.e., they drop-off exponentially as expected from standard epidemiological models.

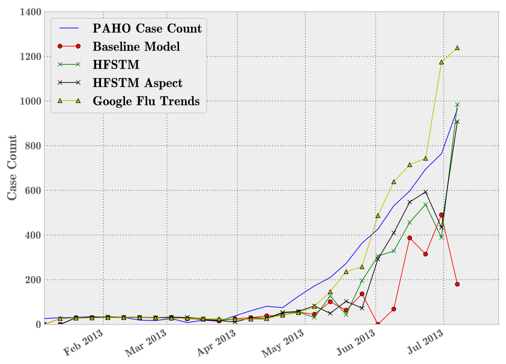
To test our hypothesis, we chose commonly occuring flu-keywords—such as enfermo (sick), mal (bad), fiebre (fever), dolor (pain)—for the analysis. Firstly, we count the total occurences of these keywords in *TestPeriod-1*. For each keyword we identify the falling part of its



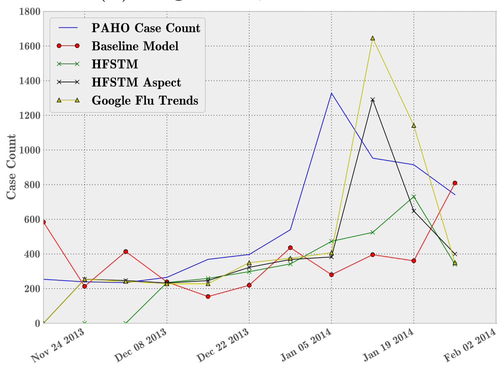
(a) All countries, *TestPeriod-1*



(b) Argentina, *TestPeriod-1*



(c) Chile, *TestPeriod-1*



(d) All countries, *TestPeriod-2*

Figure 4.6: Evaluation for the two test datatsets in 2013. Comparison of the week to week predictions against PAHO case counts using the four models: baseline model, HFSTM, HFSTM-A, and GFT (Google Flu Trend). Our models outperform the baseline; performance of HFSTM and HFSTM-A are similar, and are comparable to GFT. GFT overestimates the peak in (a), (c) and (d). (a) All countries, for *TestPeriod-1*; (b) Argentina, for *TestPeriod-1*; (c) Chile, for *TestPeriod-1*; and (d) All countries, for *TestPeriod-2*

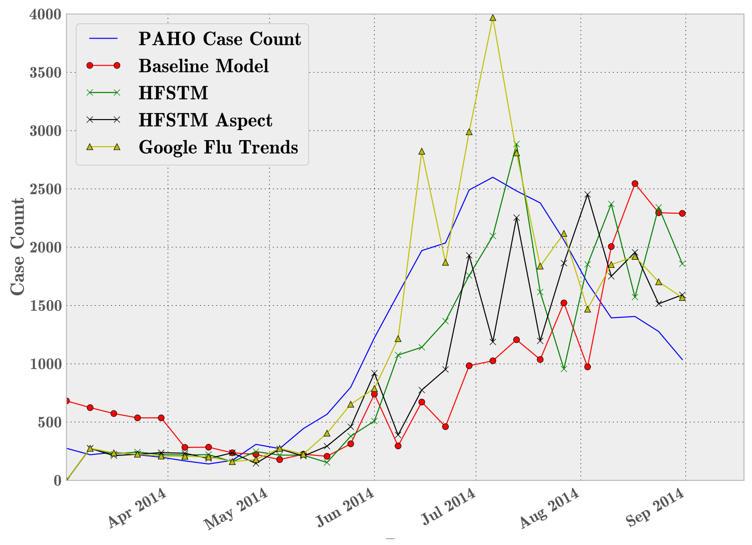


Figure 4.7: Evaluation for test dataset in 2014 (*TestPeriod-3*). Comparison of the week to week predictions against PAHO case counts using the four models. The comparison is based on all countries in the dataset. We observe that the performance of HFSTM and HFSTM-A are similar and comparable to GFT, and GFT overestimates the peak.

activity-curve. We then fit each curve with power law and exponential function. As expected from [111], Fig. 4.8 results from HFSTM and HFSTM-A (a) and (c) both show that the power-law function provides a much better fit of the falling part of the curve comparing to the exponential function (RMSE scores of exponential functions is 1.5 times higher than that of power law in both HFSTM and HFSTM-A cases).

Secondly, to study the effect of our model on the activity profiles of these keywords: we count total occurrences of these keywords in the tweets which are tweeted *only by infected* users (i.e., by those users we learn as being in I). Again, we fit the falling part of each curve with a power law and a exponential function. In contrast to the previous figure, we see that now exponential fit is much better than a power law fit; the RMSE score of power law is ~ 1.9 times higher than that of exponential functions in both HFSTM and HFSTM-A cases (see Fig. 4.8(b) and (d))—matching what we would expect from an epidemiological model like SEIS. Thus this demonstrates that finer-grained modeling can explain differences between the biological activity and the social activity which is used as its proxy.

4.3.7 Summary of Observations

In sum, the main observations from our experiments are:

1. Our models HFSTM and HFSTM-A learn both state topic distributions and transitions,

which match epidemiological intuition. The performance of HFSTM-A is robust despite an enlarged and noisy vocabulary.

2. Our models consistently get better flu *case-count* predictions than naive vocabulary assessment (the baseline model), over datasets covering multiple time-periods.
3. Our models make better flu-*peak* predictions than Google Flu Trends for the aggregated curve in both our datasets (including individual countries like Chile).
4. Our models make qualitatively comparable flu *case-count* predictions to Google Flu Trends (even beating them in some cases).
5. Our models can potentially bridge the gap between models of biological activities and their social proxies.

4.4 Discussion and Conclusion

Predicting the hidden state of a user from a sequence of tweets is highly challenging. Our proposed methods, HFSTM and HFSTM-A, have the capability to use this sparseness efficiently to produce a generative model. It satisfies the requirements of low dimensional representation of the data while retaining enough information about the system. Through extensive experiments on real tweet datasets, we showed how our methods can effectively and robustly model hidden states of a user and the associated transitions, and use it to improve flu-trend prediction, including avoiding recent errors discovered in methods like Google Flu Trends. Further, our models use public data, and our results were stable across two *different* time-periods. We also showed how our model can reconcile seemingly different behaviors from social and epidemiological models.

As mentioned in the introduction, current approaches for predicting flu using information gleaned from the Twitter data are often devoid of any epidemiological significance and hence there is a great chasm between the data driven flu trend modelling using Twitter data and the model-based, simulation-oriented epidemiological models such as SI, SIR and SEIS. Hence more broadly, our technique can act as the missing link between this uncorrelated line of research—lending a state aware nature to data-driven models, and simultaneously, it can let simulation oriented models estimate their state transition matrices by maximizing data likelihood.

4.5 Future Work

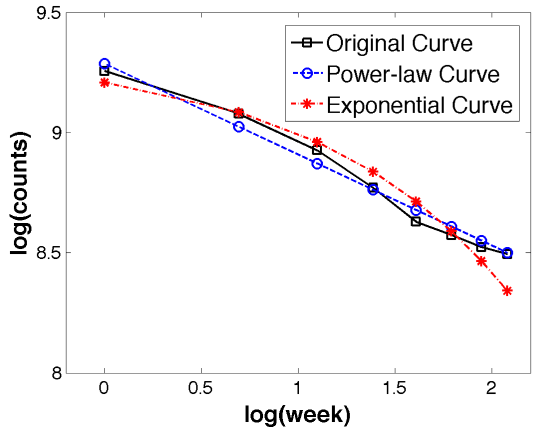
We have several directions to further extend our work.

The state transitions probabilities our models learn can be used to estimate parameters in traditional epidemiological models, such as the transmission rate, the removal rate, the infection prevalence threshold, etc. We can study how these epidemiological parameters behave in the context of Twitter, and how these social-media-derived parameters reflect on the real situation.

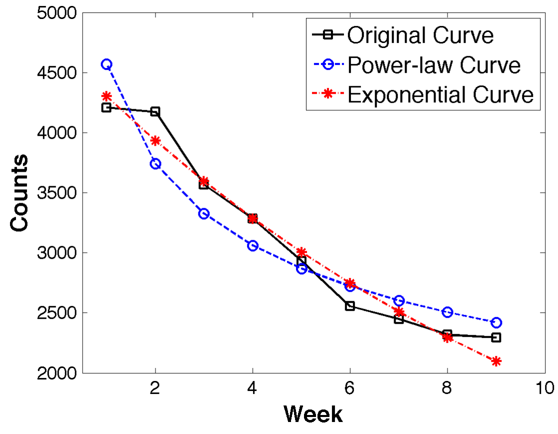
Secondly, as Twitter is a highly connected social network, we can integrate the network structure into our models and improve the results. Currently our models assume independence between Twitter users, and estimate a user's states by only looking at his/her own tweets. However in reality, people are more likely to get infected if most of their friends are infected. Hence the neighbors of a node in the network have some bias on the state transition of the node, which can be integrated into our models.

Thirdly, we can improve the efficiency of the inference algorithms. Note that our algorithms are practical enough to run on real world datasets as used in this chapter. Nevertheless it will be interesting to improve the scalability of our approach by exploring approaches like distributed EM, or other inference algorithms like MCMC.

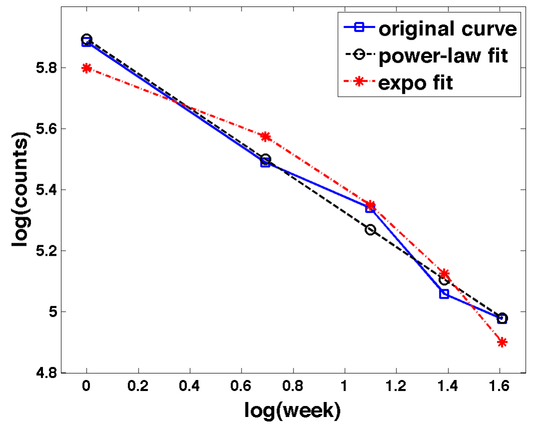
Finally, our proposed methods are general enough and can be easily extended to other domains such as monitoring organized protests where a user can go through several states of protest like 'not interested', 'ambivalent', 'active', 'resigned', etc.



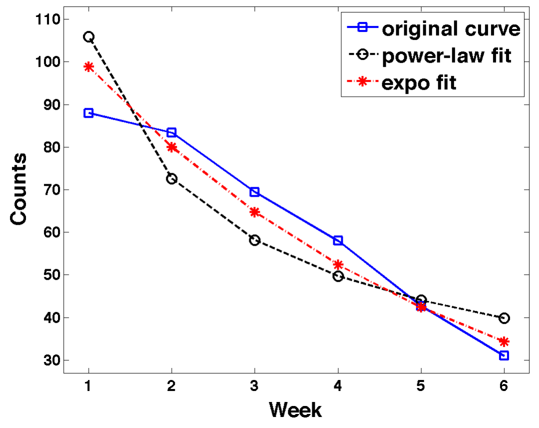
(a) Total Keyword activity (log-log) by HFSTM



(b) Keyword activity in learnt I state (lin-lin) by HFSTM



(c) Total Keyword activity (log-log) by HFSTM-A



(d) Keyword activity in learnt I state (lin-lin) by HFSTM-A

Figure 4.8: Finer grained models help bridge the gap between social and epidemiological activity models. (a), (c) Power law describes keyword activity better (in *log-log* axes to show the difference); while (b), (d) Exponential function explains well the falling part of the curves for keyword activity (note the *linear* axes). The results from HFSTM and HFSTM-A agree with each other.

Part II

Multiple Dependent Sequences

Chapter 5

Segmenting Network Sequences with Node Labels

Given a sequence of snapshots of flu propagating over a population network, can we find a segmentation, capturing when the patterns of the disease spread change, possibly due to interventions? In this chapter, we study the problem of segmenting graph sequences with labeled nodes. Memes on the Twitter network, diseases over a contact network, and movie-cascades over a social network, are all graph sequences with labeled nodes.

Most related works are on plain graphs (ignoring the label dynamics) or require much feature engineering. Instead, we propose SnapNETS, to *automatically* find segmentations of such graph sequences, with different characteristics of nodes of each label in adjacent segments. It satisfies all the desired properties (being parameter-free, comprehensive and scalable) by leveraging a principled, multi-level, and flexible framework which maps the problem to a path optimization problem over a weighted DAG.

Extensive experiments on several diverse real datasets show that it finds cut points matching ground-truth or meaningful external signals, outperforming non-trivial baselines. We also show that SnapNETS scales near-linearly with the size of the input.

5.1 Introduction

Suppose we have a sequence of Ebola infections and the associated contact network of who-can-infect-whom. Can we quickly tell a public health expert when the infection patterns change possibly due to a virus mutation? By itself, it is crucial for public health to understand the virus propagation and to design a good immunization strategy. One possible approach is to segment the sequence based on some manually selected features, such as the rate of infections. However by directly analyzing the underlying social network, and using both the

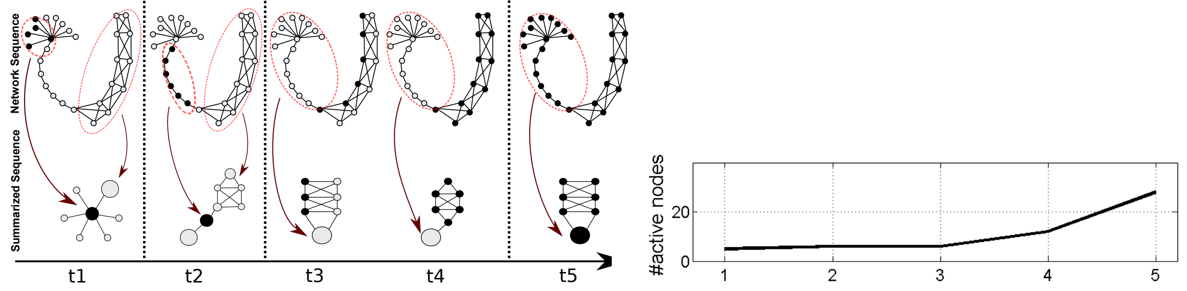


Figure 5.1: TOY EXAMPLE: SnapNETS automatically identifies four significant steps of the network sequence. The extracted time series (e.g., #active nodes) can not capture a proper segmentation. Gray nodes are inactive (i.e., label 0), and black nodes are active (i.e., label 1).

infected and uninfected nodes, we can improve the segmentation as well as its interpretability (e.g. 'disease spread in a tree like fashion among elderly till Monday, and changed to clique-like fashion among the young' and so on).

Segmenting a graph sequence is an important problem which can help us in better understanding the evolution of the dataset. It has numerous applications from epidemiology/public health to social media (rumors/memes on social networks like Twitter), anomaly detection, and cyber security (malware on computer networks). In this chapter, we study the problem of segmenting a graph sequence with varying node-label distributions. We assume binary labels and can handle dynamic graphs with varying nodes and edges. For diseases/memes, the labels can be 'infected'/'active' (1) & 'healthy'/'inactive' (0), and the network can be the underlying contact-network. Our problem is:

PROBLEM 1: SEGMENTATION

***Given*:** a sequence $\mathcal{G}$ of networks $G_1, G_2, \ldots, G_T$ with labeled nodes,

***Find*:** best segmentation c^*, which captures different patterns of node labels in $\mathcal{G}$ such that adjacent segments have different characteristics of nodes with the same label.

TOY EXAMPLE. Suppose $\mathcal{G} = \{G_1, G_2, G_3, G_4, G_5\}$ with 0, 1 labeled nodes (Fig. 5.1 top row). There are four main steps in $\mathcal{G}$: First a central node in a star and some of its spokes have the label 1; next, low degree nodes in a chain-shaped manner get label 1 (structural change). In the third segment, the label moves to another community of the graph (community change). Finally, the whole graph gets label 1 which indicates an activation rate increase in the network (rate changes). Hence $S^* = \{1, 2, 3, 5\}$. Note that even though the 'active' sub-graphs in time-step 2 and 3 are both chains, their roles in the entire graph are different and so they should belong in different segments. In time-step 2 the active chain is a bridge between two parts while the chain in time-step 3 is part of a near-clique community (role change).

Approaches / Properties	**SNAPNETS**	**Feature eng. and time series** (E.g. Li et al. 2009, Likas et al. 2003, Henderson et al. 2010)	**Plain-graph-based** (E.g. Shah et al. 2015, Koutra et al. 2014, Ferlez et al. 2008, Qu et al. 2014)
Parameter free	✓	✗	✗
Comprehensive	✓	✓	✗
Scalable	✓	✗	✓

Table 5.1: Comparison of SnapNETS with alternative approaches. A dashed cross means most approaches do not satisfy the property; similarly for the dashed check.

Any algorithm should have these desired properties:

P1. Parameter-free: Find the best number of segments and segmentation without use of parameters such as change threshold and time window.
P2. Comprehensive: Use the entire snapshot for segmentation, instead of merely active subgraphs.
P3. Scalable: The method must be scalable (i.e., scales sub-quadratically with the input size which can be millions of edges and nodes in the sequence).

This problem has been barely (if at all) studied. Most methods that we can adapt to solve this problem do not satisfy the above three properties—instead we propose SnapNETS (**Snap**ping **NET**work **S**equences) which does (Tab. 5.1 shows a brief comparison; more discussions in Sec. 5.2.1). SnapNETS is a novel multi-level approach which summarizes the given networks/labels in a very *general way* at *multiple different time-granularities*, and then converts the problem into an appropriate *optimization problem* on a data structure. We give a novel efficient algorithm for the optimization problem as well. A strong advantage of this framework is that it allows us to *automatically* find the right number of segments avoiding over or under segmentation in a very systematic and intuitive fashion. Further it gives naturally interpretable segments, enhancing its applicability. Finally, we also demonstrate SnapNETS's usefulness via multiple experiments on diverse real datasets.

5.2 Overview and Main Ideas

For sake of simplicity, we focus on the case when nodes have binary labels[1], i.e., active: 1 or inactive: 0. Also, for ease of description, we assume that the network remains constant through time and we treat the problem as one with a series of graph snapshots: though our

[1]Extending to multiple labels is interesting future work.

ideas can be easily used for other types of dynamic graphs, including with varying network structure (also shown in experiments).

Finally, we allow that nodes can even *switch* between labels freely (c.f. Fig. 5.1). This means we can handle both progressive/non-progressive scenarios: e.g., in the fundamental **S**usceptible-**I**nfected (SI) or **I**ndependent **C**ascade (IC) propagation models (where nodes once active, can not get inactive) and the 'flu-like' **S**usceptible-**I**nfected-**S**usceptible (SIS) model where infected nodes can get healthy again. Next we give some useful definitions:

Definition 5.1 (*Act-snapshot*) *$G(V, E, \mathbf{L})$ is an Act-snapshot. $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$ are sets of nodes and edges of G. $\mathbf{L} = [l_1, l_2, \ldots, l_n]$ shows the labels of nodes. l_j is 1 if v_j is active and 0 otherwise.*

Definition 5.2 (*AS-Sequence*) *$\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ is sequence of T Act-snapshots with G_i at time-step i.*

Definition 5.3 (Segment) *A segment $s_{i,j}$ is a time interval between Act-snapshots G_i and G_j, i.e., $s_{i,j} = \{[i, j) \mid i < j\}$. Set of all possible segments is $\mathcal{S} = \{s_{1,2}, s_{1,3}, \ldots\}$.*

Definition 5.4 (Segmentation) *A segmentation S of size m is a partition of time interval $[1, T]$ with m time stamps i.e. $S = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in \{1, 2, \ldots, T\}$. The set of all possible segmentations is $\mathcal{C}$.*

Definition 5.5 (*Act-set$_i$*) *Act-set$_i$ contains the active nodes in Act-snapshot G_i, i.e., Act-set$_i$ = $\{v_j | l_j = 1\}$.*

Hence our problem is to automatically segment a given *AS-Sequence*. We next explain the shortcomings of alternative approaches, and then give the big picture of our framework.

5.2.1 Shortcomings of Alternative Approaches

Two natural ways to adapt existing algorithms for this task are: (a) extract complex features from *Act-snapshots* and use time-series segmentation; and (b) extract *Act-set*s and use plain-graph-based methods.

Feature Eng. and Time Series. Converting graph sequences to time series has several drawbacks. First of all, it needs laborious feature-engineering: designing the right features to capture the pattern of graphs is a complicated task and the best choice of features may differ for different sequences [63]. Second, typical time series segmentation algorithms, which use "local" change detection, do not satisfy our desired properties. They usually need a threshold [101] (which usually depends on knowing the number of desired segments) to detect a change; or they fix *one* aggregation time period for the tracking [98]. All of these can be problematic, as it is fundamentally hard to set these parameters.

Plain-graph-based analysis. Instead of manually designing complicated features, an alternative is to use plain-graph-based methods on induced subgraphs from *Act-snapshots*. However, these approaches do not satisfy **P2**, as they typically track only the *Act-set* in each *Act-snapshot* [147, 84, 135]. As we show in Fig. 5.1, using only the active sub-graphs leads to less meaningful segmentation: the active sub-graphs at time-step 2 and 3 are both a chain of a same size, nevertheless, as discussed before the roles of these chains are different in the two snapshots. If we just track active sub-graphs, we cannot detect this difference.

5.2.2 Overview of our Method SnapNETS

In order to overcome the disadvantages we discussed before, we propose a "global" framework which looks at the entire *AS-Sequence* $\mathcal{G}$ and computes the correct segmentation. Due to **P1**, we want to examine all possible segmentations $\mathcal{C}$ over all granularities. How to do this efficiently? Our first main idea is to use a graph data structure (called the *segmentation graph* $\mathsf{G_s}$) to efficiently represent the exponential number of all segmentations in space polynomial with respect to the sequence length. The nodes mainly represent the segments in $\mathcal{S}$, while the edge weights indicate the distance ('difference') between adjacent segments. Hence any segmentation is mapped to a path between start and end time in $\mathsf{G_s}$.

How to now compute the distance between any adjacent segments $w(s_{i,j}, s_{j,k})$ (each segment will contain sets of *Act-snapshots* G_i)? We want to use the entire graph (due to **P2**), while avoiding extracting complex features. Note that despite the size of the graphs, patterns in the real-world are usually much less complex. Hence, our second main idea is to develop a *smaller* summary G_i^c which maintains important information in an efficient manner. As a result, we only need a few standard features to represent these summaries.

Finally, how to find the best path in $\mathsf{G_s}$? We need to define this best path and design an efficient algorithm to find it in $\mathsf{G_s}$. Our third main idea is to use the average longest path optimization problem on $\mathsf{G_s}$, as it intuitively regularizes the length of the path (number of segments) with the weight (difference between segments). We also develop an efficient novel algorithm DAG-ALP to find this path.

In short, we pursue 3 main goals: (1) *Summarize* G_i; (2) *Construct* $\mathsf{G_s}$ and (3) *Define and find the best segmentation.*

5.3 SnapNETs: Details

5.3.1 Goal 1: Summarizing Act-snapshots

We first propose finding a *C-graph* (i.e., G_i^c), which summarizes the structural properties and the nodes labels of each *Act-snapshot* G_i. Popular methods for graph summarization include graph sparsification [107] which try to carefully remove edges to reduce the graph's density while maintaining some properties. Nevertheless, these methods are typically designed for plain graphs and it is not straightforward to modify them for *Act-snapshots*. So we adopt a different, *merging-based* approach which reduces the *no. of nodes* instead while maintaining an intuitive and important property.

Role of Eigenvalues: In many real datasets node labels come from a diffusion/propagation process. Recent work [131] shows that important diffusion characteristics of a graph (including the so-called 'epidemic threshold') are captured by the leading eigenvalue of the adjacency matrix, for *almost all* cascade models. This naturally suggests that if the leading eigenvalue of the adjacency matrix of the summarized graph G_i^c and *Act-snapshot* are close, G_i and G_i^c will have similar properties.

Summarizing Act-snapshots via Coarsening: Motivated by the above, we want to successively merge connected nodes into 'super-nodes' (i.e., 'coarsen') while maintaining the leading eigenvalue of the adjacency matrix. Also, we want to keep the same set of labels (0/1) in the *C-graph* to keep it consistent with the *Act-snapshot*. Thus, we define the summarization problem as follows,

PROBLEM 2: *Act-snapshot* SUMMARIZATION

Given: an *Act-snapshot* $G_i(V_i, E_i, L_i)$, and remained fraction of nodes ρ.

Find: a coarsened graph $G_i^c(V_i^c, E_i^c, L_i^c)$ such that

$$\underset{|V_i^c|=\rho|V_i|}{minimizes} \; |\lambda_{G_i} - \lambda_{G_i^c}| \;\; subject\ to \;\; \textstyle\sum_{(a,b)\in E_i is\ merged} |l_a - l_b| = 0 \tag{5.1}$$

Here λ_G is the leading eigenvalue of graph G and l_a is the label of node a. This formulation allows us to be model-free and not assume any specific model (such as IC/SIS, etc.). The constraint in PROBLEM 2 maintains the 'frontier' between active and inactive nodes to help consistency and interpretability. PROBLEM 2 is similar to the graph coarsening problem (GCP [132]) whose goal is to maintain just λ_G, but without any constraint—they give an efficient algorithm for this purpose which merges edges based on a quality score. Hence, we modify that algorithm by not allowing merging of node-pairs with different labels. This works very well in practice and gives near-linear running time. Note that a better algorithm for PROBLEM 2 will only improve our results. We use the same amount of coarsening ($\rho = 0.1$) as in [132].

Type	ID	Name
Structural	f_1	Largest eigenvalue of the adjacency matrix
	f_2	Number of edges
	f_3	Entropy of the edge weight distribution
	f_4	Average clustering coefficient
Label based	f_5	Number of active nodes
	f_6	Average PageRank of active nodes
	f_7	Average degree of active nodes
	f_8	Average degree of active neighbors of active nodes

Table 5.2: Features extracted to represent each summarized *Act-snapshot* (i.e., *C-graph*).

Fig. 5.1 shows our summaries via Problem 2 for the Toy Example: The *C-graphs* clearly show the important non-trivial pattern changes in both the structural and label properties of the original graphs succinctly.

5.3.2 Goal 2: Constructing the Segmentation Graph

After summarizing the *Act-snapshots*, each segment in the *AS-Sequence* contains a set of *C-graphs*. How to find the distance between two such segments? In general, computing distances between *unlabeled* graphs is itself a challenging problem [84]. Fortunately, in our case, we can just extract *simple* features from the *C-graphs* due to their small size and complexity, and use them to compute the distance. Subsequently, we build the segmentation graph $\mathsf{G_s}$ to store the segments and distances information. Recall that $\mathsf{G_s}$ can efficiently represent all the exponential number of possible segmentations in polynomial space.

Feature extraction of C-graphs

Extracting features from G_i^c is much more efficient primarily because of their smaller size. Further our summarization maintains the relevant important properties effectively. So we do not need complex features such as "number of particular substructures" (e.g., stars, maximal cliques, ladders, etc.) used in related work.

We extracted multiple *standard* features [96] and eliminate correlated ones to get eight features for each G_i^c (see Tab. 5.2 for a description). Feature vector $\mathbf{F}_i$ contains: *Structural* features (f_1-f_4), and *Label dependent* features (f_5-f_8) (label-dependent properties). Finally, we normalize them by range normalization for a meaningful comparison between the features [96]. Thanks to our careful design, we can use very simple features for our task.

Segmentation graph We now describe how to construct $\mathsf{G_s}$ to compactly store and represent segmentations. $\mathsf{G_s}(V_s, E_s)$ is a unique weighted DAG where:

Nodes (V_s): For each segment $s_{i,j} \in \mathcal{S}$, there is one node in the graph $\mathsf{G_s}$. We add two extra nodes to the graph: a source node s and a target node t. Therefore, $V_s = \mathcal{S} \cup \{s, t\}$.

Edges (E_s): There is a directed edge from node $s_{i,j}$ to any node $s_{j,k}$. Also, the source node s links to all nodes with starting time stamp 1 and all nodes with ending time stamp t_{max} links to the target node t. Hence, $E_s = \{e(s_{i,j}, s_{j,k})\} \cup \{e(s, s_{i,j})|i = 1\} \cup \{e(s_{i,j}, t)|j = t_{max}\}$.

Edge Weights ($w(e)$): The weight of all edges from s or to t are zero. The weight of an edge from $s_{i,j}$ to $s_{j,k}$ is equal to the distance between sets of *C-graphs* in their corresponding segments, i.e., $w(e(s_{i,j}, s_{j,k})) = d(s_{i,j}, s_{j,k})$.

How to get this distance? Using the $\mathbf{F}_i$ for each G_i^c, we compute the average feature vector over all the *C-graphs* in a segment as the segment's representative, i.e., $\widehat{\mathbf{F}}_{s_{i,j}} = \frac{\sum_{a=i}^{j} \mathbf{F}_a}{(j-i+1)}$, where $\mathbf{F}_a$ is the feature vector of G_a^c in $s_{i,j}$.

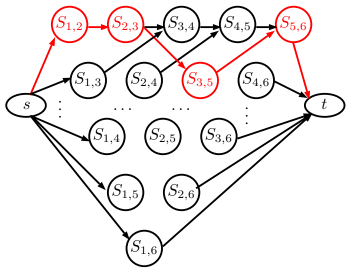


Figure 5.2: $\mathsf{G_s}$

This representation has a natural interpretation as it captures the average 'pattern' of *C-graphs* of the segment. Then the distance $d(s_{i,j}, s_{j,k})$ between '$s_{i,j}$' and '$s_{j,k}$' can be defined as $d(s_{i,j}, s_{j,k}) = ||\widehat{\mathbf{F}}_{s_{i,j}} - \widehat{\mathbf{F}}_{s_{j,k}}||_2$.

Fig. 5.2 shows the $\mathsf{G_s}$ for our Toy Example. Edge weights are not shown for clarity. Note that $\mathsf{G_s}$ is a DAG since its edges are directed and there is no cycle in it (as we cannot go back in time). Also, we need to compute the summary just once for each G_i, *not* for each segment in $\mathsf{G_s}$. We can compute the distance for every edge in $\mathsf{G_s}$ independently. Hence, we summarize *Act-snapshots* and construct the segmentation graph in *parallel.*

5.3.3 Goal 3: Finding the Best Segmentation

Let $\mathcal{P}$ be the set of all paths in $\mathsf{G_s}$ from s to t. Then,

Lemma 1 *Each path $p \in \mathcal{P}$ corresponds to a valid segmentation $S \in \mathcal{C}$ and for each $S \in \mathcal{C}$ there is a path $p \in \mathcal{P}$.*

Hence to get the best segmentation, we only need to *define* and *find* the best path in $\mathsf{G_s}$, which we discuss next.

Average longest path Note that defining the best path is a different and independent question to that of defining edge weights. We define the best segmentation as follows:

Problem 3: Finding the best segmentation

***Given*:** a segmentation graph $\mathsf{G_s}$

Find: the average longest path from s to t in $\mathsf{G_s}$, i.e.

$$S^* = \arg\max_{S \in \mathcal{P}} \frac{\sum\limits_{s_{i,j}, s_{j,k} \in \mathcal{S}} w(e(s_{i,j}, s_{j,k}))}{|S|} \tag{5.2}$$

Thus, PROBLEM 3 is the *Average*-Longest Path (ALP) problem on $\mathsf{G_s}$ (restricted to the path set $\mathcal{P}$). ALP defines the path (segmentation) quality as the average value of edge weights in the path (distance between its segments). Note that ALP is parameter-free and importantly, it also naturally *balances* the 'length' (weight) of the path (difference between segments) with # nodes in the path (# segments).

An alternative 'parameter-free' optimization would have been the Longest Path (LP) problem: which will try to find the *longest* (heaviest) path in $\mathcal{P}$ (Eq. 5.2, without the denominator). However, the LP formulation will suffer from *over-segmentation*—it is biased by the number of segments in the path, in the sense that it tends to prefer longer paths with more nodes, irrespective of the edge weights [166]. In practice our observations confirm that LP contains unnecessary edges with low weight (see Sec. 5.4). Our ALP objective is intuitive and overcomes the disadvantage of LP. Fig. 5.2 shows the ALP for TOY EXAMPLE in red.

