

琉球大学学術リポジトリ

自然災害テキスト情報源の主要記事抽出に対する教師なし分類化に関する研究

メタデータ	言語: 出版者: 琉球大学 公開日: 2015-04-14 キーワード (Ja): キーワード (En): 作成者: Enrique, Gutierrez Carlos, エンリケ, グティエレッツ カルロス メールアドレス: 所属:
URL	http://hdl.handle.net/20.500.12000/30654

Doctoral Dissertation of Engineering

**Unsupervised Classification for
Main Features Extraction in
Natural Disaster Text Sources**

March 2015

by

Carlos Enrique Gutierrez

A dissertation submitted to the Graduate School of
Engineering and Science, University of the Ryukyus,
in partial fulfillment of the requirements for the degree of

Doctor of Engineering

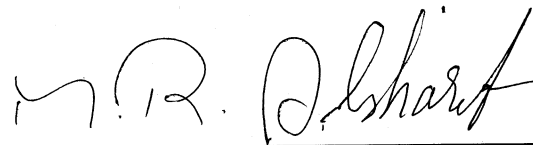


**Interdisciplinary Intelligent Systems Engineering
Graduate School of Engineering and Science
University of the Ryukyus**

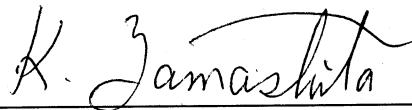
Supervisor: Prof. Mohammad Reza Alsharif

We, the undersigned, hereby, declare that we have read this thesis and we have attended the thesis defense and evaluation meeting. Therefore, we certify that, to the best of our knowledge this thesis is satisfactory to the scope and quality as a thesis for the degree of Doctor of Engineering under Interdisciplinary Artificial Intelligence Doctoral Course, Graduate School of Engineering and Science, University of the Ryukyus.

THESIS REVIEW & EVALUATION COMMITTEE MEMBERS



(Chairman) Prof. Mohammad Reza Alsharif



(Committee) Prof. Katsumi Yamashita



(Committee) Prof. Morikazu Nakamura

Abstract

After a natural disaster and during the post-disaster reconstruction time, on-line newspapers, social networks and blogs become very active describing many events interrelated. This work proposes several unsupervised models to extract main features and patterns from high dimensional text data generated during natural disasters. The main idea is to provide automatic and independent analysis tools of complex data which can be used rapidly when it is most needed.

Firstly, we explore dimensionality reduction and patterns discovering by principal components analysis; an evolutionary description of news through the time is proposed by showing the activated principal components. News are entered sequentially into a proposed algorithm and most influential patterns are shown describing their evolution. Components' meaning is extracted by finding strongest variables within each pattern, in order to improve the analysis.

Secondly, spatial and temporal properties of a news data set are extracted by self organizing maps (SOM). A model is proposed to obtain a low dimensional representation of the input data as quantization points; these new vectors are clustered on map by K-means algorithm to detect potential patterns. A semantic component is added to support interpret-ability. Temporal dependency is detected by tracking SOM units activation over the time by a time-dependency matrix, units are clustered and temporal patterns show up.

Besides that, a linear prediction model is proposed to discover trend topics on news stream by uncovering the most influential variables. Each input is classified on the fly within 2 dummy categories and entered into a linear model with shrinkage operators where strongest variables prevail while those with negligible characteristics are removed.

In addition, a random forest model composed for more than 200 decision trees is proposed to uncover predominant features from a large set of tweets, features are organized as a hierarchy of main variables where rules and an approximation of how information flows during an emergency are detected.

Furthermore, particle filtering is applied to track a set of related words, a defined topic, within a news stream. Topic' relevance is estimated through the time by using a state-space model based on uni-gram model. Sampling importance re-sampling (SIR) algorithm is used to compute the posterior distribution value using available observations. An observation based correction term is incorporated to SIR every time particles and their associated weights are generated, improving the estimation of the posterior probability.

Finally, a Bayesian model called Latent Dirichlet Allocation (LDA) is adapted to discover topics on Twitter stream text data, uncovering natural disaster related topics over the time; the inferences expose the concept of potential significant issues on real time.

Natural disaster are unexpected events with negative impact in social, economical, cultural, environmental, psychological, and technological aspects of a society. We believe the proposed models in our work can serve as an alternative to supervised models which, most of the times, requires much more time and energy for implementation.

List of Publications

International Journals Papers: Published/Accepted

1. **Carlos Enrique Gutierrez**, Mohammad Reza Asharif, K. Yamashita, M. Khosravy, “*A Tweets Mining Approach to Detection of Critical Events Characteristics using Random Forest*”, IJNGC International Journal of Next-Generation Computing, Vol. 5, No.2, pp.167-176, July 2014.
2. **Carlos Enrique Gutierrez**, Mohammad Reza Asharif, K. Yamashita, “*Uncovering Trending Topics on Data Stream by Linear Prediction Modeling*”, IJCSNS International Journal of Computer Science and Network Security, vol.14, no.5, pp.1-7, May 2014.
3. **Carlos Enrique Gutierrez**, Mohammad Reza Asharif, H. Cuiwei, M. Khosravy, R. Villa, K. Yamashita, H. Miyagi, “*Tracking a Topic in News Stream by Particle Filtering*”, ICIC Express Letters, vol.8, no.4, pp.1127-1134, April 2014.
4. **Carlos Enrique Gutierrez**, M.R. Alsharif, H. Cuiwei, M. Khosravy, R. Villa, K. Yamashita, H. Miyagi, “*Uncover News Dynamic by Principal Component Analysis*”, ICIC Express Letters, vol.7, no.4, pp.1245-1250, April 2013.

International Conferences Papers: Presented/Accepted

1. **Carlos Enrique Gutierrez**, Mohamad Reza Alsharif, Mahdi Khosravy, Katsumi Yamashita, Rafael Villa, “*Main Large Data Set Features Detection by a Linear Predictor Model*”, ICCMSE 2014, 10th International Conference of Computational Methods in Sciences and Engineering, Greece, pp.733-737, April 2014.
2. **Carlos Enrique Gutierrez**, Mohamad Reza Alsharif, K. Yamashita, R. Villa, H. Cuiwei, H. Miyagi, “*Spatial-Temporal Clustering of a Self-Organizing Map*”, Las Vegas, USA, DMIN-WorldComp 2013, pp.154-159, July 2013.
3. **Carlos Enrique Gutierrez**, M.R. Alsharif, R. Villa, K. Yamashita, H. Miyagi, “*Data Pattern Discovery on Natural Disaster News*”. Sapporo, Japan, ITC-CSCC, ISBN 978-4-88552-273-4/C3055, 2012.

4. **Carlos Enrique Gutierrez** , Mohamad Reza Alsharif , Rafael Villa, Katsumi Yamashita, He Cuiwei, Hayao Miyagi, Shiro Tamaki, “*Semantic Self-Organizing Map for Natural Disasters News*”, JC-SAT Joint Conference on Satellite Communications, Okinawa, Japan, pp.67-71, Feb. 2013.

National Conferences Papers: Presented/Accepted

1. **Carlos Enrique Gutierrez**, M.R. Alsharif, K. Yamashita, R. Villa, “*Natural Disasters Topics Detection Over Twitter Data Stream*”, Kyoto, Japan, 29th SIP symposium, pp.492-496, Nov. 2014.
2. **Carlos Enrique Gutierrez**, M.R. Alsharif, K. Yamashita, H. Miyagi, “*A correction Term Based SIR Particle Filter Algorithm*”, Shimonoseki, Japan, 28th SIP symposium, pp.385-389, Nov. 2013.
3. **Carlos Enrique Gutierrez**, M.R. Alsharif, H. Cuiwei, R. Villa, K. Yamashita, H. Miyagi, K. Kurata, “*Natural disaster online news clustering by self-organizing maps*”, Ishigaki, Japan, 27th SIP symposium, pp.246-251, 2012.

Acknowledgements

The completion of this program could not have been accomplished without the support of my Supervisor; I would like to thank Prof. Mohammad Reza Alsharif, for his great guidance and support during the PhD program.

I would like to express my very great appreciation to Prof. Hayao Miyagi for his valuable and constructive suggestions during the starting-up and development of this research work. His willingness to give his time so generously has been very much appreciated.

My grateful thanks are also extended to Prof. Katsumi Yamashita and Prof. Morikazu Nakamura for serving as the jury on my dissertation committee and for their suggestion to improve my work.

Also, thank you to all the Professors, Associate and Assistant Professors and staff of the University of the Ryukyus for their kind assistance.

A special thanks to the Japan Government, Monbukagakusho-Mext Scholarship program for providing me with the necessary funds to carry out successfully my research and PhD program.

I wish to express my gratitude to all my friends in Okinawa, for their company and help, that made my experience here more memorable.

Finally, thank you to my family for their support, love and encouragement. I dedicate this work to them.

Contents

1	Introduction	11
1.1	Introduction	11
1.2	Unsupervised extraction	12
1.3	Unsupervised Classification for Main Features Extraction	12
2	Text Patterns Discovery by Principal Component Analysis	14
2.1	Introduction	14
2.2	Principal Component Analysis	14
2.3	Text Dimensionality Reduction	16
2.4	PCA semantics	17
2.5	PCA dynamic	18
2.6	Conclusion	20
3	Spatial, Temporal and Semantic Text Properties Extraction by Self-Organized Maps	22
3.1	Introduction	22
3.2	Self-Organizing Maps	23
3.3	SOM training	23
3.4	SOM clustering	26
3.5	SOM semantics	27
3.6	SOM Temporal Analysis	30
	3.6.1 Temporal Clustering	30
3.7	Conclusion	33
4	Features Detection by Linear Predictor Model	36
4.1	Introduction	36
4.2	Linear Predictor/Regression Models	37
	4.2.1 Shrinkage methods	38
4.3	Computing Lasso Optimization	39
4.4	Lasso for Uncovering Trending Topics on Streaming Data	42
4.5	Conclusion	47
5	Features Detection by Random Forest	48
5.1	Introduction	48
5.2	Decision Trees and Random Forest	48
5.3	Mining on Tweets	50
	5.3.1 Numerical Representation of Tweets	51
	5.3.2 Random Forest Training	51
5.4	Conclusion	55

6	Tracking News Topics by Particle Filtering	58
6.1	Introduction	58
6.2	Particle Filtering	58
6.3	Tracking a Topic	60
6.4	Sampling Importance Re-sampling	61
6.5	Model Evaluation	64
6.6	A Correction Term Based SIR algorithm	65
6.7	Conclusion	67
7	Natural Disasters Topics Detection Over Twitter Data Stream	70
7.1	Introduction	70
7.2	Latent Dirichlet Allocation	71
7.3	LDA training	73
7.4	Inference	75
7.5	Conclusion	78
8	Conclusions	79

List of Tables

2.1	Top five principal components meaning	19
3.1	Examples of quantization points after SOM training	25
3.2	Examples of representative words for quantization points	26
3.3	News grouped by Cluster 4	28
3.4	Main temporal clusters detected by proposed model	35
4.1	Shrinkage simulation results	41
7.1	Main topics learned	75

List of Figures

2.1	Selection of the most important principal components by eigenvalues values	17
2.2	Two most important principal components and news associated . . .	18
2.3	Meaning Detection for First Principal Component	19
2.4	News distribution on three main components	20
2.5	Activation of principal components over the time	21
3.1	Trained SOM and Quantization points generated	25
3.2	Convergence by analyzing objective function	27
3.3	Clusters generated on SOM Map	29
3.4	Trained Semantic SOM and Quantization points	31
3.5	After spatial training, units activated are represented as vectors. They are the inputs of a time-dependent matrix T that learns over the time temporal relations among units	32
3.6	Temporal clustering example. 1^{st} cluster start with column i (most frequent unit), taking most-connected unit j after 1^{st} loop. During 2^{nd} loop unit j takes unit 3. This last unit takes unit i at 3^{rd} loop, but it is already in the cluster, therefore cluster 1 is closed	33
4.1	Strongest variables for shrinkage parameter $\delta = 10$	41
4.2	Strongest variables for shrinkage parameter $\delta = 4$	42
4.3	Proposed algorithm results after 25 iterations	45
4.4	Proposed algorithm results after 420 iterations	46
4.5	Histograms of words	47
5.1	General diagram of our implementation on tweets. Randomness is added into <i>bagging</i> and <i>node optimization</i> stages	50
5.2	Users' reaction on Twitter for a M3.3 Earthquake at Greater Los Angeles Area, California, May 8th 2014	52
5.3	Random Trees main features detection	54
5.4	Hierarchy of main features for rules detection	56
6.1	State variable R and observations Y news set	62
6.2	SIR algorithm, also known as bootstrap filter	63
6.3	Topic's relevance and its estimations by SIR particle filtering	65
6.4	Evaluation by amount of particles and standard deviation value . . .	66
6.5	SIR algorithm single iteration showing particles generated by a Gaussian function with their weights, and state and observation values . .	66
6.6	Particles values modified by a correction term	67

6.7	Comparison of estimated values between SIR and SIR with correction term	68
6.8	Comparison SIR and SIR with correction term by mean squared error	68
7.1	LDA graphical model	72
7.2	Training evaluation by log-likelihood	74
7.3	Natural disaster topics monitoring system	76
7.4	Query on Twitter for keyword “earthquake”	77
7.5	Web monitoring system of emergency topics	78
8.1	Unsupervised methods summary of advantages and disadvantages . .	80

Chapter 1

Introduction

1.1 Introduction

Natural disasters shocks society every time they occur. Trough the time several countries and organizations have been improving their predictions, readiness, and response plans in case of any scale of emergencies, from a small one to a massive contingency. Although, we are still not ready at all for what will happened in coming years, people are becoming aware of their impact on the world around them and the fact that everything we do to the world, whether destructive or creative, will eventually be reflected in that world. Sad and hard experiences, like March 2011 in Japan, demonstrated that planning and assistance must be improved, by all means. A recent and important means is web 2.0, where *news and social media* play the important role of providing information. Web 2.0 is a more social, collaborative, interactive and responsive web. It is a change in the philosophy of web companies and web developers, but more than that, Web 2.0 is a change in the philosophy of society as a whole. In [Kaplan and Haenlein, 2010] Web 2.0 is defined as ideological and technological foundations where new internet-based applications known as “social media” allow the creation and exchange of user-generated content. The social interaction among people in which they create, share or exchange information and ideas in virtual communities and networks is the predominant characteristic. A clear and new example of how web 2.0 assist during natural disasters is Google Person Finder [Inc., 2011]. This application helps people reconnect with loved ones in the aftermath of humanitarian and natural disasters. Person finder was launched during the Japanese earthquake of 2011, and utilized again during the devastation caused by Typhoon Haiyan in the Philippines. Worried friends and relatives can search for the status of a missing person, other users can fill in the gaps and, in the best cases, reunification can happen. In times when phone services are collapsed or interrupted, the web 2.0 and its online tools can be extremely beneficial.

Following disaster, people frequently feel disoriented, with a variety of thoughts and feelings such as anxiety and fear; however, they produce a huge amount of information by sharing comments on social media. Besides that, on-line newspaper and blogs become very active describing many situations interrelated. People talk about concerns, worries, and several other aspects that deserve to be analyzed; it might lead to uncover the main characteristics and patterns of an emergency situation. Discovered features may help to assess the consequences of a natural disaster and to identify needs, physical assets, damages, economic losses, and other related aspects.

The present work presents a number of contemporary and new unsupervised artificial intelligence techniques applied on static and streaming data generated during natural disasters on web 2.0.

1.2 Unsupervised extraction

Main sources on web 2.0 are text; text main disadvantage is its high-dimensionality, if we take in account that each single word represents a different variable, it is challenging to process and visualize features and relationships, fortunately, groups of variables often move together; one reason for this is that more than one variable might be measuring the same driving principle governing the system's behavior. Text data from a widespread social network called Twitter has been used after March 2011 earthquake in Japan to extract information; [Neubig et al., 2011] described a system to mine information regarding the safety of people in the disaster-stricken area by using segmentation, named entity recognition, and tweet classification. Previously [Corvey et al., 2012] described extraction of linguistic and behavioral information from tweets by named entity recognition as well. Text sources, also, has been used in an earthquake detector proposed by [Sakaki et al., 2010], where an algorithm to monitor tweets and to detect a target event is proposed based on a classifier of tweets.