DAG-ALP

ALP can be solved in poly. time on DAGs (recall $\mathsf{G_s}$ is a DAG)[2]. Current state-of-the-art algorithms [166] can solve PROBLEM 3 in $O(V_s^2.E_s)$. This is too slow for our purposes; hence, we propose a new and more efficient $O(E_s)$ algorithm for ALP on DAGs called DAG-ALP.

The main observation we use is that the ALP from s to t is the *longest* ('heaviest') path among all paths with the same number of nodes (the 'length') as the ALP. This fact leads us to calculate all the heaviest paths with different lengths in $\mathcal{P}$ and find the one which gives the maximum average edge weight. In DAG-ALP we build a queue of layers of $\mathsf{G_s}$. Each layer i contains nodes which can reach t by i steps. When we iterate through layers, we maintain the weight ($P_i(v,t)$) of the heaviest path from v to t in i steps, and the next node of v in this path (κ_v^i). Alg. 10 shows the pseudo-code of DAG-ALP.

Lemma 2 *The DAG-ALP algorithm correctly finds the average longest path* $\mathsf{G_s}$.

Lemma 3 *Time complexity of DAG-ALP is $O(|E_s|)$, where $|E_s|$ is number of edges in* $\mathsf{G_s}$.

5.3.4 The Complete Algorithm

Alg. 5 shows the final pseudo code of SnapNETS.

[2]ALP is NP-hard on general graphs.

Algorithm 5 SnapNETS

```
for each $G_i \in \mathcal{G}$, coarsen and get $G_i^c$ once (Sec. 5.3.1) do
    Compute feature $\widehat{\mathbf{F}}$ vector of segments in $\mathcal{S}$ (Sec. 5.3.2)
Generate the segmentation graph $\mathsf{G_s}$ (Sec. 5.3.2)
$S^* = $ DAG-ALP ($\mathsf{G_s}$, $\mathcal{G}$, $s$, $t$ ) (Sec. 5.3.3)
```

Algorithm 6 DAG-ALP

```
Require: $\mathsf{G_s}$, $\mathcal{G}$, $s$, $t$
Ensure: $P_{alp}$
 1: Initialize Queue
 2: $Layer_0 = \{t\}$ and Queue.push($Layer_0$)
 3: $Layer_1 = \{s_{i,j} | j = t_{max}\}$
 4: Queue.push($Layer_1$)
 5: for k = 2 to $t_{max}$ do
 6:     $Layer_k = \{s_{i,j} | T - j + 1 \geq k\} \cup \{s\}$
        Queue.push($Layer_k$)
 7: $Layer_{t_{max}+1} = s$ and Queue.push($Layer_{t_{max}+1}$)
 8: LL = Queue.pop(), $\kappa_t^0 = \varnothing$
 9: while Queue is not empty do
10:     CL = Queue.pop()
11:     for $s_{i,j} \in$ CL do
12:         $\kappa_{s_{i,j}}^{CL} = \arg\max_{s_{j,k}} P_{LL}(s_{j,k}, t) + w(e(s_{i,j}, s_{j,k}))$
13:         $P_{CL}(s_{i,j}, t) = P_{LL}(\kappa_{s_{i,j}}^{CL}, t) + w(e(s_{i,j}, \kappa_{s_{i,j}}^{CL}))$
14:     If $s$ is in CL then $lp_{CL} = P_{CL}(s, t)$
15:     LL = CL
16: $ALP = \arg\max(\frac{lp_1}{1}, \ldots, \frac{lp_{t_{max}}}{t_{max}})$
17: Extract the $P_{alp}$ using $\kappa_s^{t_{max}}$ to $\kappa_t^0$
```

Lemma 4 *SnapNETS has sub-quadratic time complexity $O(E \cdot \log E + \frac{E}{p})$, where E is the total number of edges in $\mathcal{G}$ and p is number of processors to parallelize constructing the segmentation graph $\mathsf{G_s}$.*

5.4 Experiments

We design various experiments to evaluate SnapNETS. We implemented SnapNETS in MATLAB and Python. Our experiments were conducted on a 4 Xeon E7-4850 CPU with $512GB$ of $1066Mhz$ main memory.

Datasets. We collected a number of datasets from various domains such as social and news media, epidemiology, autonomous system, and co-authorship network to evaluate SnapNETS.

See Tab. 5.3 for a summary description.

The ground truth segmentations in these datasets are non-trivial. They are induced from complex structural changes such as activation in different parts of the *Act-snapshots* (*AS Oregon*[3] and *Higgs* [47]), and varying role of active nodes, e.g., change of active nodes centrality (*BA-degree*), or activation rate changes (*Portland*[4]). Moreover, detecting the number of correct cut points is also a difficult task itself. In datasets without ground truth, we would like to find how memes/news were adopted by social media users (*Twitter* and *Memetracker* [95]) and when the co-authorship relation on a specific topic changes over time (*DBLP* [90]).

Dataset	#Nodes	#Edges	Timesteps	GT
BA-degree	500	4,900	240 units	✓
AS-PA	633	1,086	400 units	✓
AS-COM	4431	7,609	530 units	✓
AS-MIX	1899	3261	1000 units	✓
Higgs	456,626	14,855,843	7 days	✓
Portland	1,575,861	19,481,626	25 days	✓
Memetracker	960	5,001	165 days	
Twitter	126,915	5,589,083	30 days	
DBLP	369,855	1,109,452	13 years	

Table 5.3: Datasets details. (GT = Ground Truth)

Baselines. To the best of our knowledge, there is no existing algorithm which has all the desired properties. Hence, we adapt two time series based algorithms, and one plain-graph-based algorithm, as baselines. We show the details in Tab. 5.4.

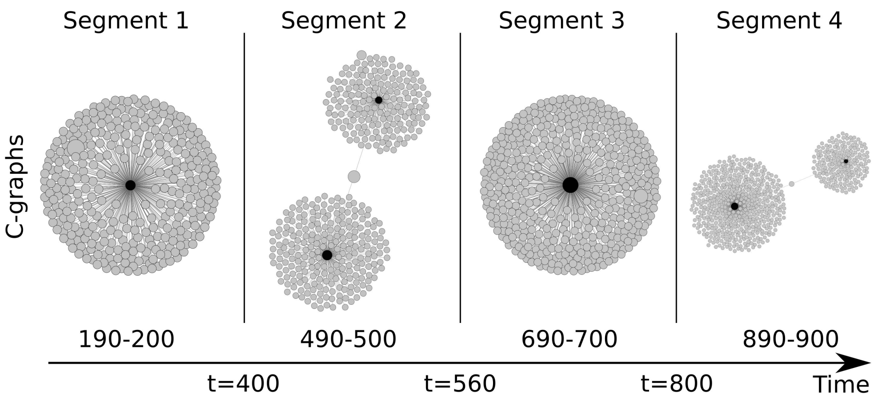
Variations: We also designed the following three variations of SnapNETS for comparison. *(1)* SN-Orig extracts features from *Act-snapshots*. *(2)* SN-LP finds the **L**ongest **P**ath instead of ALP. *(3)* SN-Greedy greedily selects edges with the largest weight from s to t, instead of ALP.

Metrics. For datasets with ground truth, we measure the performance by calculating the F_1 score of the set of detected cut-points with the ground-truth set. For others, we perform case studies. We show how SnapNETS reveals interesting patterns by matching the results with external news/events, and show how they are easily interpretable.

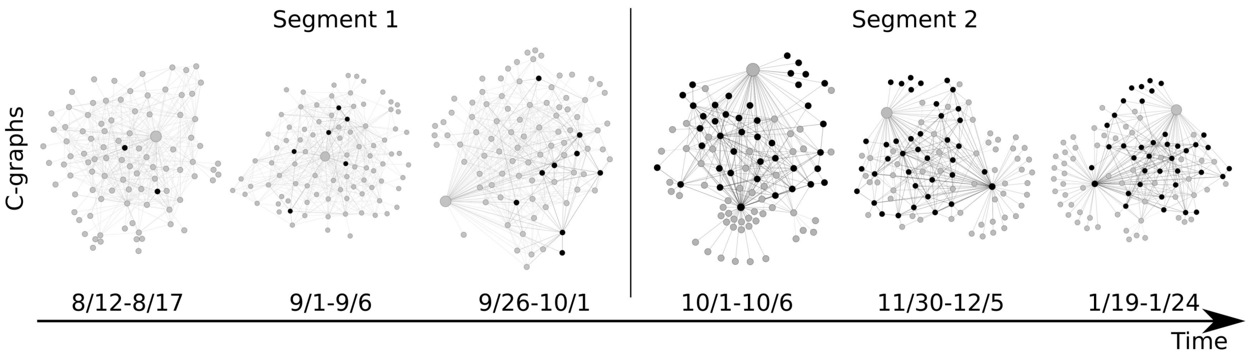
5.4.1 Segmentation Results

We give representative results here.

[3]http://topology.eecs.umich.edu/data.html
[4]http://ndssl.vbi.vt.edu/synthetic-data/



(a) *AS-MIX*



(b) *Memetracker*

Figure 5.3: Visualization of *C-graphs* for the segmentations found by SnapNETS for *AS-MIX* and *Memetracker*. The vertical lines are the detected cut points. Black nodes are active and gray ones are inactive.

Baseline	Description
Dynammo [98]	Construct time series using features in Tab. 5.2, then feed the time series and the no. of cut points (detected by SnapNETS) to *Dynammo*, and use reconstruction errors to find change points.
K-MEANS (Likas et al. 2003)	Construct time series similarly as in *Dynammo*, then it finds segmentations based on an online "local" approach, and reports a segment when a new cluster is detected.
VOG [84]	Extract active *sub-graph* (VOG does not work on labeled graphs) and use VOG to find the 10 most important sub-structures. When the set of important sub-structures changes sufficiently (above a threshold which is set to be the one with the best performance), we output a segment.

Table 5.4: Baselines description

Quantitative analysis

Data w. GT	F_1 score						
	SnapNETS	SN-ORIG	SN-LP	SN-GREEDY	*Dynammo*	K-MEANS	VOG
BA-degree	**1**	**1**	0.08	**1**	0	0.4	0.67
AS-PA	**1**	0	0.05	0.67	0	0.4	**1**
AS-COM	**0.67**	0	0.07	0.5	0.5	0.22	0
AS-MIX	**0.86**	0	0.07	0.57	0.32	0.4	0
Higgs	**1**	0	0.15	**1**	0	0.67	0
Portland	**1**	0	0.4	**1**	**1**	0.67	**1**

Table 5.5: F_1 score of the segmentation detected by SnapNETS, variations, and baselines on datasets with GT.

We see in Tab. 5.5 that SnapNETS has the best performance among baselines and variations of SnapNETS. Note that all the baselines require some input parameters: such as the number of cut points (*Dynammo*), threshold for new cluster creation (K-MEANS), and difference threshold for outputting a cut point (VOG). We test a wide range of these input values and pick the ones that give the best results. Still, their performance is clearly worse.

***AS Oregon*:** SnapNETS performs very well to differentiate between random and preferential attachment style activation (*AS-PA*) and we can accurately detect when different communities of the network get active (*AS-COM* and *AS-MIX*). Fig. 5.3(a) visualizes the *C-graphs* in the S^*, and shows that our results are easily interpretable.

***Higgs*:** SnapNETS finds the exact ground truth Jul. 4 ($F_1 = 1$). Note that according to external news, there were rumors about the Higgs boson discovery from Jul. 2-4; these rumors make the ground truth harder to detect (for example VOG finds a cut point on Jul. 2 instead of Jul. 4).

***Portland*:** SnapNETS again detects the ground truth segments ($F_1 = 1$). Other baselines have worse performance except VOG and *Dynammo* since the change of the infection pattern in this dataset is visible in the active subgraphs: an infected chain first, and then many infected stars (as the disease goes viral). SnapNETS shows its power in finding the pattern change in disease propagation.

Case studies

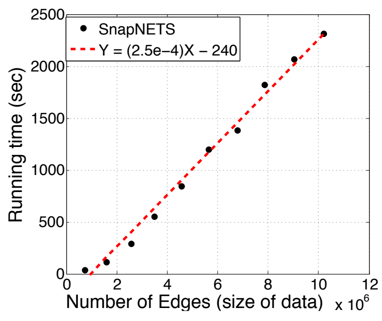
SnapNETS finds meaningful segments in multiple datasets from various domains with varied patterns of evolution (both structurally and in labels) while none of the baselines perform as well (for example, VOG finds no cut-point in *DBLP*, K-MEANS essentially gives one at the end, and *Dynammo* finds no cut-point in *Memetracker*).

***Memetracker*:** SnapNETS finds a cut point on Oct. 1, 2008, which matches the date of the televised debate between Joe Biden and Sarah Palin. In the first segment, *C-graphs* (Fig. 5.3(b)) are close to the case when all nodes have the same label—suggesting that few nodes randomly got infected ($f_5 \simeq 0.1$, $f_8 \simeq 0.2$). In the second segment, *C-graphs* are substantially sparser (i.e., f_2 dramatically dropped to 0.02) and contain large stars and leaf nodes with active centers. The size of the centers and average PageRank (f_6) in the *C-graphs* shows that important nodes such as "CNN" and "BBC" websites spread the meme to many nodes, and they are merged to form hubs.

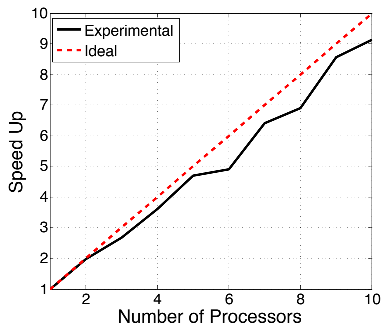
***Twitter*:** We detect two cut points at Jun. 14 (presidential election) and Jun. 27 (vote recount). In the first segment, multiple small *near-clique* structures (high $f_1 \simeq 0.8$ and low $f_2 \simeq 0.5$) of *C-graphs* shows small and highly connected groups of political activists were active. In the second segment, important nodes such as "NYtimes" in different areas of the graph became active and formed multiple large stars of active nodes in *C-graphs* (low $f_1 \simeq 0.1$ and high $f_6 \simeq 0.7$). Finally, *C-graphs* became densely connected ($f_1 \simeq 0.85$, $f_2 \simeq 0.8$) because the remaining hub and bridge nodes became active and merged, while a few small/sparse subgraphs remained inactive.

5.4.2 Scalability

We performed scalability tests, and as expected SnapNETS scales near-linearly w.r.t. the no. of edges in $\mathcal{G}$, and the running time is reasonable and practical—it is ~ 10 *times* faster than dynamic graph analysis methods such as [147]. Also, we get excellent speed-ups ($\sim$ 9X faster using 10 processors) after parallelization.



(a)



(b)

Figure 5.4: (a) Scalability of SnapNETS. (b) Speedup by parallelizing construction of $\mathsf{G_s}$.

5.5 Discussion and Conclusions

We presented SnapNETS, an intuitive and effective method to segment *AS-Sequences* with binary node labels. It is the *first* method to satisfy all the desired properties **P1**, **P2**, **P3**. It efficiently finds high-quality segmentations, detects anomalies, and gives useful insights in diverse complex datasets.

Patterns it finds: In short, SnapNETS finds segmentations where adjacent segments have different characteristics of nodes with the same label, i.e., the 'placement' and 'connection' of active/inactive nodes are different. This includes both structural (e.g., community/role/centrality) and rate changes. As a non-trivial example, we can detect both a random-vs-targeted activation and a faster-vs-slower one.

Global method: It is useful to note that SnapNETS is a 'global' method and not simply a change-point detection method. We are not just looking for local changes; rather we track the 'total variation' using $\mathsf{G_s}$. Hence this allows to find important cut-points automatically and without any specification.

Flexibility: The SnapNETS framework is very flexible, as our formulations are very general. The eigenvalue characterization is general; similarly, the $\mathsf{G_s}$-ALP formulation should be also useful for other segmentation-like problems; and DAG-ALP can be of independent interest too. Adapting SnapNETS to handle dynamic graphs with varying nodes and edges is useful as the next step. Also, extending our work to streaming and partially observed graphs, and handling more general node/edge level features, will be interesting.

Chapter 6

Segmentation with Explanation for Multi-Dimensional Time Series

Recent hurricane events have caused unprecedented amounts of damage and severely threatened our public safety and economy. The most observable (and severe) impact of these hurricanes is the loss of electric power in many regions, which causes the breakdown of many public services. Understanding the power outages and how they evolve during a hurricane provides insights on how to reduce outages in the future, and how to improve the robustness of the underlying critical infrastructure systems.

In this chapter, we propose a novel segmentation with explanations framework to help experts understand such datasets. Our method, CUT-n-REVEAL, first finds a segmentation of the outage data to capture pattern changes in the sequences. We then propose a novel explanation optimization problem to find an explanation of the segmentation, that highlights the *culprits* of the changes. Via extensive experiments, we show that our method performs consistently in multiple datasets with ground truth. We further study real county-level power outage data from several recent hurricanes (Matthew, Harvey, Irma) and show that CUT-n-REVEAL recovers important, nontrivial and actionable patterns for domain experts.

6.1 Introduction

Power outages during several recent hurricanes have caused severe impact on our national security, economy and public safety. The 2017 hurricane season was the most expensive in U.S. history resulting in huge economic losses (greater than $250 billion). Hurricane Irma caused one of the largest power outages which reportedly knocked out power to 4.5 million of Florida Power & Light's 4.9 million customers. Hence, better understanding the power outages and how they evolve during the hurricanes is a very important task for damage prevention and control.

Domain experts in critical infrastructure systems (CIS) constantly seek solutions and ideas on how to reduce the power outages during hurricanes. For example, Oak Ridge National Laboratory's (ORNL) Energy Awareness and Resiliency Standardized Services (EARSS) project developed a fully automated procedure to take wind speed and location estimates provided by hurricane forecasters and provide a geospatial estimate on the impact to the electric grid in terms of outage areas and projected duration of outages [15]. There are many such examples, including the National Infrastructure Simulation and Analysis Center (NISAC) program[1] (which provides projected outages) and ANL's HEADOUT model which quickly estimates potential customers who will lose power.

Although crucial, it is often non-trivial to analyze the cause of power outages. While the majority of power failures from national grids last only a few hours, some blackouts can last days and can impact several critical infrastructure systems like telecommunication networks, financial services, water supplies and hospitals. For example, from airlines to blood supply levels to energy and water supplies, the 2003 NE American blackout impacted a wide range of critical infrastructure and emergency management sectors in both Canada and the U.S. One of the widespread impacts of Hurricane Sandy was loss of electricity for over 8.5 million homes across the eastern US [20].

As a result, identifying how and why the power outages evolve in a certain way during historic hurricanes is a very challenging task. Identifying time-points where there is a sudden change in the evolution of number of outages can help in many aspects from identifying causes and prioritizing repair resources to predicting future outage outbreaks [50, 66]. Such a problem may be addressed using the time-series mining task of 'segmentation' (however there are some domain-based properties of such time-series which we need to take into account for segmentations). Nevertheless, merely providing cut-points via segmentation is usually not enough for directly actionable guidance. Knowing the time-point of changes, domain experts also need to know *what* is changing and *how* it is changing, which is typically deeply buried in an abundance of unrelated information. In the context of power outage data from hundreds of counties, helping power engineers understand which counties were responsible for a particular sudden change is very valuable. Knowledge of the so-called 'culprits' allows them to direct their attention to these counties, and perform faster and more targeted damage control, to prevent even larger failures.

In this chapter we address this issue via a novel segmentation-with-explanations approach. Our main contributions are:

1. We propose a segmentation algorithm for modeling power outages, that captures temporal relationships among different time stamps.

2. We propose an explanation algorithm that automatically identifies the culprit of changes for the segmentation detected.

[1] http://www.sandia.gov/nisac/

3. Via experiments on historical hurricane data, we identify when and how the outage numbers dramatically change, and we summarize how these can help us reduce the number of outages in future disaster events.

6.2 Focus and Setup

We first briefly describe our focus and setup. Large power grids usually contain thousands of generators, hundreds of thousands of transmission lines and millions of consumers. Grid components have strong interdependencies like in the transmission grid where multiple paths exist between generators and consumers and these paths typically are arranged in a mesh grid. Hence if one path or line fails, the electricity instantaneously follows an alternate path governed by Kirchhoff's voltage and current laws [65]. If the alternate path however cannot handle the overload, it in-turn fails and this failure cascades to neighboring components. Due to the well established property of cascading failure propagation in the grid [65] and the small-world properties of power grids [67], a few initial points of failure due to a hurricane quickly cause network instability in a region, potentially causing millions of people to suffer the effects of brownouts or blackouts. Due to built in failsafes and resilience mechanisms in the grid, the effect of failure of a component is often localized to a region depending on the severity of the fault or failure.

In the context of a hurricane, the grid component damage and failure is often progressive along the spatial trajectory of the hurricane. A hurricane exhibits multiple phases of varied intensity along its path, causing failures with different levels of severity at different regions in its path. We model the progression of this grid failure process as a temporal segmentation problem.

Modeling this failure process over time, across different regions (e.g., counties) affected by a hurricane, is essential for improving the resilience of critical infrastructure during future disasters.

We characterize the severity of this grid failure process by measuring the evolving number of people in a hurricane affected region (a county in our case) without power over the time period fo the hurricane. Two critical questions need to be answered for characterization of this process:

- How can we characterize the different phases of a hurricane as a function of severity of the damage to critical infrastructure like the power grid using highly dynamic, sparse customer power loss data?
- Which locations (counties) are most important for characterizing each phase of failure?

Our main goal is to help domain experts answer the aforementioned questions.

Notation: We assume we are given a set of time series $X = \{x_1, x_2, ..., x_n\}$, where each time series $x_i = [x_i(t_1), x_i(t_2), ..., x_i(t_m)]$, and $x_i(t_j)$ represents the value at time stamp t_j for the i_{th} time series. We also assume there is an underlying graph structure G that captures the relation among these time series $\{x_i\}$, and we are given the Laplacian matrix L of G. For example, in critical infrastructure systems, the number of electric outages in all counties form a set of time series, and the relation among these counties can be based on the geographical proximity: two counties that are adjacent to each other are connected to each other. We use $x_i(t_a : t_b)$ to denote $[x_i(t_a), x_i(t_{a+1}), ..., x_i(t_b)]$, and $X(t_a : t_b) = [x_1(t_a : t_b), x_2(t_a : t_b), ..., x_n(t_a : t_b)]$.

Our algorithm CUT-n-REVEAL contains two parts: detecting a good segmentation of the outage data to capture the main changes and finding the corresponding explanations for the segmentation. With this knowledge of the segmentation and the explanations for each segment, the expert has a holistic picture of the different phases of the failure process as well as the specific time series that contributed significantly to each phase change. We next describe in detail each of these tasks and our solutions.

6.3 Finding Segmentations

In this section we focus on the segmentation problem. We first describe the formal problem, and then give the overview and details of our method.

6.3.1 Our Problem

We first need to define a segmentation:

Definition 6.1 (Segmentation S**)** *A segmentation of* X *contains a set of distinct time cut points* $S = \{c_1, c_2, ..., c_k\}$*, where* $c_i \in \{t_1, t_2, ..., t_m\}$.

The cut points of S naturally divide the time period into a set of time segments. We denote such a time segment as $s_i = [c_{i-1}, c_i)$ with $c_0 = t_1$, and $c_{k+1} = t_m$. Our problem is:

Problem 6.1 ***Given*** *a set of time series* X*, the Laplacian matrix* L *of the underlying network, and a number* k*,* ***find*** *the* k*-segmentation of* S *that captures the main pattern changes in* X.

6.3.2 Overview of our Approach

We develop a model to provide simple segmentations of the continuously changing grid failure process and interpretable explanations of the segmentations. We need to isolate temporal sequences into discrete segments such that the properties of the failure process in each segment differ from neighboring segments. The process of manually or algorithmically picking reasonable segments is non-trivial as segments that are too small fail to capture significant properties of the failure process while picking segments that are too large, although capturing all failure process characteristics, do not highlight the differences between the various phases of the process. Since this process is highly dynamic and the failure dataset is highly sparse, methods based on capturing long-term correlation [62] or invariant learning [114] from the data will be unable to perform adequately.

Owing to the sparse setting at hand and the lack of any long term correlations, we decided to adopt a latent factor modeling approach which has been shown to be effective in sparse settings like recommendation systems [119, 83]. The proposed segmentation procedure learns a latent representation for each time step using the data for dynamic power loss across affected counties. The results of the segmentation procedure are designed to be sparse in nature while managing to capture all the major trends in the process. We propose to use a data driven methodology (instead of setting a constant segment size) to automatically derive a set of appropriate time segments. Through the segmentation model, we wish to group time steps with similar patterns into contiguous time segments. To this end, we propose to represent each time step in latent space so as to capture its temporal relationship with the other time steps, and merge similar time steps into temporally contiguous segments. This methodology allows us to obtain segments where time steps in a segment are related to their past and future time steps in a similar way, allowing for simpler explanations later about pattern changes.

6.3.3 Details

In line with the overarching goal of discovering the different phases of the failure process in the power grid during natural disasters, we consider these phases or segments as a collection S of disjoint sets c_i. Each set c_i contains contiguous time steps that belong to segment i. We wish to discover a collection $S = \{c1, ..., c_k\}$ that minimizes similarity between any two neighboring sets c_i, c_j. Two sets c_i, c_j are said to be neighboring sets if $c_i = \{t_l, .., t_{l+\Delta l}\}$ and c_j contains t_{l-1} or $t_{l+\Delta l+1}$.

By doing so, each segment c_i would capture a different pattern from its neighboring segments (c_{i-1}, c_{i+1}), thus the segmentation S captures pattern changes in the time series. We employ the normalized cut framework which has been shown to work well in subspace clustering and segmentation tasks [148].

The normalized cut algorithm generates segments such that the similarity of time steps within each segment are maximized while the similarity between time steps in different segments is minimized.

The question of how to represent each time step, for effective similarity calculation between time steps, still remains. One straight-forward way is to use the data values at t_i and t_j across all time series and check how similar the values are. However, in this case, much of the information relating to the continuous evolution of the failure process will be lost.

In an effort to find a more principled approach to capture the similarity between different time steps in the failure process, we adopt the formulation provided by Tierney et al. [160] for video scene segmentation for our purposes of modeling the hurricane failure process. The model represents each time step in the data X, using a latent representation as a function of other important time steps. It is through this latent representation V that we attempt to capture the dynamics in the data X.

$$\begin{aligned} &\min_{V} \frac{1}{2}||X - XV||_F^2 + \lambda_1||V||_1 + \lambda_2||VR||_{1,2} \\ &\textbf{s.t.} \qquad \textbf{diag}(V) = 0 \end{aligned} \tag{6.1}$$

where V is an m by m matrix, and the i^{th} column represents the latent representation of time step i in terms of all the other time steps.

The first term in Eq. 6.1 calculates the reconstruction error between X and XV while the second term introduces sparsity into the latent representation, enforcing that each time step be explained as a function of a small subset of other important time steps. Finally in the third term, R is an m by $m-1$ lower triangular matrix with -1 on the diagonal and 1 on the second diagonal.

$$R = \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix} \tag{6.2}$$

The term VR calculates the difference of each time step with its previous time step in the latent V space. This term essentially serves as a smoothness constraint penalizing the dissimilarity of neighboring time steps. The $l_{1,2}$ norm term forces whole column similarity between two columns of V between neighboring time steps in V as opposed to just element-wise similarity in the case of a simpler l_1 norm on VR.

The solution to equation 6.1 can be obtained by applying the alternating direction method of multipliers (ADMM) [26]. Using this approach does not guarantee convergence but in our experiments, the algorithm always converged. To separate each term in equation 6.1, we assign $K = V$ and $U = KR$.

The Lagrangian formulation is given by equation 6.3

$$\begin{aligned}\mathcal{L}(V,K,U) = &\frac{1}{2}||X - XK||_F^2 + \lambda_1||V||_1 + \lambda_2||U||_{1,2} \\ &+ \langle G, V-K\rangle + \frac{\beta_1}{2}||V-K||_F^2 + \langle F, U-KR\rangle \\ &+ \frac{\beta_2}{2}||U-KR||_F^2\end{aligned} \tag{6.3}$$

Solving equation 6.3 yields a temporal weight matrix $V \in R^{m\times m}$ from which we derive an affinity W. The affinity matrix is then segmented using the normalized cuts procedure as mentioned previously to obtain the set of segments S.

6.4 Finding Explanations

Despite the sparsity of the segmentation procedure in the previous section, it is often not possible to identify the cause for each segment due to many simultaneously changing time series throughout the failure process. In this scenario, it is beneficial for a domain expert to know the subset of 'culprit' time series that were influential in forming a particular segment.

6.4.1 Our Problem

Assuming we are given a segmentation S of X, containing a set of time segments and the corresponding cut points $\{c_i\}$, which counties/time series are the most important for characterizing the different patterns in the segments? How to design explanations of the segmentations in an intuitive, easy to understand way? The cut points in the segmentation and our latent V matrix (which instead only captures the temporal relation among *time stamps*) do not provide answers to this question. A desired explanation of the segmentation should be simple yet effective enough that it gives direct guidance to prevent or curtail the effect of the failure of critical infrastructure in future disasters.

We provide answers to these questions by introducing an explanation vector e_i for each cut point c_i in the segmentation. Each e_i is an n by 1 vector, where the j_{th} value represents the importance of the j_{th} time series/counties in explaining the cut point. Intuitively, if a time series x_j shows very different patterns before and after the cut point c_i, we consider it important in explaining why c_i is a good cut point. On the other hand, if time series remains constant/unchanged across c_i, it does not provide useful information in terms of the cut point c_i and should have low values in e_i. In the hurricane outage data where there are hundreds of time series/counties, such explanation vectors are able to highlight the culprit time series/counties where major changes happen at the cut point. This explanation also serves as a guide for future resource allocation policies of maintenance and emergency personnel.

Definition 6.2 (Cut point explanations E**)** *$E = \{e_1, e_2, ..., e_k\}$, where e_i is an n by 1 non-negative explanation vector. $||e_i||_1 = 1$ and e_{ij} represents the importance of time series j for explaining the cut point c_i.*

Now we give the problem we solve.

Problem 6.2 ***Given*** *a set of time series X, the Laplacian matrix L of the underlying network, a number k, and the k-segmentation of S,* ***find*** *the associated explanations E, that capture the main pattern changes in X.*

6.4.2 Overview of our Approach

Existing time series segmentation algorithms do not provide any explanation of the result in an automatic principled way. Only recently there has been a push toward making complex machine learning model outputs quantifiable, explainable and simple [140]. To design good explanations specifically for hurricane outage data, we consider the characteristics of the data, as well as the requirements from the domain experts. We formulate an optimization problem that automatically learns explanations considering the underlying geographical relation between counties, revealing a small number of truly important counties as the culprits for the domain experts.

6.4.3 Details

We want to design an optimization problem that automatically finds good $\{e_i\}$. Assume that we have a function $d(S, i)$, which takes a segmentation S and a cut point index i as inputs, and returns an n by 1 vector which captures the difference of each time series before and after the i_{th} cut point c_i in S. We want e_i to give higher weights on time series with higher $d(S, i)_j$ values (therefore higher difference across cut point c_i). A straight-forward way is to maximize the weighted sum in the following way.

$$\begin{aligned} \arg\max_{E} & \sum_{i=1}^{k} e_i^T d(S, i) \\ \text{subject to} \quad & 0 \le e_{ij} \le 1, \\ & ||e_i||_1 = 1 \end{aligned} \tag{6.4}$$

The above formulation guarantees that we give higher importance to time series/counties with larger difference across the corresponding cut points. However, it treats each county as independent and ignores their geographical relation (some counties are close in distance, and some are far away). Such a geographical relation is important to capture because a hurricane trajectory is continuous and it usually hits counties that are close to each other at the same time. Hence, the importance of counties should be geographically smooth in the sense that adjacent counties should have similar importance. Another drawback of the above formulation is that it does not correctly reflect our requirement of a 'simple' explanation: we only need a few culprit counties and want to avoid having high importance for too many counties. Due to these considerations, we improve the above formulation by adding a geographical and a sparsity constraint on e_i. The final optimization problem we solve is shown below.

Given: A set of time series X,L, a segmentation S, α, λ.

Find: $E = \{e_i\}$ such that

$$\begin{aligned} &\arg\max_{E} \sum_{i=1}^{k} [e_i^T d(S,i) - \alpha e_i^T L e_i] - \lambda \sum_{i=1}^{k} ||e_i||_1 \\ &\text{subject to} \quad 0 \leq e_{ij} \leq 1, \\ &\qquad\qquad\quad ||e_i||_1 = 1 \end{aligned} \tag{6.5}$$

The geographical smoothness is introduced in the second term using the Laplacian matrix L (obtained from the underlying network behind the counties). This term basically minimizes the difference of e_i for adjacent counties. The third term is an $L1$ norm regularization on e_i, which introduces sparsity in e_i. This leads to the simplicity in our explanations: only a few important counties will have non-zero values in e_i to explain the c_i, making the results much simpler to interpret.

One question remains unsolved: How should we design the distance function $d(S,i)$ to capture the difference of time series across a cut point? Our idea is to look at a time window before the cut point c_i and a time window after c_i, and calculate the difference of these two time windows as the difference of the time series across c_i. Assume that w_{ij}^- represents the sub-sequence of x_j in the time window before c_i, and w_{ij}^+ represents the sub-sequence in the time window after c_i. We then calculate the difference of w_{ij}^- and w_{ij}^+ using simple, standard time series features: the mean value (f_1), the standard deviation (f_2), the maximum value (f_3) and the minimum value (f_4).

$$D(S,i)_j = \frac{1}{4} \sum_{z=1}^{4} |f_z(w_{ij}^-) - f_z(w_{ij}^+)| \tag{6.6}$$

Note that $|f_z(w_{ij}^-) - f_z(w_{ij}^+)|$ are usually in different magnitudes for different features, hence as a preprocessing step which we do not elaborate in the equation, we perform a min-max normalization of $|f_z(w_{ij}^-) - f_z(w_{ij}^+)|$ across all time series to make them in the same scale. As both w_{ij}^- and w_{ij}^+ are of a short length (a deliberate setting since the pattern changes that justify the choice of a particular cut points usually lie in the local area), these simple features are enough to capture the main pattern difference.

Finally, to solve Eq. 6.6, we optimize each e_i separately. For each e_i, the optimization can be re-written as a Quadratic Programming problem in the following way.

$$\begin{aligned} \arg\min_{e_i} \quad & \alpha e_i^T L e_i - [d(S,i)^T - \lambda i^T] e_i \\ \text{subject to} \quad & 0 \leq e_{ij} \leq 1, \\ & ||e_i||_1 = 1 \end{aligned} \tag{6.7}$$

The QP problem is well studied in the literature, and it is NP-hard in its general form. In our case, where the QP is convex to e_i, it can be solved in polynomial time using an Interior Point method [178], and we use the existing Matlab function (quadprog) to solve the problem.

6.5 Empirical Study

We implement CUT-n-REVEAL in Python and Matlab (we will release the code for research purposes). Our experiments were conducted on a 4 Xeon E7-4850 CPU with 512 GB of 1066Mhz main memory.

6.5.1 Setup

Dataset. We collect datasets from different domains with the ground truth segmentations to quantitatively evaluate our performance, and we run CUT-n-REVEAL on three hurricane outage datasets to show our case study results. For efficiency purposes, we perform a standard rolling average as a preprocessing step to all the data. The final statistics of the datasets are shown in Tab. 6.1.

ChickenDance: A "chicken" dance motion is recorded as a sequence of 4-dimensional data points. This was extracted by Matsubara et.al [109] and is originally from a CMU motion capture database[2]. We have the ground-truth segmentation here based on motions in the dances.

WalkJog: We use a dataset adapted from the REALDISP Activity Recognition Dataset [14, 55], where motions of walking, jogging and running are recorded by sensors. We use a sub-dataset with the walking motion and the jogging motion (as these motions show obviously different patterns), and we aim to detect when a different motion happens.

NILM: Non Intrusive Load Monitoring dataset. This dataset consists of real power measurements for various household appliances like lamps, laptops, and refrigerators, recorded through the use of MAU (Measurement and Actuation Units) connected between the device

[2]`http://mocap.cs.cmu.edu`

and the wall-socket. A detailed account of this dataset and the data acquisition methodology has been presented in [139]. We use a twenty-four hour snapshot of the NILM data to evaluate our performance, and use the time when a device switches state as the ground truth segmentation.

Hurricane Outage data: ORNL has developed several grid situational awareness products over the last decade such as VERDE, EARSS and EAGLE-I[3] for different stakeholders like DOE and FEMA, primarily for emergency management. For example, the National Outage Map within EAGLE-I collects distribution outage data of all the customers from utility websites every 15 minutes. Due to the recent coverage expansion (with more utilities exposing data from their Outage Management Systems), in this chapter, we consider the more recent hurricane outage data, namely for Matthew, Harvey and Irma since it covers nearly 90% of the population in the hurricane affected areas.

Baselines To the best of our knowledge, there is no algorithm that finds explanations for segmentations in the way we do. In the experiments, we mainly select state-of-the-art multivariate time series segmentation algorithms to compare against.

Autoplait [109] is a Hidden Markov Model based algorithm that discovers the different regimes in co-evolving time series. Each regime can be thought of as the segments for our problem.

TICC [62] is a subsequence clustering algorithm for multivariate time series to discover repeated patterns. It clusters time stamps into segments that can be well interpreted by the same model. Each cluster of the time stamps becomes the segments for our problem.

Dynammo [98] reconstructs data values in time series by discovering latent variables and their dynamics, and then uses the spikes of the reconstruction errors to find cut points for the data.

Table 6.1: Datasets used.

Dataset	**#Timestamps**	**#Time series**
NILM	721	5
ChickenDance	322	4
WalkJog	500	2
Harvey	264	250
Irma	169	271
Matthew	252	369

[3]Outage Data source: EAGLE-I https://eagle-i.doe.gov/

6.5.2 Quantitative Evaluation

We compare CUT-n-REVEAL performance with the baselines on the datasets with ground truth segmentations: *NILM, ChickenDance* and *WalkJog*. We evaluate the detected cut points by calculating the F1 score based on the ground truth cut points (as in [109]). Higher F1 scores show better segmentation precision. We show the results in Tab. 6.2.

Table 6.2: Evaluation on ground truth datasets

Method / Dataset	CUT-n-REVEAL	AutoPlait	TICC	Dynammo
NILM	0.8	0.4	**0.81**	0.75
ChickenDance	**0.8**	0.73	0.5	0.57
WalkJog	**1.0**	0.0	0.25	0.0

As shown in the table, except for the *NILM* data, CUT-n-REVEAL significantly outperforms the baselines (achieving the best F1 scores) in most of the cases. Even in *NILM*, our performance is comparable to TICC's (a difference of only 0.01). This showcases the effectiveness of CUT-n-REVEAL at identifying and extracting different patterns in time series. In the following, we visualize our segmentations and show that they capture the main pattern changes in the data.

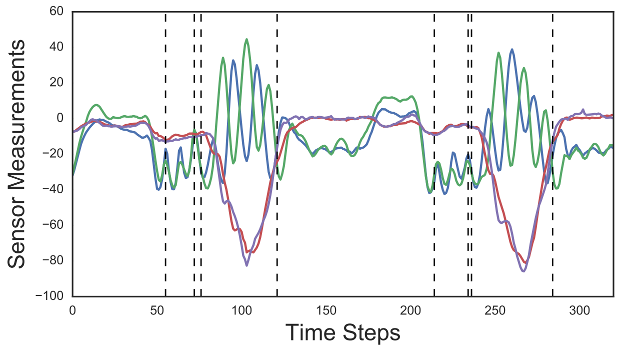


Figure 6.1: The discovered segmentation results for *ChickenDance*.

We show our segmentation result for *ChickenDance* in Fig. 6.1. CUT-n-REVEAL is able to isolate most of the different data trends successfully. It detects precisely 6 cut points from the 7 ground truth cuts, with two false positive cuts at time steps 72 and 234, which are very close to ground truth cuts. In Fig. 6.2, for *WalkJog*, we see that CUT-n-REVEAL correctly separates the sequences of data generated due to walking, from those due to jogging. The cut point discovered by our method lies in a 1% cut point location tolerance window with respect to the ground truth cut point. Finally for the *NILM* dataset, the results (Fig. 6.3) are able to identify accurately (F1 score=0.8) the different residential daily usage patterns (segments 1 - 9 enumerated from left to right of the figure).

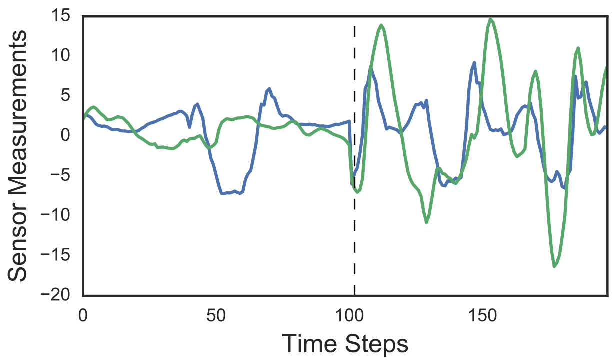


Figure 6.2: The discovered segmentation (vertical black dashed line) results for the *WalkJog* dataset.

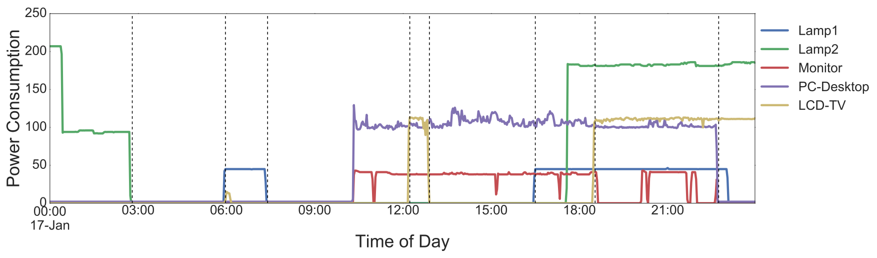


Figure 6.3: The discovered segmentation (vertical black dashed line) results for the *NILM* dataset.

6.5.3 Case Studies: NILM

Monitoring the energy consumption pattern of a home is essential for understanding patterns of power wastage and overuse. Hence it is interesting to study in-depth the patterns as shown by our segmentation for the *NILM* dataset. In Figure 6.3, each segment captures a state change of one or multiple devices. Segments 1 and 2 represent a possible nightly routine with all electronic devices except possibly a night lamp (Lamp2) turned off. Segment 4 represents the power consumption during the first half of the work day and segment 5 captures the event of the television being turned on, possibly during a lunch break. Segment 6 could be equated with segment 4 representing the second half of the work day. Segment 7 represents the beginning of the evening routine. This segment is also the period of peak demand for the day. If a user wished to lower their energy bill, it is apparent that the energy consumption in this segment would have to be reduced or distributed to neighboring segments. Segments 8 and 9 can be considered evening recreational activity. The segmentation algorithm thus allows a resident to gain an abstracted view of their energy usage patterns as a function of their daily activities with the devices in each segment highlighted. In order to further our performance evaluation, and to understand if our results are robust to noise (common in such datasets), we also included another device (a refrigerator) that consumes significant energy relative to the other appliances, and has a consistent cyclic daily usage pattern. The usage pattern with segmentation on the new dataset is shown in Figure 6.4. Interestingly CUT-n-REVEAL is quite robust in the presence of such devices. It is still able to effectively capture the device state changes of all appliances in the household like before, with the difference being the additional segments during periods of isolated operation of the refrigerator.

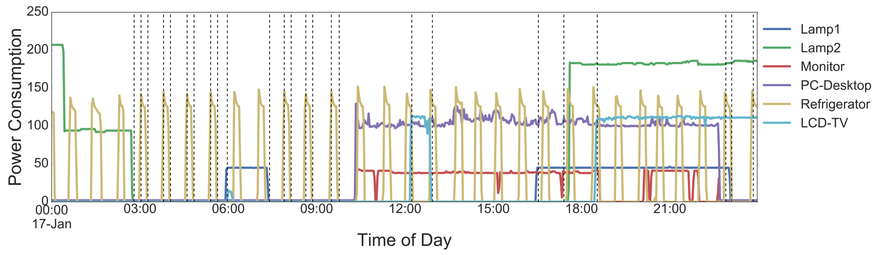


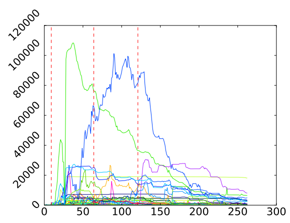
Figure 6.4: *NILM* with noisy cyclic patterns

6.5.4 Case Studies: Hurricanes

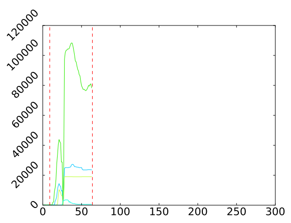
We run CUT-n-REVEAL on outage data from three recent hurricanes. We show that, thanks to CUT-n-REVEAL, we can find the segmentations which capture important outage pattern changes, and our explanations correctly identify the culprit counties where major events happen.

Hurricane Harvey. We show our segmentation and explanation results in Fig. 6.5. The segmentation and all the time series are shown in Fig. 6.5(a). In Fig. 6.5(b)(c)(d), we visualize for each cut point the most important time series whose e_i values sum over 0.8 (their importance in explaining the corresponding cut point is over 80%). In Fig. 6.5(e)(f)(g), we visualize the entire e_i vector on a map, where the color of a county shows the importance of that county in explaining the cut point.

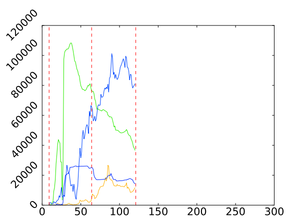
We detect three cut points for the Harvey data (see Fig. 6.5(a)). The first cut point captures the date (Aug. 25) when the hurricane strengthened before its landfall. Soon after this cut point, the number of outages starts to rise in several counties. In Fig. 6.5(b)(e), we observe that CUT-n-REVEAL correctly highlights the counties with the steepest rise for the first cut point (Nueces, San Patricio and Aransas), and all of them are near Harvey's landfall area. For the second cut point (around Aug. 27), our explanations capture two different patterns: the number of outages in Nueces County (green line in Fig. 6.5(c) and the red county in Fig. 6.5) experience major decrease, while the number of outages in the Victoria, Matagorda and Montgomery counties start to rise. This cut point correctly shows that the impact of the hurricane is moving, and our explanation identifies the direction of the move (which is consistent with the trajectory of Hurricane Harvey). Finally for the last cut point (around Aug. 30), while the outages of many counties are decreasing, our algorithm correctly detects a small set of counties (Orange, Jefferson and Hardin) that experience a sudden outage increase (Fig. 6.5(d)(g)). The main reason for this increase is due to the rising water of the Neches River, which causes the city to lose service from its major pump stations. Note that this information is deeply buried in the entire time series and it is hard to directly observe in Fig. 6.5(a) with hundreds of time series why such a cut point makes sense. Through our explanations, we perfectly isolate these culprit counties in Fig. 6.5(d)(g).



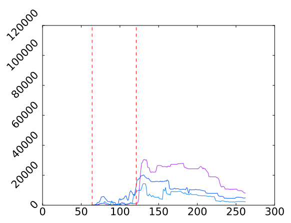
(a) Harvey segmentation



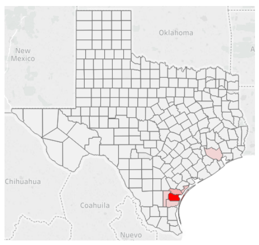
(b) Important time series for c_1



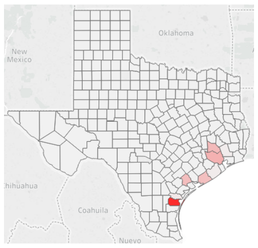
(c) Important time series for c_2



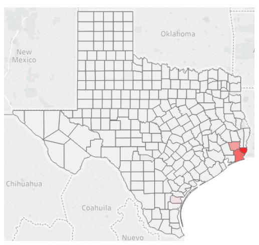
(d) Important time series for c_3



(e) Explanation e_1



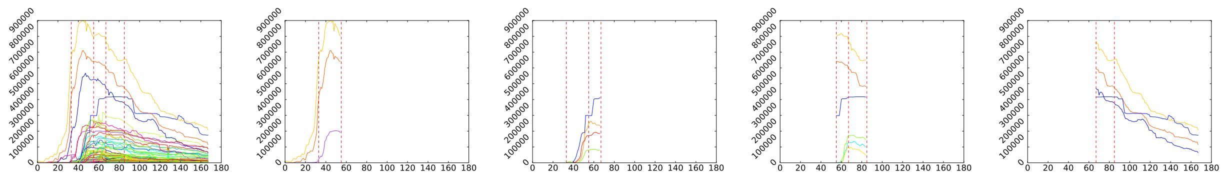
(f) Explanation e_2



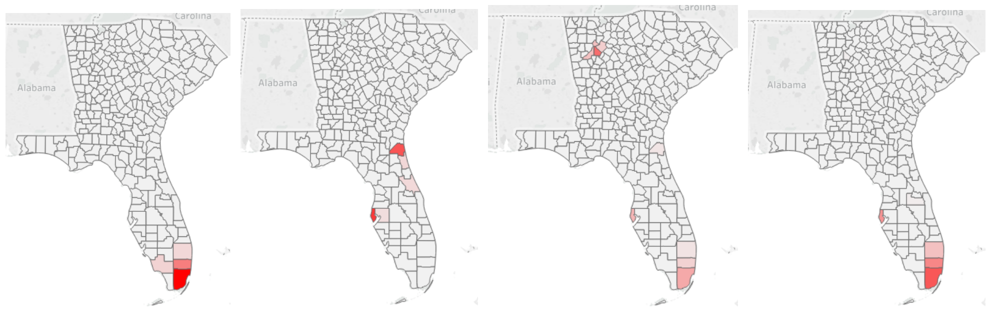
(g) Explanation e_3

Figure 6.5: Segmentation and the corresponding explanations for Harvey. (a) The segmentation and all the time series. (b)(c)(d) The most important time series (that contribute over 80% importance in e_i) for each of the cut point. (e)(f)(g) e_i visualizations for each cut point. Counties with higher e_i values are more important for the cut point, and are marked with a color closer to red.

Hurricane Irma. We make similar plots for Irma in Fig. 6.6. The first cut point (around Sep. 10) again captures the date of the hurricane's landfall. Interestingly, the second and third cut point of the segmentation are very close to each other, and by directly reading the raw data with hundreds of time series in Fig. 6.6(a), it's impossible to make sense of these two cut points. On the other hand, the explanations learned by CUT-n-REVEAL precisely show why these two cut points are detected: they are related to outage increases in different locations that are far away from each other. The second cut point shows the outage increase in the Pinellas, Duval and St. Johns counties, while the third cut point captures the outage outbreak in the DeKalb, Fulton and Gwinnett counties (the highlighted area in the top of Fig. 6.6(h)). Finally, the last cut point (around Sep. 12) correctly captures the date when hurricane Irma weakened into a Category 2 storm, and the number of outages in many counties decrease.



(a) Irma segmentation (b) Important time series for c_1 (c) Important time series for c_2 (d) Important time series for c_3 (e) Important time series for c_4



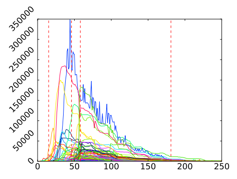
(f) Explanation e_1 (g) Explanation e_2 (h) Explanation e_3 (i) Explanation e_4

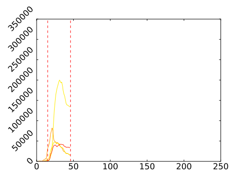
Figure 6.6: Segmentation and the corresponding explanations for Irma. See detailed discussions in Sec. 6.5.4.

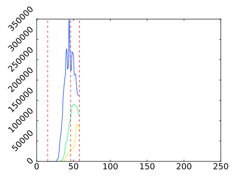
Hurricane Matthew. In Fig. 6.7, we show our segmentation and explanations in similar ways for Hurricane Matthew. The first cut point (around Oct. 2) corresponds to the landfall of Matthew and a set of counties in the east coast (Brevard, Palm Beach, etc.) start to have increasing power outages. The second cut point (around Oct. 4) and the corresponding explanations show the moving direction of the hurricane. Counties like Duval, Chatham, Horry and Charleston, which are to the north of the landfall location, start to experience power outages (Chatham County experiences a peak of outage at the cut point). Soon after Oct. 4, in Fig. 6.7(h), we observe that Horry County (highlighted with the red color), which is influenced in the previous cut point, has now become severely affected and the influence has spread to neary counties as well. This can be also observed in Fig. 6.7(d) where a set of time series suddenly increase. Finally, the last cut point (around Oct. 9) shows that the power outage impact of Matthew on these previously affected counties has mostly abated.

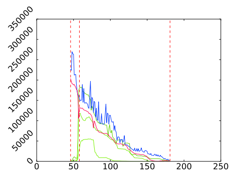
6.6 Conclusions

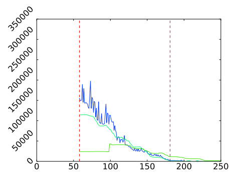
In this chapter, we have developed a combined framework for providing simple interpretable explanations for failure processes like critical infrastructure outages. We evaluated the performance of our methodology against state-of-the-art segmentation and time series clustering

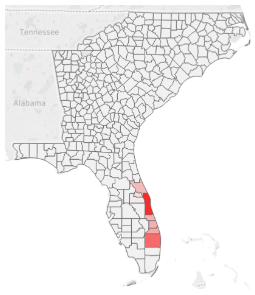

(a) Matthew segmentation


(b) Important time series for c_1

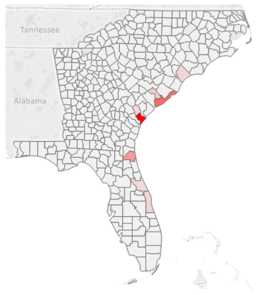

(c) Important time series for c_2


(d) Important time series for c_3

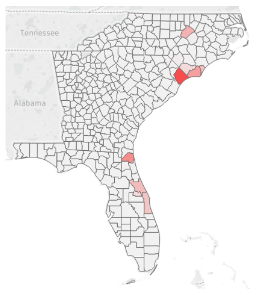

(e) Important time series for c_4



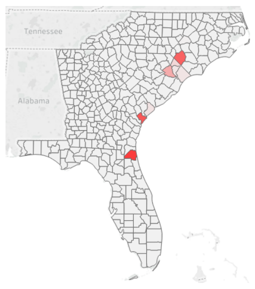
(f) Explanation e_1



(g) Explanation e_2



(h) Explanation e_3



(i) Explanation e_4

Figure 6.7: Segmentation and the corresponding explanations for Matthew. See detailed discussions in Sec. 6.5.4.

procedures on ground truth datasets. We have also conducted extensive analysis on the failure of the power grid during three hurricane events along with conducting a case study on the applicability of temporal segmentation to understanding residential energy usage patterns. There are many avenues for future work. Methodologically, we can study performing a joint learning of segmentations and explanations to leverage both spatial and temporal information simultaneously. We are also exploring integrating CUT-n-REVEAL with existing analysis tools, such as the URBANNET toolkit [37] in use at national labs and power utilities.

Chapter 7

Modeling Influence among Tweet Sequences

Microblogging websites, like Twitter and Weibo, are used by billions of people to create and spread information. This activity depends on various factors such as the friendship links between users, their topic interests, and the social influence between them. Making sense of these behaviors is very important for fully understanding and utilizing these platforms.

Most prior works on modeling social-media either ignore the effect of social influence, or consider its effect only on link formation or post generation. In contrast, in this chapter we propose PoLIM, which *jointly* models the effect of influence on both link and post generation, leveraging weak supervision. We also give PoLIM-FIT, an efficient parallel inference algorithm for PoLIM which scales to large datasets. In our experiments on a large Twitter corpus, we detect meaningful topical communities, celebrities, as well as the influence strengths pattern among them. Further, we find that there is a significant portion of posts and links that are caused by influence, and this portion increases when the data focuses on a specific event. We also show that differentiating and identifying the influenced content benefits other quantitative downstream tasks as well, like predicting future tweets and link formation.

7.1 Introduction

Modeling microblogging data, such as Twitter and Weibo, has attracted great attention in recent years [33, 133, 170]. Social media data is easy to obtain, and understanding how the data is generated provides insights to many applications such as community detection, influence maximization, public-health surveillance, etc. Ultimately we want to understand how people generate content and how online information diffuses across the underlying social network.

This problem is compounded by the fact that due to social influence, an individual's behavior and attributes may conform to her neighbors' [86, 106]. In the context of social media, social influence can be thought of as a latent factor, that may alter users' posting and linking behavior. For example, a Twitter user may follow Barack Obama simply because he is a celebrity with high popularity, regardless of his/her own interests. Similarly, a Twitter user may retweet a close friend because of their mutual friendship regardless of whether the tweet content is interesting to her. Without understanding how such latent social influence affects the generation of posts and links, we cannot fully and correctly understand the complex information patterns.

To this end, we propose a novel generative model PoLIM (**Po**st and **L**ink level **I**nfluence **M**odel) which extracts and models the latent influence that affects the generation of *both* posts and links. We assume the existence of some weak supervision (like tweets with the 'RT' label in Twitter), that are more likely to be affected by social influence [115]. We use this weak supervision ('RT') to guide the inference of PoLIM, which then generalizes and learns the latent influence for all the posts as well as the follower-followee links. Modeling the extent of this latent influence for both posts and links helps us learn better topic interests for users and communities. This helps us achieve higher performance on other downstream tasks too, such as predicting the link formation and retweet generation in the future. Informally, our goal of proposing PoLIM can be stated as:

Problem 7.1 *Given a microblogging dataset (such as Twitter and Weibo), with an underlying directed friendship network $G(E, V)$, and set of textual posts $\mathcal{T}_i$ from each user n_i, identify social influence among users, and the posts and friendship connections that are caused by social influence.*

Most of the existing models for social media datasets either completely ignore the effect of social influence (they assume all behaviors are driven by self interest), or only consider its effect on one of the two aspects (links or posts). The most closely related model is COLD by Hu et al. [77]: although they propose a model that covers social influence, links and posts (like us), they only use social influence to control link generation (in contrast, we use it to control each link and post). Moreover we learn this social influence at multiple granularities including at the community and individual level, helping us better understand content generation. Doing so ultimately helps us to get better prediction and analysis of the diffusion. For example, we show in our experiments how PoLIM can achieve better link prediction and retweet volume prediction than state-of-the-art competitors which do not perform this integrated modeling. In summary, our main contributions are:

1. We propose a novel model PoLIM which jointly models post and link generation via latent social influence and weak supervision. We also develop a parallel inference algorithm PoLIM-FIT to efficiently learn the parameters.
2. We use PoLIM to analyze a large 2009 Twitter dataset (containing more than 27 million tweets). We find multiple meaningful topical communities with different latent

influence strength patterns. We also find interesting patterns among the influential users of different communities.

3. Based on the specificity of the communities and tweets, we detect that significant portions of posts and links are affected by social influence. We also demonstrate that differentiating these posts from those by self interest leads to better performance on concrete downstream tasks like predicting future links and retweets.

7.2 Model Formulation

We formulate our proposed model in this section. Our main hypotheses are 1) social influence controls both the post generation and the social link formation, and 2) we have some supervision on users' posts, which are good indicators of whether the post is generated by social influence or not. Given these hypotheses, we formulate our model **Po**st And **L**ink level **I**nfluence **M**odel (PoLIM) to jointly model the post content, social structure and the social influence, using weak supervision from the influence indicators. We first explain the main concepts in PoLIM in the following. The notations we used are shown in Tab. 7.1. For simplicity, we only show the most important symbols, and skip the prior parameters (α, β, η, etc.) and those symbols explained in the text (λ, c^*, c', etc.).

Table 7.1: Terms and symbols

Symbol	Definition and Description
Act-snapshot	The follower-followee network
N	Number of users in *Act-snapshot*
n_i	User i
T_i	Number of posts by n_i
Z	Number of topics
W	Number of unique words
K	Number of communities
E_i	Number of links from n_i
c_i	The i_{th} community
θ_i	The topic interest of c_i: Zx1 probability vector
ϕ_i	The word distribution for $TopicM_i$: Wx1 probability vector
$TopicM_i$	The i_{th} topic
I	K by K influence matrix
$N(n_i)$	Neighbors of n_i in *Act-snapshot*
t_{ij}	The j_{th} tweet of n_i
e_{ij}	The j_{th} followee of n_i
v_i	The probability that n_i is not influenced by another user
ρ	The probability that a user follows someone in her own community (vs. random following)
u	The probability of generating word from the background topic
r	The switch value for each tweet; when $r = 0$, the tweet is caused by self interest, otherwise social influence
ϵ	The switch value for each link; when $\epsilon = 0$, the link is caused by social influence
A_i	The $N \times 1$ celebrity vector for c_i
M	The community distribution: Kx1 probability vector

7.2.1 Main Concepts

We denote the social network as a directed graph *Act-snapshot*, where each node represents a user n_i in the social network, and a directed edge represents a following relation. Every user n_i has a sequence of posts $\mathcal{T}_i = \{t_{i1}, t_{i2}, ...\}$, and each post t_{ij} contains a sequence of words. In the following, we define the most important concepts in PoLIM.

Communities. Communities sharing similar interests naturally exist in social networks. Hence, to make the model expressive but still tractable, in PoLIM, we define the following community concept to aggregate users with the same topic interests.

Definition 7.1 (Community) *A community c_i is a group of users, who share the same topic interest θ_i ($Z \times 1$ vector), where θ_{ij} represents the probability of generating a post with topic* TopicM_j.

In spite of the above definition, a user can still be influenced by another user. In the following, we define the user-to-user influence through the lens of the community concept. Given an instance that n_a in c_1 is influenced by n_b in c_2 to generate a post/link, we decompose such an influence to two steps: *community-to-community influence*, and *user-in-community selection.*

Community-to-community influence. When a user n_a in community c_i gets influenced by another user, we first select where (which community) the influence comes from. We define the following influence matrix I to capture the probability of a community being influenced by another.

Definition 7.2 (Influence matrix) *An influence matrix I is a K by K matrix, where I_{ij} represents the probability of a user n_a in c_i being influenced by some user in c_j given that n_a is influenced by another user.*

Note that the diagonal entries of I can be non-zero, i.e., we allow a user to be influenced by someone in the same community. This matches the fact that one may be influenced by someone with either different, or similar interests. For example, n_a in the data mining community may be influenced by n_b in the politics community to retweet political news, and n_a can also retweet his/her colleague n_c from the same community about a new paper in the field. Both cases (intra- and inter- community influence) are commonly seen in social media [17].

User-in-community selection. Once we decide which community the influence comes from, we select an influencer from the influencing community. Notice that different members in a community have different powers to influence other users (for example, a dean's tweets are more likely to be retweeted than a student's); we define the following 'celebrity' vector to capture users' popularity in his/her community.

Definition 7.3 (Celebrity vector) *Each community c_i has a celebrity vector A_i (a U by 1 vector), which shows the popularity of users in c_i.*

A user with higher popularity in a community is more influential than other members in affecting others' behaviors, and attracts more followers and retweets. Naturally, a user can only be a celebrity in her own community (having non-zero values in A_i), while she can still be influential in other communities via social influence I.

7.2.2 Our Model PoLIM

Now we describe our PoLIM model. The graphical representation of our model is shown in Fig. 7.1: as highlighted, the main feature of PoLIM is that social influence contributes to the generation of each post and link. We will first explain how each post and link can be generated without any supervision (generation process shown in Alg. 7), and then explain why and how we introduce the weak supervision l to the model.

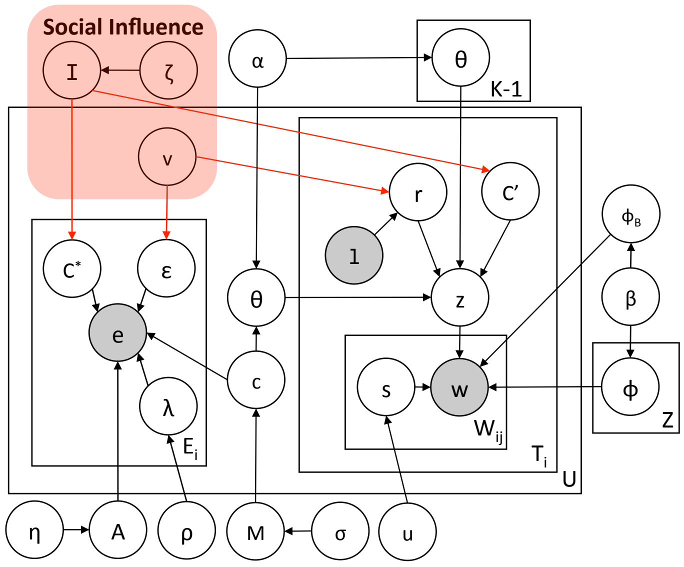


Figure 7.1: Plate diagram of PoLIM.