Most of the related works propose algorithm and systems that depend on a *supervised* learning. Named entity recognition requires a big amount of annotations on training set to extract the desired information. In [Neubig et al., 2011], after tsunami hit Japan coast, a big effort of a group of volunteers allowed to create a reliable training data set in a relative short time. Classifiers of tweets also required of a training data set to recognize the new input. When it comes to emergencies, time and speed are everything; to gather the information, to organize the volunteer's team, to build the tools, to prepare the training data, to implement system in production; are tasks that might be thought in an *unsupervised* way. Our work explores unsupervised tools and adapts them in order to get main features and patterns of natural disaster text sources.

1.3 Unsupervised Classification for Main Features Extraction

In the following chapters, it is shown the application of classic methods, such as *Principal Component Analysis*, *Self-Organized Maps*, and *linear Models*, as well as modern techniques like *Random Forests*, *Particle Filtering* and *Latent Dirichlet Allocation* to extract in a purely *unsupervised way* main characteristics of an emergency event.

The application of the mentioned algorithm includes for each case novel adaptations and complementary algorithms to generate new insides from data. The techniques were carefully adapted to process huge amount of data effectively and to reduce the dimensionality of input data set as much as possible. In addition, visualization and output of the topology and features extracted are organized using weights or hierarchies.

In real situations, to guaranty decision making process, it is needed to extract and organize most important information as soon as possible to avoid adding more confusion; we believe that the proposed tools in this work can contribute to generate an *unsupervised model* for faster natural disaster assessment without the large load of preparing reliable training data.

Chapter 2

Text Patterns Discovery by Principal Component Analysis

2.1 Introduction

After a natural disaster, many aspects and events are described by newspapers online, that could be deeply analyzed by data mining and machine learning methods. It may lead us to discover dynamic of variables and text patterns. Text sources are considered high-dimensional data, therefore it is needed a manageable method not only to discover patterns but to reduce the dimensionality of the entire data set. In this chapter we describe Principal Component Analysis (PCA) and how this well-known process can be used to reach these two objectives, obtaining at the same time a good visualization of the inherent structure and topology of the data collection. PCA allows the discovery of new variables known as principal components. In the following sections it is described how strongest principal components are related to text patterns discovery and it is presented a simple but novel algorithm to obtain the semantic meaning of components. Text sources, mainly, are pre-processed and arranged as numerical matrices, it will allow to apply PCA and transform the data set in an meaningful output described at the end of the chapter.

2.2 Principal Component Analysis

Principal component analysis is a mathematical method that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Each principal component is a linear combination of the original variables, and all of them are orthogonal to each other, therefore there is no redundant information. The principal components as a whole form an orthogonal basis for the data space. By using this method it is possible to identify patterns in data, and express them in such a way as to highlight their similarities and differences. Its advantage is that once you have found these patterns, it is possible to compress the data by reducing the number of dimensions, without much loss of information [Smith, 2002].

Let's X and Y be $m \times n$ matrices related by a linear transformation P ($n \times n$); m indicates the observation number with n variables. X is the initial data set and Y

is a re-representation of X . PCA re-express the initial data as a linear combination of its basis vectors:

$$Y = XP \quad (2.1)$$

Let's define the following:

p_i are column vectors of P .

x_i are row vectors of X .

y_i are row vectors of Y .

Therefore, the explicit form of equation 2.1 is:

$$Y = \begin{bmatrix} x_1 \cdot p_1 & x_1 \cdot p_2 & \dots & x_1 \cdot p_n \\ x_2 \cdot p_1 & x_2 \cdot p_2 & \dots & x_2 \cdot p_n \\ \dots & \dots & \dots & \dots \\ x_m \cdot p_1 & x_m \cdot p_2 & \dots & x_m \cdot p_n \end{bmatrix} \quad (2.2)$$

Each row of has the form:

$$y_i = [x_i p_1 \dots x_i p_n] \quad (2.3)$$

We recognize that each coefficient of y_i is a dot product of x_i with the corresponding column in P , in other words, the j^{th} coefficient of y_i is a projection on to the j^{th} column of P . P represents a change of basis [Shlens, 2014]. Geometrically, it is a rotation and a scale which again transforms X into Y .

By assuming linearity, the problem reduces to find the appropriate change of basis, the columns vectors p_i of P , also known as the principal components of X . But first, let's define S_x as the covariance matrix of X . S_x is a simple way to quantify redundancy by calculating the spread between variables. X , usually, is in mean deviation form because the means have been subtracted off or are zero.

$$S_x = \frac{1}{n-1} X^T X \quad (2.4)$$

S_x is a square symmetric $n \times n$ matrix. Its diagonal terms are the variance of particular variables. The off-diagonal terms are the covariance between variables. From S_x the eigenvectors with their corresponding eigenvalues are calculated. Eigenvectors are a special set of vectors associated with a linear system of equations (i.e., a matrix equation), also known as characteristic vectors, proper vectors, or latent vectors [Marcus, 1988]. Each eigenvector is paired with a corresponding factor so-called eigenvalue by which the eigenvector is scaled when multiplied by its matrix. A non-zero vector p_i is an eigenvector of the covariance matrix S_x if there is a factor λ_i such that:

$$S_x p_i = \lambda_i p_i \quad (2.5)$$

Generalizing:

$$S_x P = \Lambda P \quad (2.6)$$

The full set of eigenvectors is as large as the original set of variables. In PCA, the

eigenvectors of S_x are the principal components of X . Matrix P ($n \times n$) contains n eigenvectors, arranged in a way such as p_1 is the principal component with the largest variances (the most important, the most “principal”); p_2 is the 2nd most important, and so on. In addition, the eigenvalues contained in diagonal matrix Λ are arranged in descending order $\lambda_1 > \lambda_2 > \dots > \lambda_n$ and they represent the variance of X captured by the principal components. This last relation is used for dimensionality reduction when the amount of variables is large, as in the case of text data sources.

$$S_x p_i = \sigma_i^2 p_i \tag{2.7}$$

with

$$\sigma_1^2 > \sigma_2^2 > \dots > \sigma_n^2$$

2.3 Text Dimensionality Reduction

Large variances have important dynamics; therefore, principal components with larger associated variances represent data patterns while those with lower variances represent noise. It is common to consider only the first few principal components whose variances exceed 80% of the total variance of the original data. In order to identify the most important components of a text data set, it is needed to represent text by numbers, in that way the co-variance matrix can be calculated and its eigenvectors and eigenvalues are determined.

To demonstrate PCA usability on text, a data set containing 421 text files (news) is represented as a numerical matrix. The representation is simple and is explained as a sequence of steps:

1. Extraction from each text file of words, creating a dictionary and computing words frequency.
2. Deletion of special characters, numbers, symbols, and meaningless words such as conjunctions, prepositions and adverbs.
3. Application of Stemming process [Porter, 1980] for reducing inflected (or sometimes derived) words to their stem, base or root form. The general idea underlying stemming is to identify words that are the same in meaning but different in form by removing suffixes and endings; for instance, words such as “expanded”, “expanding”, “expand”, and “expands” are reduced to the root word, “expand”. This step is optional but important if a strong dimensionality reduction is needed.
4. Arrange the data set as a matrix of (files \times words), where each element $x_{i,j}$ is a number equal to the frequency of word j at file i . The expected results are, mostly, a high-dimensional matrix.

The obtained high-dimensional matrix is modified by subtracting off the mean for each variable (variables are synonym of words in case of text), and the covariance matrix and its principal component are computed. The selection of the most important components constitutes definitely the dimensionality reduction sought. As an experiment we processed the mentioned news data set, it has the distinction

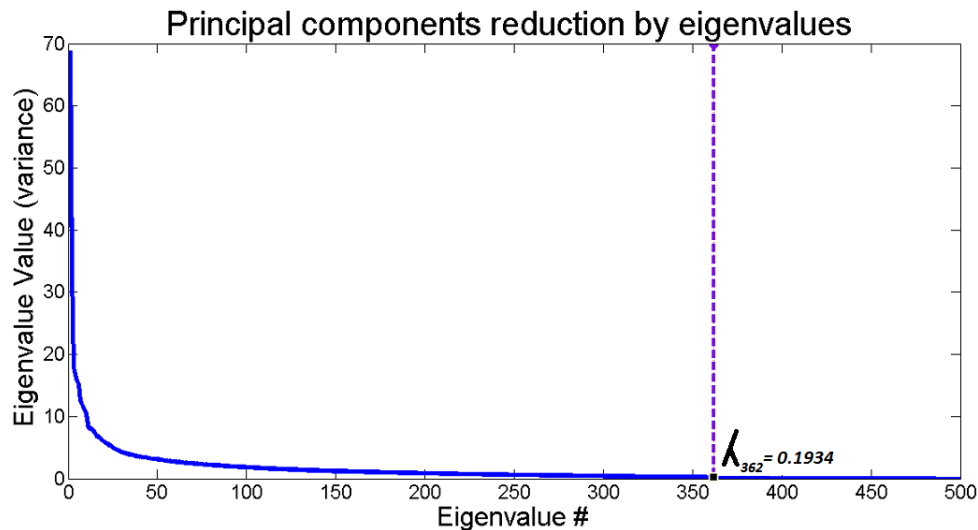


Figure 2.1: Selection of the most important principal components by eigenvalues values

of belonging to the many generated news during last tsunami/earthquake in Japan. As result, a set of 9961 principal components, matrix P , and the corresponding 9961 eigenvalues, diagonal matrix Λ , were obtained. Figure 2.1 shows values of λ which makes possible to define how many principal components of matrix P describe the variability of X . It displays a “ranking of importance” for the principal components based on their eigenvalues. According to the corresponding variances, we would say that the first 362 principal components (out of 9961) describe almost all the variability of the news data set. Let’s take, for instance, the l largest eigenvalues $\lambda_1 > \lambda_2 > \dots > \lambda_l$, ($l < n$); and truncate the associated matrix P at column $l = 362$. It implies a strong dimensionality reduction, in the order of 96%. Figure 2.1 indicates that important dynamics exists along the few first components. This new “change of basis” or reduced ($n \times l$) matrix is called \hat{P} .

2.4 PCA semantics

Principal components with larger associated variances represent data patterns which are mainly a group of variables moving together as one variable, the principal component. Figure 2.2 shows how natural disaster news are distributed along the 1st and 2nd principal components as if they were dots with (x, y) coordinates. Clearly, news related to nuclear emergency are located along one component and news related to crisis in middle east along the other one. PCA semantics seeks for an interpretation or meaning of those components; originally without dimensionality reduction words encapsulates the meaning, however, after transformation that meaning is lost. In order to detect principal components meaning, we extract the most representative variables from each of them, providing a pseudo-summary. We introduce below algorithm as an “easy-to-implement” alternative:

1. for $i = 1$ to l
 - (a) Extract the i -column p_i from \hat{P} (p_i is a principal component)
 - (b) Calculate the absolute value of each element of p_i

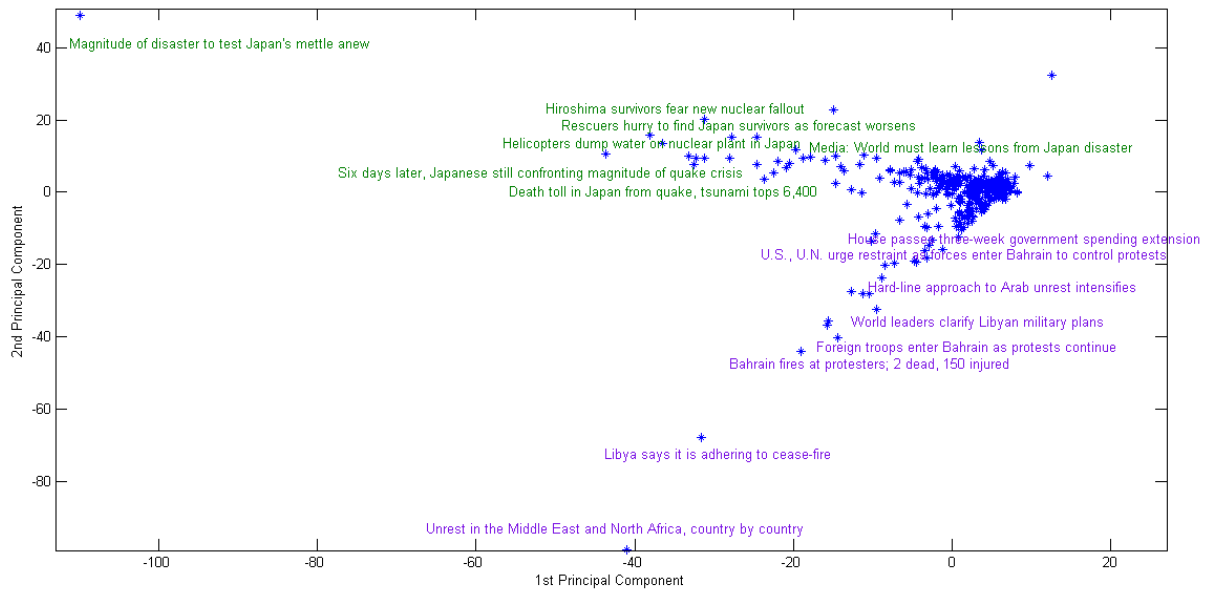


Figure 2.2: Two most important principal components and news associated

- (c) Search for the largest k elements within p_i and take the corresponding k indexes
- (d) Look for the k words from dictionary by matching the k indexes obtained in previous step

2. end

By this process, 362 different “meanings” are obtained, one for each data pattern, said in other words, one meaning for each principal component. Figure 2.3 shows the results of previous algorithm after one iteration, that is, for the 1st and most important data pattern with $k = 5$. Large p_i elements values represent high related variables (words). The set of k words is considered a pseudo-meaning and summary of the data pattern; 1st data pattern is summarized as {‘nuclear’, ‘reactor’, ‘plant’, ‘radiation’, ‘power’}. Any value of k can be selected, in this case 5 was chosen arbitrarily. Experimental results of our proposed algorithm are described in table 2.1. It shows the main words for the first five principal components discovered from news during the 1st week immediately after 2011 earthquake-tsunami in Japan. It is quickly recognized that components 1, 3 and 4 refer to Japan nuclear emergency and components 2 and 5 talks about the crisis in Libya and Bahrain. As a general assumption the entire data set of news is summarized by the meanings of the most important components.

2.5 PCA dynamic

Previously, we observed how news were dispersed on two main components; each news should be more “attracted” to a certain data pattern than to any other, meaning there is a winner component for each news. In order to detect that relation the original data set X ($m \times n$) is re-expressed as Y ($m \times l$) using a new “change of

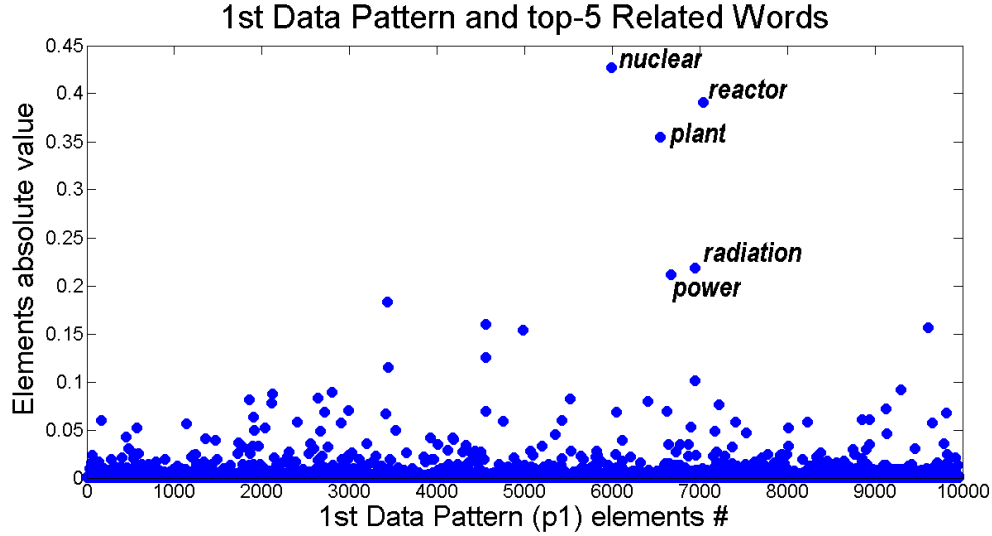


Figure 2.3: Meaning Detection for First Principal Component

1 st component	2 nd component	3 rd component	4 th component	5 th component
nuclear	government	Japan	nuclear	Bahrain
reactor	force	reactor	people	zone
plant	Gadhafi	nuclear	power	Gadhafi
radiation	security	fuel	water	government
power	Bahrain	power	fall	protest

Table 2.1: Top five principal components meaning

basis" matrix \hat{P} ($n \times l$):

$$Y = X\hat{P} \quad (2.8)$$

Equation 2.8, as a dot product, shows how matrix Y compresses X and contains the distribution of the news along the most important l patterns. Figure 2.4 serves to highlight that distribution, showing similarities and differences. The plot provides information about how close news are regarding to the main three data components.