To generate the data, we first initialize $\{\phi, \theta, I, A, c, u, v, \rho\}$ using the prior parameters (α, β, etc.). We use the standard conjugate priors (Beta and Dirichlet distribution) for Bernoulli and Multinomial distribution to generate these parameters for ease of inference (a widely adopted setup in topic modeling [116, 19]).

Generating Posts. Considering the characteristics of posts on microblogging websites such as Twitter and Weibo, we make the following two assumptions about the posts: 1) each post has only one latent topic (also commonly used in the past work [182, 133, 134]) and 2) each word in a post can be generated either by the corresponding latent topic, or by a background topic which generates common words like 'I', 'and', 'to', 'for', etc. Each user n_i has a probability v_i to behave from his/her self interest. Using this v_i probability, we draw the switch values r and ϵ, which represent whether the corresponding post and link are generated from social influence or not. If a post is generated from self interest, we draw a topic from the user's own topic interest (the same as the topic interest of the community he/she belongs to), and generate the post accordingly. On the other hand, if the post is influenced by another user, we select an influencing community c' according to the influence matrix I, and generate the post based on the topic interest of c'.

Generating Links. Similarly for the link generation, we first draw the switch value ϵ. If $\epsilon = 0$, the link is generated from social influence, and we first choose an influencing community c^* according to I, and then choose an influencing user in c^* to follow based on the corresponding A^* vector. When $\epsilon = 1$, differently from the post generation above, the link is considered either generated from self interest or random following (decided by the switch value λ). If the link is generated from self interest, we select a celebrity user in $n_i's$ own community to follow based on the corresponding A vector; otherwise, we choose a random user in the network to follow.

Adopting Weak Supervision. To correctly learn social influence, a big challenge for PoLIM is to learn when a post/link is influenced (namely to learn the switch values r, ϵ correctly). In general, distinguishing social influence from other compounding variables is a very hard task [9, 5]. Without any guidance, the change of the data likelihood caused by an arbitrary change of these switch values, can be undesirably compensated by updating the other parameters accordingly. In this chapter, motivated by the usage of aspects in topic models [129], we assume the existence of good influence indicators/markers l for each post. Intuitively, if $l = 1$, it suggests that the post is *likely* (not necessarily) generated by influence, and we are more likely to learn $r = 1$ for the post, and vice versa. Retweets have been regularly used as an proxy for influence in Twitter social influence studies [115] in past. Hence in our experiments, we simply use the RT label as a weak influence indicator ($l = 1$ if the tweet contains the RT label), and bias the learning of r using the following equations (with $\tau = 0.1$):

$$p(r = 0|l = 0) = 1 - \tau + \tau v$$
$$p(r = 0|l = 1) = \tau \cdot v$$

The probability of $p(r = 1|l)$ can be calculated accordingly. Note that this (weak) supervision only applies to r; however, it also affects the learning of ϵ because both switch values are controlled by the same influence probability v. Therefore, although we use l as the weak supervision, we would be able to leverage both the posts and links information to learn beyond l and extract social influence that best describes the data.

7.3 PoLIM-FIT: Model Inference

The main parameters in PoLIM are $\{\theta, I, A, \phi, \mathit{TopicM}, r, s, c', c^*, c, \epsilon, \lambda, u, v, \rho\}$ (other prior parameters can be inferred once these parameters are learned). To fit PoLIM on real datasets,

Algorithm 7 Generative process for PoLIM

```
1: Initialize $\phi$, $\theta$, $I$, $A$, $c$, $u$, $v$, $\rho$ using prior parameters like $\alpha$, $\beta$, etc.
2: //Generate tweets
3: for each user $n$ do
4:     for each tweet $t$ do
5:         Choose an indicator $r \sim Ber(v_n)$ //Bernoulli distribution
6:         if r=0 //from self interest then
7:             Choose a topic $z \sim Multi(\theta_n)$ //Multinomial distribution
8:         else
9:             Choose an influencing community $c' \sim I_n$//from social influence
10:            Choose a topic $z \sim Multi(\theta_{c'})$
11:        for each word w do
12:            Choose an indicator $s \sim Ber(u)$
13:            if s=0 then
14:                Choose a word $w \sim Multi(\phi_B)$//background word
15:            else
16:                Choose a word $w \sim Multi(\phi_z)$
17: //Generate links
18: for each user $n$ do
19:     for each link l of user $n$ do
20:         Choose $\epsilon \sim Ber(1 - v_n)$
21:         if $\epsilon = 0$ then
22:             Choose a community $c^* \sim I_n$//from social influence
23:             Choose an influencing user in the community to follow $e_{n,l} \sim Multi(A_{c^*})$
24:         else
25:             Choose $\lambda \sim Ber(\rho)$
26:             if $\lambda = 0$ then
27:                 Choose an influencing user from $n$'s own community to follow $e_{n,l} \sim Multi(A_{c_n})$ //self interest
28:             else
29:                 Choose a random user to follow.
```

Algorithm 8 Pseudo-code for PoLIM-FIT

```
1: Initialize prior parameters like $\alpha$, $\beta$, etc.
2: Sample values for $\{TopicM, r, s, c', c^*, c, \epsilon, \lambda, u, v, \rho\}$.
3: Repeat step 2 until convergence.
4: Sample values for the marginalized variables $\{\theta, I, A, \phi\}$.
```

we propose PoLIM-FIT to automatically learn all these parameters from the data in linear time. Further, we improve the efficiency of PoLIM-FIT by parallelization.

Both the likelihood function and the posterior distributions in such graphical models are intractable for exact inference [11]. Hence, we propose a Collapsed-Gibbs-Sampling-based [58] algorithm PoLIM-FIT to learn the model parameters. We show the pseudo-code in Alg. 8. PoLIM-FIT first marginalizes several parameters ($\{\theta, I, A, \phi\}$) when sampling other parameters ($\{TopicM, r, s, c', c^*, c, \epsilon, \lambda, u, v, \rho\}$). Once the sampling process converges to stable values, we then estimate the marginalized parameters from the sampled values. As an example, in the following we show the final sampling equations for two of the important parameters in PoLIM. [1]

Community assignment c_i. Each user n_i has a unique community assignment c_i. Given the

[1]See detailed equations, additional content in Appendix C.

other parameters, c_i is sampled using the following probabilities.

$$p(c_i|\ldots) \propto \prod_{t_{ij},r=0} \frac{\alpha + N_{-n_i,c_i}^{TopicM_{ij}}}{Z\alpha + N_{-n_i,c_i}} \cdot \prod_{t_{ij},r=1} \frac{\zeta + N_{-n_i,c_i}^{c'_{c_i}}}{K\zeta + N_{-n_i,c_i}} \cdot$$
$$\prod_{e_{ij},\epsilon=1,\lambda=0} \frac{\eta + N_{-n_i,c_i}^{e_{ij}}}{|c_i|\eta + N_{-n_i,c_i}} \cdot \prod_{e_{ij},\epsilon=0} \frac{\zeta + N_{-n_i,c_i}^{c_i^*}}{K\zeta + N_{-n_i,c_i}} \cdot M(c_i)$$

where $N_{-n_i,c_i}^{TopicM_{ij}}$ denotes the number of times the topic $TopicM_{ij}$ is generated by a user in c_i; $N_{-n_i,c_i}^{c'_{c_i}}$ denotes the number of times that c'_{c_i} is influenced by c_i; and $N_{-n_i,c_i}^{e_{ij}}$ is the number of times that e_{ij} is chosen from c_i by other users to follow. All the $-n_i$ means excluding n_i in the counting. Intuitively, it goes over all the tweets and edges from n_i, and calculates the likelihood of the user belonging to c_i given all the switch values. Note that the edges caused by random following are not used in the equation because their likelihoods are independent of c_i.

Latent topic TopicM_{ij}. Each post t_{ij} has a latent topic $TopicM_{ij}$ which can be sampled as:

$$p(TopicM_{ij}|\ldots) \propto \prod_{w_{ijt},s_{ijt}=1} \frac{\beta + N_{-t_{ij},TopicMij}^{w_{ijt}}}{W\beta + N_{-t_{ij},TopicMij}} \cdot$$
$$\left(\frac{\alpha + N_{-n_i,c_i}^{TopicM_{ij}}}{Z\alpha + N_{-n_i,c_i}}\right)^{\mathbb{1}(r_i=0)} \cdot \left(\frac{\alpha + N_{-n_i,c'_i}^{TopicM_{ij}}}{Z\alpha + N_{-i,c'_i}}\right)^{\mathbb{1}(r_i=1)}$$

where $\mathbb{1}()$ is an indicator function. Basically, we first go over all the words in t_{ij} that are not generated by the background topic ($s_{ijt} = 1$), and calculate the likelihood of t_{ij} being generated by $TopicM_{ij}$; then based on the r value, if t_{ij} is generated from influence, we calculate the likelihood of c_i generating the topic $TopicM_{ij}$, otherwise the post is influenced, and we calculate the likelihood of the influencing community c'_i generating $TopicM_{ij}$.

In sum, PoLIM-FIT has a linear time complexity of $O(R(WZ + TK + EK + N))$, where R is the number of iterations the algorithm runs. Note that theoretically, sampling the marginalized parameters I is quadratic in the number of communities, but since it only needs to be sampled once after the sampling process converges, it is not the bottleneck of the running time.

Speeding Up & Parallelization. To further improve the running time, we implement a parallel version of PoLIM-FIT. Since many of PoLIM's parameters are based on users, we allocate data from different users to different worker processes, and each individual process samples the parameters related to its users simultaneously. The counters used in the sampling process (such as $N_{-n_i,c_i}^{TopicM_{ij}}$) are maintained as global variables that are shared by all the processors. The final time complexity for our sampling algorithm is therefore reduced to $O(\frac{R}{p}(WZ + TK + EK + N))$, where p is the number of processes.

7.4 Empirical Study

We implement PoLIM in Java (we will release the code for research purposes). Our experiments were conducted on a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory. We design various experiments in this section to answer the following questions. Essentially we want to check if our model can help in objective downstream related tasks, and if it gives insightful patterns in the data.

Q1 [Downstream task 1] Can PoLIM improve the performance of link prediction?

Q2 [Downstream task 2] Can PoLIM improve the performance of retweet volume prediction?

Q3 In the entire dataset, what is the extent of social influence?

Q4 Does PoLIM find meaningful topics?

Q5 What are the influence strengths among different communities? Who are the celebrities?

Q6 Can PoLIM help understand the content generation and communities during a specific event?

7.4.1 Setup

Dataset. We use a Twitter dataset *Tweets-Whole* collected over a 7-month period during 2009 [175]. We preprocess this data by first filtering out users that do not have at least 15 tweets in each month. Then for each tweet from these users, we perform standard tokenization, stemming, lemmatization, infrequent words removal, and get our final dataset. In addition to this large complete data, for fast performance comparison purposes (which requires multiple runs for cross-validations with different parameter settings), we generate three sample datasets. We randomly sample 2%, 5%, and 20% of the users based on their degrees in the social network. Finally, to analyze more specific events, we create an *Tweets-Iran* dataset by extracting tweets that contain keywords in the 2009 Iran Election (such as 'iran', 'iran elect', 'neda', etc.[1]).

Baselines. To the best of our knowledge, there are no existing methods which model how social influence affects the generation of *both* posts and links as we do. As a result, unlike PoLIM which can predict both links and retweets, most state-of-the-art algorithms can only be used for either one of the task. We list all the baselines in the following.

1. *COmmunity Level Diffusion* (COLD) [77] is the **state-of-the-art** generative model that covers social influence, posts and links as we do. However, the social influence it models only contributes to the link generation, while all the posts are still considered generated from users' own interests.

Table 7.2: Datasets used.

Dataset	**#Users**	**#Edges**	**#Tweets**
Tweets-Whole	$46.5K$	$2.1M$	$27.5M$
Tweets-2%	$0.9K$	$28K$	$0.7M$
Tweets-5%	$2K$	$0.1M$	$1.8M$
Tweets-20%	$9.2K$	$0.7M$	$6.5M$
Tweets-Iran	$3K$	$40K$	$62K$

2. *Mixed Membership Stochastic Blockmodel* (MMSB) [6] combines dense connectivity (blockmodel) with node specific variability (mixed-membership). Each user's membership is a mixture of communities with different weights.

3. *Topical Affinity Propagation* (TAP) [155] is a popular model which finds topical influence between users given the network and users' topic interests. For our experiments, we feed TAP with the topic interests learned by PoLIM. We then combine topic posterior and the topical influence probabilities to calculate the probability of a tweet being influenced.

4. EMP is a baseline we designed for the retweet prediction task. It uses the empirical retweet ratio in the training data as its estimation of retweet probability in the testing data.

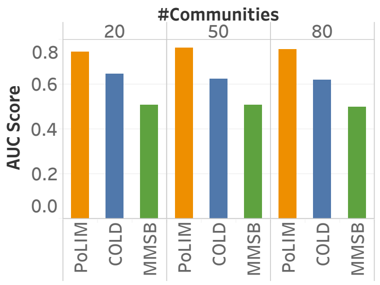
7.4.2 Q1: Link Prediction

We show that the parameters learned from PoLIM can be used to predict whether a user will follow another user well. In this experiment, we design a 5-fold cross validation on *Tweets-2%*, *Tweets-5%*, and *Tweets-20%*: in each instance, we leave out 20% of the links as the test set. Further, we randomly choose 1% of the non-existing links (nodes that are not connected) and include them in the test set. We then train our model on the remaining links, and evaluate the link prediction performance on the test set. To calculate the probability of user n_a in c_i following user n_b in c_j, we use:

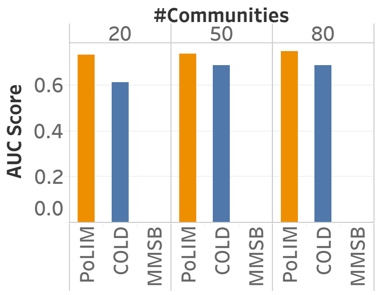
$$\Pr(n_a \to n_b) = v_a \rho A_i(n_b) + (1 - v_a) I_{ij} A_j(n_b) + v_a (1-\rho) \frac{1}{N} \tag{7.1}$$

The first term represents the case where n_a behaves from self interest and chooses a followee from his/her own community c_i; the second term measures the probability that n_a is influenced by c_j, and chooses a celebrity in c_j to follow; and finally, the third term represents the probability of a random following. Given the $\Pr(n_a \to n_b)$ value calculated from Eq. 7.1, we can compare it with a discrimination threshold value between 0 and 1 to predict whether the link exists or not. Hence, we use the AUC (area under the curve) metric for evaluation, which calculates the performance (true positive rate divided by false positive rate) at various threshold settings, and the area under this performance curve.

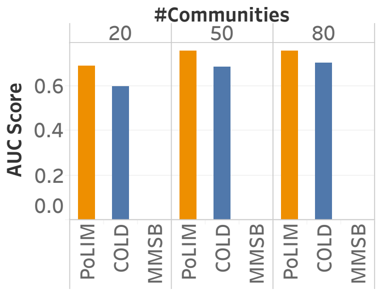
As we can see in Fig. 7.2, in all three settings where we vary the number of communities from low to high while keeping the number of topics constant (20), PoLIM consistently outperforms the baseline algorithms. MMSB performs the worst among all three, and it does not finish (converge) even in a day for the larger *Tweets-5%* and *Tweets-20%*. Note that our method also outperforms COLD in all different settings. This is expected since PoLIM combines both community-level influence (I) and personalized influence (v), while COLD only contains the former. Hence PoLIM predicts the link generation with higher accuracy.



(a) *Tweets-2%*



(b) *Tweets-5%*



(c) *Tweets-20%*

Figure 7.2: Link prediction results. Higher the AUC, better the performance (MMSB does not finish for *Tweets-5%* and *Tweets-20%*).

7.4.3 Q2: Retweet Volume Prediction

In this section, we show that PoLIM captures well how social influence affects the tweet generation. We design a retweet volume prediction task, where we train our model on data in a training time period, and then use the model parameters to predict how many tweets in the testing time period are retweets. We use a 7-fold cross validation on *Tweets-2%*, *Tweets-5%* and *Tweets-20%*. Repeatedly we leave one month's tweets for testing, and train PoLIM on the other six months' data. For each tweet (from user n_a in community c_i) in the testing data, we can calculate the probability of the tweet being generated from social influence using:

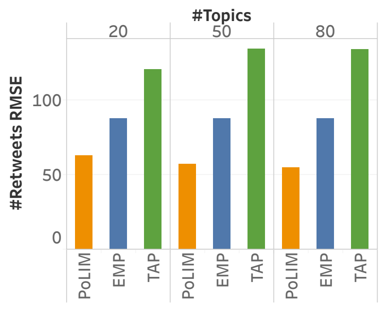
$$\begin{aligned}&\Pr(r=1|t) \propto \Pr(t|r=1) * \Pr(r=1)\\ &= (1-v_a)\sum_j I_{ij} \sum_{TopicM} \theta_j(TopicM) \prod_{w\in t}[u\phi_B(w) + (1-u)\phi_{TopicM}(w)] \qquad (7.2)\end{aligned}$$

where r is the switch value for the tweet ($r = 1$ means the tweet is caused by social influence). The latter part of the equation basically goes over all possible influencing communities and topics and calculates the likelihood of the tweet. Similarly we have:

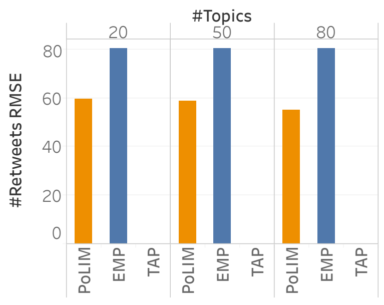
$$\Pr(r = 0|t) \propto v_a \sum_{TopicM} \theta_i(TopicM) \prod_{w \in t} [u\phi_B(w) + (1-u)\phi_{TopicM}(w)] \tag{7.3}$$

We then normalize these probabilities to get the final influence probability for the tweet. If this probability is greater than 0.5, we predict it as a retweet. Finally, we calculate the average RMSE value between all users' predicted number of retweets and the ground truth number obtained from the RT labels. Note that the word 'RT' is only used to obtain the labels for tweets; the word itself is filtered as stopword from the text corpus for training and testing.

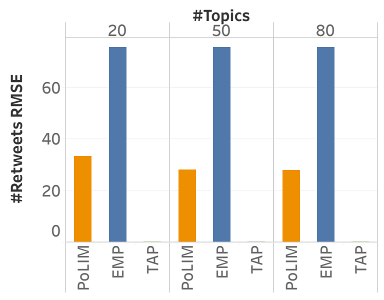
We see in Fig. 7.3, PoLIM achieves the best performance in all three datasets with different number of topics (number of communities fixed as 20). This is mainly because PoLIM directly models for each tweet if it is influenced. In contrast, TAP is designed to capture the overall topical influence and authority based on users' topic interests and their connections, and it turns out to perform badly when it is applied on a lower level for each individual tweet (it does not converge in a day for *Tweets-5%* and *Tweets-20%*). EMP also performs worse because it does not adapt to the text content of the testing tweets. Note that we can not use COLD as a baseline here, as it makes no distinction between retweets and tweets, and hence can not predict the number of retweets.



(a) *Tweets-2%*



(b) *Tweets-5%*



(c) *Tweets-20%*

Figure 7.3: Retweet volume prediction results. Lower the RMSE, better the performance (TAP does not finish for *Tweets-5%* and *Tweets-20%*).

7.4.4 Q3: Identifying Influenced Content

We identify the portion of tweets and links in *Tweets-Whole* that are caused by social influence, and show that PoLIM indeed learns beyond the weak supervision (i.e., the retweet labels). After running PoLIM on *Tweets-Whole*, similarly we use Eq. 7.1, Eq. 7.2, and Eq. 7.3 to calculate the influence probability of a tweet/link. We find that among a total of 27.5M tweets, there is a significant portion ($\sim 4.7\%$) of ~ 1.3M tweets which are not retweets but are actually identified as being influenced, and among a total of 2.1M connections, there is a large portion (27.3%) of $\sim$ 574K links that are affected by social influence. This shows the impact of social influence on both the tweet and link generation. Further, when we run PoLIM on *Tweets-Iran*, these portions of influenced tweets and links increase to 54% and 50.7% respectively. This shows that as the communities are all about the same event (Iran election), they have higher influence among them. We will discuss these communities more in Sec. 7.4.7.

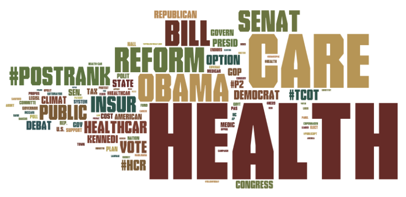
We want to stress that RT labels are used only as a weak supervision, and our model can learn beyond that and find non-retweets (tweets with no RT) that are likely to be caused by social influence. In Tab. 7.3, we show some examples of these tweets. They show a clear sign of getting influenced: they are either related to an external event (such as a live broadcast, or an interaction with a friend), or related to a URL about a specific product (like a converter tool, a demo link, etc.). Mixing these tweets that are actually caused by social influence with the other tweets for topic analysis would lead to inaccurate estimation of a user's interest, and may potentially harm other applications such as link prediction. To validate this, we run one of the baseline algorithms COLD (which does not handle influence the same way as we do) on the same link prediction task as in Sec. 7.4.2. By removing these influenced tweets from the *Tweets-Iran*, we observe an average AUC improvement of $\sim$ 0.013, over the same 5-fold cross validation.

Table 7.3: Example non-retweets that are influenced.

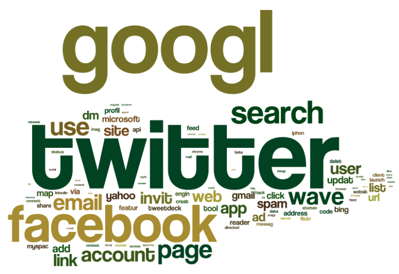
Tweet
Softwarevorstellung: Free YouTube to iPod and PSP Converter 3.1.4 http://bit.ly/u4eFv
My Twitter profile is worth $307 http://tweetvalue.com
Obama now speaking at #nerdprom2 on @cspan & http://cnn.com
Broadcasting live now! See me at http://bit.ly/14NLmL
@Welshbybirth Thanks for the #followfriday!
Hello from Chile (#foofighterslive live on http://bit.ly/3zR5dr)
Flash CS5 demo from Adobemax http://bit.ly/8qEyM
I have closed my account at this site that gets you followers: http://morefollowers.info

7.4.5 Q4: Quality of Topics

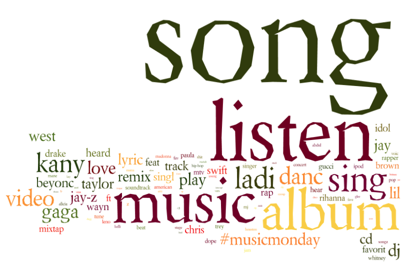
We show in Fig. 7.4 that PoLIM learns high quality topics on the entire tweet dataset *Tweets-Whole*. Due to the lack of space, we only show six example topics, and there are many other topics covering different domains such as economy, Iran election, traffic, sports, design, religious, energy, education, etc. Note that all the words in the word clouds are stemmed and lemmatized.



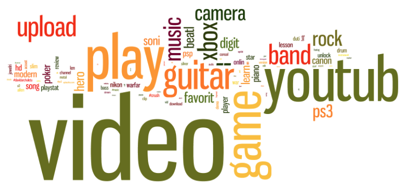
(a) Obama care



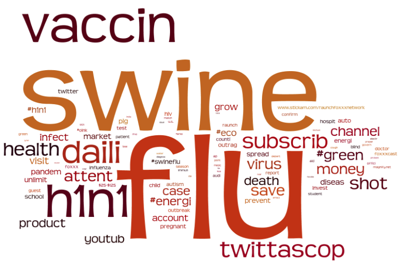
(b) Tech



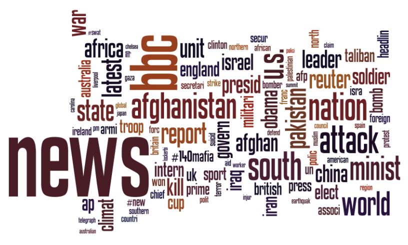
(c) Music



(d) Video



(e) Disease



(f) News

Figure 7.4: Word clouds for topics learned by PoLIM on the entire tweet data *Tweets-Whole* (visualized using Wordle, an online word cloud generator). For each topic, we show the top 100 words with highest weights. Each word is already stemmed and lemmatized. The layout of the word cloud is randomized, and the size of the word is proportional to its weight in the topic.

7.4.6 Q5: Influence Analysis

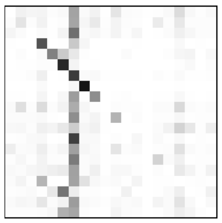
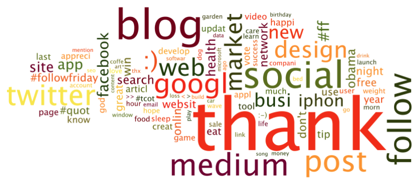


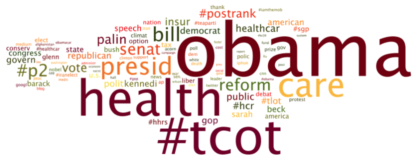
Figure 7.5: Matrix I learnt; darker the color, higher the value.

In this section, we examine in detail the influence PoLIM learned. At the community level, we first analyze the influence strengths among different communities, and the celebrities inside each community. Then, we analyze the influence at the individual level, where we show users' tendencies of being influenced.

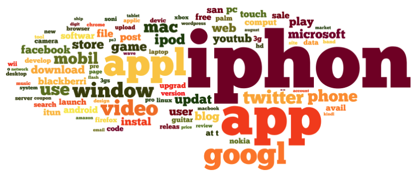
Community influence and celebrities. We show the social influence and the communities learned by PoLIM on *Tweets-Whole* in Fig. 7.5 and Fig. 7.6. We make several interesting observations. First, we observe from Fig. 7.5 that there exists a set of communities (c_3 to c_8) with high in-community influence, highlighted by the dark color of the diagonal of I. This means that for users in these communities, even when they are influenced, they tend to be influenced by users within the same community. Second, we identify a very influential community c_6 that influences almost all the other communities (the corresponding column in I has consistent high values). In the following, we look into several of these interesting communities for more detailed analysis.



(a) Topic interests of c_6



(b) Topic interests of c_3



(c) Topic interests of c_{10}

User Screenname	A_6 Value
mashable	0.0092
chrisbrogan	0.0051
problogger	0.0045
nansen	0.0041
brooksbayne	0.0038
the_gman	0.0036
chrispirillo	0.0034
garyvee	0.0034
stephenkruiser	0.0033
dcagle	0.0032
tedmurphy	0.0032

(d) Celebrities in c_6

User Screenname	A_3 Value
timoreilly	0.1446
nprnews	0.0820
anamariecox	0.0818
bbctech	0.0236
markknoller	0.0209
gleonhard	0.0187
politicalticker	0.0185
harrislacewell	0.0149
howardlindzon	0.0138
joanwalsh	0.0129
whitehouse_rss	0.0126

(e) Celebrities in c_3

User Screenname	A_{10} Value
engadget	0.0662
journalismnews	0.0432
shanselman	0.0350
hatebu	0.0343
twfeed	0.0326
pvponline	0.0317
crackberry	0.0237
theiphoneblog	0.0214
whiteafrican	0.0160
watch_akiba	0.0156
shauninman	0.0142

(f) Celebrities in c_{10}

Figure 7.6: The topic distributions for several communities and the celebrities (users with the highest A_i values) in these communities. For the most important topics in each community, we annotate them with the top three words (stemmed and lemmatized) in those topics.

First, we check the most influential community c_6. We plot the word clouds as the summaries for the communities we found in Fig. 7.6. The size of each word is proportional to a weighted importance of the word calculated by using the topic distribution of the community (θ) and the word distribution for the topic (ϕ). In Fig. 7.6(a), we observe that many frequent words used in c_6, such as 'design', 'busi', 'market', 'facebook' come from different disciplines and topics. In fact, the topic distribution of c_6 has a 'flat' shape, showing that this community is interested in a wide range of topics, including economy, politics, food, animal and Iphone. Combined with the fact that c_6 has influence over almost all other communities, this depicts the type of users who are influential in the social network, who respond to a wide range of topics, and they are more likely to be well-known figures/companies. This is confirmed by Fig 7.6(d)), where we show the celebrities (the users with top A_i values) in the community. The individuals with highest A_i values in c_6 are mashable (media and entertainment company), chrisbrogan (very highly rated influencer online), and problogger (a popular blog website). All of these celebrities in c_6 are famous online users with high influence over a wide range of topics. PoLIM correctly groups them as communities, and our influence matrix correctly captures their high influence over the other communities.

On the other hand, PoLIM also detects communities with very focused topic interests. Take c_3 for an example, we see that in Fig. 7.6(b)(e), c_3 shows a focused interest in Obama health care. Moreover, the 'local' celebrities in c_3, such as timeoreilly, politicalticker, whitehouse_rss, are consistent with the topic interest. We make similar observations for c_{10} in Fig. 7.6(c)(f), where the main topic interests are related to technology (keywords like 'window', 'appl', 'iphone', 'app', etc.), and the celebrities in c_{10} (such as engadget, theiphoneblog, crackberry) also show similar technology focus.

Celebrity structures. We examine the celebrity values in each community and find different celebrity structures for different communities. By examining the histogram and entropy of the A values for the top 20 celebrities in the communities, we make an interesting observation. For a community about a specific event, such as c_7 which is mainly about the Iran election, the importance/authority are more spread out to multiple users; while for a community about a general topic, like c_{14} which focuses on garden and art, a few users would be the leading authority and the others are much less influential. This leads to an insight that when a new topic/event emerges, different perspectives/arguments of the subject can be discussed, which offers more chances for users to be noticed and hence become influential. On the other hand, for a very developed and general topic, the 'heat' of the discussion has decreased to a stable level, and the authority has started to concentrate rather than diverge.

Individual's influence tendency. At the individual level, we show that PoLIM correctly learns the probability of a person being influenced. We pick two users (mashable and robertgoodwin) and check their actual tweets to further analyze our results PoLIM learns a $v = 0.98$ for mashable, which means that 98% of the times, mashable makes posts or follows users out of self interest, and it is rarely influenced by others, as one would expect from a large media content generator. From its tweets in Fig. 7.7, we do see that most of them are original tweets, and it covers many different topics. In contrast, user robertgoodwin, an IT

Time in secs	**Tweets from user mashable**
1302781.0	tweet win free vip ticket #140conf
1303395.0	you'r watch nba final 4 way make nba final social
1338254.0	hunch launch reinvent make decis
1346474.0	onli two hour left tweet win free vip ticket #140conf
1354831.0	reddit start job board whi
1361533.0	tip you'r new twitter know someon tri twitter list section
1369526.0	youtub continu time ad choic
1370893.0	green tweet 75+ environmentalist follow twitter #ecomonday
1392419.0	realiti tv show ink twitter name tattoo

(a) User mashable, $v = 0.98$, mostly original tweets

Time in secs	**Tweets from user robertgoodwin**
1426918.0	rt doe anybodi know good view launch beach ani beach melbourn area #nasa #sts127
1427333.0	rt @nasa launch offici edt wednesday guess space.com wa wrong 5:20 launch
1648039.0	brighter note you'r ever cocoa beach check italian courtyard sicilian thick crust pizza great
1798689.0	rt nasa might found defect caus leak juli 11
2034576.0	rt @nasa lcross live stream happen live stream lunar orbit
2141796.0	rt stun pictur hole cloud astronaut wit volcano erupt
2142023.0	insan relax condo loan rule haven't learn anyth rt wait ?...
2211875.0	rt @aldotcom nasa fund restor billion support

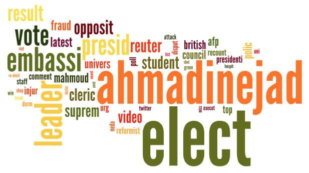
(b) User robertgoodwin, $v = 0.37$, mostly influenced by NASA (note the rt's)

Figure 7.7: Tweets from two example users with different v values learned.

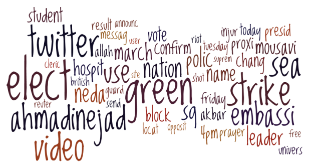
professional who works for NASA, shows a clear sign of influence from NASA in his tweets (60% of the tweets are retweets). PoLIM correctly learns a low $v = 0.37$ to capture this fact.

7.4.7 Q6: Case Study on Iran Election

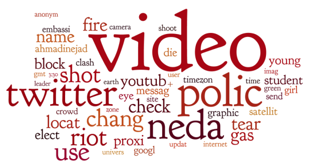
PoLIM can help understand and detect finer-grained communities even for a specific topic. In this experiment, we run PoLIM with 15 topics and 4 communities on *Tweets-Iran*. We show the word clouds (plotted in the same way as in Sec. 7.4.6) as the summaries for the communities we found, in Fig. 7.8. We observe that each community corresponds to a specific aspect of the event. c_1 is about the results or progress of the election itself; c_2 is about the Iranian green movement, a political movement that arose after the election to demand the removal of Mahmoud Ahmadinajad from office; c_3 is related to the video of the death of Neda, a student of philosophy, during the protest; and finally c_4 is mainly about the online petition during the election. These results show that PoLIM can also be applied on a specific event to discover communities with subtle difference.



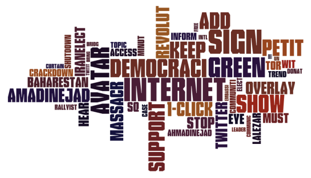
(a) c_1: Election news/progress



(b) c_2: Iranian green movement



(c) c_3: The video of Neda's death



(d) c_4: Online petition

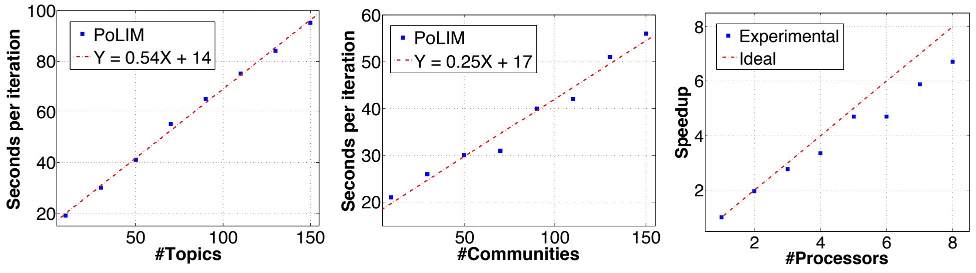
Figure 7.8: Word clouds for the communities detected in Iran election.

7.4.8 Scalability

Finally, to examine the running time of PoLIM, we vary two of the input parameters: the number of topics and communities. We observe that in Fig. 7.9(a)(b), the running time scales linearly w.r.t. both #topics and #communities as expected from the complexity of the algorithm. We also observe we get *near-linear* speedup from parallelizing the inference algorithm.

7.5 Discussion and Conclusions

To summarize our observations, PoLIM learns real communities with meaningful topic interests, as well as the celebrities with high influence in each communities. Broadly, it extracts



(a) Varying #Topics (b) Varying #Communities (c) Parallelization

Figure 7.9: PoLIM scales linearly w.r.t. the number of topics and communities, and parallelization gives near-linear speed-ups.

well the social influence strength among different communities. At the same time, at a finer level, it also correctly learns a person's tendency of being influenced.

Our experimental results also confirm the existence of social influence in real datasets, and its impact on various applications. We find that the same latent social influence governs a significant portion of posts and links in Twitter, and such a portion tends to increase when the dataset is about more specific events. By differentiating these posts and links from those caused by self interests, PoLIM is able to learn more precise topic interests, and therefore achieve better performance in other tasks such as predicting future links and retweets.

Note that one of the most important hypotheses we used in PoLIM is that retweets are more likely to be caused by social influence. This hypothesis then guides the learning of social influence in PoLIM in a weakly supervised fashion. While retweets are good indicators of influence in Twitter, they may not be available for other social media websites. In these cases, as future work, we may design other indicators such as the replies, mentions, or even train a low-cost low-accuracy feature-based classifier as the weak supervision for PoLIM.

Chapter 8

Conclusion and Future Work

In this research, we exploit the within-sequence and across-sequence correlations to improve the segmentation and prediction performance in a variety of problems from different domains. We summarize the main idea and conclusions in each chapter below.

- **Chapter 3: 'Temporal closeness' for better segmenting multi-dimensional value sequences.** We use the 'temporal closeness' between data values (the similarity of the data values' temporal distribution in the sequence) for segmenting multi-dimensional value sequences. Our algorithm is general: each data value can have arbitrary time stamps, and it improves the segmentation performance.
- **Chapter 4: Latent state modeling for better tweet prediction.** We capture the state transition behind tweet sequences to better summarize and understand the Twitter data. When we apply our models on a flu-related tweet dataset, we show that the results detected by our algorithms help fill the gap between phenomenological methods for disease surveillance and epidemiological models, reconciling some contrasting behaviors between epidemiological and social models.
- **Chapter 5: Exploiting the correlation network for better segmenting multiple value sequences.** We use the correlation behind sequences to convert multiple sequences to an attributed network sequence. Our segmentation algorithm considers nodes with all possible label values rather than focusing on only one value (as is done by most community-based dynamic graph algorithms). Further, our algorithm works in a parameter-free manner that automatically detects the number of segments in the optimal segmentation.
- **Chapter 6: Spatial relations for better explaining the segmentation results.** We design a framework which not only gives the segmentation of the sequences, but also an intuitive explanation of the segmentation itself. Such explanations (which respect the spatial distance behind different time series) help us better understand the detected

changes by pin-pointing the culprit sequences. It also facilitates quick utilization and deployment of the segmentation results in real applications.

- **Chapter 7: Social influence modeling for better summarizing and understand Social Media data.** We jointly model tweets and links in the Twitter data, as well as how each tweet and link is affected by social influence. Such fine-grained modeling introduces big challenges to inferring the model parameters; at the same time, it leads to better understanding of the data and more precise prediction for both the tweets and links.

Our work has been consistently adopted in real applications. The URBANNET toolkit that we developed for monitoring, simulating and analyzing Critical Infrastructure Systems is being used and licensed by Oak Ridge National Lab experts. We are also planning on merging the CUT-n-REVEAL framework for analyzing power outages with the current URBANNET toolkit. Our HFSTM and HFSTM-A models for flu case surveillance also contribute to the EMBERS project in the Discovery Analytics Center at Virginia Tech, which aims to forecast significant events using open source data.

8.1 Future Work

Our algorithms and models have achieved good performance in multiple the real applications in different domains. However, there are still many directions to explore. We discuss several such directions for further extending our work in the following.

8.1.1 Integrating Additional Temporal and Spatial Correlations

One natural way to extend our work is to expand our algorithms and models for more types of correlations. In the context of social media, the Twitter data contains rich information such as the time stamp of a tweet and the geo-location where the tweet is generated. This information can be integrated into our current models (HFSTM, HFSTM-A, and PoLIM) to make the models more comprehensive. For example, we can combine a Hawkes process with HFSTM to model the exact time stamps of tweets. It may enrich our state transition analysis by answering how much time a user typically stay in a state. We can also try to model location-specific topics: the same topic (such as technology) has different work distributions in different locations. By integrating such location information, we would be able to see how people in different locations may use different words in the same state.

8.1.2 Consolidating Generative Modeling and Segmentation

Another direction to extend our work is to consolidate our generative models for text sequences with the segmentation algorithms for multi-dimensional value sequences. There are several possible ways to do this. One idea to unify these two lines of work is to model the segmentations themselves. By directly modeling the segmentation, the model will automatically learn segmentations of the text sequence as well. Another idea is to use popular embedding methods (such as word2vec) to convert text/sentences to multi-dimensional value representations. Therefore we would be able to use the same segmentation framework for both value and text sequences.

8.1.3 Connecting Our Generative Models to Real Applications

In the past decade, there have been much work on modeling social media data, each having different hypotheses and models. The typical metrics we have used to evaluate and compare these different models include direct comparison of the likelihood of the data, or indirect comparison of the models' performance in applications such as link and tweet prediction. However, sometimes we find these metrics inadequate in fully justifying the models and the underlying hypothesis. For example, a typical cross-validation for link/tweet prediction would save part of the data for testing and the rest for training, which is not a realistic setting in practice. To design a more rigorous evaluation metric, one idea is to tie the model to a specific application. As we have done for HFSTM and HFSTM-A, the models are designed for a specific task of flu trend prediction; the evaluation of the model thus becomes straight-forward: the model that gives the least fitting error is the best model for the task. Similarly, as future work, we can attach PoLIM to a real application to evaluate the model more rigorously. For example, applying the social influence detected by PoLIM for product promotion/meme propagation is an interesting extension that we can explore.

8.1.4 Introducing Additional Domain Constraints from CIS

As critical infrastructure systems are interdependent in very complicated ways, we inevitably make simplified assumptions in our work. For example, in our CUT-n-REVEAL framework, we only consider the geo-spatial constraint between counties in finding the explanation of the segmentation: counties that are near each other should have similar importance. However, different counties can be correlated in more subtle ways based on the underlying CIS structures. The major power generator in one county may depend on the water supply in another county for its cooling system. Therefore, expanding our framework to allow easy adoption of additional domain constraints is a very useful extension of our work.

Bibliography

[1] H. Achrekar, A. Gandhe, R. Lazarus, S.-H. Yu, and B. Liu. Predicting flu trends using Twitter data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 702–707. IEEE, 2011.

[2] C. Aggarwal and K. Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):10, 2014.

[3] C. C. Aggarwal and N. Li. On node classification in dynamic content-based networks. In *SDM*, 2011.

[4] C. C. Aggarwal and S. Y. Philip. Online analysis of community evolution in data streams. In *SDM*, pages 56–67. SIAM, 2005.

[5] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)*, 6(2):9, 2012.

[6] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

[7] R. Albert, I. Albert, and G. L. Nakarado. Structural vulnerability of the North American power grid. *Phys. Rev. E*, 69:025103, 2004.

[8] S. E. Amiri, L. Chen, and B. A. Prakash. Snapnets: Automatic segmentation of network sequences with node labels. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3–9, 2017.

[9] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *KDD*. ACM, 2008.

[10] R. M. Anderson and R. M. May. *Infectious Diseases of Humans.* Oxford University Press, 1991.

[11] M. Andrews and G. Vigliocco. The Hidden Markov Topic Model: A Probabilistic Model of Semantic Representation. *Topics in Cognitive Science*, 2(1):101–113, 2010.

[12] E. Aramaki, S. Maskawa, and M. Morita. Twitter Catches the Flu: Detecting Influenza Epidemics Using Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1568–1576, 2011.

[13] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. Com2: Fast automatic discovery of temporal ('comet') communities. In *PAKDD*, 2014.

[14] O. Banos, M. A. Toth, M. Damas, H. Pomares, and I. Rojas. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors*, 14(6):9995–10023, 2014.

[15] A. M. Barker, E. B. Freer, O. A. Omitaomu, S. J. Fernandez, S. Chinthavali, and J. B. Kodysh. Automating natural disaster impact analysis: An open resource to visually estimate a hurricane's impact on the electric grid. In *Southeastcon*, pages 1–3, 2013.

[16] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. KDD '12, pages 280–288, 2012.

[17] V. Belák, S. Lam, and C. Hayes. Cross-community influence in discussion fora. In *ICWSM*, 2012.

[18] E. Beretta and Y. Takeuchi. Global stability of an SIR epidemic model with time delays. *Journal of mathematical biology*, 33(3):250–260, 1995.

[19] B. Bi, Y. Tian, Y. Sismanis, A. Balmin, and J. Cho. Scalable topic-specific influence analysis on microblogs. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 513–522. ACM, 2014.

[20] E. S. Blake, T. B. Kimberlain, R. J. Berg, J. P. Cangialosi, and J. L. Beven Ii. Tropical Cyclone Report: Hurricane Sandy. *National Hurricane Center*, 12:1–10, 2013.

[21] S. Blasiak and H. Rangwala. A Hidden Markov Model Variant for Sequence Classification. In *The 21nd International Joint Conference on Artificial Intelligence*, pages 1192–1197, 2011.

[22] D. Blei, L. Carin, and D. Dunson. Probabilistic Topic Models. *Signal Processing Magazine, IEEE*, 27(6):55–65, 2010.

[23] D. Blei and J. Lafferty. Dynamic Topic Models. In *the 23rd International Conference on Machine Learning*, pages 113–120, 2006.

[24] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[25] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 946–957. SIAM, 2014.

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[27] S. P. Brennan, A. Sadilek, and H. A. Kautz. Towards understanding global spread of disease from everyday interpersonal interactions. In *IJCAI*, pages 2783–2789, 2013.

[28] S. V. Buldyrev, N. W. Shere, and G. A. Cwilich. Interdependent networks with identical degree of mutually dependent nodes. *Physical Review*, 83(1), 2011.

[29] D. Butler. When Google got Flu Wrong. *Nature*, 494(7436):155–156, 2013.

[30] P. Chakraborty, P. Khadivi, B. Lewis, A. Mahendiran, J. Chen, P. Butler, E. Nsoesie, S. Mekaru, J. Brownstein, M. Marathe, and N. Ramakrishnan. Forecasting a Moving Target: Ensemble Models for ILI Case Count Predictions. SDM '14, 2014.

[31] C. Chen, J. He, N. Bliss, and H. Tong. On the connectivity of multi-layered networks: Models, measures and optimal control. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 715–720. IEEE, 2015.

[32] L. Chen, S. E. Amiri, and B. A. Prakash. Automatic Segmentation of Data Sequences. In *AAAI, 2018*.

[33] L. Chen, K. S. M. T. Hossain, P. Butler, N. Ramakrishnan, and B. A. Prakash. Flu Gone Viral: Syndromic Surveillance of Flu on Twitter Using Temporal Topic Models. In *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pages 755–760, 2014.

[34] L. Chen, K. S. M. T. Hossain, P. Butler, N. Ramakrishnan, and B. A. Prakash. Syndromic surveillance of Flu on Twitter using weakly supervised temporal topic models. *Data Min. Knowl. Discov.*, 30(3):681–710, 2016.

[35] L. Chen, N. Muralidhar, S. Chinthavali, N. Ramakrishnan, and B. A. Prakash. Segmentations with explanations for outage analysis. *Computer Science Technical Reports TR-18-02, VTechWorks*, 2018.

[36] L. Chen and B. A. Prakash. Modeling influence using weak supervision: A joint link and post-level analysis. *Computer Science Technical Reports TR-18-03, VTechWorks*, 2018.

[37] L. Chen, X. Xu, S. Lee, S. Duan, A. G. Tarditi, S. Chinthavali, and B. A. Prakash. Hotspots: Failure cascades on heterogeneous critical infrastructure networks. In *CIKM*, pages 1599–1607. ACM, 2017.

[38] W. Chen, T. Lin, Z. Tan, M. Zhao, and X. Zhou. Robust influence maximization. In *KDD*, pages 795–804, 2016.

[39] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[40] X. C. Chen, K. Steinhaeuser, S. Boriah, S. Chatterjee, and V. Kumar. Contextual time series change detection. In *SDM*, 2013.

[41] S. Chiappa. A Bayesian Approach to Switching Linear Gaussian State-Space Models for Unsupervised Time-Series Segmentation. *ICMLA*, pages 3–9, 2008.

[42] N. A. Christakis and J. H. Fowler. Social Network Sensors for Early Detection of Contagious Outbreaks. *PLoS ONE*, (9), 09 2010.

[43] P. Cohen, B. Heeringa, and N. Adams. Unsupervised segmentation of categorical time series into episodes. In *ICDM*, pages 99–106. IEEE, 2002.

[44] R. Crane and D. Sornette. Robust Dynamic Classes Revealed by Measuring the Response Function of a Social System. In *PNAS*, 2008.

[45] A. Culotta. Towards detecting influenza epidemics by analyzing Twitter messages. In *Proceedings of the first workshop on social media analytics*, pages 115–122. ACM, 2010.

[46] H. Daneshmand, M. Gomez-Rodriguez, L. Song, and B. Schoelkopf. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *ICML*, pages 793–801, 2014.

[47] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 2013.

[48] S. Duan, S. Lee, S. Chinthavali, and M. Shankar. Reliable communication models in interdependent critical infrastructure networks. In *Resilience Week (RWS), 2016*, pages 152–157. IEEE, 2016.

[49] L. Dueas-Osorio, J. I. Craig, and B. J. Goodno. Seismic response of critical interdependent networks. *Earthquake Engineering and Structural Dynamics*, 36(2):285–306, 2007.

[50] R. Eskandarpour and A. Khodaei. Machine learning based power grid outage prediction in response to extreme events. *IEEE Transactions on Power Systems*, 32(4):3315–3316, 2017.

[51] W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *SIGMOD*, 2012.

[52] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik. Monitoring network evolution using MDL. In *ICDE*. IEEE, 2008.

[53] D. Fernandez and M. Lutz. Urban flood hazard zoning in Tucumán Province, Argentina, using GIS and multicriteria decision analysis. *Engineering Geology*, 111(1):90–98, 2010.

[54] A. Gensler, T. Gruber, and B. Sick. Blazing Fast Time Series Segmentation Based on Update Techniques for Polynomial Approximations. *ICDM Workshops*, pages 1002–1011, 2013.

[55] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh. Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *ICDM*, pages 117–126. IEEE, 2017.

[56] J. Ginsberg, M. Mohebbi, R. Patel, L. Brammer, M. Smolinski, and L. Brilliant. Detecting Influenza Epidemics using Search Engine Query Data. *Nature*, 457(7232):1012–1014, 2008.

[57] N. Glance, M. Hurst, and T. Tomokiyo. Blogpulse: Automated trend discovery for weblogs. *WWW 2004 workshop on the weblogging ecosystem: Aggregation, analysis and dynamics*, 2004.

[58] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[59] A. Gruber, M. Rosen-Zvi, and Y. Weiss. Hidden Topic Markov Models. *Artificial Intelligence and Statistics (AISTATS)*, 2007.

[60] P. D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[61] M. F. Habib, M. Tornatore, and B. Mukherjee. Cascading-failure-resilient interconnection for interdependent power grid-optical networks. In *OFC*, 2015.

[62] D. Hallac, S. Vare, S. Boyd, and J. Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *KDD*, pages 215–223. ACM, 2017.

[63] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong. Metric forensics: A multi-level approach for mining volatile graphs. In *KDD*, 2010.

[64] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42, 2000.

[65] P. Hines, K. Balasubramaniam, and E. C. Sanchez. Cascading failures in power grids. *IEEE Potentials*, 28(5), 2009.

[66] P. D. Hines, I. Dobson, and P. Rezaei. Cascading power outages propagate locally in an influence graph that is not the actual grid topology. *IEEE Transactions on Power Systems*, 32(2):958–967, 2017.

[67] Å. J. Holmgren. Using graph models to analyze the vulnerability of electric power networks. *Risk analysis*, 26(4):955–969, 2006.

[68] L. Hong, D. Yin, J. Guo, and B. Davison. Tracking Trends: Incorporating Term Volume into Temporal Topic Models. In *the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 484–492, 2011.

[69] K. Hou, W. c. Feng, and S. Che. Auto-Tuning Strategies for Parallelizing Sparse Matrix-Vector (SpMV) Multiplication on Multi- and Many-Core Processors. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2017.

[70] K. Hou, W. Liu, H. Wang, and W. Feng. Fast Segmented Sort on GPUs. In *Proceedings of the 2017 International Conference on Supercomputing*, ICS '17. ACM, 2017.

[71] K. Hou, H. Wang, and W. c. Feng. Delivering Parallel Programmability to the Masses via the Intel MIC Ecosystem: A Case Study. In *the 43rd International Conference on Parallel Processing Workshops*, pages 273–282, Sept 2014.

[72] K. Hou, H. Wang, and W. Feng. ASPaS: A Framework for Automatic SIMDization of Parallel Sorting on x86-based Many-core Processors. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, ICS '15, pages 383–392, New York, NY, USA, 2015. ACM.

[73] K. Hou, H. Wang, and W. Feng. GPU-UniCache: Automatic Code Generation of Spatial Blocking for Stencils on GPUs. In *Proceedings of the ACM Conference on Computing Frontiers*, CF '17. ACM, 2017.

[74] K. Hou, H. Wang, and W. Feng. A Framework for the Automatic Vectorization of Parallel Sort on X86-based Processors. *IEEE Trans. Parallel Distrib. Syst. (TPDS)*, 2018.

[75] K. Hou, H. Wang, W. Feng, J. Vetter, and S. Lee. Highly Efcient Compensation-based Parallelism for Wavefront Loops on GPUs. In *IEEE Int. Parallel and Distrib. Process. Symp. (IPDPS)*, 2018.

[76] K. Hou, H. Wang, and W. C. Feng. AAlign: A SIMD Framework for Pairwise Sequence Alignment on x86-Based Multi-and Many-Core Processors. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 780–789, May 2016.

[77] Z. Hu, J. Yao, B. Cui, and E. Xing. Community level diffusion extraction. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1555–1569. ACM, 2015.

[78] X. Huang, S. Shao, H. Wang, S. V. Buldyrev, H. E. Stanley, and S. Havlin. The robustness of interdependent clustered networks. *EPFA*, 101(1), 2013.

[79] J. Jacquez and C. Simon. The stochastic SI model with recruitment and deaths I: Comparison with the closed SIS model. *Mathematical biosciences*, 117(1):77–125, 1993.

[80] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146. ACM, 2003.

[81] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM'01*.

[82] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM Trans. Knowl. Discov. Data*, 3(4), 2009.

[83] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[84] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. VOG: Summarizing and Understanding Large Graphs. In *SDM*, 2014.

[85] H.-P. Kriegel, P. Kroger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. ICDM, 2005.

[86] T. La Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, pages 601–610. ACM, 2010.

[87] A. Lamb, M. J. Paul, and M. Dredze. Separating fact from fear: Tracking flu infections on twitter. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2013.

[88] V. Lampos and N. Cristianini. Nowcasting events from the social web with statistical learning. *ACM TIST*, 3(4):72, 2012.

[89] V. Lampos, T. De Bie, and N. Cristianini. Flu Detector: Tracking Epidemics on Twitter. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, pages 599–602, 2010.

[90] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *KDD*, 2010.

[91] D. M. Lazer, R. Kennedy, G. King, and A. Vespignani. The parable of Google Flu: Traps in big data analysis. *Science*, 343(6176):1203–1205, 2014.

[92] C.-T. Lee. GIS Application in Landslide Hazard Analysis–An Example from the Shihmen Reservoir Catchment Area in Northern Taiwan. In *Pacific Neighborhood Consortium (PNC) 2008 Annual Meeting program*, 2008.

[93] K. Lee, A. Agrawal, and A. Choudhary. Real-Time Digital Flu Surveillance using Twitter Data. In *Proceedings of the SDM Workshop on Data Mining for Medicine and Healthcare (DMMH)*, 2013.

[94] K. Lee, A. Agrawal, and A. Choudhary. Real-Time Disease Surveillance using Twitter Data: Demonstration on Flu and Cancer. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 1474–1477. ACM, 2013.

[95] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, 2009.

[96] G. Li, M. Semerci, B. Yener, and M. J. Zaki. Effective graph classification based on topological and label attributes. *Statistical Analysis and Data Mining*, 2012.

[97] J. Li and C. Cardie. Early Stage Influenza Detection from Twitter. *CoRR*, abs/1309.7340, 2013.

[98] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD*, 2009.

[99] M. Li and J. Muldowney. Global stability for the SEIR model in epidemiology. *Mathematical Biosciences*, 125(2):155–164, 1995.

[100] W. Li, A. Bashan, S. V. Buldyrev, H. E. Stanley, and S. Havlin. Cascading failures in interdependent lattice networks: the critical role of the length of dependency links. *Physical Review Letters*, 108(22), 2011.

[101] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[102] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. Mining topic-level influence in heterogeneous networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 199–208. ACM, 2010.

[103] L. Liu, J. Tang, J. Han, and S. Yang. Learning influence from heterogeneous social networks. *Data Mining and Knowledge Discovery*, 25(3):511–544, 2012.

[104] W. Liu, A. Kan, J. Chan, J. Bailey, C. Leckie, J. Pei, and R. Kotagiri. On compressing weighted time-evolving graphs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2319–2322. ACM, 2012.

[105] X. Liu, Z. Lin, and H. Wang. Novel Online Methods for Time Series Segmentation. *Knowledge and Data Engineering, IEEE Transactions on*, 20(12):1616–1626, 2008.

[106] P. V. Marsden and N. E. Friedkin. Network studies of social influence. *Sociological Methods & Research*, 22(1):127–151, 1993.

[107] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD*, 2011.

[108] M. Mathioudakis and N. Koudas. TwitterMonitor: trend detection over the Twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 1155–1158, New York, NY, USA, 2010. ACM.

[109] Y. Matsubara, Y. Sakurai, and C. Faloutsos. AutoPlait: Automatic Mining of Co-evolving Time Sequences. SIGMOD '14, pages 193–204, 2014.

[110] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa. Fast mining and forecasting of complex time-stamped events. KDD '12, pages 271–279, 2012.

[111] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 6–14, 2012.

[112] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen. CSI: Community-Level Social Influence Analysis. In *Proceedings, Part II, of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8189*, ECML PKDD 2013, 2013.

[113] A. Mueen and E. Keogh. Online discovery and maintenance of time series motifs. KDD '10, pages 1089–1098, 2010.

[114] N. Muralidhar, C. Wang, N. Self, M. Momtazpour, K. Nakayama, R. Sharma, and N. Ramakrishnan. Illiad: InteLLigent Invariant and Anomaly Detection in Cyber-Physical Systems. *ACM TIST*, 9(3):35, 2018.

[115] S. A. Myers, C. Zhu, and J. Leskovec. Information diffusion and external influence in networks. In *KDD*. ACM, 2012.

[116] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 542–550. ACM, 2008.

[117] H.-V. Nguyen and J. Vreeken. Linear-time detection of non-linear changes in massively high dimensional time series. In *SDM*. SIAM, 2016.

[118] M. H. Nguyen and F. Torre. Maximum margin temporal clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 520–528, 2012.

[119] X. Ning and G. Karypis. SLIM: Sparse linear methods for top-n recommender systems. In *ICDM*, pages 497–506. IEEE, 2011.

[120] M. Ouyang. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability Engineering and System Safety*, 121:43–60, 2014.

[121] PAHO. Epidemic Disease Database, Pan American Health Organization. `http://ais.paho.org/phip/viz/ed\_flu.asp`, Dec. 2012.

[122] A. Pal and S. Counts. Identifying topical authorities in microblogs. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 45–54. ACM, 2011.

[123] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB'05*.

[124] M. Parandehgheibi and E. Modiano. Robustness of interdependent networks: the case of communication networks and the power grid. In *GLOBECOM*, pages 2164–2169, 2013.

[125] M. Parandehgheibi and E. Modiano. Robustness of bidirectional interdependent networks: Analysis and design. *arXiv preprint arXiv:1605.01262*, 2016.

[126] D. Patnaik, S. Laxman, B. Chandramouli, and N. Ramakrishnan. Efficient episode mining of dynamic event streams. ICDM '2012.

[127] M. Paul and M. Dredze. You Are What You Tweet: Analyzing Twitter for Public Health. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*, 2011.

[128] M. Paul and R. Girju. A Two-dimensional Topic-aspect Model for Discovering Multifaceted Topics. *Urbana*, 51:61801, 2010.

[129] M. J. Paul and M. Dredze. You are what you tweet: Analyzing twitter for public health. *Icwsm*, 20:265–272, 2011.

[130] P. Pederson, D. Dudenhoeffer, S. Hartley, and M. Permann. Critical infrastructure interdependency modeling: a survey of US and international research. *Idaho National Laboratory*, pages 1–20, 2006.

[131] B. A. Prakash, D. Chakrabarti, N. C. Valler, M. Faloutsos, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. *KAIS*, 2012.

[132] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *KDD*, 2014.

[133] M. Qiu, F. Zhu, and J. Jiang. It is not just what we say, but how we say them: LDA-based behavior-topic model. In *2013 SIAM International Conference on Data Mining (SDM13). SIAM*. SIAM, 2013.

[134] Q. Qu, C. Chen, C. S. Jensen, and A. Skovsgaard. Space-time aware behavioral topic modeling for microblog posts. *IEEE Data Eng. Bull*, 38(2):58–67, 2015.

[135] Q. Qu, S. Liu, C. S. Jensen, F. Zhu, and C. Faloutsos. Interestingness-driven diffusion process summarization in dynamic networks. In *ECML PKDD*, pages 597–613, 2014.

[136] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. KDD '12, pages 262–270, 2012.

[137] N. Ramakrishnan, P. Butler, S. Muthiah, N. Self, R. Khandpur, P. Saraf, W. Wang, J. Cadena, A. Vullikanti, G. Korkmaz, et al. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1799–1808. ACM, 2014.

[138] N. Ramakrishnan, S. Tadepalli, L. T. Watson, R. F. Helm, M. Antoniotti, and B. Mishra. Reverse engineering dynamic temporal models of biological processes and their relationships. *Proceedings of the National Academy of Sciences*, 107(28):12511–12516, 2010.

[139] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz. On the accuracy of appliance identification based on distributed load metering data. In *SustainIT (2012)*, pages 1–9. IEEE, 2012.

[140] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016.

[141] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *Control Systems, IEEE*, 21(6):11–25, 2001.

[142] D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704, 2011.

[143] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[144] A. Sadilek, H. Kautz, and V. Silenzio. Predicting Disease Transmission from Geo-Tagged Micro-Blog Data. In *the 26th AAAI Conference on Artificial Intelligence*, 2012.

[145] A. Samé and G. Govaert. Online Time Series Segmentation Using Temporal Mixture Models and Bayesian Model Selection. *ICMLA '12*, 1:602–605.

[146] P. Sarkar, D. Chakrabarti, and M. I. Jordan. Nonparametric link prediction in dynamic networks. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[147] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *KDD*, 2015.

[148] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[149] M. Shokoohi-Yekta, Y. Chen, B. Campana, B. Hu, J. Zakaria, and E. Keogh. Discovery of meaningful rules in time series. In *KDD'15*, pages 1085–1094.

[150] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. SIGIR, pages 208–215, 2000.

[151] N. Spasojevic, J. Yan, A. Rao, and P. Bhattacharyya. LASTA: Large Scale Topic Assignment on Multiple Social Networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1809–1818, New York, NY, USA, 2014. ACM.

[152] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic Author-topic Models for Information Discovery. In *The 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315, 2004.

[153] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. KDD, 2007.

[154] S. Swarup, S. G. Eubank, and M. V. Marathe. Computational epidemiology as a challenge domain for multiagent systems. AAMAS '14, 2014.

[155] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816. ACM, 2009.

[156] N. Tassetti, A. Bernardini, and E. S. Malinverni. Use of remote sensing data and GIS technology for assessment of landslide hazards in Susa valley, Italy. *EARSeL eProceedings*, 7(1):59–67, 2008.

[157] N. Tatti. Fast sequence segmentation using log-linear models. *Data Mining and Knowledge Discovery*, 27(3):421–441, 2013.

[158] N. Tatti and J. Vreeken. The long and the short of it: Summarising event sequences with serial episodes. KDD '12, pages 462–470, 2012.

[159] W. W. Thompson, L. Comanor, and D. K. Shay. Epidemiology of seasonal influenza: use of surveillance data and statistical models to estimate the burden of disease. *Journal of Infectious Diseases*, 194(Supplement 2):S82–S91, 2006.

[160] S. Tierney, J. Gao, and Y. Guo. Subspace clustering for sequential data. In *Proceedings of IEEE CVPR*, pages 1019–1026, 2014.

[161] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[162] M. Toyoda, Y. Sakurai, and Y. Ishikawa. Pattern discovery in data streams under the time warping distance. *The VLDB Journal*, 22(3):295–318, 2013.