Some news are highly correlated to certain components than others; let's call y_i to the row vectors of Y ; each y_i represents a single news, and its elements $y_i(j)$ are projections of its words on l principal components. Given a news y_i , the largest value of $y_i(j)$ indicates its most correlated data pattern. For instance, news such as *Helicopters dump water on nuclear plant* and *Japan vows to resume aerial, ground efforts to avert nuclear crisis* have largest values on 1st component (its largest value is $y_i(1)$), while news such as *Unrest in the Middle East and North Africa, country by country* and *Libya says it is adhering to cease-fire* have largest value on 2nd component (its largest value is $y_i(2)$). Values of $y_i(j)$ can be used as a measurement of the impact of individual news onto discovered pattern. With this in mind, we present the news sequentially as a data stream, showing at each step an interpretation of the most correlated principal component. An algorithm is proposed to extract the mentioned component and display its meaning:

1. for $i = 1$ to m
 - (a) extract the i -row y_i from Y (y_i is a single news)

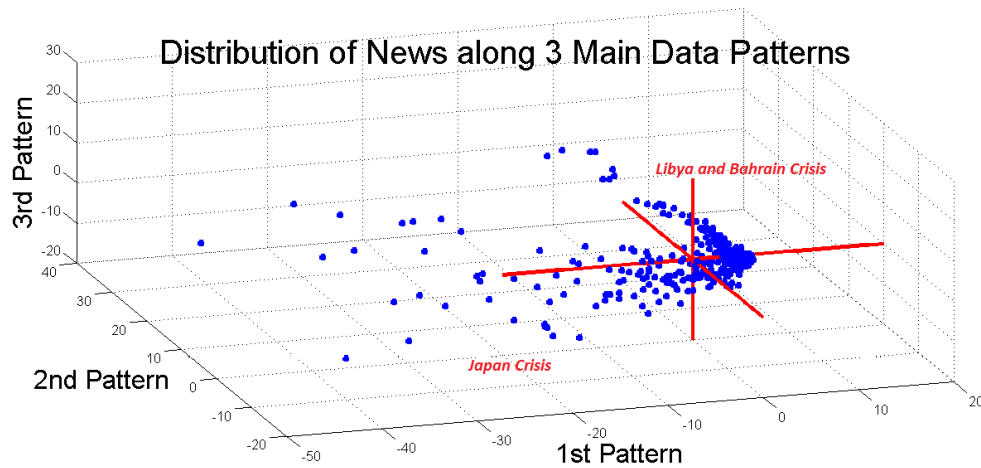


Figure 2.4: News distribution on three main components

- (b) calculate the absolute value of each element $y_i(j)$
- (c) search for the largest element within y_i and take the corresponding k index.
- (d) look for the meaning of the k principal component by matching the k index
- (e) display news title, its largest value within y_i and the meaning found at previous step

2. end

Every time a news is read, our proposed algorithm fires the meaning of the component with the highest value, displaying an evolution curve for each component. Figure 2.5 shows an example for natural disaster data set. The algorithm draws a tracking line and a dot every time one of the main three data patterns is activated by the incoming news. The tracking lines, as curves, show the dynamic of the principal components.

2.6 Conclusion

PCA is easily applied on text sources only if there was a previous data transformation to numerical matrices, it is an obvious disadvantage of PCA. Then, the obtained high-dimensional matrices can be compressed, transformed, and arranged to uncover components dynamic. The discovered components meanings serve as summary for each of them. Main patterns meanings summarize the whole text data set; highest values of $y_i(j)$ serves as an indicator of relevant component on news. The proposed algorithms in this chapter can be effortlessly applied in a larger set of data to provide automatically a general view based on previously seen principal components. A new input can be transformed in a numerical n -dimensional v vector, then transformed by a dot product $v (1 \times n) \hat{P} (n \times l)$, obtaining a reduced input $v (1 \times l)$ that activates a certain principal component.

PCA can be useful when emergencies events occur. In those cases, on internet, an overflow of news is generated describing most of them, many situations inter-related

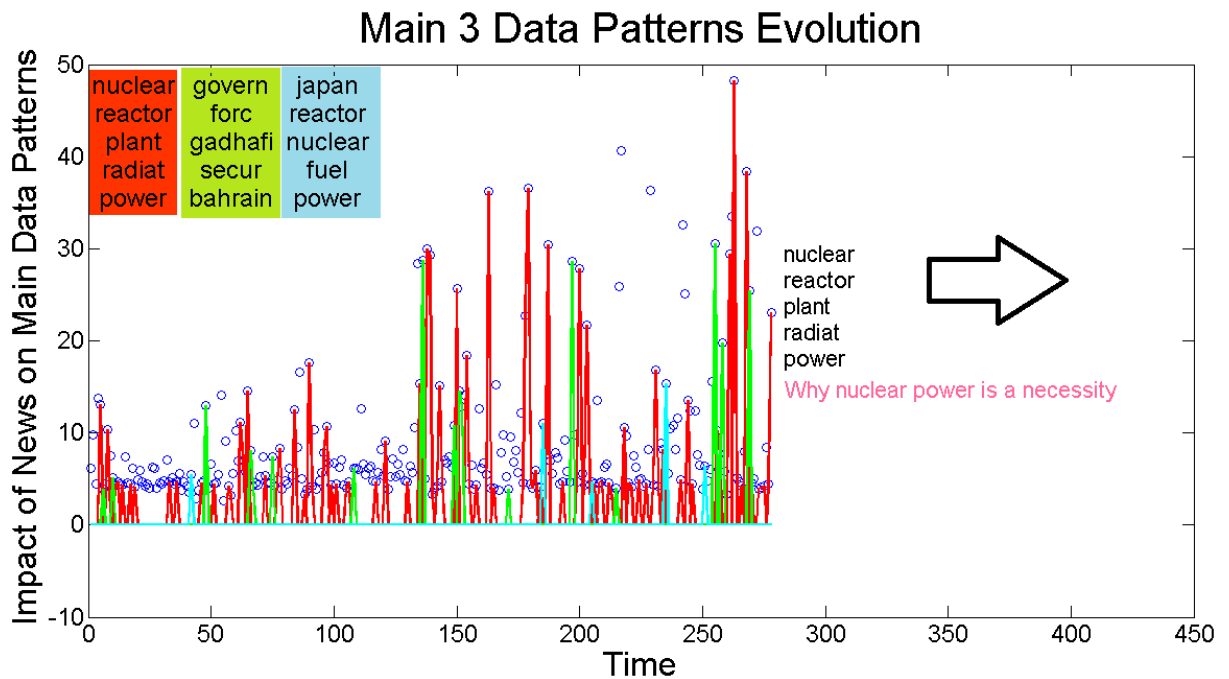


Figure 2.5: Activation of principal components over the time

as the events evolve through time. An analyst can observe the sequence of meanings and understand how news and their topics evolve through time. Although this method is simple to apply, the component and its meaning fired by only one index of an input with many indexes produces sometimes errors for the inferred meaning. For example, the method fired the meaning {'nuclear', 'reactor', 'plant', 'radiation', 'power'} for news "Christians in Egypt stage protest". Clearly the algorithm output is not 100% reliable, however PCA applied on text during emergencies is extremely useful for getting a first impression on large sources, then later, more powerful algorithms must dig deeper to improve the inferences.

Chapter 3

Spatial, Temporal and Semantic Text Properties Extraction by Self-Organized Maps

3.1 Introduction

Exploration of text properties includes unsupervised classification tasks also known as clustering. Clustering is the task of grouping a data set in such a way that inputs in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and essential when large amount of texts demand to be evaluated as could be the case of the search for patterns in texts generated from news and comments in social networks during natural disasters.

One of the most used unsupervised techniques in neural networks is Self-Organizing Maps (SOM); this method is used to produce a low-dimensional representation of the input data space by mapping high dimensional vectors into a 2-dimensional grid. SOM is composed by fewer units than input vectors, this property reduces the complexity when input data set is high dimensional and large amounts become more manageable; training stage produces not only a visual representation but also a set of quantization points. The mentioned SOM properties fits perfectly with the characteristics needed to process and analyze huge amount of text during natural disasters. Sequential news could be quantized in a SOM, then later quantization points could be clustered into groups and interpreted spatially, temporally and mainly semantically.

This chapter describes a novel approach to extract relations within text by applying SOM learning to classify input vectors according to how they are related spatially and temporally; in addition a clustering is applied on SOM map to organize quantization points as groups. Groups are analyzed semantically and in order to get descriptions and meanings which again divide the map semantically. Besides spatial properties, temporal features are analyzed by tracking SOM units' activation over the time, discovering temporal associations among data items. Uncovered spatial-temporal representations can be used for predictive modeling, search of sequential patterns, and mainly for understanding.

3.2 Self-Organizing Maps

A *Self-Organizing Map* is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional discretized representation of the input space of the training samples, called *map*. SOM consists of components called *units*. Associated with each unit there is a weight vector of the same dimension as the input data and a position vector in the map space. In learning stage an input vector is presented to the SOM at each step. These vectors constitute the *environment* of the network. The procedure for placing a vector from data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector. The most popular model of SOM is the model proposed by *Teuvo Kohonen* [Kohonen, 1988] called *Kohonen networks*. Kohonen algorithm introduces a model that is composed of two interacting subsystems. One of these subsystems is a competitive neural network that implements the winner-take-all function. The other subsystem modifies the local synaptic plasticity of the neurons in learning [Rojas, 1996]. Kohonen learning uses a neighborhood function ϕ , whose value $\phi(i, k)$ represents the strength of the coupling between unit i and unit k during the training process. The learning algorithm for Kohonen networks is as follows [Rojas, 1996]:

1. Start: The n -dimensional weight vectors w_1, w_2, \dots, w_m of the m computing units are selected at random. An initial radius of the neighborhood r , a learning constant η , and a neighborhood function ϕ are selected.
2. Step 1: Select an input vector y using the desired probability distribution over the input space.
3. Step 2: The unit k with the maximum excitation is selected (that is, for which the *Euclidean distance* between w_i and y is minimal, $i = 1, 2, \dots, m$).
4. Step 3: The weight vectors are updated using the neighborhood function ϕ and the following rule:

$$w_i = w_i + \eta\phi(i, k)(y - w_i) \quad (3.1)$$

for

$$i = 1, 2, \dots, m$$

5. Step 4: Stop if the maximum number of iterations has been reached; otherwise modify η and ϕ as scheduled and continue with step 1.

By repeating this simple process several times, it is expected to arrive at a uniform distribution of weight vectors for the input space.

3.3 SOM training

Self Organizing Maps applied on text helps to uncover spatial relations within a data set, as first result, SOM provides an understandable map of the input space. Huge text inputs could be organized by quantization points on a reduced SOM map.

To demonstrate it, a SOM network of 2–dimensional 10x10 units is feed with a set of news generated within a week after a natural disaster. News are represented as high dimensional vectors where each element is the frequency of a certain word from a predefined dictionary. Also, it is possible to transform each news using a PCA transformation as described on chapter 2.

Generally, in SOM, variables are normalized by dividing each of them by its standard deviation; after normalization a Kohonen’s network is fed with the training set y_i , for $i = 1, 2, \dots, m$. The network is a square grid with 10 rows and 10 columns, its size is defined base on the amount of input vectors and the expected amount of quantization points. Considering our input training vectors set, we assume that a grid of 100 units may produce a reasonable amount of quantization points and a suitable visualization. After setting the weight vectors w_1, w_2, \dots, w_{100} with random values, we perform several iterations. At each iteration, the complete set of training vectors is entered into the network once. The radius of the neighborhood r and the learning constant η are reduced according to the following schedule:

1. start $r = 20, \eta = 0.1$
2. at 200 iterations $r = 15, \eta = 0.01$
3. at 1000 iterations $r = 10, \eta = 0.01$
4. at 2500 iterations $r = 1, \eta = 0.001$

The neighborhood function $\phi(i, k)$ is defined by:

$$\phi(i, k) = \exp - \left(\frac{|i - k|}{r} \right)^2 \quad (3.2)$$

Where i is the position of the i^{th} unit and k is the position of the unit with the maximum excitation. The neighborhood function ϕ changes according to schedule, producing larger corrections at the beginning of the training that at the end. Figure 3.1 shows the SOM map after training and the 95 quantization points generated.

Convergence of the network is evaluated empirically; we cannot ensure that neighborhood function and the schedule are the best for our network, however, figure 3.1 shows a network in a stable state after 3000 iterations, at this stage the map does not change and weight vectors experiment very small updates. Quantization points are represented as dots with different sizes expressing the amount of input vectors captured by weight vectors. For instance, w_{36} at coordinates (6, 4) capture more inputs than w_{18} at (8, 2). As an example, table 3.1 displays 2 quantization points and the news captured by them.

Each quantization point captures news related spatially, meaning their *Euclidean distance* is minimal; therefore, a quantization point is by itself a group and represents a reduced set of input vectors. SOM training concludes with the generation of the mentioned quantization points. In addition, each of them is interpreted and characterized by a set of main words within the group to provide an idea of their content. As a preliminary step of our analysis, the most representative k words from each quantization point are extracted; any value of k can be selected, in our case we choose 7 arbitrarily. For q_i (from q_1 to q_n), q_i is a vector captured by a certain quantization point and n is the number of q_i captured by the mentioned

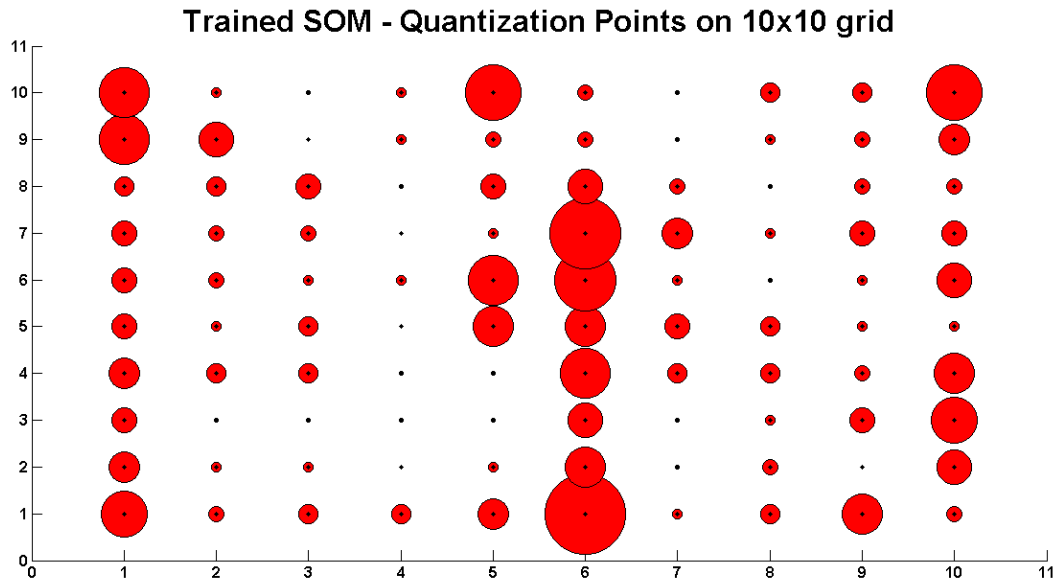


Figure 3.1: Trained SOM and Quantization points generated

Weight (quantization point)	Vector	Coordinates at the map	Some news captured
w_{41}		(1, 5)	*'Will Japan face a mental health crisis?', *'Obama pledges to help Japan rebuild; U.S. issues larger radiation zone', *'Don't panic the people', *'West Coast officials, Obama: Don't worry about radiation risk n U.S.', *'A radioactive hazard zone? Chernobyl's example'
w_{91}		(1, 10)	*'Tensions aside, China sends rescue team, money and supplies to Japan', *'Rescuers hurry to find Japan survivors as forecast worsens', *'Doctors, aid workers get to work in Japan', *'California student from Japan finds family alive on YouTube', *'Amid disaster, Japan's societal mores remain strong', *'Welfare groups race to rescue Japan's abandoned animals', *'Disaster is heavy burden to bear for Japan's elderly', *'Magnitude of disaster to test Japan's mettle anew'

Table 3.1: Examples of quantization points after SOM training

Weight Vector (quantization point)	Representative words
w_{41}	{'radiation', 'health', 'Japan', 'people', 'plant', 'Chernobyl', 'nuclear'}
w_{91}	{'Japan', 'people', 'tsunami', 'rescue', 'disaster', 'team', 'kosaka'}

Table 3.2: Examples of representative words for quantization points

quantization point; the words frequencies for q_i are accumulated and the k largest elements are extracted from dictionary. As result, table 3.2 shows an example for quantization points w_{41} and w_{91} .