[163] V. S. Tseng, C.-H. Chen, P.-C. Huang, and T.-P. Hong. A cluster-based genetic approach for segmentation of time series and pattern discovery. *IEEE Congress on Evolutionary Computation*, pages 1949–1953, 2008.

[164] W. Ugulino, D. Cardador, K. Vega, E. Velloso, R. Milidiu, and H. Fuks. Wearable computing: Accelerometers' data classification of body postures and movements. In *SBIA 2012*, pages 52–61. 2012.

[165] M. Van Leeuwen and A. Siebes. Streamkrimp: Detecting change in data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 672–687. Springer, 2008.

[166] J. Waggoner, S. Wang, D. Salvi, and J. Zhou. Handwritten text segmentation using average longest path algorithm. WACV, 2013.

[167] H. Wang, W. Liu, K. Hou, and W. Feng. Parallel Transposition of Sparse Data Structures. In *Proceedings of the 2016 International Conference on Supercomputing*, ICS '16, pages 33:1–33:13, New York, NY, USA, 2016. ACM.

[168] P. Wang, H. Wang, and W. Wang. Finding semantics in time series. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 385–396, 2011.

[169] X. Wang and A. McCallum. Topics Over Time: a non-Markov Continuous-time Model of Topical Trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 424–433, 2006.

[170] Y. Wang, E. Agichtein, and M. Benzi. TM-LDA: efficient online modeling of latent topic transitions in social media. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 123–131. ACM, 2012.

[171] G. Weiss. Timeweaver: A genetic algorithm for identifying predictive patterns in sequences of events. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 718–725. Citeseer, 1999.

[172] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[173] C.-W. Wu, Y.-F. Lin, P. S. Yu, and V. S. Tseng. Mining high utility episodes in complex event sequences. KDD '13, pages 536–544, 2013.

[174] K. S. Xu, M. Kliger, and A. O. Hero, III. Tracking communities in dynamic social networks. SBP'11, pages 219–226, Berlin, Heidelberg, 2011. Springer-Verlag.

[175] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, 2011.

[176] J. Yang, J. J. McAuley, J. Leskovec, P. LePendu, and N. Shah. Finding progression stages in time-evolving event sequences. In *WWW '14*, 2014.

[177] S.-H. Yang, A. Kolcz, A. Schlaikjer, and P. Gupta. Large-scale High-precision Topic Modeling on Twitter. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1907–1916, New York, NY, USA, 2014. ACM.

[178] Y. Ye and E. Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical programming*, 44(1-3):157–179, 1989.

[179] D. Zhang, H. Wang, K. Hou, J. Zhang, and W. Feng. pDindel: Accelerating InDel Detection on a Multicore CPU Architecture with SIMD. In *2015 IEEE 5th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*, pages 1–6, Oct 2015.

[180] N. R. Zhang and D. O. Siegmund. Model selection for high-dimensional, multi-sequence change-point problems. *Statistica Sinica*, 2012.

[181] S. Zhao, L. Zhong, J. Wickramasuriya, and V. Vasudevan. Human as Real-Time Sensors of Social and Physical Events: A Case Study of Twitter and Sports Games. *CoRR*, abs/1106.4300, 2011.

[182] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing Twitter and traditional media using topic models. In *Advances in Information Retrieval*, pages 338–349. Springer, 2011.

[183] Y. Zhu, X. Yan, L. Getoor, and C. Moore. Scalable text and link analysis with mixed-topic link models. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 473–481. ACM, 2013.

Appendix A

DASSA (Chapter 3)

A.1 Preliminaries

First we provide the symbols used in the paper in Table A.1.

Symbol	Description
$\mathbb{D}$	A set of ordered pair $(\mathbf{x}_i, t_i)$. i is the index of each data point. It also used as discretized data sequence later in the paper.
$\mathbf{x}_i$	A $d \times 1$ vector of data points
t_i	Time stamp of each data point. $t_{min} \leq t < t_{max}$
X	The set of unique data values $(\mathbf{x}_i)$ in $\mathbb{D}$
N	Number of data points
d	Number of dimensions
k	Number of blocks in each dimension when discretizing the dataset
$\tilde{x}$	A data cluster
$\tilde{X}$	The set of clusters
s_{min}	The minimum length of a time segment
Y	The set of all possible time segments, considering s_{min} and s_{max}
$y_{i,j}$	A time segment in Y, with the starting time c_i and ending time c_j
$D_{KL}[p\|\|q]$	Kullback-Leiber divergence of p & q
s_{mdl}	The step size used in calculating $Cost_T$
$Cost_T$	The total MDL cost

Table A.1: Symbols and notation.

Algorithm 9 Pseudo-code of Cluster

```
Require: $\mathbb{D}$
Ensure: Data cluster $\tilde{X}$, and cluster membership $p(\tilde{x}|x)$
 1: Initialize the data value set $X$, and time segment set $Y$.
 2: Initialize $p(\mathbf{x}, y)$, $p(\mathbf{x})$, $p(y|\mathbf{x})$. Set $\tilde{X}$ as $X$.
 3: while $C_T$ is decreasing do
 4:     Find the $(\tilde{x}_i, \tilde{x}_j)$ pair which minimizes $\delta I(\tilde{x}_i, \tilde{x}_j)$.
 5:     Merge those two $(\tilde{x}_i, \tilde{x}_j)$ to $\tilde{x}_*$.
 6:     Update $\tilde{X} = \{\tilde{X} - \{\tilde{x}_i, \tilde{x}_j\}\} \cup \{\tilde{x_*}\}$, and the $\delta I$, $p(\tilde{x_*}|x)$, $p(\tilde{x_*})$, $p(y|\tilde{x_*})$ values.
```

Algorithm 10 Pseudo-code of DAG-ALP

```
Require: a weighted DAG $\mathcal{G}$ (V, E, W), $h$, $s$, $t$
Ensure: Average longest path
 1: $L_0 = \{s\}$; $\pi(s, 0) = \varnothing$; $p_w(s, 0) = 0$; $L_1, L_2, ..., L_h = \varnothing$//Initialize the first layer with $s$. The parent node
    of $s$ in this layer is None. The path weight to $s$ with length 0 is 0.
 2: for $i = 0$ to $h - 1$ do
 3:     for $v$ that can be reached from nodes $n$ in $L_i$ do
 4:         if $v \notin L_{i+1}$ then
 5:             $L_{i+1} = L_{i+1} \cup v$, $\pi(v, i+1) = u$, $p_w(v, i+1) = p_w(u, i) + w(u, v)$
 6:         else
 7:             if $p_w(v, i+1) < p_w(u, i) + w(u, v)$ then
 8:                 $\pi[v, i+1] = u$, $p_w(v, i+1) = p_w(u, i) + w(u, v)$// updating the path
 9:     if $t \in L_{i+1}$ then
10:         $lp_{i+1} = p_w(t, i+1)$ // find the longest paths with length i+1
11: $ALP = argmax(\frac{lp_1}{1}, \ldots, \frac{lp_h}{h})$. Backtrack with $\pi$ to extract the path.
```

A.2 Pseudo-code

We show the pseudo-code for Cluster algorithm in Alg. 9, and DAG-ALP algorithm in Alg. 10.

A.3 Proofs

Here we prove several lemmas that we mentioned in the main paper. Recall we have the following desired property for edge weight function.

Property 2 *For any three consecutive segments u, v, t, if $v = [c_i, c_j)$, $v_1 = [c_i, c_k)$, $v_2 = [c_k, c_j)$ (i.e. v can be further divided to two segments v_1 and v_2), then $w(e(u, v)) + w(e(v, t)) \leq w(e(u, v_1)) + w(e(v_1, v_2)) + w(e(v_2, t))$.*

We now prove the following lemma.

Lemma 5 *Our edge weight function $w(e(y_a, y_b)) = D_{EU}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$ satisfies property 2.*

Proof 1 *We know that $n_v = n_{v_1} + n_{v_2}$, where n_y is the number of data observations in segment y. We assume $\mathbf{P}_y = p(\tilde{x}|y)$, we have*

$$\mathbf{P}_v = \frac{n_{v1}}{n_v}\mathbf{P}_{v_1} + \frac{n_{v2}}{n_v}\mathbf{P}_{v_2} = \frac{n_{v1}}{n_v}(\mathbf{P}_{v_1} - \mathbf{P}_{v_2}) + \mathbf{P}_{v_2}$$

Similarly, we can also write

$$\mathbf{P}_v = \frac{n_{v1}}{n_v}\mathbf{P}_{v_1} + \frac{n_{v2}}{n_v}\mathbf{P}_{v_2} = \mathbf{P}_{v_1} + \frac{n_{v2}}{n_v}(\mathbf{P}_{v_2} - \mathbf{P}_{v_1})$$

Therefore,

$$\begin{aligned}
&w(e(u,v)) + w(e(v,t)) = ||\mathbf{P}_u - \mathbf{P}_v||_2 + ||\mathbf{P}_v - \mathbf{P}_t||_2 \\
&= ||\mathbf{P}_u - \mathbf{P}_{v_1} - \frac{n_{v2}}{n_v}(\mathbf{P}_{v_2} - \mathbf{P}_{v_1})||_2 + \\
&||\frac{n_{v1}}{n_v}(\mathbf{P}_{v_1} - \mathbf{P}_{v_2}) + \mathbf{P}_{v_2} - \mathbf{P}_t||_2 \\
&\leq ||\mathbf{P}_u - \mathbf{P}_{v_1}||_2 + \frac{n_{v2}}{n_v}||\mathbf{P}_{v_1} - \mathbf{P}_{v_2}||_2 \\
&+ \frac{n_{v1}}{n_v}||\mathbf{P}_{v_1} - \mathbf{P}_{v_2}||_2 + ||\mathbf{P}_{v_2} - \mathbf{P}_t||_2 \\
&= w(e(u,v_1)) + w(e(v_1,v_2)) + w(e(v_2,t))
\end{aligned}$$

The inequality is because of the triangle inequality of Euclidean distance.

We also prove that if we use the well-known distribution distance measure KL divergence instead of Euclidean distance, it does not satisfy property 2 in general.

Lemma 6 *An edge weight function $w(e(y_a, y_b)) = D_{KL}(p(\tilde{x}|y_a), p(\tilde{x}|y_b))$ does not satisfy property 2 in general.*

Proof 2 *We can prove this by constructing a counter example. We can assign $\mathbf{P}_u - \mathbf{P}_{v_1}$, $\mathbf{P}_{v_1} - \mathbf{P}_{v_2}$, $\mathbf{P}_{v_2} - \mathbf{P}_t$ accordingly so that the triangle inequality does not hold for these vectors, and then following the proof above, we naturally get a case where $w(e(u,v)) + w(e(v,t)) > w(e(u,v_1)) + w(e(v_1,v_2)) + w(e(v_2,t))$.*

Lemma 7 *ALP problem (decision version) on general graphs is NP-complete.*

Proof 3 *The ALP decision problem is: given a weighted graph G, a source node s, a target node t, does G have an ALP from s to t that have at least average weight of c. Given an ALP solution, we can easily check if it is a path from s to t, and its average weight in poly-time. So the problem is in NP. Now we make a reduction from LP (decision version) to this ALP problem. Given G, s, t, assume that we know the LP in G has at least total weight of c. We now construct G' by adding to G a node t', an edge from t to t' with weight -c. Since we know the LP in G has at least total weight of c, we know that the ALP in G' has at least average weight of* 0*. On the other hand, if we know G' has an ALP with at least average weight of* 0*, it means that G has a path from s to t with weight larger than or equal to c. Hence the LP in G must have at least total weight of c.*

Lemma 8 *DAG-ALP correctly finds the average longest path.*

Proof 4 *We first prove that ALP is the longest path given the corresponding path length. If not, there exists a path with larger sum of weights while having the same path length, which gives a better average weights and leads to a contradiction. Now we prove by reduction that DAG-ALP finds longest paths of different length from s to t. The base case where we are at the first two layers are clearly true. The reduction step can be proved by suboptimality of the problem: given a longest path P(s,t), and any sub-path P(s,a), this sub-path P(s,a) must also be the longest path from s to a given the corresponding path length. If not, there exists a path from s to a with larger sum of weights while having the same length. Combining this alternative path with the rest path from a to t, we have a path with larger average weights.*

Lemma 9 *The time complexity of DAG-ALP is $O(|E|)$.*

Proof 5 *The time complexity of DAG-ALP is equal to the number of edges are visited during the algorithm which can be calculated as follows:*

$$
\begin{aligned}
&\textit{Layer 1: } (h-1) && = (h-1) \\
&\textit{Layer 2: } (h-2)+(h-3)+\ldots+1 && = \frac{(h-1)(h-2)}{2} \\
&\textit{Layer 3: } (h-3)+(h-4)+\ldots+1 && = \frac{(h-2)(h-3)}{2} \\
&\textit{Layer 4: } h-4)+\ldots+1 && = \frac{(h-3)(h-4)}{4} \\
&\vdots && \vdots \\
&\textit{Layer h-1: } \; 1 && = 1
\end{aligned}
$$

Therefore, the number of visited edges is,

$$\begin{aligned}
&(h-1)+\\
&\underbrace{\frac{(h-1)(h-2)}{2}+\frac{(h-2)(h-3)}{2}}_{(h-2)^2}+\\
&\underbrace{\frac{(h-3)(h-4)}{2}+\frac{(h-4)(h-5)}{2}}_{(h-4)^2}+\\
&\ldots+1\\
&=(h-1)+\sum_{i=1}^{\frac{h}{2}}(h-2i)^2=O(h^3)=O(|E|)
\end{aligned}$$

A.4 Additional Experiments

Here we provide the omitted results in the experiment section of the main paper.

A.4.1 Datasets

1)*Portland.* NDSSL [154] provides several realistic population networks which have been used in national public health studies. We select 7 most representative features for each person to construct the data values. We simulate a flu-like disease propagation (using SI model) on the contact network by seeding two nodes as infected at different times. We expect to discover the time the second seed is injected.

2)*ChickenDance.* In each of the two chicken dance datasets (*ChickenDance* 1 and *ChickenDance* 2), a "chicken" dance motion is recorded as a sequence of 4-dimensional data points [109]. We have the ground-truth segmentation here based on motions in the dances.

3)*Twitter.* We combine Twitter API with Datasift's collection service[1] to construct this dataset. We first collect tweets generated from three different countries (*Peru, Paraguay, Argentina*) from Apr to Aug, 2013 using Datasift. Then we extract flu-related tweets [**?**] and build keyword trends of 12 infection-related keywords. We aim to understand change of these word mentions with respect to time.

4)*Ebola*[2] contains Ebola disease reports for a number of towns, and different infection types in towns, such as 'suspected', 'confirmed', etc. We use the town name and the infection type

[1]`http://datasift.com/`
[2]`http://health.gov.sl`

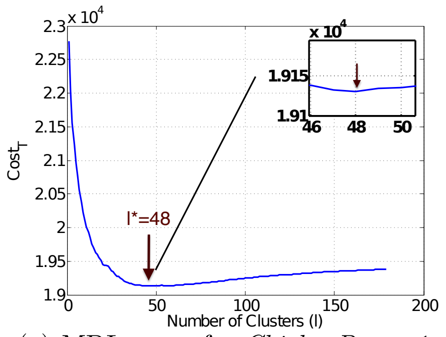
as features. Combined with the date we have a categorical data sequence. We aim to find the infection patterns from this sequence.

5)*PUC-Rio.* is a human motion recognition dataset [164], where subjects perform different actions while wearing accelerometers. For our experiment, we concatenate data from different actions, and construct a data sequence where every once a while, the subject changes her action.

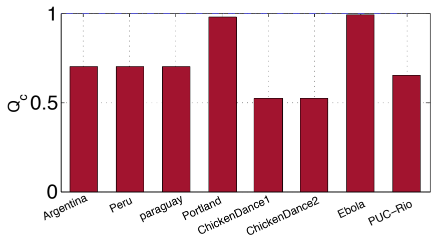
A.4.2 Quality of Clusters ($\tilde{X}$)

Here we examine the number and quality of data clusters found by DASSA.

Number of clusters (MDL) Fig. A.1(a) shows the $Cost_T$ values with respect to the number of clusters for the *ChickenDance* 1 (MDL curves for other datasets are similar). We see that the MDL curve is indeed near-convex, and it suggests an optimal number of clusters which minimizes $Cost_T$.



(a) MDL curve for *ChickenDance* 1



(b) Q_c scores

Figure A.1: (a) MDL curves of *ChickenDance* 1: $Cost_T$ vs number of clusters l. (b) Q_c scores. Note $Q_c > 0.5$ for all datasets—indicates high quality clusters.

Quality of clusters found by DASSA Cluster finds data values which are 'temporally similar' to each other. To measure the quality of the clusters ($\tilde{X}$), we calculate how close in time data values in the same cluster ($\{\mathbf{x}_i | \mathbf{x}_i \in \tilde{x}_j\}$) appear in the sequence. We represent the occurrence of each data value $\mathbf{x}_i$ by a $1 \times |Y|$ vector $v_{\mathbf{x}_i}$, where $v_{\mathbf{x}_i}(j) = 1$ if $\mathbf{x}_i$ occurs in the j_{th} segment in Y, and 0 otherwise. We then calculate the well-known Silhouette score [143] using Eq. A.1, where $A_{\mathbf{x}_i}$ is the average JD (*Jaccard* distance) from $v_{\mathbf{x}_i}$ to data points in the same cluster; $B_{\mathbf{x}_i}$ are the average JD's from $v_{\mathbf{x}_i}$ to data points in other clusters. The Silhouette score for each data point $\mathbf{x}_i$ implies how much $\mathbf{x}_i$ belongs to $\tilde{x}$. We take the average over all $\mathbf{x}_i$ and normalize it to $[0, 1]$. Higher values of Q_c indicate the better clustering.

$$Sil(\mathbf{x}_i) = \frac{\min(B_{\mathbf{x}_i}) - A_{\mathbf{x}_i}}{\max(\min(B_{\mathbf{x}_i}), A_{\mathbf{x}_i})};\ Q_c = \sum_i \frac{Sil(\mathbf{x}_i)+1}{2N} \tag{A.1}$$

Results: We show the Q_c values for different datasets (after running DASSA) in Fig. A.1(b). It can be seen that in all datasets the $Q_c > 0.5$ which means our clustering algorithm gives high-quality clusters capturing the temporal similarity between data values.

A.4.3 Finding the Best Path (ALP vs LP)

We show that the average longest path (ALP) optimization gives better segmentation paths than simply using longest path (LP) optimization. Here we run (1) DASSA which uses ALP; (2)*LP* (replacing ALP by LP in DASSA) for the datasets with ground-truth segmentation. We compare the set of ground-truth cut points and the set of detected cut points from DASSA by calculating the F_1 *Score* as also used in other segmentation literature [109].

Results: The results are shown in Tab. A.2: DASSA performs much better than *LP* on all datasets with ground truth. Especially, we get *perfect* segmentations on three of the datasets. For datasets without ground truth, we compare the segmentation length. As expected, we observe over-segmentation from *LP*. For instance, in *Ebola* (duration = 48 days) and *ChickenDance* 1 (duration = 1535 time-units) the number of segments with *LP* is 12 and 11 while with DASSA it is just 2 and 8.

A.4.4 More Results.

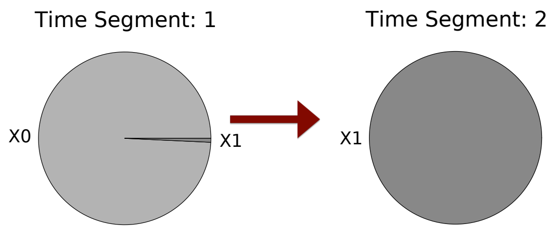
Portland: We show the pie-chart for the cluster ($\tilde{x}$) distribution, as well as the most frequent values in each segment in Fig. A.2(a). We see how the clusters are completely different across segments in this dataset. And as described in the main paper, The results of the most frequent values in the two segments indicate that elder people, with higher incomes, larger number of workers in family, and more vehicles are infected first. Then younger people with lower incomes, fewer vehicles get infected. It illustrates that DASSA is capable of detecting the pattern of disease propagation.

ChickenDance: As shown in Fig. A.4, DASSA captures all the distinct chicken dance motions precisely. In contrast, the cut points detected by *EMP* and *TopicM* do not correctly find the time when a different motion takes place: they either miss the correct cut points, or have unnecessary additional cut points.

Additionally we observe that, segments with the same motion label do not necessarily share the exact same data values. For example, the two beaks in Fig. A.4(c) have similar patterns

Dataset	**DASSA**	*TopicM*	*EMP*	*LP*	*Dynammo*
ChickenDance 1	**1**	0.85	0.76	0.63	0.57
ChickenDance 2	**1**	0.6	0.90	0.54	0.71
Portland	**1**	1	0.66	0	**1**
PUC-Rio	**0.66**	0.46	0.25	0.44	0.25

Table A.2: F_1 score for datasets with ground-truth.



(a) Cluster distributions ($p(\tilde{x}_i|y)$) for *Portland*.

	age	Y	X	Income	Size	# Workers	# Vehicles
Segment:1	4.0	4.0	4.0	10.0	0.0	3.0	5.0
	4.0	3.0	4.0	10.0	0.0	3.0	2.0
	4.0	4.0	2.0	10.0	2.0	5.0	5.0
	4.0	3.0	4.0	10.0	2.0	3.0	2.0

	age	Y	X	Income	Size	# Workers	# Vehicles
Segment:2	1.0	5.0	7.0	6.0	5.0	1.0	2.0
	4.0	5.0	7.0	3.0	0.0	1.0	1.0
	4.0	6.0	6.0	7.0	1.0	5.0	4.0
	2.0	5.0	5.0	7.0	0.0	3.0	2.0

(b) High frequent feature values for *Portland*.

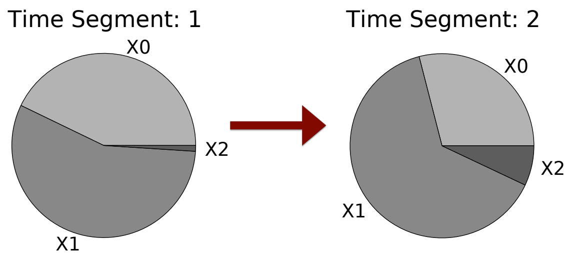
Figure A.2: Pie-chart of $p(\tilde{x}_i|y)$ for each segment, and the associated most frequent data values for *Portland*. DASSA learns $l^* = 2$.

but the data values inside the segments are different. Hence, we do not expect the $\tilde{x}_i$ in the two segments (with the same motion label) to be exactly the same. Interestingly, the shape of the co-occurring proportion still captures the similarity between two segments with the same labels. Consider segment 1 and segment 5 in Fig. A.4—although the $\tilde{x}_i$ values are not the same, the shapes/proportions in the two pie-charts are still similar. This captures the fact that the two segments have similar patterns, yet the data values are not exactly the same.

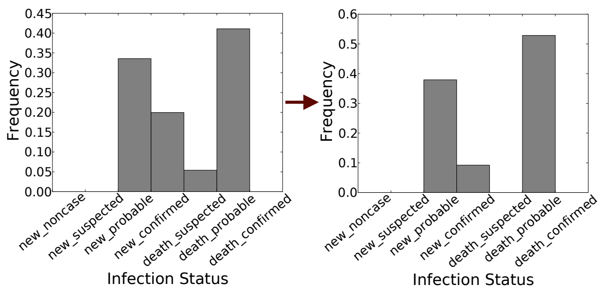
Twitter: We also observe in Fig. A.5(f) and Fig. A.5(h) that the important words across segments are different. *Dynammo*, when given the same number of cut points as detected by DASSA, produces less meaningful cut points.

Ebola: We see in Fig. A.3(a) that the $p(\tilde{x}_i|y)$ in the two segments are different. We explore the feature values in the two segments detected by DASSA. In Fig. A.3(b) we see that the death and newly confirmed cases reduce significantly from segment 1 to segment 2, which shows a sign of increased caution for the disease. We also notice from the change of distribution of towns that at first the infection mostly occurs in town 2 and 3 which are "Kono" and "Kambia" in Sierra-Leone. Then it spreads to other towns (e.g. town 9 which is "Bo" in sierra-leone). DASSA *automatically* finds a segmentation that captures this disease propagation pattern; giving a better understanding of the situation.

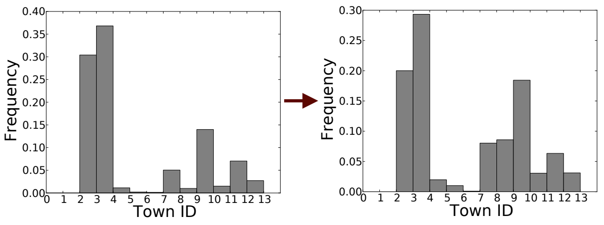
In conclusion, for the three datasets (*Portland*, *ChickenDance*, *PUC-Rio*) where we have the ground truth, DASSA discovers the exact ground truth segmentation in most of the cases; for the other datasets (*Twitter*, *Ebola*), DASSA finds a segmentation with a high Q_{seg}. We provide an interpretation of the segmentation and explain how they reveal the interesting patterns in the data sequences.



(a) Cluster distributions ($p(\tilde{x}_i|y)$) for each segment.

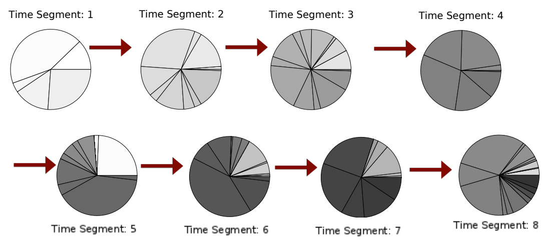


(b) Distribution of infection status.

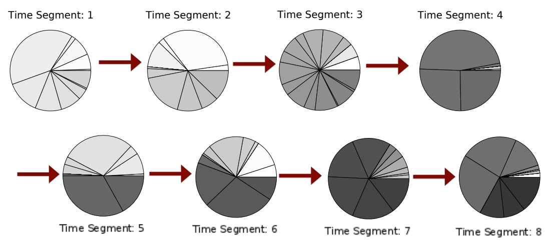


(c) Distribution of towns.

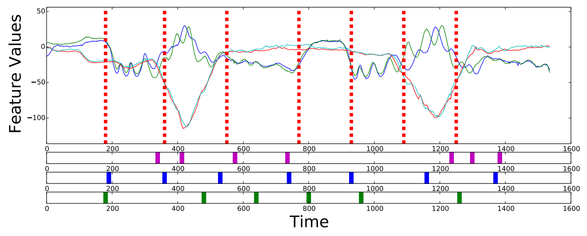
Figure A.3: Results from the *Ebola* datasets. Note that at first infection mostly occurs in towns 2 and 3 but it gradually spread over other towns in second segments. (a) also indicates the death and new confirmed cases are reduced significantly over time.



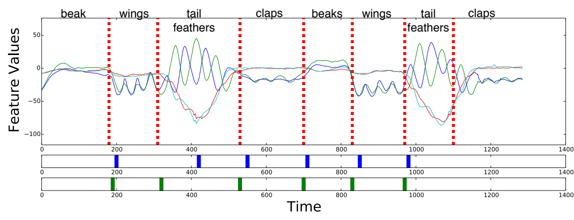
(a) Cluster distributions $p(\tilde{x}_i|y)$ for *ChickenDance* 1.



(b) Cluster distributions $p(\tilde{x}_i|y)$ for *ChickenDance* 2.

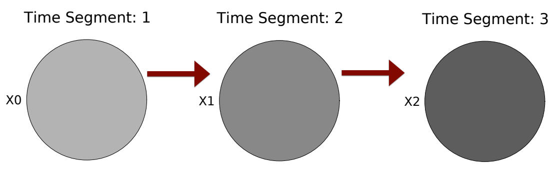


(c) Segmentation of *ChickenDance* 1.

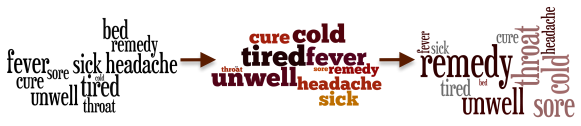


(d) Segmentation of *ChickenDance* 2.

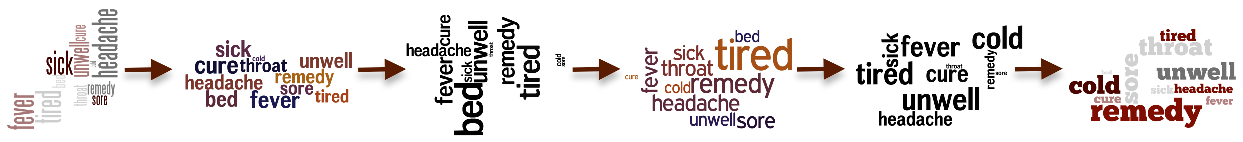
Figure A.4: Pie-charts of $p(\tilde{x}_i|y)$. Same colors represent the same $\tilde{x}_i$ value. (c) and (d) Cut points of the segmentation in *ChickenDance* 1 and *ChickenDance* 2 with DASSA. The cut points of *TopicM* (blue cut points) and *EMP* (green cut points) are shown below the DASSA segmentation. DASSA learns $l^* = 48$.



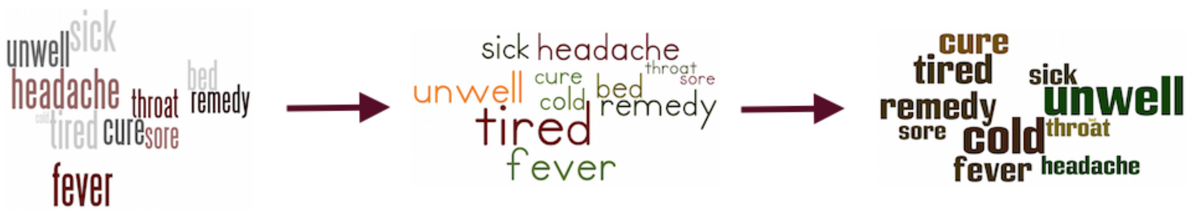
(a) Cluster distributions $p(\tilde{x}_i|y)$ for *Peru*.



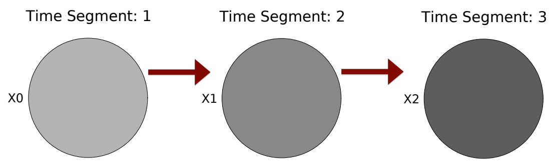
(b) Word cloud for *Peru* from DASSA.



(c) Word cloud for *Peru* from *TopicM*.

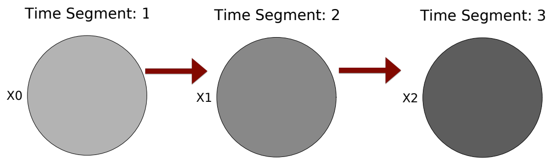


(d) Word cloud for *Peru* from *Dynammo*.

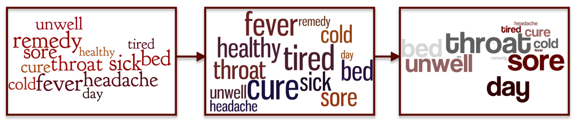


(e) Cluster distributions $p(\tilde{x}_i|y)$ for *Paraguay*.

(f) Word cloud for *Paraguay*



(g) Cluster distributions $p(\tilde{x}_i|y)$ for *Argentina*.



(h) Word cloud for *Argentina*

Figure A.5: (a), (e) and, (g) are pie-charts of $p(\tilde{x}_i|y)$ for each segment of *Peru*, *Paraguay* and *Argentina*, and the corresponding word clouds for then are shown in (b), (f) and, (h) respectively. The results for *TopicM* and *Dynammo* for Peru are also shown in c and d.

Appendix B

HFSTM and HFSTM-A (Chapter 4)

In the model we have three kinds of topic distributions: background, non-flu topic, and state. We test them step by step, by first testing the model where only state words are considered, and then pulsing the background noise part to the model and then the non-flu topic part. We'll introduce the inference for each of them in this sheet. We start with a model where only state words are considered.

Let K, T, N, and U be the number of states, number of tweets per user, number of words per tweet, and total number of users. Let $O =< O_1, O_2, \ldots, O_T >$ and $S =< S_1, S_2, \ldots, S_T >$ the observed sequences of tweets and hidden states respectively for a particular user.