3.4 SOM clustering

When the number of SOM units is large, to facilitate quantitative analysis of the map, similar units are clustered. To produce summaries and quantitative descriptions of data interesting groups of map units must be detected from the SOM. While its properties are certainly interesting, even more useful summaries can be prepared if the SOM is separated in groups and these are studied separately. It should be emphasized that the goal here is not to find an optimal clustering for the data but to get good insight into the cluster structure of the data for data mining purposes. Clustering on SOM map is performed as a way to group together weight vectors that are similar. Similar weight vectors suggest similar input vectors with a certain relationship among them. Clustering of the SOM can be seen as a 2^{nd} level grouping, quantization points are the 1^{st} level. The reason of 2^{nd} level grouping on weight vectors is that SOM can be used for categorization of new input data; any new input will fire a weight vector already associated to a certain 1^{st} and 2^{nd} level group, allowing classification.

K-means is one of the most used clustering methods on data mining, it is a simple iterative method to partition a given data set into a number of clusters [Wu et al., 2008]. In our implementation the method is applied on a set of l -dimensional weight vectors $L = \{w_i | i = 1, 2, \dots, m\}$ where the amount of clusters k is calculated by the general rule $\sqrt{\frac{m}{2}}$. The algorithm begins by picking k vectors w_i randomly as the initial centroids and iterates between two steps until the convergence:

1. 1^{st} step: each weight vector is assigned to its closest centroid, by calculating the *Euclidean distance*.
2. 2^{nd} step: the new k centroids are relocated by calculating the center (mean) of all weight vectors into the cluster.

The algorithm minimizes the within-cluster sum of squares also known as *objective function*, which is used to verify its convergence:

$$\arg \min \sum_{i=1}^k \sum_{w_j \in S_i} |w_j - \mu_i|^2 \quad (3.3)$$

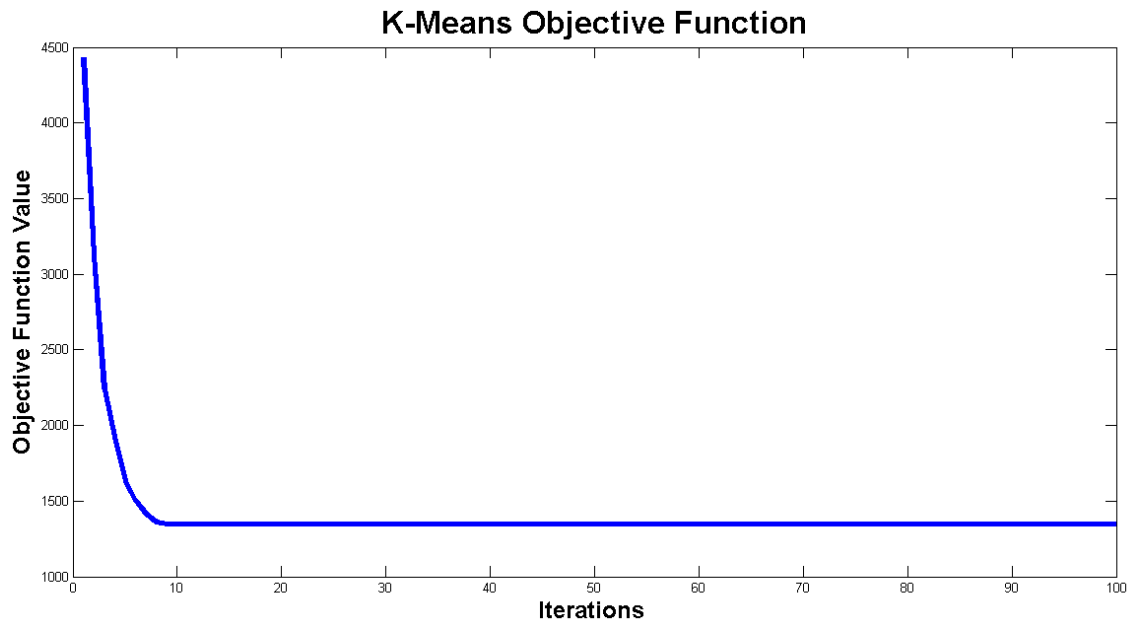


Figure 3.2: Convergence by analyzing objective function

Where μ_i is the centroid of the cluster S_i . Figure 3.2 shows the convergence of the method by plotting the objective function. k clusters are generated and no new assignments is performed from iteration 10.

Experimental results show 7 clusters generated on the set of weight vectors. Figure 3.3 displays the clusters found that can be considered as news patterns. The results are interpreted intuitively by reading news titles; table 3.3 presents news within cluster 4. It is possible to obtain a set of words that characterizes the cluster, in this case the words are {'Japan', 'tsunami', 'earthquake', 'people', 'quake', 'water', 'area', 'rescue'}. These words are the 8 most frequent on cluster 4.

Figure 3.3 demonstrates that plain text can be represented as numerical vectors and serve as input data for a SOM network. A trained SOM produced a representation of the news set into a 2 dimensional map. This representation is a finite number of quantization points and clusters. The groups found are *data patterns* that contain spatially related news.

3.5 SOM semantics

Beside relationships that can be found by a computation of a *spatial distance* between vectors; one of the most important problems in the theory of artificial neural networks is to find an effective, simple and adaptive system able to find "meanings" from raw data. SOM is a unique artificial neural network that can be used to reflect, at least in a basic form, meaning-cognitive representations and relations within raw data. Previously, the self-organizing process ensured clusters mapped to a common localized domain in the map. Now, the semantic factor is added by the placement of the main words that qualifies the clusters.

Semantic relatedness between inputs cannot be detected by comparing or calculating any metric from their attributes. Our proposed solution is to present, after

Cluster	Input Vectors Captured (News)
4	<p>*Weight vector 81: 'Widespread destruction from Japan earthquake, tsunamis', 'Tsunami warnings and advisories remain across Pacific region', 'U.S., Canada threatened by tsunami', 'Japan prepared well for tsunami', 'How a tsunami can strike within minutes', 'Cruise ships at sea safe from tsunami's destruction', 'As U.S. damage measured, emergency declared in California counties', 'Obama declares bond with Japan 'unshakeable'', 'Quake moved Japan coast 8 feet, shifted Earth's axis', 'Rescuers hurry to find Japan survivors as forecast worsens'</p> <p>*Weight vector 82: 'Fire on water: Japan, world watches tsunami strike live', 'Quake survivors pack roads, stores outside hard-hit area', 'Japan quake: It could have been even worse', 'Concern about food, fuel in wake of Japan disasters', 'Quakes not increasing, but human risk is', 'Not business as usual as Japan strives for normality'</p> <p>*Weight vector 91: 'Tensions aside, China sends rescue team, money and supplies to Japan', 'Rescuers hurry to find Japan survivors as forecast worsens', 'Doctors, aid workers get to work in Japan', 'California student from Japan finds family alive on YouTube', 'Amid disaster, Japan's societal mores remain strong', 'Welfare groups race to rescue Japan's abandoned animals', 'Disaster is heavy burden to bear for Japan's elderly', 'Magnitude of disaster to test Japan's mettle anew'</p> <p>*Weight vector 92: 'Even after rescue, survivors struggle to come to grips with disaster'</p> <p>*Weight vector 93: '10 most beautiful waterfalls'</p>

Table 3.3: News grouped by Cluster 4

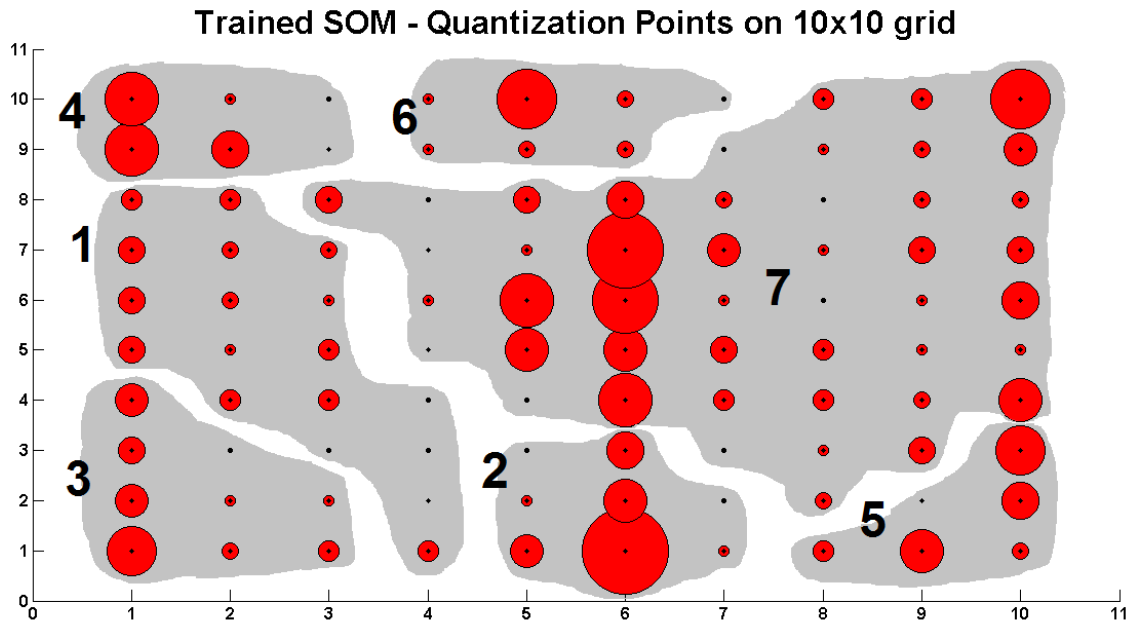


Figure 3.3: Clusters generated on SOM Map

training, not only input vectors but the *context* where they may be located, in that way the map should reflect their logic or semantic similarities. Context is a background, environment, framework, setting, or situation surrounding an event or occurrence. In linguistic it is defined as words and sentences that occur before or after a word or sentence and imbue it with a particular meaning. We assume that the most frequent words have strong correlations with contexts that surround events described in our set of news. Hence we choose to represent each most frequent k^{th} word by a n -dimensional vector, whose k^{th} component has a fixed value equal to k^{th} word's total frequency and whose remaining components are zero. The words are presented to the network and the strongest responsive units are detected and labeled with those words. The responses on the map show how the network captured the relations among the news (figure 3.4). News related to earthquake-tsunami in Japan are distributed on the left side, while those related to Middle East and Libya on the right. Middle units captured a variety of topics. Earthquake-tsunami news are differentiated in sub-categories, corresponding to more specialized items such as radiation, health, energy. The labels uncover the semantic relation between items; they show the contexts where the news items are located. Each news incorporates frequent words in its representation as vector; with a sufficient amount of training the inputs leave memory traces on the same units at which later the words individually converge. Therefore, a meaningful topographic map is obtained by adding 100 most frequent words, showing logical similarities among inputs. It is possible to add more words which will enrich the map and will add details to the semantic relations, but for simplicity and good visualization, we chose only 100 words. Our application emphasizes the spatial arrangement of the units and the separation of the information into different areas. The semantic map gives a meaning to that segregation, although in a high level of semantic, it does completely on unsupervised way.

At this step SOM developed automatically and organized spatially the formation of a *memory* in a way that its layout forms an image of the most important concepts

and their relations. Although the role of the contexts as the most frequent words still very simple, their inclusion enabled to form an interesting semantic structure in which news items are grouped according to similarity. The concept used for the context needs to be improved and extended, incorporating the time dimension due that any perception of meaning, knowledge, logic, etc. usually occurs over the time.

3.6 SOM Temporal Analysis

Exploration of temporal properties of text sources related to natural disaster by using SOM can be achieved by uncovering temporal dependencies of SOM units', and monitoring units activation over the time. Previously SOM stored information throughout its grid in a way such that space structures of the input data set were discovered, now we introduce a temporal learning stage to uncover temporal relations. Spatial-Temporal relations are used in predictive modeling, search of sequential patterns, and mostly used for understanding. In natural disasters case, discovered dependencies would describe causes and contexts of an issue, showing how it evolved or moved over the time.

One of the most interesting problem is to find an effective and simple method able to discover temporal relations; in case of SOM, a temporal component can be added to the map by the analysis of temporal dependencies between units, with the introduction of a time-dependent matrix that stores unit-to-unit and neighborhood-to-unit temporal relations. The main idea is to identify temporal sequences of spatial patterns that are likely to occur one after another.

Previous spatially-trained SOM is fed once more with the input data set and temporal sequences of activated units are monitored and stored in a time-dependent matrix. The temporal aspect comes from movements or changes of the input data. Every time a unit k fires, our model creates a vector d of dimension m , where m is the total amount of SOM units. The element $d(k)$ has a fixed value equal to 1 and the remaining components are zero. Vectors d are the inputs of the time-dependent matrix denoted as T . In order to analyze temporal proximity properties of units, matrix T is created with m rows and m columns and its elements are initialized to 0. Rows correspond to units activated at time $(t - 1)$ and columns to the units at time (t) . It is proposed a model that memorizes the previously activated unit, in a way such that for an input d at time t , the matrix T is updated by increasing $T(i, j)$ an amount equal to a , where i is the unit fired on time $(t - 1)$, and j is the unit fired on time (t) . The value added to $T(i, j)$ corresponds to a transition value a from the past unit to the current unit. Neighbors of unit i are also considered, the reasoning is that if unit j is frequently followed by unit i , the model considers that there is a high probability that neighbors of i follow unit j as well. In that case, matrix T is updated by a scale down increment $(a * \beta)$ in elements $T(N_i, j)$, where N_i denotes neighbors of i . Figure 3.5 displays the process by which a time-dependent matrix T receives inputs and learns temporal relations among units over the time.

3.6.1 Temporal Clustering

After T is built, the process continues by clustering matrix T . The aim is to generate coherent clusters, which means, we seek to detect clusters where the units have high



Figure 3.4: Trained Semantic SOM and Quantization points

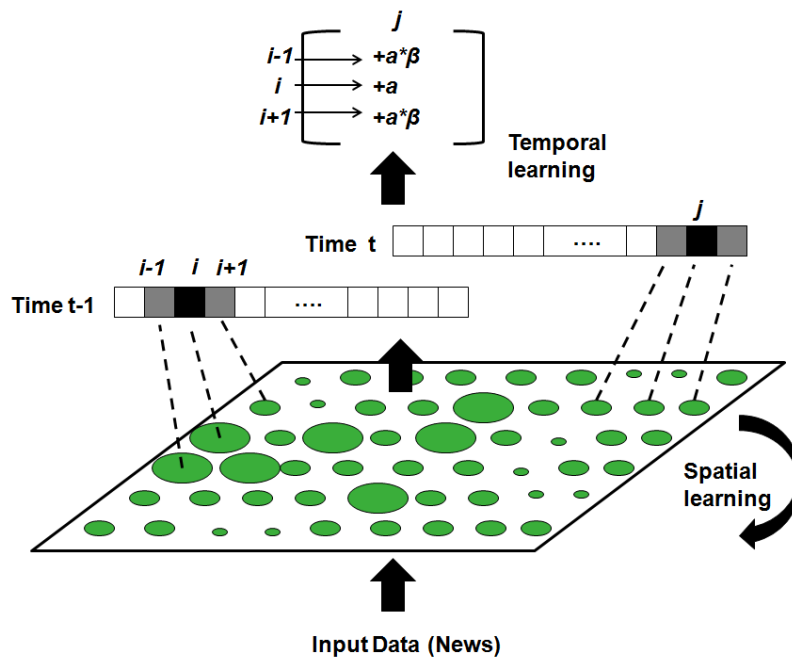


Figure 3.5: After spatial training, units activated are represented as vectors. They are the inputs of a time-dependent matrix T that learns over the time temporal relations among units

probability to follow each other through the time; these clusters are called temporal clusters and they contain groups of units that are likely to represent the evolution on time of a certain topic or event. Vector C of dimension m , where m is the total amount of SOM units, stores the number of news pooled for each unit of the SOM. C and matrix T are used by the algorithm described below to detect temporal clusters. In addition, figure 3.6 illustrates the clustering process.