Here is a list of symbols that we will use.

1. ϵ: the prior for the binary state switching variable, which determines whether state of a tweet is drawn from the transition probability matrix or simply copied from the state of the previous tweet (a number in $(0, 1]$)
2. π: initial state probability (size is $1 \times K$)
3. η: tansition probability matrix (size is $K \times K$)
4. ϕ: word distrtibution for each state (size is $K \times W$, where W is the total number of keywords for all of the states)
5. w_{tn}: the nth word in the tth tweet

We want to learn these parameters given the tweet sequence. For compact notation we use $H = (\epsilon, \pi, \eta, \phi)$. We use forward backward procedure for which we define forward variable $A_t(i)$ and backward variable $B_t(i)$ as follows.

$$A_t(i) = P(O_1, O_2, \ldots, O_t, S_t = i|H) \tag{B.1}$$

$$B_t(i) = P(O_{t+1}, \ldots, O_T|S_t = i, H) \tag{B.2}$$

Forward variables:

Initialization is as follows:

For $1 \leq i \leq K$:

$$\begin{aligned} A_1(i) &= P(O_1, S_1 = i|H) = P(O_1|S_1 = i, H)P(S_1 = i|H) \\ &= \prod_{n=1}^{N} P(w_{1n}|S_1 = i, H)\pi_i = \pi_i \prod_{n=1}^{N} \phi^i(w_{1n}) \end{aligned}$$

For $K + 1 \leq i \leq 2K$: $A_1(i) = 0$

Induction is as follows:

For $1 \leq j \leq K$:

$$\begin{aligned} A_t(j) &= P(O_1, O_2, \ldots, O_t, S_t = j|H) \\ &= \left(\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij} \right) P(O_t|S_t = j, H) \\ &= \left(\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij} \right) \prod_{n=1}^{N} \phi^j(w_{tn}) \end{aligned}$$

For $K + 1 \leq j \leq 2K$:

$$\begin{aligned} A_t(j) &= P(O_1, O_2, \ldots, O_t, S_t = j|H) \\ &= \left(A_{t-1}(j) + A_{t-1}(j - K)\right)(1 - \epsilon)P(O_t|S_t = j, H) \\ &= \left(A_t(j) + A_t(j - K)\right)(1 - \epsilon) \prod_{n=1}^{N} \phi^j(w_{tn}) \end{aligned}$$

Backward variables: Initialization is as follows:

For $1 \leq i \leq 2K$:

$$B_T(i) = 1$$

Induction is as follows:

For $1 \leq i \leq K$:

$$\begin{aligned}
B_t(i) &= P(O_{t+1}, \ldots, O_T | S_t = i, H) \\
&= \left(\sum_j^K \epsilon \eta_{ij} P(O_{t+1} | S_{t+1} = j, H) B_{t+1}(j) \right) + (1-\epsilon) P(O_{t+1} | S_{t+1} = i+K, H) B_{t+1}(i+K) \\
&= \left(\sum_j^K \epsilon \eta_{ij} \prod_{n=1}^N \phi^j(w_{(t+1)n}) B_{t+1}(j) \right) + (1-\epsilon) \prod_{n=1}^N \phi^i(w_{(t+1)n}) B_{t+1}(i+K)
\end{aligned}$$

For $K+1 \leq i \leq 2K$:

$$\begin{aligned}
B_t(i) &= P(O_{t+1}, \ldots, O_T | S_t = i, H) \\
&= \left(\sum_j^K \epsilon \eta_{ij} P(O_{t+1} | S_{t+1} = j, H) B_{t+1}(j) \right) + (1-\epsilon) P(O_{t+1} | S_{t+1} = i, H) B_{t+1}(i) \\
&= \left(\sum_j^K \epsilon \eta_{ij} \prod_{n=1}^N \phi^j(w_{(t+1)n}) B_{t+1}(j) \right) + (1-\epsilon) \prod_{n=1}^N \phi^{i-K}(w_{(t+1)n}) B_{t+1}(i)
\end{aligned}$$

Let $\gamma_t(i)$ be the probability of being in state S_i at for tth tweet given the observed tweet sequence O and other model parameters. For each user the size of γ is $2K \times T$.

For $1 \leq i \leq 2K$:

$$\gamma_t(i) = P(S_t = i | O, H) = \frac{A_t(i) B_t(i)}{P(O|H)} = \frac{A_t(i) B_t(i)}{\sum_{i=1}^{2K} A_t(i) B_t(i)}$$

Let $\xi_t(i,j)$ be the probability of being in state S_i at time t, and state S_j at time $t+1$, given O and other model parameters.For each user the size of ξ is $K^2 \times (T-1)$.

$$\xi_t(i,j) = P(S_t = i, S_{t+1} = j | O, H) = \frac{P(S_t = i, S_{t+1} = j, O|H)}{P(O|H)}$$

For $\xi_t(i,j)$, we define follwoing terms.

For $1 \le i \le 2K$ and $1 \le j \le K$

$$T_1 = A_t(i)\epsilon\eta_{ij}P(O_{t+1}|S_{t+1} = j, H)B_{t+1}(j) = A_t(i)\epsilon\eta_{ij}\prod_{n=1}^{N}\phi^i(w_{(t+1)n})B_{t+1}(j) \tag{B.3}$$

$$\text{Note: when } i > K \text{ then } \eta_{ij} = \eta_{(i-K)j} \tag{B.4}$$

$$T_2 = A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i+K, H)B_{t+1}(i+K)$$
$$\text{for } 1 \le i \le K \text{ and } K+1 \le j \le 2K \tag{B.5}$$

$$T_3 = A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i, H)B_{t+1}(i)$$
$$\text{for } K+1 \le i \le 2K \text{ and } K+1 \le j \le 2K \tag{B.6}$$

$$T_1 = A_t(i)\epsilon\eta_{ij}P(O_{t+1}|S_{t+1} = j, H)B_{t+1}(j) = A_t(i)\epsilon\eta_{ij}\prod_{n=1}^{N}\phi^i(w_{(t+1)n})B_{t+1}(j)$$
$$\text{for } 1 \le i \le 2K \text{ and } 1 \le j \le K \tag{B.7}$$

$$\text{Note: when } i > K \text{ then } \eta_{ij} = \eta_{(i-K)j} \tag{B.8}$$

$$T_2 = A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i+K, H)B_{t+1}(i+K)$$
$$\text{for } 1 \le i \le K \text{ and } K+1 \le j \le 2K \tag{B.9}$$

$$T_3 = A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i, H)B_{t+1}(i)$$
$$\text{for } K+1 \le i \le 2K \text{ and } K+1 \le j \le 2K \tag{B.10}$$

$$\xi_t(i,j) = \frac{T_1}{\sum_i\sum_j(T_1+T_2+T_3)} \tag{B.11}$$

$$\xi_t(i,j) = \frac{T_2}{\sum_i\sum_j(T_1+T_2+T_3)} \tag{B.12}$$

$$\xi_t(i,j) = \frac{T_3}{\sum_i\sum_j(T_1+T_2+T_3)} \tag{B.13}$$

B.1 Estimation of Parameters

For estimating ϵ:

$$\epsilon = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{K}\xi(i,j)}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{2K}\xi(i,j)}$$

For estimating π:

$$\pi_i = \frac{\sum_{u=1}^{U}\gamma_1(i)}{\sum_{u=1}^{U}\sum_{i=1}^{K}\gamma_1(i)} \quad \text{for } 1 \leq i \leq K$$

For estimating η:

$$\eta_{ij} = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}(\xi_t(i,j)+\xi_t(i+K,j))}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{j=1}^{K}(\xi_t(i,j)+\xi_t(i+K,j))} \quad \text{for } 1 \leq i \leq K,\, 1 \leq j \leq K$$

For estimating ϕ:

$$\phi_(^i w) = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{\substack{1\leq n\leq N\\ \&\\ w=w_{tn}}}(\gamma_t(i)+\gamma_t(i+K))}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{w=1}^{W}\sum_{\substack{1\leq n\leq N\\ \&\\ w=w_{tn}}}(\gamma_t(i)+\gamma_t(i+K))} \quad \text{for } 1 \leq i \leq K$$

B.1.1 With Background Noise

We have switch variable: l. If If $l = 1$, the word is generated by states, if $l = 0$ it's generated by background.

For $l_i = 1$, w_i is generated by states.

$$\begin{aligned}
P(l_i = 1|\lambda, H, w) &= \frac{P(l_i = 1|\lambda, H)P(w|l_i = 1, \lambda, H)}{P(w|\lambda, H)} \\
&= \frac{\lambda P(w_i|\lambda, H, l_i = 1, w_{-i})P(w_{-i}|\lambda, H, l_i = 1)}{P(w_i|\lambda, H, w_{-i})P(w_{-i}|\lambda, H)} \\
&= \frac{\lambda[\sum_{state} P(w_i|\lambda, H, state, w_{-i})P(state|\lambda, H, w_{-i})]}{\sum_{l_i} P(w_i|\lambda, H, w_{-i}, l_i)P(l_i|\lambda)} \\
&= \frac{\lambda[\sum_{state} \phi_{state}(w_i)\gamma_i(state)]}{\lambda[\sum_{state} \phi_{state}(w_i)\gamma_i(state)] + (1-\lambda)\phi_{Bak}(w_i)}
\end{aligned}$$

For $l_i = 0$, w_i is generated by background.

$$\begin{aligned}
P(l_i = 0|\lambda, H, w) &= \frac{P(l_i = 0|\lambda, H)P(w|l_i = 0, \lambda, H)}{P(w|\lambda, H)} \\
&= \frac{(1-\lambda)\phi_{Bak}(w_i)}{\lambda[\sum_{state} \phi_{state}(w_i)\gamma_i(state)] + (1-\lambda)\phi_{Bak}(w_i)}
\end{aligned}$$

Forward variable: Initialization is as follows:

For $1 \leq i \leq K$:

$$\begin{aligned}
A_1(i) &= P(O_1, S_1 = i|H) = P(O_1|S_1 = i, H)P((S_1 = i|H) \\
&= \pi_i \prod_{n=1}^{N} P(w_{1n}|S_1 = i, H) = \pi_i \prod_{n=1}^{N} [(1-\lambda)\phi_{Bak}(w_{1n}) + \lambda\phi^i(w_{1n})]
\end{aligned}$$

For $K + 1 \leq i \leq 2K$: $A_1(i) = 0$

Induction is as follows:

For $1 \leq j \leq K$:

$$\begin{aligned}
A_t(j) &= P(O_1, O_2, \ldots, O_t, S_t = j|H) \\
&= (\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij})P(O_t|S_t = j, H) \\
&= (\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij}) \prod_{n=1}^{N} [(1-\lambda)\phi_{Bak}(w_{tn}) + \lambda\phi^j(w_{tn})]
\end{aligned}$$

For $K+1 \leq j \leq 2K$:

$$\begin{aligned}A_t(j) &= P(O_1, O_2, \ldots, O_t, S_t = j|H) \\ &= (A_{t-1}(j) + A_{t-1}(j-K))(1-\epsilon)\prod_{n=1}^{N}[(1-\lambda)\phi_{Bak}(w_{tn}) + \lambda\phi^j(w_{tn})]\end{aligned}$$

Backward variable: Initialization is as follows:

For $1 \leq i \leq 2K$:

$$B_T(i) = 1$$

Induction is as follows:

For $1 \leq i \leq K$:

$$\begin{aligned}B_t(i) &= P(O_{t+1}, \ldots, O_T|S_t = i, H) \\ &= \left(\sum_{j}^{K} \epsilon\eta_{ij} P(O_{t+1}|S_{t+1} = j, H) B_{t+1}(j)\right) \\ &\quad + (1-\epsilon)P(O_{t+1}|S_{t+1} = i+K, H)B_{t+1}(i+K) \\ &= \left(\sum_{j}^{K} \epsilon\eta_{ij} \prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^j(w_{(t+1)n}))B_{t+1}(j)\right) \\ &\quad + (1-\epsilon)\prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^i(w_{(t+1)n}))B_{t+1}(i+K)\end{aligned}$$

For $K+1 \leq i \leq 2K$:

$$\begin{aligned}B_t(i) &= P(O_{t+1}, \ldots, O_T|S_t = i, H) \\ &= \left(\sum_{j}^{K} \epsilon\eta_{ij} P(O_{t+1}|S_{t+1} = j, H) B_{t+1}(j)\right) \\ &\quad + (1-\epsilon)P(O_{t+1}|S_{t+1} = i, H)B_{t+1}(i) \\ &= \left(\sum_{j}^{K} \epsilon\eta_{ij} \prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^j(w_{(t+1)n}))B_{t+1}(j)\right) \\ &\quad + (1-\epsilon)\prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^{i-K}(w_{(t+1)n}))B_{t+1}(i)\end{aligned}$$

γ is the same as in the base case:

$$\gamma_t(i) = P(S_t = i|O, H) = \frac{A_t(i)B_t(i)}{P(O|H)} = \frac{A_t(i)B_t(i)}{\sum_{i=1}^{2K} A_t(i)B_t(i)}$$

ξ need to be changed as follows:

$$\xi_t(i, j) = P(S_t = i, S_{t+1} = j|O, H) = \frac{P(S_t = i, S_{t+1} = j, O|H)}{P(O|H)}$$

For $1 \leq i \leq 2K$ and $1 \leq j \leq K$:

$$\begin{aligned} T_1 &= A_t(i)\epsilon\eta_{ij}P(O_{t+1}|S_{t+1} = j, H)B_{t+1}(j) \\ &= A_t(i)\epsilon\eta_{ij}\prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^j(w_{(t+1)n}))B_{t+1}(j) \end{aligned}$$

For $1 \leq i \leq K$ and $K + 1 \leq j \leq 2K$:

$$\begin{aligned} T_2 &= A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i + K, H)B_{t+1}(i + K) \\ &= A_t(i)(1-\epsilon)\prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^i(w_{(t+1)n}))B_{t+1}(i + K) \end{aligned}$$

For $K + 1 \leq i \leq 2K$ and $K + 1 \leq j \leq 2K$:

$$\begin{aligned} T_3 &= A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i, H)B_{t+1}(i) \\ &= A_t(i)(1-\epsilon)\prod_{n=1}^{N}((1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda\phi^{i-K}(w_{(t+1)n}))B_{t+1}(i) \end{aligned}$$

$$\begin{aligned} \xi_t(i, j) &= \frac{T_1}{\sum_i\sum_j(T_1 + T_2 + T_3)} \\ \xi_t(i, j) &= \frac{T_2}{\sum_i\sum_j(T_1 + T_2 + T_3)} \\ \xi_t(i, j) &= \frac{T_3}{\sum_i\sum_j(T_1 + T_2 + T_3)} \end{aligned}$$

Estimation of parameters:

All other estimation remains the same

For estimating ϵ:

$$\epsilon = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{K}\xi(i,j)}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{2K}\xi(i,j)}$$

For estimating π:

$$\pi_i = \frac{\sum_{u=1}^{U}\gamma_1(i)}{\sum_{u=1}^{U}\sum_{i=1}^{K}\gamma_1(i)} \quad \text{for } 1 \le i \le K$$

For estimating η:

$$\eta_{ij} = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}(\xi_t(i,j)+\xi_t(i+K,j))}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{j=1}^{K}(\xi_t(i,j)+\xi_t(i+K,j))} \quad \text{for } 1 \le i \le K,\, 1 \le j \le K$$

Except for the following two:

Estimation for λ:

$$\lambda = \frac{\sum_u \sum_t \frac{1}{N_t}\sum_{n=1}^{N_t} P(l_{tn}=1|\lambda,H,w)}{UT}$$

Estimation for ϕ:

$$\phi^i_(w) = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{\substack{1\le n\le N\\ \&\\ w=w_{tn}}} P(l_{tn}=1|\lambda,H,O)\,(\gamma_t(i)+\gamma_t(i+K))}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{w=1}^{W}\sum_{\substack{1\le n\le N\\ \&\\ w=w_{tn}}} P(l_{tn}=1|\lambda,H,O)\,(\gamma_t(i)+\gamma_t(i+K))} \quad \text{for } 1 \le i \le K$$

$$\phi_{Bak} = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{\substack{1\le n\le N\\ \&\\ w=w_{tn}}} P(l_{tn}=0|\lambda,H,O)}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{w=1}^{W}\sum_{\substack{1\le n\le N\\ \&\\ w=w_{tn}}} P(l_{tn}=0|\lambda,H,O)}$$

B.1.2 With Both Background Noise and Topics

We have two switch variables: l, x. If $l = 1$, the word is generated either by states or topics, if $l = 0$ it's generated by background. If $x = 0$, the word is generated by topics, if $x = 1$ it's by states.

For $l_i = 1$, which means that w_i is generated by either state or topics.

$$\begin{aligned}
P(l_i = 1|\lambda, c, H, w) &= \frac{P(l_i = 1|\lambda, c, H)P(w|l_i = 1, \lambda, c, H)}{P(w|\lambda, c, H)} \\
&= \frac{\lambda P(w_i|\lambda, c, H, l_i = 1, w_{-i})P(w_{-i}|\lambda, c, H, l_i = 1)}{P(w_i|\lambda, c, H, w_{-i})P(w_{-i}|\lambda, c, H)} \\
&= \frac{\lambda \sum_{x_i}[P(w_i|\lambda, c, H, l_i = 1, x_i, w_{-i})P(x_i|\lambda, c, H, l_i = 1, w_{-i})]}{\sum_{l_i}[P(w_i|\lambda, c, H, l_i, w_{-i})P(l_i|\lambda, c, H, w_{-i})]} \\
&= \frac{\lambda[(\sum_{topic} \phi_{topic}(w_i)P(topic|x_i = 0, l_i = 1, \lambda, c, H, w_{-i}))(1 - c) + (\sum_{state} \phi_{state}(w_i)\gamma_i(state))c]}{\lambda[(\sum_{topic} \phi_{topic}(w_i)P(topic|...))(1 - c) + (\sum_{state} \phi_{state}(w_i)\gamma_i(state))c] + (1 - \lambda)\phi_{Bak}(w_i)}
\end{aligned}$$

For $l_i = 0$, w_i is generated by background.

$$\begin{aligned}
P(l_i = 0|\lambda, c, H, w) &= \frac{P(l_i = 0|\lambda, c, H)P(w|l_i = 0, \lambda, c, H)}{P(w|\lambda, c, H)} \\
&= \frac{(1 - \lambda)\phi_{Bak}(w_i)}{\lambda[(\sum_{topic} \phi_{topic}(w_i)P(topic|...))(1 - c) + (\sum_{state} \phi_{state}(w_i)\gamma_i(state))c] + (1 - \lambda)\phi_{Bak}(w_i)}
\end{aligned}$$

For $x_i = 0$, w_i is generated by topics.

$$\begin{aligned}
P(x_i = 0|\lambda, c, H, w) &= \frac{P(x_i = 0|\lambda, c, H)P(w|x_i = 0, \lambda, c, H)}{P(w|\lambda, c, H)} \\
&= \frac{(1 - c)P(w_i|\lambda, c, H, x_i = 0, w_{-i})P(w_{-i}|\lambda, c, H, x_i = 0)}{P(w_i|\lambda, c, H, w_{-i})P(w_{-i}|\lambda, c, H)} \\
&= \frac{(1 - c)\sum_{l_i}[P(w_i|\lambda, c, H, x_i = 0, l_i, w_{-i})P(l_i|\lambda, c, H, x_i = 0, w_{-i})]}{\sum_{x_i} P(w_i|\lambda, c, H, w_{-i}, x_i)P(x_i|\lambda, c, H, w_{-i})} \\
&= \frac{(1 - c)[(\sum_{topic} \phi_{topic}(w_i)P(topic|x_i = 0, l_i = 1, \lambda, c, H, w_{-i}))\lambda + \phi_{Bak}(w_i)(1 - \lambda)]}{(1 - c)[(\sum_{top} \phi_{top}(w_i)P(top|...))\lambda + \phi_{Bak}(w_i)(1 - \lambda)] + c[(\sum_{sta} \phi_{sta}(w_i)\gamma_i(sta))\lambda + \phi_{Bak}(w_i)(1 - \lambda)]}
\end{aligned}$$

For $x_i = 1$, w_i is generated by states.

$$\begin{aligned}
P(x_i = 1|\lambda, c, H, w) &= \frac{P(x_i = 1|\lambda, c, H)P(w|x_i = 1, \lambda, c, H)}{P(w|\lambda, c, H)} \\
&= \frac{c[(\sum_{sta} \phi_{sta}(w_i)\gamma_i(sta))\lambda + \phi_{Bak}(w_i)(1 - \lambda)]}{(1 - c)[(\sum_{top} \phi_{top}(w_i)P(top|...))\lambda + \phi_{Bak}(w_i)(1 - \lambda)] + c[(\sum_{sta} \phi_{sta}(w_i)\gamma_i(sta))\lambda + \phi_{Bak}(w_i)(1 - \lambda)]}
\end{aligned}$$

Forward variable: Initialization is as follows:

For $1 \le i \le K$:

$$\begin{aligned}
A_1(i) &= P(O_1, S_1 = i|H) \\
&= P(O_1|S_1 = i, H)P((S_1 = i|H) \\
&= \pi_i \prod_{n=1}^{N} P(w_{1n}|S_1 = i, H) \\
&= \pi_i \prod_{n=1}^{N} \{(1-\lambda)\phi_{Bak}(w_{1n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{1n})P(top|\ldots) + c\phi^i(w_{1n})]\}
\end{aligned}$$

For $K+1 \le i \le 2K$: $A_1(i) = 0$

Induction is as follows:
For $1 \le j \le K$:

$$\begin{aligned}
A_t(j) &= P(O_1, O_2, \ldots, O_t, S_t = j|H) \\
&= (\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij})P(O_t|S_t = j, H) \\
&= (\sum_{i}^{2K} A_{t-1}(i)\epsilon\eta_{ij}) \prod_{n=1}^{N} \{(1-\lambda)\phi_{Bak}(w_{1n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{1n})P(top|\ldots) + c\phi^j(w_{1n})]\}
\end{aligned}$$

For $K+1 \le j \le 2K$:

$$\begin{aligned}
&A_t(j) = P(O_1, O_2, \ldots, O_t, S_t = j|H) \\
&= (A_{t-1}(j) + A_{t-1}(j-K))(1-\epsilon) \prod_{n=1}^{N} \{(1-\lambda)\phi_{Bak}(w_{1n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{1n})P(top|\ldots) + c\phi^j(w_{1n})]\}
\end{aligned}$$

Backward variable: Initialization is as follows:

For $1 \le i \le 2K$:

$$B_T(i) = 1$$

Induction is as follows:

For $1 \leq i \leq K$:

$$\begin{aligned}
B_t(i) &= P(O_{t+1}, \ldots, O_T | S_t = i, H) \\
&= \left(\sum_j^K \epsilon \eta_{ij} P(O_{t+1} | S_{t+1} = j, H) B_{t+1}(j) \right) \\
&\quad + (1-\epsilon) P(O_{t+1} | S_{t+1} = i+K, H) B_{t+1}(i+K) \\
&= \left(\sum_j^K \epsilon \eta_{ij} \prod_{n=1}^N \{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{(t+1)n}) P(top|\ldots) + c\phi^j(w_{(t+1)n})]\} B_{t+1}(j) \right) \\
&\quad + (1-\epsilon) \prod_{n=1}^N \{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{(t+1)n}) P(top|\ldots) + c\phi^i(w_{(t+1)n})]\} B_{t+1}(i+K)
\end{aligned}$$

For $K+1 \leq i \leq 2K$:

$$\begin{aligned}
B_t(i) &= P(O_{t+1}, \ldots, O_T | S_t = i, H) \\
&= \left(\sum_j^K \epsilon \eta_{ij} P(O_{t+1} | S_{t+1} = j, H) B_{t+1}(j) \right) \\
&\quad + (1-\epsilon) P(O_{t+1} | S_{t+1} = i, H) B_{t+1}(i) \\
&= \left(\sum_j^K \epsilon \eta_{ij} \prod_{n=1}^N \{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{(t+1)n}) P(top|\ldots) + c\phi^j(w_{(t+1)n})]\} B_{t+1}(j) \right) \\
&\quad + (1-\epsilon) \prod_{n=1}^N \{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top} \phi_{top}(w_{(t+1)n}) P(top|\ldots) + c\phi^{i-K}(w_{(t+1)n})]\} B_{t+1}(i)
\end{aligned}$$

γ is the same as in the base case:

$$\begin{aligned}
\gamma_t(i) &= P(S_t = i | O, H) \\
&= \frac{A_t(i) B_t(i)}{P(O|H)} \\
&= \frac{A_t(i) B_t(i)}{\sum_{i=1}^{2K} A_t(i) B_t(i)}
\end{aligned}$$

Define z as follows:

$$\begin{aligned}
z_{t,n}(i) &= P(T_{tn} = i | l_{tn} = 1, x_{tn} = 0, w_{tn}, H) \\
&= \frac{P(w_{tn} | T_{tn} = i, H, l_{tn} = 1, x_{tn} = 0) P(T_{tn} = i | l_{tn} = 1, x_{tn} = 0, H)}{P(w_{tn} | l_{tn} = 1, x_{tn} = 0, H)} \\
&= \frac{\phi_{top=i}(w_{tn}) P(T_{tn} = i | l_{tn} = 1, x_{tn} = 0, H)}{\sum_i [\phi_{top=i}(w_{tn}) P(T_{tn} = i | l_{tn} = 1, x_{tn} = 0, H)]}
\end{aligned}$$

ξ need to be changed as follows:

$$\begin{aligned}\xi_t(i,j) &= P(S_t = i, S_{t+1} = j|O, H)\\ &= \frac{P(S_t = i, S_{t+1} = j, O|H)}{P(O|H)}\end{aligned}$$

For $1 \le i \le 2K$ and $1 \le j \le K$:

$$\begin{aligned}T_1 &= A_t(i)\epsilon\eta_{ij}P(O_{t+1}|S_{t+1} = j, H)B_{t+1}(j)\\ &= A_t(i)\epsilon\eta_{ij}\prod_{n=1}^{N}\{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top}\phi_{top}(w_{(t+1)n})P(top|\ldots) + c\phi^{j}(w_{(t+1)n})]\}B_{t+1}(j)\end{aligned}$$

For $1 \le i \le K$ and $K+1 \le j \le 2K$:

$$\begin{aligned}T_2 &= A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i+K, H)B_{t+1}(i+K)\\ &= A_t(i)(1-\epsilon)\prod_{n=1}^{N}\{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top}\phi_{top}(w_{(t+1)n})P(top|\ldots) + c\phi^{i}(w_{(t+1)n})]\}B_{t+1}(i+K)\end{aligned}$$

For $K+1 \le i \le 2K$ and $K+1 \le j \le 2K$:

$$\begin{aligned}T_3 &= A_t(i)(1-\epsilon)P(O_{t+1}|S_{t+1} = i, H)B_{t+1}(i)\\ &= A_t(i)(1-\epsilon)\prod_{n=1}^{N}\{(1-\lambda)\phi_{Bak}(w_{(t+1)n}) + \lambda[(1-c)\sum_{top}\phi_{top}(w_{(t+1)n})P(top|\ldots) + c\phi^{i-K}(w_{(t+1)n})]\}B_{t+1}(i)\end{aligned}$$

$$\begin{aligned}\xi_t(i,j) &= \frac{T_1}{\sum_i\sum_j(T_1+T_2+T_3)}\\ \xi_t(i,j) &= \frac{T_2}{\sum_i\sum_j(T_1+T_2+T_3)}\\ \xi_t(i,j) &= \frac{T_3}{\sum_i\sum_j(T_1+T_2+T_3)}\end{aligned}$$

Estimation of parameters:

All other estimation remains the same

For estimating ϵ:

$$\epsilon = \frac{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{K}\xi(i,j)}{\sum_{u=1}^{U}\sum_{t=1}^{T}\sum_{i=1}^{2K}\sum_{j=1}^{2K}\xi(i,j)}$$

For estimating π:

$$\pi_i = \frac{\sum_{u=1}^{U} \gamma_1(i)}{\sum_{u=1}^{U} \sum_{i=1}^{K} \gamma_1(i)} \quad \text{for } 1 \le i \le K$$

For estimating η:

$$\eta_{ij} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} \left(\xi_t(i,j) + \xi_t(i+K,j)\right)}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{j=1}^{K} \left(\xi_t(i,j) + \xi_t(i+K,j)\right)} \quad \text{for } 1 \le i \le K,\ 1 \le j \le K$$

For estimating λ:

$$\lambda = \frac{\sum_u \sum_t \frac{1}{N_t} \sum_{n=1}^{N_t} P(l_{tn} = 1|\lambda, c, H, w)}{UT}$$

Except for the following:

Estimation for c:

$$c = \frac{\sum_u \sum_t \frac{1}{N_t} \sum_{n=1}^{N_t} P(l_{tn} = 1|\lambda, c, H, w) P(x_{tn} = 1|\lambda, c, H, w)}{\sum_u \sum_t \frac{1}{N_t} \sum_{n=1}^{N_t} P(l_{tn} = 1|\lambda, c, H, w)}$$

Estimation for ϕ:

$$\phi^{i}_{(}w) = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 1|\lambda, c, H, O) P(x_{tn} = 1|\lambda, c, H, O) \left(\gamma_t(i) + \gamma_t(i+K)\right)}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{w=1}^{W} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 1|\lambda, c, H, O) P(x_{tn} = 1|\lambda, c, H, O) \left(\gamma_t(i) + \gamma_t(i+K)\right)}$$

for $1 \le i \le K$

$$\phi_{Bak}(w) = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 0|\lambda, c, H, O)}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{w=1}^{W} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 0|\lambda, c, H, O)}$$

$$\phi_{Topic}(w) = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 1|\lambda, c, H, O) P(x_{tn} = 0|\lambda, c, H, O) z_{t,n}(Topic)}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{w=1}^{W} \sum_{\substack{1 \le n \le N \\ \& \\ w = w_{tn}}} P(l_{tn} = 1|\lambda, c, H, O) P(x_{tn} = 0|\lambda, c, H, O) z_{t,n}(Topic)}$$

$$P(T_{tn} = i | l_{tn} = 1, x_{tn} = 0, H) = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{n=1}^{N_t} P(l_{tn} = 1 | \lambda, c, H, O) P(x_{tn} = 0 | \lambda, c, H, O) z_{t,n}(i)}{\sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{n=1}^{N_t} P(l_{tn} = 1 | \lambda, c, H, O) P(x_{tn} = 0 | \lambda, c, H, O)}$$

B.1.3 With Aspects

The only thing we need to change is the value of c and λ according to the equation in our paper. And use the new value of c andλ in same inferences introduced above.

Appendix C

PoLIM (Chapter 7)

C.1 Additional Experiments

Keywords used to obtain *Tweets-Iran*. iran, iranian, iranians, ahmadinejad, tehran, mahmoud, moussavi, mir hossein, neda, green movement, green ribbon, mousavi, iranelection.

Individual's influence tendency.

Time in secs	Tweets from user mashable
1302781.0	tweet win free vip ticket #140conf
1303395.0	you'r watch nba final 4 way make nba final social
1338254.0	hunch launch reinvent make decis
1346474.0	onli two hour left tweet win free vip ticket #140conf
1354831.0	reddit start job board whi
1361533.0	tip you'r new twitter know someon tri twitter list section
1369526.0	youtub continu time ad choic
1370893.0	green tweet 75+ environmentalist follow twitter #ecomonday
1392419.0	realiti tv show ink twitter name tattoo

(a) User mashable, $v = 0.98$, mostly original tweets

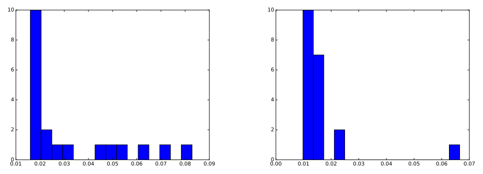
Time in secs	Tweets from user robertgoodwin
1426918.0	rt doe anybodi know good view launch beach ani beach melbourn area #nasa #sts127
1427333.0	rt @nasa launch offici edt wednesday guess space.com wa wrong 5:20 launch
1648039.0	brighter note you'r ever cocoa beach check italian courtyard sicilian thick crust pizza great
1798689.0	rt nasa might found defect caus leak juli 11
2034576.0	rt @nasa lcross live stream happen live stream lunar orbit
2141796.0	rt stun pictur hole cloud astronaut wit volcano erupt
2142023.0	insan relax condo loan rule haven't learn anyth rt wait ?...
2211875.0	rt @aldotcom nasa fund restor billion support

(b) User robertgoodwin, $v = 0.37$, mostly influenced by NASA (note the rt's)

Figure C.1: Tweets from two example users with different v values learned

At the individual level, we show that PoLIM correctly learns the probability of a person being influenced. We pick two users (mashable and robertgoodwin) and check their actual tweets to further analyze our results. PoLIM learns a $v = 0.98$ for mashable, which means that 98% of times, mashable makes posts or follow users out of self interests, and it is rarely influenced by others; as one would expect from a large media content generator. From its tweets, we do see that most of them are original tweets, and it covers many different topics. In contrast, user robertgoodwin, an IT professional who works for NASA, shows a clear sign of influence from NASA in his tweets (60% of the tweets are retweets). PoLIM correctly learns a low $v = 0.37$ to capture this fact.