1. Step 1: find from vector C the most frequent unit that is not yet part of a cluster. The most frequent unit is the one with the highest corresponding value in C .
2. Step 2: pick the unit that is most-connected to the most frequent unit. The model finds the most-connected unit by finding the highest value in the column of matrix T that corresponds to the current unit. Add the most connected unit to the cluster only if it is not part of a cluster.
3. Step 3: repeat step 2 for the most connected unit. Then recursively computes step 2 on its most connected unit, and so on, until no new unit is added.
4. Step 4: all these units are added to a new temporal cluster.
5. Step 5: go to step 1 and find the most frequent unit that is not yet part of a cluster.

Once temporal clusters are formed, they are interpreted as frequent news topics and events evolving over the time. Each unit captures a subset of news; therefore, the five most frequent words are taken from each unit as a description of the unit's topic, results are shown in table 3.4.

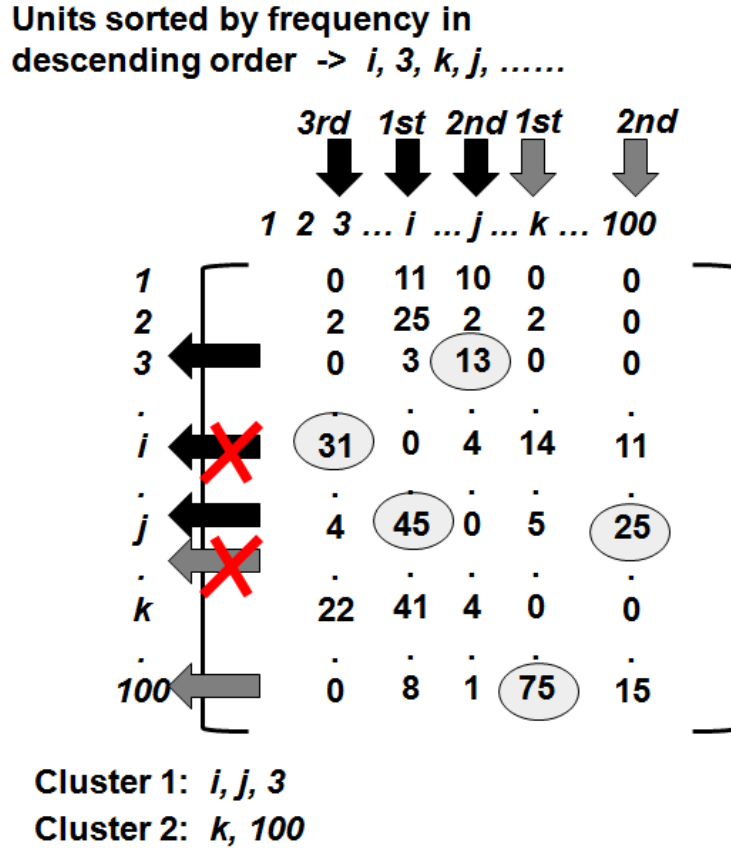


Figure 3.6: Temporal clustering example. 1st cluster start with column i (most frequent unit), taking most-connected unit j after 1st loop. During 2nd loop unit j takes unit 3. This last unit takes unit i at 3rd loop, but it is already in the cluster, therefore cluster 1 is closed

Temporal clusters represent a high-level perception of meaning, knowledge, logic, etc, over the time. They can be interpreted as an image or memory of frequent sequences of topics. Main topics, those that remain in the time, come to light, while volatile topics are not displayed. For instance, temporal cluster of units (1,6), (1,3) shows that topic $\{ 'radiat', 'japan', 'airlin', 'nuclear', 'flight' \}$ follows frequently to topic $\{ 'nuclear', 'plant', 'power', 'energi', 'reactor' \}$. A natural inference is that, during the disaster, there was a transition from issue *nuclear-radiation-flights* to issue *energy-power-reactor*, and that transition was frequently mentioned. The proposed SOM developed automatically the formation of a temporal *memory* uncovering the most important relations.

3.7 Conclusion

A SOM has been trained producing a spatial representation of the news set into a 2 dimensional map. This representation is a finite number of quantization points that group similar input vectors. Similar quantization points were grouped in clusters by applying *k-means* on weight vectors. Frequent words on map enabled to form a semantic structure. Time dimension was considered on SOM temporal learning, where groups of units were discovered having a high time-dependency. Temporal

clusters detection was possible by the utilization of a time-dependent matrix that stores the transitions from a SOM unit to another; this matrix is the model's perception of frequent events over the time. Our model was able to discover temporal dependencies, and results could be improved by what model is exposed to. Enough input must change and flow continuously through time for a suitable learning. The model can be scaled with diverse input data without complexity due its finite set of quantization points. The time-dependent matrix also can be modified assigning a memory to it, in a way that it does not remember only the last fired unit at time $(t - 1)$, but the last k units fired at times $(t - 1), (t - 2), (t - 3), \dots, (t - k)$, expanding its ability to detect unknown temporal relations. An improvement to consider is that neighbors of unit i fired at time $(t - 1)$ that follow unit j fired at time (t) are considered, but we do not evaluate the potential temporal relation among neighbors of i with neighbors of j . In addition, matrix T was updated by a scale down increment $(a.\beta)$ in elements $T(N_i, j)$, where N_i denotes neighbors of i ; we assigned empirical amounts to transition value a and parameter β . If a memory is provided to matrix T , a and β should vary on time. Temporal clustering algorithm also can be improved considering, for example, not only the most-connected unit, but the *2nd* most-connected, the *3rd* most-connected. Temporal clusters can be used to make predictions. The model computes for a new input y a spatial distribution on its m units, and a temporal distribution on its c temporal clusters. Our application emphasizes the spatial-temporal arrangement of the units and the segregation of the information into separate areas. Temporal clusters give an idea of how frequent events evolve over the time on an entirely unsupervised way.

Temporal (Units activated in temporal order)	Cluster frequently activated in temporal order)	5 most frequent words for each unit in temporal order
(5, 7)		'lodg', 'sweat', 'trial', 'particip', 'ray'
(6, 2)		'court', 'charg', 'attorney', 'case', 'judg'
(6, 1)		'polic', 'investig', 'depart', 'alleg', 'suspect'
(2, 9)		'tokyo', 'earthquak', 'power', 'japan', 'quak'
(1, 9)		'tsunami', 'japan', 'earthquak', 'warn', 'area'
(6, 7)		'moon', 'year', 'last', 'look', 'zune'
(6, 8)		'seavey', 'bike', 'week', 'appl', 'kate'
(5, 10)		'your', 'like', 'want', 'peopl', 'just'
(5, 1)		'investig', 'crash', 'driver', 'polic', 'william'
(10, 10)		'aristid', 'haiti', 'spend', 'return', 'elect'
(9, 1)		'bahrain', 'forc', 'govern', 'secur', 'saudi'
(1, 1)		'reactor', 'plant', 'radiat', 'fuel', 'nuclear'
(6, 3)		'yale', 'school', 'clark', 'polic', 'sentenc'
(6, 4)		'polic', 'accord', 'offic', 'baghdad', 'video'
(10, 2)		'zone', 'gadhafi', 'council', 'unit', 'resolut'
(2, 4)		'nuclear', 'plant', 'power', 'japan', 'disast'
(3, 8)		'earthquak', 'japan', 'school', 'might', 'peopl'
(10, 4)		'gadhafi', 'govern', 'libyan', 'presid', 'libya'
(9, 7)		'state', 'unit', 'hispan', 'medic', 'marijuana'
(10, 3)		'gadhafi', 'zone', 'forc', 'libyan', 'libya'
(7, 1)		'palestinian', 'isra', 'author', 'hama', 'gaza'
(4, 1)		'reactor', 'meltdown', 'nuclear', 'possibl', 'radiat'
(3, 1)		'reactor', 'plant', 'nuclear', 'explos', 'tuesday'
(10, 6)		'afghanistan', 'diplomat', 'petraeus', 'pakistan', 'court'
(1, 6)		'radiat', 'japan', 'airlin', 'nuclear', 'flight'
(1, 3)		'nuclear', 'plant', 'power', 'energi', 'reactor'
(3, 2)		'plant', 'reactor', 'nuclear', 'japan', 'agenc'
(8, 1)		'protest', 'forc', 'govern', 'demonstr', 'secur'
(9, 3)		'forc', 'bahrain', 'govern', 'gadhafi', 'intern'
(2, 2)		'power', 'nuclear', 'reactor', 'plant', 'daiichi'
(1, 7)		'japan', 'food', 'govern', 'japanes', 'spaniard'
(2, 1)		'reactor', 'plant', 'japanes', 'report', 'nuclear'
(7, 4)		'offici', 'defens', 'peopl', 'rain', 'civil'
(8, 6)		'releas', 'record', 'anonym', 'execut', 'donat'
(7, 6)		'right', 'inmat', 'maryland', 'bill', 'california'
(8, 9)		'head', 'earli', 'educ', 'start', 'a'childhood'
(8, 7)		'obama', 'conyer', 'presid', 'kenni', 'critic'
(7, 3)		'lucia', 'attack', 'accord', 'anti', 'baker'
(4, 10)		'citi', 'just', 'parad', 'your', 'peopl'

Table 3.4: Main temporal clusters detected by proposed model

Chapter 4

Features Detection by Linear Predictor Model

4.1 Introduction

The amount of data in our world has been exploding; data sets grow in exponential sizes in part because they are increasingly being gathered by ubiquitous information-sensing devices and social media. Large data sets become complex and difficult to process using on-hand database management tools or traditional data processing applications. Analyzing such data sets is one of the keys of leading companies and one of the most active research fields. Several issues need to be addressed to capture the full potential of big data; one of them is to find correlations from a vast amount of variables. The exploration of effective methods capable to detect the most important variables (features) from a large set of variables is focused in this chapter on a linear model; which aims to retain a subset of the most important variables based on their correlation with the desired output values. The main idea is to obtain a linear predictor that transforms the original data set showing its most important features. From a set with a large number of features, a smaller subset of variables is desired, considering the balance between the prediction error and the emerged subset.

Primary goal is not to create a prediction model, but to reduce the dimensionality of the input data, reduction that can be used for further purposes. Previously in chapter 2, experiments using PCA principal component analysis has been explained to compress a set of documents; the obtained orthogonal transformation generated a new set of values of linearly uncorrelated variables. In that case, each principal component is a linear combination of the original variables; the real meaning of the features disappeared, eliminating the chances to use the transformation results in applications that require the *semantic* of the data. On the other hand, a linear prediction model is a powerful alternative for compression with no loss of the essence and context of the data, essential characteristics for further analysis.

As an application on real data, we present an adaptation of a linear prediction model to discover trend topics on a news stream. Natural disasters text sources are used and main features are extracted to analyze and find out patterns to make better decisions during difficult times. Our algorithm shows the potential application of linear models on torrents of data to help governments and organizations to understand what information truly count. A linear model includes the capability of shrinkage on input vectors; showing after some iterations variables with strongest

characteristics, while those with negligible characteristics are discarded. Due to the dynamic and uninterrupted characteristics of the input, the output targets to expose the evolution of the most significant variables over the time.

4.2 Linear Predictor/Regression Models

Since many years ago, linear predictor models (or regression models) are still being effective and actively used in a diverse area of applications, such as data forecasting, speech recognition, model-based spectral analysis, signal restoration, and others. In machine learning field, a linear predictor is a linear function of a set of coefficients and independent variables, whose value is used to predict the outcome of a dependent variable. Functions of this type are common in linear regression, where the coefficients are known as regression coefficients. They also appear in various types of linear classifiers, such as perceptrons, support vector machines, and linear discriminant analysis.

A linear predictor model assumes its function as linear in the inputs X_1, X_2, \dots, X_n . In a general case, it is desired to predict an output $Y = f(X)$, so the model is defined as follows:

$$f(X) = \theta_0 + \sum_{j=1}^d X_j \theta_j \quad (4.1)$$

Where $\theta_0, \theta_1, \dots, \theta_d$ are the unknown coefficients, and d is the dimension of input vectors X . Typically there is a set of training data $(X_1, y_1), \dots, (X_n, y_n)$ from where parameters θ are estimated. The most popular estimation method is the least squares, in which the coefficients θ are defined in order to minimize the quadratic cost between the output training data and the model predictions.

$$J(\theta) = \sum_{i=1}^n (y_i - f(x_i))^2 \quad (4.2)$$

$$J(\theta) = \sum_{i=1}^n \left(y_i - \theta_0 - \sum_{j=1}^d x_{ij} \theta_j \right)^2 \quad (4.3)$$

Each $x_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$ is a vector of feature measurements for the i^{th} case. Let's denote by X the $n \times (d + 1)$ matrix, where each row is an input vector, the first column is filled with ones, and similarly let's denote as y to the n -vector of outputs in the training set. The quadratic cost can be written in a matrix notation, such as:

$$J(\theta) = (y - X\theta)^T (y - X\theta) \quad (4.4)$$

Optimization of 4.4 by differentiating with respect to θ results:

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta} &= -2X^T y + 2X^T X \theta \\ 0 &= -2X^T y + 2X^T X \theta \\ \hat{\theta} &= (X^T X)^{-1} X^T y \end{aligned} \quad (4.5)$$

Model's predictions are calculated as follows:

$$\hat{y} = X \hat{\theta} = X (X^T X)^{-1} X^T y \quad (4.6)$$

One of the most famous results in statistics asserts that the least squares estimates of the parameters have the smallest variance among all linear unbiased estimates [Hastie et al., 2005]. In case of text data during emergencies, we have high-dimensional inputs coming continuously, therefore, it is needed to estimate a high-dimensional vector of coefficients θ which is almost impractical for a streaming data processing system; an automatic and unsupervised selection of the most influential variables using shrinkage methods is useful in that case.

4.2.1 Shrinkage methods

Ridge regression shrinks the θ 's by imposing a penalty on their size. The ridge coefficients minimize a penalized quadratic cost between the output training data and the model predictions. Its solution is given by:

$$\hat{\theta}^{ridge} = \arg \min_{\theta} \left(\sum_{i=1}^n \left(y_i - \theta_0 - \sum_{j=1}^d x_{ij} \theta_j \right)^2 + \delta^2 \sum_{j=1}^d \theta_j^2 \right) \quad (4.7)$$

Where $\delta^2 \geq 0$ is a complexity parameter that controls the amount of shrinkage, for larger values of δ the amount of shrinkage increases. Ridge method compresses θ 's toward zero and each other. Writing equation 4.7 in matrix form, we have:

$$J(\theta) = (y - X\theta)^T (y - X\theta) + \delta^2 \theta^T \theta \quad (4.8)$$

And the solution of the above regularized quadratic cost function is:

$$\hat{\theta} = (X^T X + \delta^2 I_d)^{-1} X^T y \quad (4.9)$$

Where I_d is the $d \times d$ identity matrix. The ridge coefficients minimize a penalized quadratic cost between the output training data and the model predictions. Equation 4.9 shows how ridge adds the penalty down the diagonal, introducing bias but reducing the variance of the estimate. This penalty was incorporated due issues with results for equation 4.5, if matrix X is not full rank $X^T X$ is not invertible and there is no unique solution; this problem does not occur with ridge regression, this was the main motivation for ridge method when it was introduced.

Another biased regression technique is *Least Absolute Selection and Shrinkage Operator*, also known as *lasso*, it is like ridge method but with big differences in the penalty term. The estimation by lasso is as follows:

$$\hat{\theta}^{lasso} = \arg \min_{\theta} \left(\sum_{i=1}^n \left(y_i - \theta_0 - \sum_{j=1}^d x_{ij} \theta_j \right)^2 + \delta^2 \sum_{j=1}^d |\theta_j| \right) \quad (4.10)$$

In this case, the ridge penalty in equation 4.7 is replaced by the lasso penalty $\delta^2 \sum_{j=1}^d |\theta_j|$ also known as L_1 norm. Due to the new penalty form, the solutions for estimation of θ 's are nonlinear in y . Lasso in matrix form is as follows:

$$J(\theta) = (y - X\theta)^T (y - X\theta) + \delta^2 \sum_{j=1}^d |\theta_j| \quad (4.11)$$

A ridge solution can be hard to interpret because it is not sparse, which means no θ 's are set exactly to 0. Lasso penalty introduces a very important change; some of the coefficients may be shrunk exactly to zero [Tibshirani, 1996]. If complexity parameter δ is sufficiently large, some θ 's are driven to zero, leading to a sparse model.

This is the core of the proposed algorithm explained in next section; lasso applied consecutively on a text data set, takes care of the variables selection for us removing irrelevant features. Computing lasso solution is a quadratic programming problem; it is needed to optimize equation 4.11 where several variables are subjected to linear constraints. For example, the derivative containing an absolute value. Next section describes our proposed solution for this problem.

4.3 Computing Lasso Optimization

The *Lasso* has been studied and used in many applications over the last years. Least angle regression algorithm [Efron et al., 2004] and the solution path of the generalized lasso [Tibshirani, 2011] are two relevant algorithms to solve lasso quadratic programming problems. Our proposed algorithm is a close form of lasso solution, designed to provide scalability from batch data mode to streaming data. Expression 4.11 can be expressed as:

$$J(\theta) = \sum_{i=1}^n \left(y_i - \theta_0 - \sum_{j=1}^d x_{ij}\theta_j \right)^2 + \delta^2 \sum_{j=1}^d |\theta_j| \quad (4.12)$$

Replacing $\sum_{j=1}^d x_{ij}\theta_j$ by its equivalent in matrix notation $X_i^T\theta$ it is obtained:

$$J(\theta) = \sum_{i=1}^n (y_i - \theta_0 - X_i^T\theta)^2 + \delta^2 \sum_{j=1}^d |\theta_j| \quad (4.13)$$

Differentiating the 1st term of the above sum with respect to one generic feature coefficient θ_j we have:

$$\frac{\partial J(\theta)^{1^{st}term}}{\partial \theta_j} = \sum_{i=1}^n 2 \left(y_i - \theta_0 - X_{i-j}^T\theta_{-j} - X_{i_j}\theta_j \right) (-X_{i_j}) \quad (4.14)$$

Where $X_{i-j}^T\theta_{-j}$ is same as $X_i^T\theta$ but excluding the feature j and its coefficient, and $X_{i_j}\theta_j$ are the variable and coefficient for feature j .

$$\frac{\partial J(\theta)^{1^{st}term}}{\partial \theta_j} = \sum_{i=1}^n 2X_{i_j}^2\theta_j - \sum_{i=1}^n 2 \left(y_i - \theta_0 - X_{i-j}^T\theta_{-j} \right) (-X_{i_j}) \quad (4.15)$$

Denoting $\sum_{i=1}^n 2X_{i_j}^2 = z_1$ and $\sum_{i=1}^n 2 \left(y_i - \theta_0 - X_{i-j}^T\theta_{-j} \right) (-X_{i_j}) = z_2$, equation 4.15 is expressed as:

$$\frac{\partial J(\theta)^{1^{st}term}}{\partial \theta_j} = z_1\theta_j - z_2 \quad (4.16)$$

The 2nd term $\delta^2 \sum_{j=1}^d |\theta_j|$ is differentiated in the following way:

$$\frac{\partial J(\theta)^{2^{nd}term}}{\partial \theta_j} = \delta^2 \frac{\partial |\theta_j|}{\partial \theta_j} = \begin{cases} -\delta^2, & \text{if } \theta_j < 0 \\ (-\delta^2, \delta^2), & \text{if } \theta_j = 0 \\ \delta^2, & \text{if } \theta_j > 0 \end{cases} \quad (4.17)$$

Written together equations 4.16 and 4.17 and equating to zero, we have:

$$0 = z_1 \theta_j - z_2 + \delta^2 \frac{\partial |\theta_j|}{\partial \theta_j} \quad (4.18)$$

Therefore estimation for the feature coefficient $\hat{\theta}_j$ will be:

$$\hat{\theta}_j = \frac{z_2 - \delta^2 \frac{\partial |\theta_j|}{\partial \theta_j}}{z_1} = \begin{cases} \frac{z_2 + \delta^2}{z_1}, & \text{if } \theta_j < 0 \\ \left(\frac{z_2 + \delta^2}{z_1}, \frac{z_2 - \delta^2}{z_1} \right), & \text{if } \theta_j = 0 \\ \frac{z_2 - \delta^2}{z_1}, & \text{if } \theta_j > 0 \end{cases} \quad (4.19)$$

Based on equation 4.15 it is known that z_1 is always positive, therefore $\theta_j < 0$ is same as $z_2 < -\delta^2$, and equivalently $\theta_j > 0$ same as $z_2 > \delta^2$.

A proposed algorithm that implements expression 4.19 iteratively until convergence is described below:

1. Initialize coefficients by ridge method (equation 4.9)

$$\hat{\theta} = (X^T X + \delta^2 I_d)^{-1} X^T y$$

2. Calculate z_1 (equation 4.15)
3. Initialize a value for δ (amount of shrinkage)
4. Repeat until coefficients $\hat{\theta}$'s get stable:
 - (a) for each feature j perform:
 - i. Calculate z_2 (equation 4.15)
 - ii. In case $z_2 < -\delta^2$ update $\hat{\theta}_j = \frac{z_2 + \delta^2}{z_1}$
 - iii. In case $z_2 > \delta^2$ update $\hat{\theta}_j = \frac{z_2 - \delta^2}{z_1}$
 - iv. Otherwise, update $\hat{\theta}_j = 0$
 - (b) end

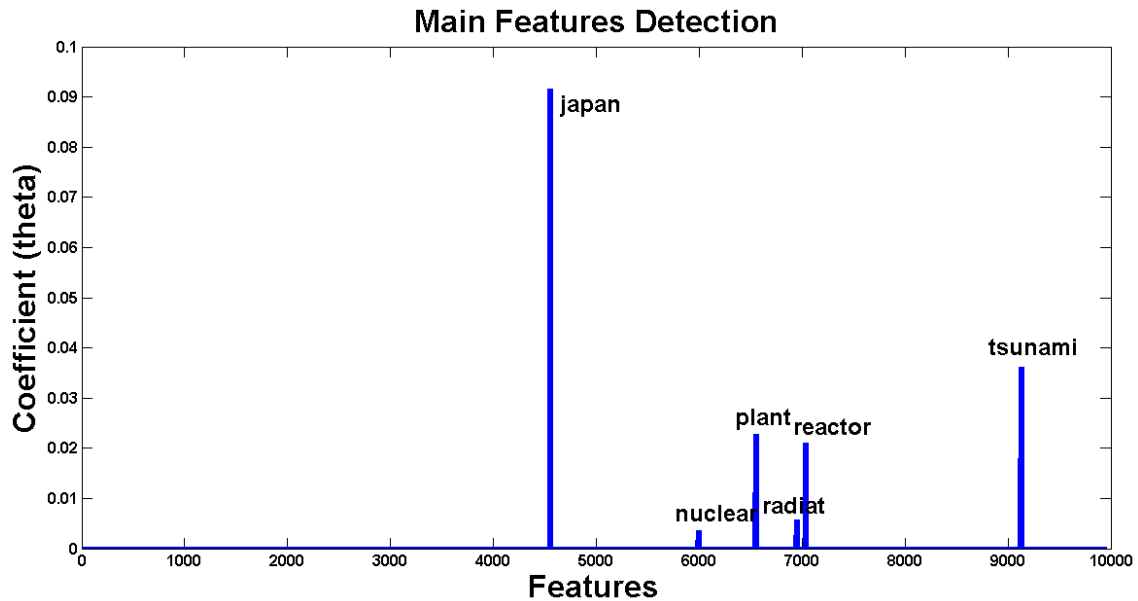
5. Change value of δ and iterate from step 4

6. end

Above algorithm allows for identification of the relevant variables for a set of inputs; some modifications are implemented in next section to adapt its operation to a real problem like streaming data. Algorithm described has been run several times for different values of the complexity parameter that controls the amount of shrinkage. Simulation results are shown in table 4.1, compression ratio is defined as the ratio between the amount of original variables (9962 in our experiment) and the final subset of strongest variables. For example, for $\delta = 0.2$, the error prediction

Shrinkage parameter δ	Prediction Squared Error (equation 4.4)	Compression ratio (initial variables / final variables)
0.2	0.0068	6.17
0.5	0.1741	13.77
1	1.7651	25.61
4	24.3671	262.16
10	38.6543	1660.33

Table 4.1: Shrinkage simulation results

Figure 4.1: Strongest variables for shrinkage parameter $\delta = 10$

is very slight, but the compression is low. Although a compression ratio of 6.17 implies a 6 times reduction, from 9962 to 1613 features, for some systems and applications to manage 1600 variables could still be complex and costly in resources and time. In the opposite case, for $\delta = 10$, the error prediction is high and the compression results are considerable. For this scenario, figure 4.1 shows the final subset of features that evidence strong compression results. The issue in this case is the prediction error, some systems and applications cannot afford higher degradation levels of the accuracy. The final decision involves balancing the error against the compression considering future applications of the data. Adding new testing data and re-computing values presented in table 4.1, help to verify the generalization of the final subset, and to support a closing decision.

At this stage, it was obtained a manageable algorithm to select automatically the most important features from a high dimensional data set. The algorithm is straightforward, and it is a derivation of the shared application between a linear prediction model besides *ridge* and *lasso* shrink methods. Different compression ratios can be obtained by the modification of a shrink parameter, high linear prediction accuracy is correlated with low compression ratios, and on the contrary, strong compression harmonizes with a degradation of the prediction accuracy. Our proposed solution can be used as an alternative for systems that are struggling with issues related to high dimensional input data. It produces an input model that is interpretable and

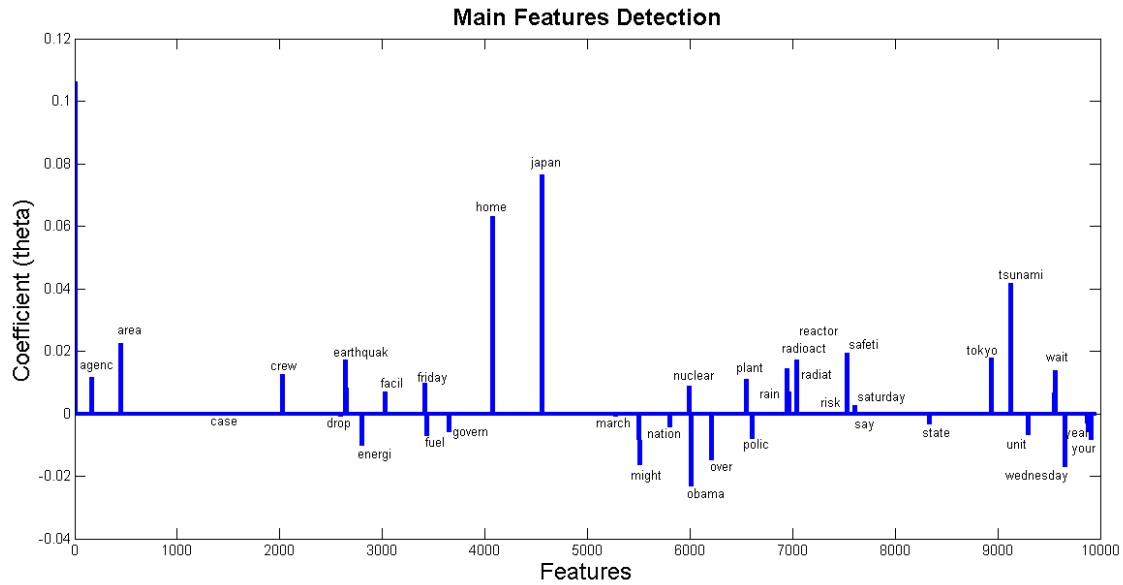


Figure 4.2: Strongest variables for shrinkage parameter $\delta = 4$

has possibly lower prediction error than the full model. An application is shown in next section on a stream news data set.

4.4 Lasso for Uncovering Trending Topics on Streaming Data

In the past few years the tracking of breaking topics has become an active research area in computer science. Twitter introduced *Twitter Trends* in 2008, defined them as the topics that are being talked about more right now than they were previously. The trends show the most breaking news of a set of breaking news generated from millions of tweets around the world. Algorithms to process such data stream must operate continuously, processing each new input in real time; using computational resources' memory and storage space up to a reasonable threshold. That is one of the main characteristic that differentiates data stream algorithms from batch algorithms. Many algorithms are based on stream elements' frequency; showing as output the most frequent elements. A big issue related to counting is that it is needed a ranking of frequent elements extracted from a maintained full list of elements and their counters. Sampling techniques also are not enough efficient, they represent an approximate solution of what is a trending topic; the issue is related to whether samples include or not real trending topics. Twitter's algorithm for determining trends is a private algorithm, but some information provided by the company indicates that topics break into trends when the volume associated to that topic at a given moment dramatically increases. In this case information's velocity increases quickly enough, compared to a baseline level and historical appearances/velocities associated. We propose an algorithm to find the most important topics on a news stream, discovering trends and showing their evolution over the time. Our algorithm is based on a linear prediction model with shrinkage operators that retain the most important variables, to our knowledge this is the first time that linear prediction models are used for uncovering trending topics. With a large number of

fixed features, around 10,000 in our experiments, we obtain automatically smaller subsets of the strongest variables over the time. In this paper we will refer to variables/features as potential trends. Important to clarify that our main goal is not to create a prediction model, but to uncover the strongest topics of a news stream. The algorithm pretends to be a powerful alternative for processing data stream, where answers can be derived from a sequence of main features detected over the time, that approximate the output for the stream as a whole.

Our algorithm works based on two important assumptions:

1. Lasso regression is used for regression-prediction in linear models, therefore it is needed a training set with a defined output $Y = f(X)$. Every time an input comes we “classify” it “on the fly” assigning a dummy category at random; the category is selected from the set $\{-1, 1\}$. We assume a continue stream of data as a fictitious training data.
2. We assume a subset of sequential inputs vectors as a segment; its size is an important parameter in our trend topics detection system, due to it has close relation with the computing power and memory availability. During simulations explained later we use segments of size = 50; every time an input vector comes the system computes its results; when the vector number 50 comes the system restarts its parameters.

The algorithm described below detects main variables from a set of fixed variables that can be thought as a model’s constrain, but if we guide the system’s application, for example, to discover trends topics during emergencies, it is viable to get satisfying insides from our proposed solution. The fixed variables set used throughout simulations was a predefined dictionary of 10,000 words related to natural disasters. We want to emphasize, lasso is incorporated as part of the system not only to uncover main variables, but to comprise the *high-dimensional* problem of processing text. Reasonable to suppose that most of the coefficients of the words are exactly equal to 0. Aside from segment’s size, the shrinkage can be controlled; more or less variables can be shown depending on the problem, by varying complexity parameter δ . Proposed algorithm is shown below:

1. Initialize parameters: select value of δ and segment’s size.
2. Repeat until needed
 - (a) Receive input vector from data stream
 - (b) Assign, at random, a dummy category from set $\{-1, 1\}$ to input vector.
 - (c) If size of segment is reached, initialize z_1 and z_2 to zero (ready for next segment).
 - (d) Calculate z_1 (equation 4.15) (this calculation is accumulative over iterations, up to segment’s size)
 - (e) for each feature j perform:
 - i. Calculate z_2 (equation 4.15) (This calculation is performed for current segment; it is accumulative over iterations, up to segment’s size)
 - ii. In case $z_2 < -\delta^2$ update $\hat{\theta}_j = \frac{z_2 + \delta^2}{z_1}$

iii. In case $z_2 > \delta^2$ update $\hat{\theta}_j = \frac{z_2 - \delta^2}{z_1}$

iv. Otherwise, update $\hat{\theta}_j = 0$

(f) end

3. Iterate from step 2 until needed.

4. end

Above algorithm does not include considerations about velocity of the trend topics, but it shows how coefficients θ for each important feature evolve over the time.

Experiments were performed in a conventional laptop, with 4 GB of RAM and CPU @ 2.60 GHz. Segment's size was set to 50 inputs and amount of shrinkage $\delta = 7$. We reproduce a news stream environment using a huge collection of news linked to March 2011 earthquake / tsunami in Japan, sorted by date and time. News are entered into the system sequentially; for every new input, the system matches it to a set of 10,000 fixed features predefined. Thus, every news is transformed in a high-dimensional vector where each element is a number equal to the frequency of each term in the dictionary. Figure 4.3 shows simulation results up to 25 iterations; main features are immediately detected. Firstly, we can observe main topics such as "earthquake", "Japan", among others are exposed as significant in figure 4.3 major subplot. Horizontal axis enumerates the established 10,000 features, while vertical axis displays value of features coefficient θ 's. Lasso regression penalizes θ 's for worthless topics, shrinking them gradually to zero, and at the same time rewards to trend topics coefficients stabilizing them. In our system, coefficients θ 's can be understood as a weighting value for each feature; in that sense, previous figure exposes "earthquake" as the most important topic, followed by "tsunami", "Japan" and others. Individual subplots show values of θ 's for trend topics over the time; we believe that θ 's model behavior of real trends topics, at the beginning when main issues are discovered they experiment a peak whose value gets stable after some time, that value may reach zero if the topic is not relevant anymore. For example, "tsunami" makes a peak at time 4; its weight $\theta_{tsunami}$ is strong because it just was discovered, through some interactions it becomes stable; although with a very small value it still pertinent.