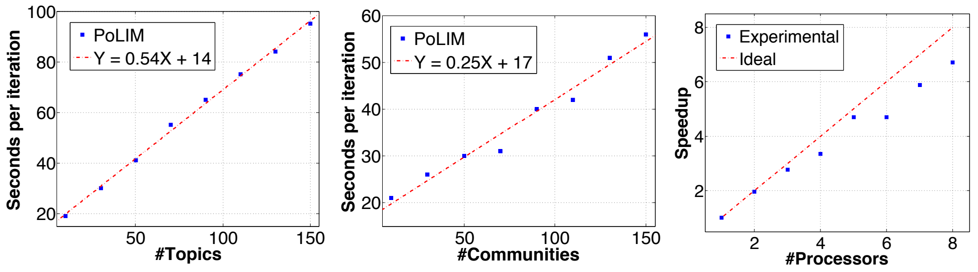
Celebrity structures. We examine the celebrity values in each community and find different celebrity structures for different communities. By examining the histogram and entropy of the A values for the top 20 celebrities in the communities, we make an interesting observation. For a community about a specific event, such as c_7 which is mainly about the Iran election, the importance/authority are more spread out to multiple users; while for a community about a general topic, like c_{14} which focuses on garden and art, a few users would have the leading authority and the others are much less influential. This leads to an insight that when a new topic/event emerges, different perspectives/arguments of the subject can be discussed, which offers more chances for users to be noticed and hence become influential. On the other hand, for a very developed and general topic, the 'heat' of the discussion has decreased to a stable level, and the authority has started to concentrate rather than diverge.



(a) c_7 (Iran election) (b) c_{14} (garden and art)

Figure C.2: Histograms of the top 20 A values in two different communities.

Scalability Finally, to examine the running time of PoLIM, we vary two of the input parameters: the number of topics and communities. In Fig. C.3(a)(b), the running time scales linearly w.r.t both #topics and #communities as expected from the complexity of the algorithm. We also observe we get *near-linear* speedup from parallelizing the inference algorithm. in Fig. C.3(c).



(a) Varying #Topics (b) Varying #Communities (c) Parallelization

Figure C.3: PoLIM scales linearly w.r.t topics and communities and parallelization gives near-linear speed-ups.

C.2 Switching Parameters

Probabilities of the switching parameters:

$$p(s=0)=u \tag{C.1}$$
$$p(r=0)=v \tag{C.2}$$
$$p(\epsilon=0)=\mu=1-v \tag{C.3}$$
$$p(\lambda=0)=\rho \tag{C.4}$$

When $s=0$, the word is generated from background distribution. When $r=0$, the topic of the tweet is generated from the user's own topic interest. When $\epsilon=0$, the user selects celebrity in an influencing community to follow. When $\lambda=0, \epsilon=1$, the user chooses a celebrity in her own community to follow. When $\lambda=1, \epsilon=1$, the user chooses a random user to follow.

After introducing the weak supervision, $p(r=0)$ changes accordingly:

$$p(r=0|l=0)=1-\tau+\tau\cdot v \tag{C.5}$$
$$p(r=0|l=1)=\tau\cdot v \tag{C.6}$$

where $l=1$ means that the tweet is a retweet or a reply. When we observe this, we bias the probability of the tweet being influenced by others.

C.3 Gibbs Sampling

In deriving the sampling equations, let $\Theta=\{\alpha,\beta,u,v,\zeta,\rho,\eta,\sigma,\xi,\theta,\phi_z,\phi_B,I,A,M\}$, and $\Theta'=\{\alpha,\beta,u,v,\zeta,\rho,\eta,\sigma,\xi\}$. Note that the equations here are more elaborated than the equations in the main paper. We expand most of the counters to more detailed situations, but they are actually the same.

Sampling z:

We now derive the conditional probability of $p(z_i=j|z_{-i},w,r,s,y,c,\Theta')$.

$$p(z_i=j|z_{-i},w,r,s,y,c,\Theta')\propto p(w_i|z_i=j,z_{-i},w_{-i},r,s,y,c,\Theta')p(z_i=j|z_{-i},w_{-i},r,s,y,c,\Theta') \tag{C.7}$$

We factorize w_i to each word in the tweets w_{it}, the left term can be written as the product of all $p(w_{it}|\cdot)$. Also we only need to consider the words that have $s=1$ since the others wouldn't change the probability of z:

$$p(w_{it}|z_i=j,z_{-i},w_{-i},r,s,y,c,\Theta')=\int_{\phi_j}p(w_{it}|\phi_j)p(\phi_j|z_{-i},w_{-i},r,s,y,c,\Theta')d\phi_j \tag{C.8}$$

In the above equation, we can write

$$p(\phi_j|z_{-i},w_{-i},r,s,y,c,\Theta')\propto p(w_{-i}|z_{-i},\phi_j,r,s,y,c,\Theta')p(\phi_j|z_{-i},r,s,y,c,\Theta') \tag{C.9}$$
$$\sim Dirichlet(\beta+N^{w}_{-i,j,s=1}) \tag{C.10}$$

where $N^{w}_{-i,j}$ is the number of times that word w is assigned to j and is not generated by background topic ($s = 1$). The reason of the above equation is that the Dirichlet distribution is a conjugate prior to the multinomial likelihood distribution. Taking this result back to Eq. C.8, the probability can be written as the expectation of the Dirichlet distribution.

$$p(w_i|z_i = j, z_{-i}, w_{-i}, r, s, y, c, \Theta') = \prod_{w_{it}, s_{it}=1} \frac{\beta + N^{w_{it}}_{-i,j,s=1}}{W\beta + N_{-i,j,s=1}} \tag{C.11}$$

where W is the number of words in the vocabulary, and $N_{-i,j}$ is the number of total words that are assigned to topic j.

Now we look at the right term of Eq. C.7. When $r = 0$, the topic is generated from the user d's own interest. We have

$$p(z_i = j|z_{-i}, w_{-i}, r, s, y, c, \Theta') = \int_{\theta_{c_d}} p(z_i = j|z_{-i}, \theta_{c_d}, w_{-i}, r, s, y, c, \Theta')p(\theta_{c_d}|z_{-i}, w_{-i}, r, s, y, c, \Theta')d\theta_{c_d} \tag{C.12}$$

where c_d is the community user d belongs to. The right term of the above equation can be written as:

$$p(\theta_{c_d}|z_{-i}, w_{-i}, r, s, y, c, \Theta') = p(z_{-i}|\theta_{c_d}, w_{-i}, r, s, y, c, \Theta')p(\theta_{c_d}|w_{-i}, r, s, y, c, \Theta') \tag{C.13}$$

$$\sim \mathit{Dirichlet}(\alpha + N^{z}_{-i,r=0,c_d} + N^{z}_{-i,r=1,c_y=c_d}) \tag{C.14}$$

where $N^{z}_{-i,r=0,c_d}$ is the number of time topic z is generated by θ_{c_d} and $r = 0$, and $N^{z}_{-i,r=1,c_y=c_d}$ is the number of time topic z is generated by θ_{c_d}, which is an influence from user y who's in community c_d. Then Eq. C.12 can be written by the expectation of the Dirichlet distribution.

$$p(z_i = j|z_{-i}, w_{-i}, r, s, y, c, \Theta') = \frac{\alpha + N^{z=j}_{-i,r=0,c_d} + N^{z=j}_{-i,r=1,c_y=c_d}}{Z\alpha + N_{-i,r=0,c_d} + N_{-i,r=1,c_y=c_d}} \tag{C.15}$$

where Z is the number of total topics. Similarly we can derive the case when $r = 1$.

$$p(z_i = j|z_{-i}, w_{-i}, r, s, y, c, \Theta') = \frac{\alpha + N^{z=j}_{-i,r=0,c_d=c_y} + N^{z=j}_{-i,r=1,c_y}}{Z\alpha + N_{-i,r=0,c_d=c_y} + N_{-i,r=1,c_y}} \tag{C.16}$$

Notice now we are looking at the topics generated from θ_{c_y} instead of θ_{c_d} since $r = 1$ indicates that the topic is generated under another user's influence.

Having both the left term and right term, we rewrite Eq. C.7 as:

$$p(z_i = j|z_{-i}, w, r, s, y, c, \Theta') \propto \prod_{w_{it}, s_{it}=1} \frac{\beta + N^{w_{it}}_{-i,j,s=1}}{W\beta + N_{-i,j,s=1}} \cdot \\ (\frac{\alpha + N^{z=j}_{-i,r=0,c_d} + N^{z=j}_{-i,r=1,c_y=c_d}}{Z\alpha + N_{-i,r=0,c_d} + N_{-i,r=1,c_y=c_d}})^{\mathbb{1}(r_i=0)} \cdot (\frac{\alpha + N^{z=j}_{-i,r=0,c_d=c_y} + N^{z=j}_{-i,r=1,c_y}}{Z\alpha + N_{-i,r=0,c_d=c_y} + N_{-i,r=1,c_y}})^{\mathbb{1}(r_i=1)} \tag{C.17}$$

Sampling s:

We look at the following conditional probability:

$$\begin{aligned}p(s_i = 1|s_{-i}, z, w, r, y, \Theta') &\propto p(w_i|s_i = 1, s_{-i}, z, w_{-i}, r, y, \Theta')p(s_i = 1|s_{-i}, z, w_{-i}, r, y, \Theta')\\ &= (1-u)\int_{\phi_z} p(w_i|s_i = 1, s_{-i}, z, \phi_z, w_{-i}, r, y, \Theta')p(\phi_z|s_i = 1, s_{-i}, z, w_{-i}, r, y, \Theta')d\phi_z\\ &= (1-u)\frac{\beta + N^{w_i}_{-i,z,s=1}}{W\beta + N_{-i,z,s=1}} \qquad (C.18)\end{aligned}$$

For $s_i = 0$, we have

$$\begin{aligned}p(s_i = 0|s_{-i}, z, w, r, y, \Theta') &\propto p(w_i|s_i = 0, s_{-i}, z, w_{-i}, r, y, \Theta')p(s_i = 0|s_{-i}, z, w_{-i}, r, y, \Theta')\\ &= u\int_{\phi_B} p(w_i|s_i = 0, s_{-i}, z, \phi_B, w_{-i}, r, y, \Theta')p(\phi_B|s_i = 0, s_{-i}, z, w_{-i}, r, y, \Theta')d\phi_z\\ &= u\frac{\beta + N^{w_i}_{-i,s=0}}{W\beta + N_{-i,s=0}} \qquad (C.19)\end{aligned}$$

Sampling r: In the following we use r_i to denote the r value for the i_{th} tweet from user d.

$$\begin{aligned}p(r_i = 1|z, w, r_{-i}, y, c, \Theta') &\propto p(z_i|r_i = 1, z_{-i}, w, r_{-i}, y, c, \Theta')p(r_i = 1|z_{-i}, w, r_{-i}, y, c, \Theta')\\ &= (1-v_d)\int_{\theta_{c_y}} p(z_i|r_i = 1, \theta_{c_y}, z_{-i}, w, r_{-i}, y, c, \Theta')p(\theta_{c_y}|r_i = 1, z_{-i}, w, r_{-i}, y, c, \Theta')d\theta_{c_y}\\ &= (1-v_d)\frac{\alpha + N^{z_i}_{-i,r=0,c_d=c_y} + N^{z_i}_{-i,r=1,c_y}}{Z\alpha + N_{-i,r=0,c_d=c_y} + N_{-i,r=1,c_y}} \qquad (C.20)\\ p(r_i = 0|z, w, r_{-i}, y, c, \Theta') &\propto p(z_i|r_i = 0, z_{-i}, w, r_{-i}, y, c, \Theta')p(r_i = 0|z_{-i}, w, r_{-i}, y, c, \Theta')\\ &= v_d\int_{\theta_{c_d}} p(z_i|r_i = 0, \theta_{c_d}, z_{-i}, w, r_{-i}, y, c, \Theta')p(\theta_{c_d}|r_i = 0, z_{-i}, w, r_{-i}, y, c, \Theta')d\theta_{c_d}\\ &= v_d\frac{\alpha + N^{z_i}_{-i,r=0,c_d} + N^{z_i}_{-i,r=1,c_y=c_d}}{Z\alpha + N_{-i,r=0,c_d} + N_{-i,r=1,c_y=c_d}} \qquad (C.21)\end{aligned}$$

Note that the value of v_d will be biased by the supervision based on the equations we show at the beginning.

Sampling ϕ:

Similarly we can write the conditional distribution for ϕ.

$$\begin{aligned}p(\phi_z|z, w, s, \Theta') &\propto p(w|\phi_z, z, w, s, \Theta')p(\phi_z|z, s, \Theta')\\ &\sim Dir(\beta + N^{w_i}_{s=1,z}) \qquad (C.22)\end{aligned}$$

where $N^{w_i}_{s=1,z}$ is the number of times that w_i is assigned to topic z and the word is not generated from the background. Hence for every word w, we have

$$\phi_z(w) = \frac{\beta + N^{w}_{s=1,z}}{W\beta + N_{s=1,z}} \qquad (C.23)$$

For ϕ_B, we have

$$p(\phi_B|z, w, s, \Theta') \propto p(w|\phi_B, z, w, s, \Theta')p(\phi_B|z, s, \Theta') \\ \sim Dir(\beta + N^{w_i}_{s=0}) \tag{C.24}$$

$$\phi_B(w) = \frac{\beta + N^{w}_{s=0}}{W\beta + N_{s=0}} \tag{C.25}$$

where $N^{w_i}_{s=0}$ is the number of times word w_i is generated from the background word distribution.

Sampling y:

In sampling the above parameters, we only use the community information of the influencing user y (due to our assumption that all community members share the same topic interest). Hence we only need to sample c_y here.

$$p(c_{y_i}|w, e, \epsilon, \lambda, z, r, c_{y_{-i}}, c, c^*, \Theta') \propto p(z_i|r, z_{-i}, w, c_{y_i}, c_{y_{-i}}, c, \Theta')p(c_{y_i}|r, z_{-i}, w, e, \epsilon, \lambda, c_{y_{-i}}, c, c^*, \Theta') \tag{C.26}$$

If $r_i = 0$, the tweet is generated from user's own topic interes. The left term of the above equation can be written as:

$$p(z_i|r, z_{-i}, w, c_{y_i}, c_{y_{-i}}, c, \Theta') = \int_{\theta_{c_d}} p(z_i|\theta_{c_d}, r_i = 0, r_{-i}, z_{-i}, c, \Theta')p(\theta_{c_d}|r_i = 0, r_{-i}, z_{-i}, c, \Theta')d\theta_{c_d} \\ = \frac{\alpha + N^{z_i}_{-i,r=0,c_d} + N^{z_i}_{-i,r=1,c_y=c_d}}{Z\alpha + N_{-i,r=0,c_d} + N_{-i,r=1,c_y=c_d}} \tag{C.27}$$

If $r_i = 1$, the left term can be written as:

$$p(z_i|r, z_{-i}, w, c_{y_i}, c_{y_{-i}}, c, \Theta') = \int_{\theta_{c_y}} p(z_i|\theta_{c_y}, r_i = 1, r_{-i}, z_{-i}, c, \Theta')p(\theta_{c_y}|r_i = 1, r_{-i}, z_{-i}, c, \Theta')d\theta_{c_d} \\ = \frac{\alpha + N^{z_i}_{-i,r=0,c_d=c_y} + N^{z_i}_{-i,r=1,c_y}}{Z\alpha + N_{-i,r=0,c_d=c_y} + N_{-i,r=1,c_y}} \tag{C.28}$$

The right term in Eq. C.26 is expanded as:

$$p(c_{y_i}|r, z_{-i}, w, e, \epsilon, \lambda, c_{y_{-i}}, c, c^*, \Theta') = \int_{I_{c_d}} p(c_{y_i}|I_{c_d}, r, z_{-i}, w, c_{y_{-i}}, c, \Theta')p(I_{c_d}|r, z_{-i}, w, e, \epsilon, \lambda, c_{y_{-i}}, c, c^*, \Theta')dI_{c_d} \\ \propto \int_{I_{c_d}} p(c_{y_i}|I_{c_d})p(e|I_{c_d}, \epsilon, \lambda, c^*, c_{y_{-i}}, c, \Theta')p(I_{c_d}|c^*, c_{y_{-i}}, \Theta')dI_{c_d} \\ \propto \int_{I_{c_d}} p(c_{y_i}|I_{c_d})p(I_{c_d}|c^*, c_{y_{-i}}, \Theta')dI_{c_d} \\ \propto \int_{I_{c_d}} p(c_{y_i}|I_{c_d})p(c^*, c_{y_{-i}}|I_{c_d}, \Theta')p(I_{c_d}|\Theta')dI_{c_d} \\ \propto \frac{\zeta + N^{c_y=c_{y_i}}_{-i,r=1,c_d} + N^{c^*=c_{y_i}}_{\epsilon=0,c_d}}{K\zeta + N_{-i,r=1,c_d} + N_{\epsilon=0,c_d}} \tag{C.29}$$

where K is the number of communities. Note that the probablity of $p(e|I_{c_d}, c^*)$ does not depend on I_{c_d} anymore when c^* are known. Hence it can be directly removed from the above steps.

Sampling c:

$$p(c_d|e, w, z, r, c_{-d}, c^*, c^y, \epsilon, \lambda, \Theta') \propto p(z_d, e_d|c_d, z_{-d}, e_{-d}, r, c_{-d}, c^*, c^y, \epsilon, \lambda, \Theta')p(c_d|z_{-d}, e_{-d}, r, c_{-d}, c^*, c^y, \epsilon, \lambda, \Theta')$$
$$\propto p(z_d|c_d, z_{-d}, r, c_{-d}, \Theta')p(e_{d,\epsilon=0,\lambda=1}|c_d, e_{-d}, c_{-d}, c^*, \Theta')p(c_d|z_{-d}, e_{-d}, r, c_{-d}, c^*, c^y, \epsilon, \lambda, \Theta')$$
$$\propto \int_{\theta_{c_d}} p(z_d|\theta_{c_d}, r)p(\theta_{c_d}|c_d, z_{-d}, r, c_{-d}, \Theta')d\theta_{c_d} \cdot$$
$$\int_{A_{c_d}} p(e_{d,\epsilon=0,\lambda=1}|A_{c_d}, \epsilon, \lambda)p(A_{c_d}|c_d, e_{-d}, c_{-d}, c^*, \Theta')dA_{c_d} \cdot$$
$$p(c^*_d, c^y_d|c_d, z_{-d}, e_{-d}, r, c_{-d}, c^*_{-d}, c^y_{-d}, \epsilon, \lambda, \Theta')p(c_d|z_{-d}, e_{-d}, r, c_{-d}, c^*_{-d}, c^y_{-d}, \epsilon, \lambda, \Theta')$$
$$\propto \int_{\theta_{c_d}} p(z_d|\theta_{c_d}, r)p(z_{-d}|\theta_{c_d}, c_d, r, c_{-d}, \Theta')p(\theta_{c_d}|c_d, r, c_{-d}, \Theta')d\theta_{c_d} \cdot$$
$$\int_{A_{c_d}} p(e_{d,\epsilon=0,\lambda=1}|A_{c_d}, \epsilon, \lambda)p(e_{-d}|A_{c_d}, c_d, c_{-d}, c^*, \Theta')p(A_{c_d}|c_d, c_{-d}, c^*, \Theta')dA_{c_d} \cdot$$
$$\int_{I_{c_d}} p(c^*_d, c^y_d|I_{c_d}, c_d, r, \epsilon, \lambda, \Theta')p(I_{c_d}|c_d, c_{-d}, c^*_{-d}, c^y_{-d}, r, \epsilon, \lambda, \Theta')dI_{c_d} \cdot M(c_d)$$
$$\propto \prod_{d,r=0} \frac{\alpha + N^{z_d}_{-d,r=0,c_d} + N^{z_d}_{-d,r=1,c_y=c_d}}{Z\alpha + N_{-d,r=0,c_d} + N_{-d,r=1,c_y=c_d}} \cdot$$
$$\prod_{e_{d,\epsilon=1,\lambda=0}} \frac{\eta + N^{e_{d,\epsilon=1,\lambda=0}}_{-d,\epsilon=1,\lambda=0,c_d} + N^{e_{d,\epsilon=1,\lambda=0}}_{-d,\epsilon=0,c^*=c_d}}{|c_d|\eta + N_{-d,\epsilon=1,\lambda=0,c_d} + N_{-d,\epsilon=0,c^*=c_d}} \cdot$$
$$\prod_{d,r=1} \frac{\zeta + N^{c^y_d}_{-d,r=1,c_d}}{K\zeta + N_{-d,r=1,c_d}} \cdot$$
$$\prod_{d,\epsilon=0} \frac{\zeta + N^{c^*_d}_{-d,\epsilon=0,c_d}}{K\zeta + N_{-d,\epsilon=0,c_d}} \cdot M(c_d)$$

Sampling c^*:

$$p(c^*_i|w, e, \epsilon, \lambda, z, r, c, c^*_{-i}, c_y, \Theta') \propto p(e_i|c^*, e_{-i}, \epsilon, \lambda, c, \Theta')p(c^*_i|w, e_{-i}, c^*_{-i}, z, r, \epsilon, \lambda, c, c_y, \Theta')$$

If $\epsilon_i = 0$:

$$p(e_i|c^*, e_{-i}, \epsilon, \lambda, c, \Theta') \propto \int_{A_{c^*_i}} p(e_i|A_{c^*_i}, c^*_i)p(A_{c^*_i}|c^*_i, e_{-i}, \epsilon, \lambda, c, \Theta')dA_{c^*_i}$$
$$\propto \frac{\eta + N^{e_i}_{-i,\epsilon=0,c^*_i} + N^{e_i}_{-i,\epsilon=1,\lambda=0,c_d=c^*_i}}{|c|\eta + N_{-i,\epsilon=0,c^*_i} + N_{-i,\epsilon=1,\lambda=0,c_d=c^*_i}}$$

For other cases, this probability wouldn't change when we change c^*. We now look at the right term:

$$p(c^*_i|w, e_{-i}, c^*_{-i}, c_y, z, r, \epsilon, \lambda, c, \Theta') = \int_{I_{c_d}} p(c^*_i|I_{c_d})p(I_{c_d}|w, e_{-i}, c^*_{-i}, c_y, z, r, \epsilon, \lambda, c, \Theta')dI_{c_d}$$
$$\propto \int_{I_{c_d}} p(c^*_i|I_{c_d})p(c^*_{-i}, c_y|I_{c_d}, \Theta')p(I_{c_d}|\Theta')dI_{c_d}$$
$$\propto \frac{\zeta + N^{c^*_i}_{-i,\epsilon=0,c^*_i} + N^{c^*_i}_{r=1,c_y=c^*_i}}{K\zeta + N_{-i,\epsilon=0,c^*_i} + N_{r=1,c_y=c^*_i}}$$

Sampling A_c:

Note that we assume members in the same community share the same topic interest, hence the A_c doesn't have an impact on the word generation process.

$$\begin{aligned}&p(A_c|e, w, z, r, c, c^*, c^y, \epsilon, \lambda, \Theta')\\&\quad\propto p(e|A_c, c, c^*, \epsilon, \lambda, \Theta')p(A_c|c, c^*, \epsilon, \lambda, \Theta')\\&\quad\sim Dir(\eta + N^{e_i}_{\epsilon=1,\lambda=0,c} + N^{e_i}_{\epsilon=0,c^*=c})\end{aligned}$$

And we have

$$A_c(e_i) = \frac{\eta + N^{e_i}_{\epsilon=1,\lambda=0,c} + N^{e_i}_{\epsilon=0,c^*=c}}{|c|\eta + N_{\epsilon=1,\lambda=0,c} + N_{\epsilon=0,c^*=c}}$$

Since we use hard membership in PoLIM, theoretically the counts of $N^{e_i}_{\epsilon=1,\lambda=0,c}$ for e_i not in c should be 0. However, this would make it difficult for the algorithm to improve from a bad initialization. Hence, we use a discount factor $\gamma = 0.1$ here, for each instance of e_i not in c, it is counted as γ instead of 1 for the first half of the iterations. Then the γ value linearly decreases to 0 in the following iterations so that the final results are consistent with our hard membership assumption.

Sampling ϵ:

$$p(\epsilon_i = 0|\epsilon_{-i}, e, \lambda, \pi, c, c^*, \Theta') \propto p(e_i|\epsilon_i = 0, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta')p(\epsilon_i = 0|\epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta') \tag{C.30}$$

The left term of Eq. C.30 can be written as:

$$\begin{aligned}&p(e_i|\epsilon_i = 0, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')\\&\quad= \int_{A_{c_i^*}} p(e_i|A_{c_i^*}, \epsilon_i = 0, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')p(A_{c_i^*}|\epsilon_i = 0, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')dA_{c_i^*}\\&\quad\propto \frac{\eta + N^{e_i}_{-i,\epsilon=1,\lambda=0,c=c_i^*} + N^{e_i}_{-i,\epsilon=0,c_i^*}}{|c_i^*|\eta + N_{-i,\epsilon=1,\lambda=0,c=c_i^*} + N_{-i,\epsilon=0,c_i^*}}\end{aligned} \tag{C.31}$$

The right term of Eq. C.30 is:

$$p(\epsilon_i = 0|\epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta') = \mu$$

For $\epsilon_i = 1$, if $\lambda_i = 0$, the left term of Eq. C.30 can be similarly written as:

$$\begin{aligned}&p(e_i|\epsilon_i = 0, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')\\&\quad= \int_{A_{c_i}} p(e_i|A_{c_i}, \epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')p(A_{c_i}|\epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \Theta')dA_{c_i}\\&\quad\propto \frac{\eta + N^{e_i}_{-i,\epsilon=1,\lambda=0,c_i} + N^{e_i}_{-i,\epsilon=0,c^*=c_i}}{|c_i|\eta + N_{-i,\epsilon=1,\lambda=0,c_i} + N_{-i,\epsilon=0,c^*=c_i}}\end{aligned} \tag{C.32}$$

The right term of Eq. C.30 is $1-\mu$.

Similarly, If $\lambda_i = 1$, we have:

$$\begin{aligned}
p(\epsilon_i = 1|\epsilon_{-i}, e, \lambda, \pi, c, c^*, \Theta') &\propto p(e_i|\epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta')p(\epsilon_i = 1|\epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta') \\
&= (1-\mu)p(e_i|\epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta') \\
&= (1-\mu)\int_\pi p(e_i|\pi, \epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta')p(\pi|\epsilon_i = 1, \epsilon_{-i}, e_{-i}, c, c^*, \lambda, \pi, \Theta')d\pi \\
&\propto= (1-\mu)\frac{\xi + N^{e_i}_{-i,\epsilon=1,\lambda=1}}{D\xi + N_{-i,\epsilon=1,\lambda=1}}
\end{aligned}$$

Sampling λ:

$$\begin{aligned}
p(\lambda_i = 0|\lambda_{-i}, e, \epsilon, \pi, c, c^*, \Theta') &\propto p(e_i|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')p(\lambda_i = 0|\lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&= \rho \cdot p(e_i|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')
\end{aligned}$$

If $\epsilon_i = 0$,

$$\begin{aligned}
&p(e_i|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&\quad= \int_{A_{c^*}} p(e_i|A_{c^*}, \lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')p(A_{c^*}|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')dA_{c^*} \\
&\quad\propto \frac{\eta + N^{e_i}_{-i,\epsilon=0,c^*} + N^{e_i}_{-i,\epsilon=1,\lambda=0,c_d=c^*}}{|c^*|\eta + N_{-i,\epsilon=0,c^*} + N_{-i,\epsilon=1,\lambda=0,c_d=c^*}}
\end{aligned}$$

If $\epsilon_i = 1$,

$$\begin{aligned}
&p(e_i|\lambda_i = 1, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&\quad= \int_{A_{c_i}} p(e_i|A_{c_i}, \lambda_i = 1, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')p(A_{c_i}|\lambda_i = 1, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')dA_{c_i} \\
&\quad\propto \frac{\eta + N^{e_i}_{-i,\epsilon=0,c^*=c_i} + N^{e_i}_{-i,\epsilon=1,\lambda=0,c_i}}{|c_i|\eta + N_{-i,\epsilon=0,c^*=c_i} + N_{-i,\epsilon=1,\lambda=0,c_i}}
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
p(\lambda_i = 1|\lambda_{-i}, e, \epsilon, \pi, c, c^*, \Theta') &\propto p(e_i|\lambda_i = 1, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')p(\lambda_i = 1|\lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&= (1-\rho) \cdot p(e_i|\lambda_i = 1, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')
\end{aligned}$$

If $\epsilon_i = 0$,

$$\begin{aligned}
&p(e_i|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&\quad= \int_{A_{c^*}} p(e_i|A_{c^*}, \lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')p(A_{c^*}|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta')dA_{c^*} \\
&\quad\propto \frac{\eta + N^{e_i}_{-i,\epsilon=0,c^*} + N^{e_i}_{-i,\epsilon=1,\lambda=0,c_d=c^*}}{|c^*|\eta + N_{-i,\epsilon=0,c^*} + N_{-i,\epsilon=1,\lambda=0,c_d=c^*}}
\end{aligned}$$

If $\epsilon_i = 1$,

$$\begin{aligned}
&p(e_i|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') \\
&\quad = \int_\pi p(e_i|\pi, \lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') p(\pi|\lambda_i = 0, \lambda_{-i}, e_{-i}, \epsilon, \pi, c, c^*, \Theta') d\pi \\
&\quad \propto \frac{\xi + N^{e_i}_{-i,\epsilon=1,\lambda=1}}{D\xi + N_{-i,\epsilon=1,\lambda=1}}
\end{aligned}$$

Sampling M:

$$p(M|e, w, \epsilon, \lambda, r, y, c, c^*, c^y, \Theta') \propto p(c|M, \Theta') p(M|\Theta') \sim Dir(\sigma + N^{c_i})$$

Sampling I:

$$p(I|c^y, c^*, c, \Theta') \propto p(c^y|I, c, \Theta') p(c^*|I, c, \Theta') p(I|c, \Theta') \sim Dir(\zeta + N^{c_i}_{c^y} + N^{c_i}_{c^*})$$

Sampling θ:

$$p(\theta_{c_i}|z, c, c^y, r, \Theta') \propto p(z|\theta_{c_i}, c, c^y, r, \Theta') p(\theta_{c_i}|c, c^y, r, \Theta') \sim Dir(\alpha + N^{z_i}_{r=0,c_i} + N^{z_i}_{r=1,c^y=c_i})$$

Sampling ρ:

For all these parameters, we assume conjugate Beta prior, hence we have

$$p(\rho_i|\lambda, \Theta') \propto p(\lambda|\rho_i, \Theta') p(\rho_i|\Theta') \sim Beta(0.5 + N^{\lambda_i}_{\epsilon=1})$$

Sampling v:

$$p(v_i|\epsilon, r, \Theta') \propto p(\epsilon, r|v_i, \Theta') p(v_i|\Theta') \sim Beta(0.5 + N_e^{\epsilon_i=1} + N_d^{r_i=0})$$

Sampling u:

$$p(u|s, \Theta') \propto p(s|u, \Theta') p(u|\Theta') \sim Beta(0.5 + N^{s_i}_w)$$