Along our simulations, trend topics displayed peaks following a sequence comparable with how real facts occurred. At figure 4.3 our system detected the peaks' sequence: "tsunami", "earthquake", "plant", "Japan", "Tokyo", "area", "quake", "warn". Sequences analysis can be useful for future prevention plans and readiness. Figure 4.4 shows algorithm results up to iteration 420; in this case previously trends were replaced by weighty current issues, such as "nuclear", "plant", "people", "government". As mentioned previously, some issues might require a deeper analysis, in that case the amount of shrinkage can be modified to lower values of δ to display more detailed trending topics. Mainly, trending topics are found by counting inputs with repeated hashtags (#) tagged by users, as mentioned previously, Twitter's trend status is defined by a combination of velocity and volume of tweets containing the hashtag. Our proposal does not use hashtags; neither performs a systematic counting of terms. Experiments exposed at figures 4.3 and 4.4 are applied to a large set of sequential news including a major amount of terms; this characteristic demonstrates the scalability of our algorithm to different sizes of inputs vectors. Figure 4.5 shows histograms of words after 25 and 420 iterations. This plot can be

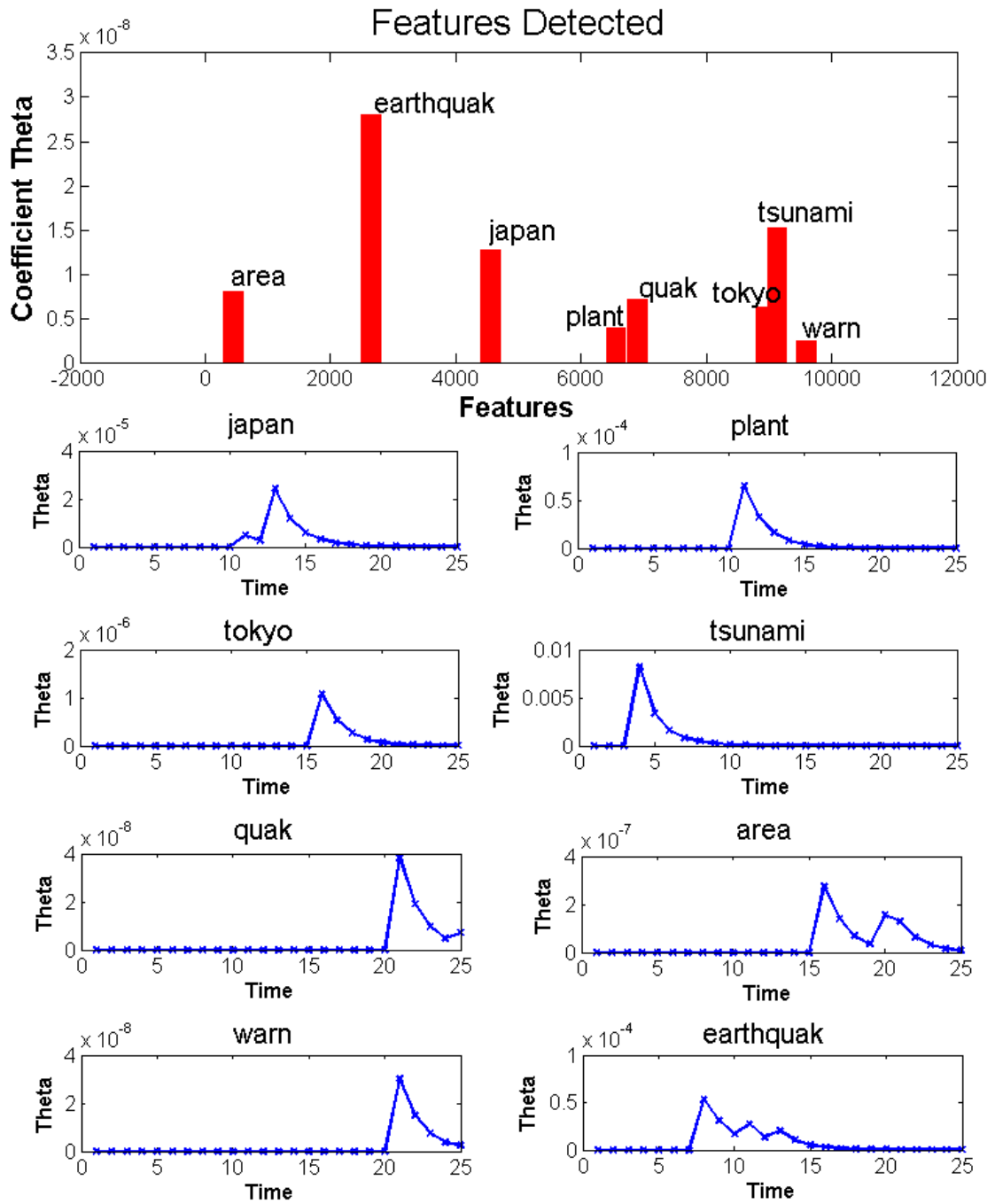


Figure 4.3: Proposed algorithm results after 25 iterations

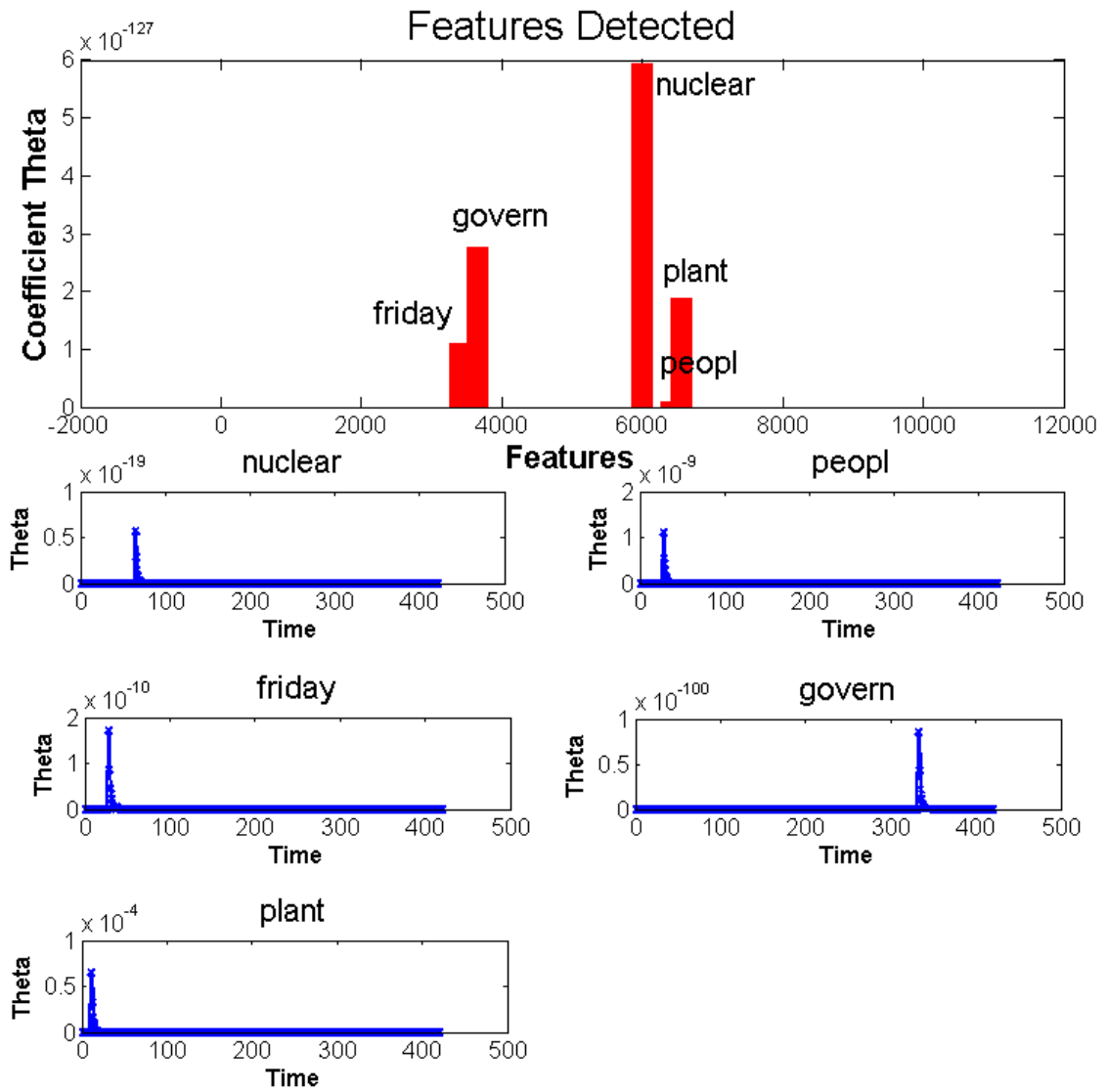


Figure 4.4: Proposed algorithm results after 420 iterations

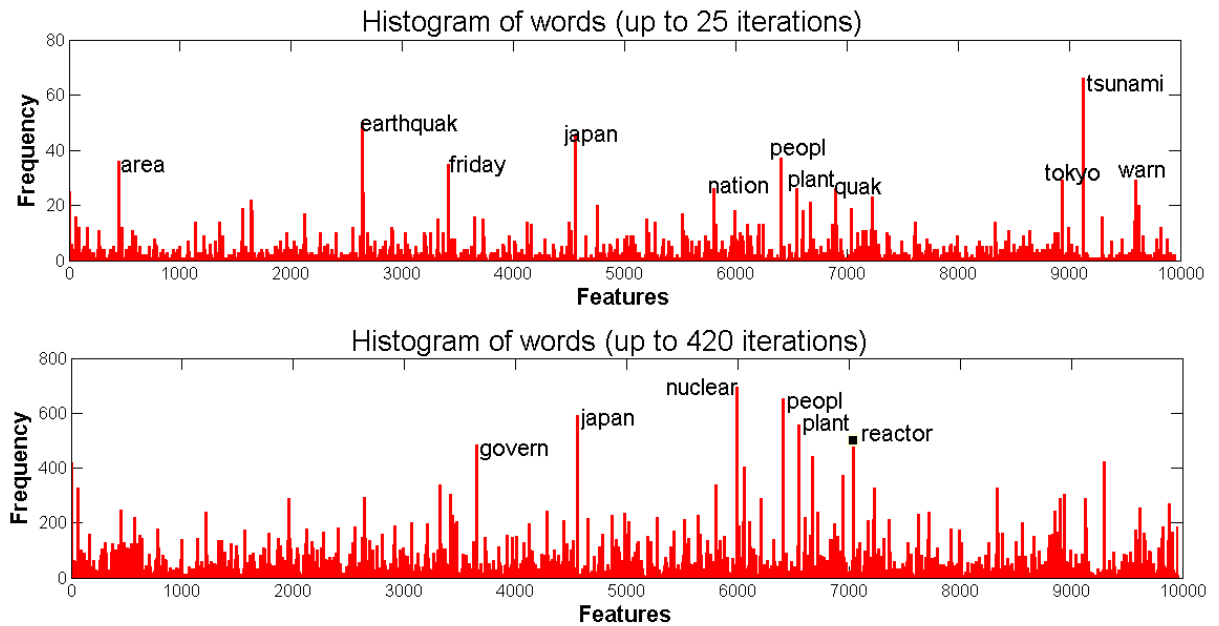


Figure 4.5: Histograms of words

compared with figures 4.3 and 4.4 respectively; counting of terms produces similar results, for example after 25 iterations our system delivers trending topics "area", "earthquake", "Japan", "plant", "quake", "Tokyo", "tsunami", "warn" which are comparable with most frequent words for 1st histogram "area", "earthquake", "Friday", "Japan", "nation", "people", "plant", "quake", "Tokyo", "tsunami", "warn"; equivalent conclusions can be obtained observing the results after 420 iterations. However, our solution incorporates a novel component, the weighting values θ 's for each topic. This is an important issue of Twitter; its topics list constantly changes and does not apply a weight or relevancy to a particular topic. Our algorithm not only shows a significance value for discovered trends, but also how they behave over the time.

4.5 Conclusion

A system to uncover trend topics from a data stream was proposed; as far as we know it is the 1st time a linear regression / prediction model with shrinkage operators is used for detecting main topics on data stream. Our algorithm makes intense use of lasso model and adapts it to process streaming data; in a way that a trend is understood and modeled as a coefficient θ that serve as topic's weight. Coefficients θ 's and their evolution over time represents the "lifetime" of main topics detected. Although the amount of features to detect is bounded to a dictionary of 10,000 variables, our algorithm does not need special characters, like "#", to recognize a trend topic. The algorithm is not based on probability neither counting of variables occurrences. It is a pure application of a linear prediction model.

Chapter 5

Features Detection by Random Forest

5.1 Introduction

The adoption of random forest model to detect and extract from a large set of text sources main features and related rules of a sudden critical event is a novel application analyzed and proposed in this chapter. After a number of several applications, mainly for computer vision [Nowozin et al., 2011] [Bosch et al., 2007], its performance has been proved to achieve excellent results. Random forest's generalization accuracy on unseen data is verified at [Ho, 1995]; and its full randomization is demonstrated to be computationally efficient at [Geurts et al., 2006]. A unified random decision forest model has been presented at [Criminisi et al., 2012] for classification, regression, density estimation, manifold learning, semi-supervised learning, and active learning. An unsupervised random forest model, like a density estimation model, can be used to detect main features and split the data set automatically. A set of predominant features in text sources are words patterns; in case of natural disasters text sources those features could represent an approach to behavioral patterns and concerns of the affected user's community, town, city, country. The problem of main features extraction from a large set of text sources is a large data mining problem, where a vast amount of text require to be processed immediately. Random forest has the advantage to take a huge amount of data, split it in small sub-sets and assign them to individual *decision trees*. Trees process the information in parallel, and after a relatively short time of learning, outputs are combined to provide a final and unified result.

5.2 Decision Trees and Random Forest

Decision trees have been studied for long time, many algorithms have been created to get a trained optimal decision tree, mainly for regression and classification [Breiman et al., 1984]. A decision tree is a classifier expressed as a recursive partition of the input space. The decision tree consists of a node called "root", "internal" or "test" nodes, and "leaves" or "terminal" nodes. Each test node split the input space into two or more sub-spaces according to a test function of the input attributes values. On the contrary, random forests are an active research field in the present years, used

for classification, regression, pattern recognition and density estimation. One of its famous successful implementations is the Microsoft Kinect for Xbox 360. A random decision forest is an ensemble of randomly trained decision trees, where all tree outputs are combined by averaging their class posteriors. This combination is the argumentation of random forest's popularity; on classification tasks they produce much higher accuracy on previously unobserved data, improving substantially its generalization. Let's denote a generic data point by a vector $v = (x_1, x_2, x_3, \dots, x_d) \in R^d$, where each x_i is a measurement of the i^{th} feature and d is the dimensionality. In most of the problems, the dimensionality of input vectors is high; however, in random forest it is not necessary to compute all d dimensions of v , they are randomly sampled from the set of all possible features by a function $\phi(v) : R^d \rightarrow R^{d'}$ with $d' \ll d$.

Random forest training consists of optimizing the parameters of the test functions for each internal node, in order to maximize an acquired *energy function*. Test functions split the data in different sets where each set has an associated *Shannon* entropy defined by:

$$H(S) = - \sum_{c \in C} p(c) \log(p(c)) \quad (5.1)$$

Where S is the training data set, and c is a category from the set of all defined categories C in case of labeled data. Having the entropy, the information gain is solved mathematically as:

$$I = H(S) - \sum_{i \in \{Left, Right\}} \frac{|S^i|}{|S|} H(S^i) \quad (5.2)$$

S is better class separated when the information gain is higher; the information gain is the *energy function* used to optimize the test function that produces the highest confidence. Each internal node j is associated with a binary test function denoted as:

$$h(v, \theta_j) \in \{0, 1\} \quad (5.3)$$

Where 0 and 1 indicate respectively "false" and "true", input data is evaluated and sent to left or right according to the output of function above. In a general model $\theta_j = (\phi, \beta, \tau)$, where ϕ is a function that selects some features out of the entire set of features, β defines a geometric primitive used to separate the data, and τ are the thresholds used for the inequalities in test functions. For most of the cases, equation 5.3 is reduced to pick the best variable or split point among the d' features, which maximizes the information gain. Training a tree is achieved by optimizing each internal node test function parameters by:

$$\theta_j^* = \arg \max I_j \quad (5.4)$$

Randomness is essential in this method (figure 5.1), individual component trees provides a de-correlated prediction; this characteristic helps to improve robustness with respect to noisy. Randomness is included during training by:

1. bagging: random selection of the training data.
2. randomized node optimization: from the entire set of all possible features we chose randomly a subset of θ .

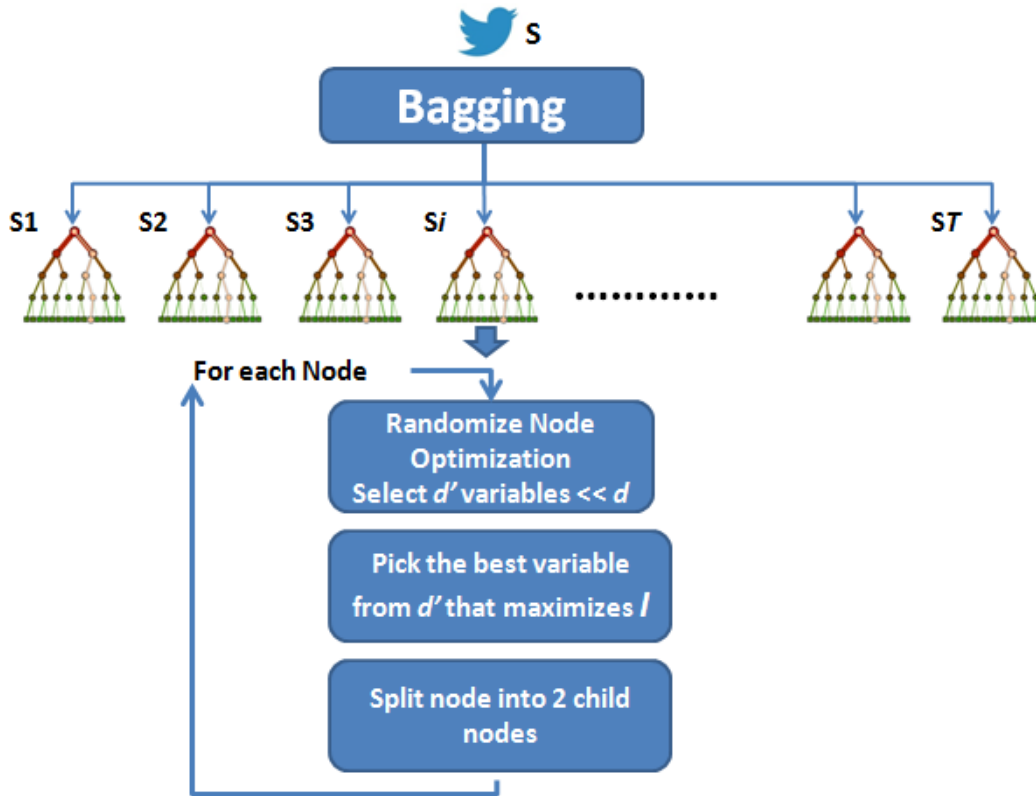


Figure 5.1: General diagram of our implementation on tweets. Randomness is added into *bagging* and *node optimization* stages

During training, information that is useful for prediction is learned for all nodes. In classification, each leaf stores the empirical distribution over the classes associated to the subset of training data that has reached that leaf. In a forest with T trees where $t \in \{1, 2, \dots, T\}$, trees combine their outputs into a single forest prediction by an averaging operation as follows:

$$p(c|v) = \frac{1}{T} \sum_{t=1}^T p_t(c|v) \quad (5.5)$$

Random forest most common usage is “supervised classification” tasks, next section explains our implementation on *tweets* using unlabeled data with unsupervised learning, also known as clustering forest or density forest.

5.3 Mining on Tweets

The real world critical events are reflected on Twitter by an exponential growth of posts or comments by the users with a direct relation to that particular event. We recorded tweets during a minor earthquake in California on 8th May 2014; people felt the earthquake and immediately reported brief text updates on Twitter. This situation is well recognized by analyzing figure 5.2; it shows how at 06:41 UTC time the number of tweets per minute increases rapidly, returning after 25 minutes approximately to an average number. Our paper focuses on that 25 minutes time

interval to extract information, where a huge amount of tweets were created as an instantaneous reaction to a potential emergency.

Tweets are free text micro blogging posts of no more than 140 characters, used by millions of people around the world; with one important characteristic, its real-time nature. Although their length per post is limited, the variety of words that can be used is high. Groups of variables often move together; our previous work at [Gutierrez et al.], [Gutierrez et al., 2012] and [Gutierrez et al., 2013] exposed that, on text, more than one variable is measuring the same driving principle governing the system's behavior.

5.3.1 Numerical Representation of Tweets

A dictionary of words is created from the IPCC special report on *Managing the Risks of Extreme Events and Disasters to Advance Climate Change Adaptation* [Field, 2012] and the *Terminology on Disaster Risk Reduction* of the United Nations [ISDR, 2009]. The dictionary is a compound of the most influential terms required to evaluate an emergency situation sorted by the frequency found at the mentioned documents, which gives us a relative weight for each term. In addition, we created a second dictionary composed by most frequent words found on tweets that contain terms such as *tsunami*, *earthquake*, *quake*, *flood*, *cyclone*, *avalanche*, *blizzard*, *landslide*, *typhoon*, etc.

Having a dictionary, each tweet is processed online as follows:

1. Special characters, numbers, symbols, and meaningless words such as conjunctions, prepositions and adverbs were removed.
2. Porter stemming process [Porter, 1980] is applied to reduce inflected or derived words to their stem, base or root form. Both dictionaries are composed by roots terms.
3. Random forest selects from the dictionaries words at random and transforms a tweet in a reduced dimension vector where each element is a number equal to the frequency found for each term. We chose to have 2 representations of each tweet to compare results using different dictionaries.

The result is an unlabeled collection of numerical vectors used to train individual decision trees independently and in parallel.

5.3.2 Random Forest Training

A random forest with $T = 100$ trees and deep $D = 4$ is trained by using algorithm below:

1. Repeat until complete T trees:
 - (a) Query on Twitter using keywords *natural disaster*, *tsunami*, *earthquake*, *quake*, *flood*, etc.
 - (b) Assign query's results to set S with size N and create $i : 1$ to 20 trees by the following steps:

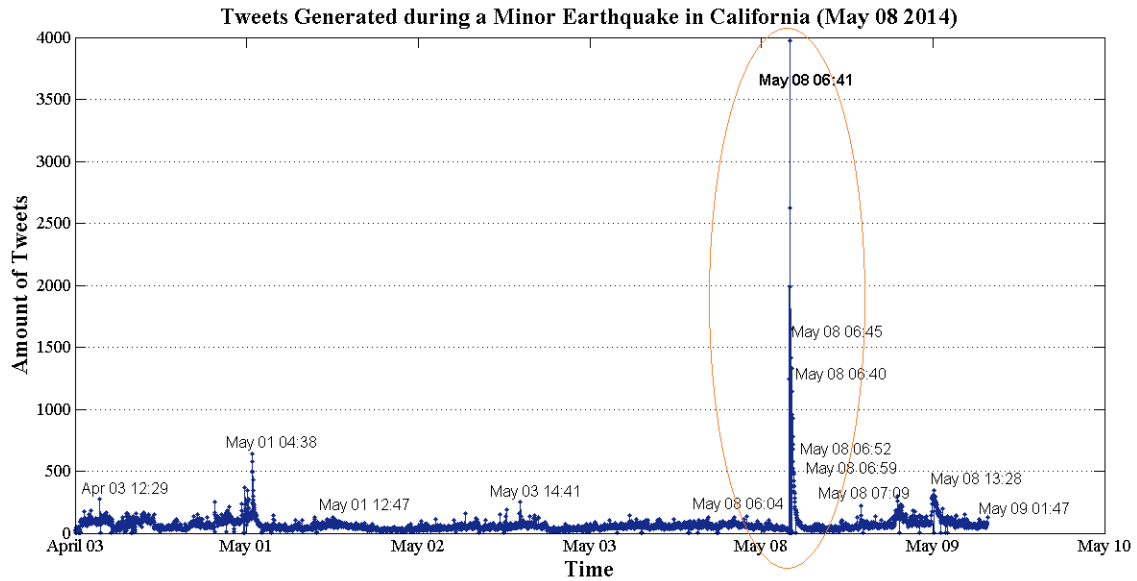


Figure 5.2: Users’ reaction on Twitter for a M3.3 Earthquake at Greater Los Angeles Area, California, May 8th 2014

- i. Perform *bagging*, by drawing uniformly at random a sample S_i of size $M < N$ from S .
 - ii. Grow a decision tree t_i by recursively repeating steps below for each internal node:
 - A. Perform randomized node optimization by selecting d' variables at random from d .
 - B. Transform sampled tweets into numerical d' -dimensional vectors.
 - C. Pick the best variable among the d' variables. The best split feature maximizes the information gain for the node under training.
 - D. Split the node into two child nodes.
 - E. If the maximum deep is reached (parameter D), close tree t_i and create next tree.
2. Group split variables learned at each node for each tree and build a histogram of main variables.

Although the proposed algorithm is shown as “sequential”, it is a completely distributed algorithm. It is possible to assign a query’s result to some trees in a machine, another subset of data to trees in another machine, and so on. In today’s world, when data sets are too large, it is unwise to load all data into memory. Even more, any emergency situation produces a flood of data to be processed. Our proposed algorithm takes several queries and split their results in small subsets. This model can be applied to process streams in real-time; an online random forest for classification has been proposed at [Saffari et al., 2009] that takes ideas from [Geurts et al., 2006] and implements an on-line decision tree growing procedure. Proposed algorithm doesn’t implement an on-line growing strategy, it manages the streaming data by querying Twitter at different intervals of time.

Breiman’s approach [Breiman, 2001] is used to introduce a way of injecting randomness in the forest by randomly sampling the training data. As well, the technique is known as *bagging*. Trees takes at random a subset containing approximately a 10% of query’s result. Nodes were trained by using a subset of features of interests randomly selected based on their weights at the dictionary.

Each tree is trained by searching intensely the best split variable for each generated subset, resulting in maximizing the information gain. Unlike classification, where it is possible to use equation 5.1 to obtain the entropy, an unsupervised form of entropy is used, where each subset that reaches a node is explained by a multivariate *Gaussian* distribution, then the differential entropy of a d -variate *Gaussian* is defined as:

$$H(S) = \frac{1}{2} \log ((2\pi e)^d |\Lambda(S)|) \quad (5.6)$$

Where $\Lambda(S)$ is a $d \times d$ covariance matrix and $|\Lambda(S)|$ is its determinant. Being d high; covariance matrix calculation returns values close to zero or zero, causing error and small negative values for the determinant. This problem is solved in our implementation by the application of two strategies:

1. Reducing d by randomized node optimization.
2. Adding a small bias λ to the diagonal of the covariance matrix before computing its determinant. We perform $\Lambda(S) + \lambda I$, where I is the identity matrix.

The information gain based on the unsupervised entropy is expressed as:

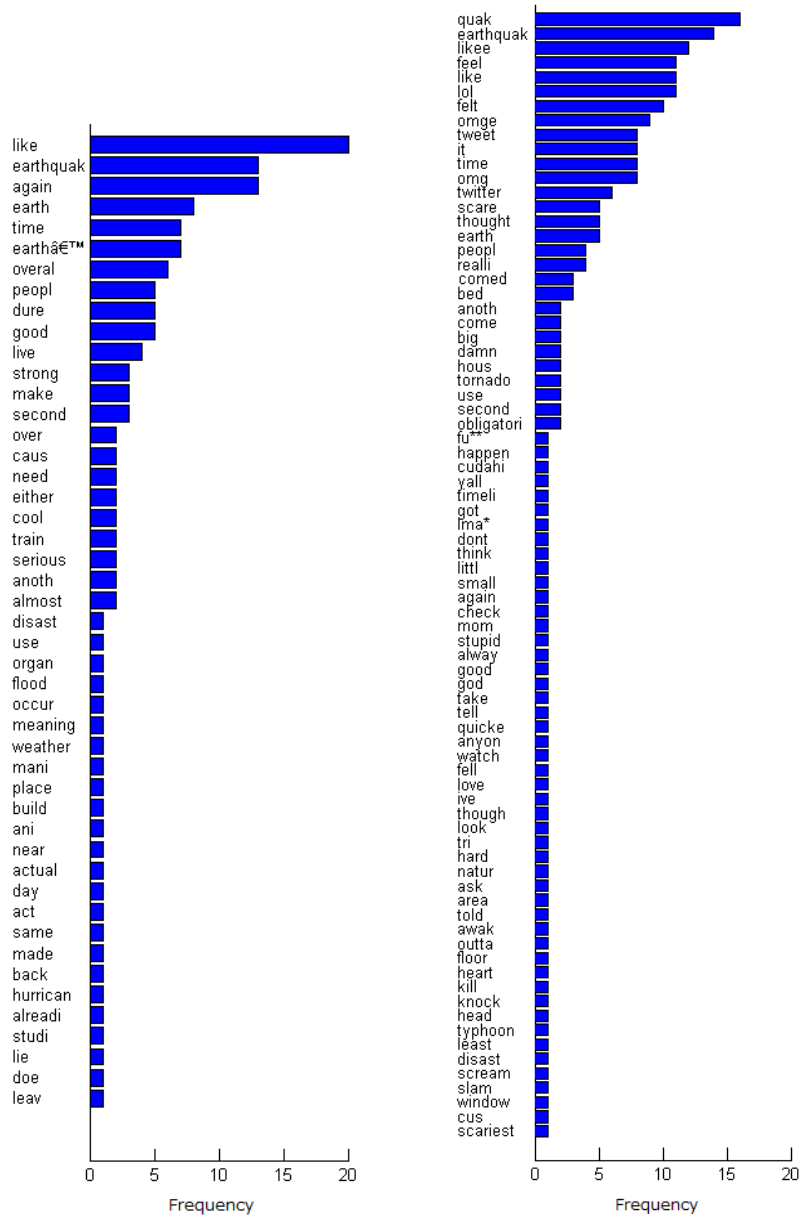
$$I_j = \log (|\Lambda(S_j)|) - \sum_{i \in \{Left, Right\}} \frac{|S_j^i|}{|S_j|} \log (|\Lambda(S_j^i)|) \quad (5.7)$$

Where $|\cdot|$ indicates the determinant or the cardinality of the subsets. By optimizing previous equation, the process results in a tree that splits a subset into several clusters which are assumed to have a *Gaussian* distributions.

Application’s goal is not to analyze the generated clusters, neither to study any way to perform prediction. Instead, it searches to identify what is the information most useful to characterize the entire emergency data set. Split features play an important role; they contain the information needed to split training data set into a number of compact clusters. Therefore, for our application, the set of split features constitutes the set of main characteristics detected.

Two random forests were trained with $T = 50$ trees and deep $D = 4$, with different dictionaries as mentioned previously. The selection of D is tied to amount of main variables desired to obtain, higher values of D implies higher processing time. In each case, a bunch of features is presented and the model selects some of them as nodes in the trees. Selected features essentially “decide” on the data set; they are the best choices to provide information about an unsupervised clustering that minimizes the entropy about the emergency event. Main features help to assess the emergency by answering, for example, what is the best to ask? what is the best to evaluate?. After training, each tree generate a matrix F_{t_i} of $d \times (2^{D+1} - 1)$ (features \times nodes), where element $F_{t_i}(k, j)$ is equal to 1 if feature k is the split variable of node j , otherwise 0. Main variables are calculated as follows:

$$Histogram(k) = \sum_{j=1}^{(2^{D+1}-1)} \sum_{i=1}^T F_{t_i}(k, j) \quad (5.8)$$



(a) Results using a dictionary of risk assessment standard words

(b) Results using a dictionary of common terms used in Twitter

Figure 5.3: Random Trees main features detection

Equation 5.8 basically tells us the importance of each word when decisions are made on an emergency event; originally we have many features at each dictionary, now we have discovered naturally the most important characteristics in a reasonable amount of time. Figure 5.3 shows different features discovered, 1st case gives a formal evaluation on standards $\{like, earthquake, again, earth, time, people, strong, serious, \dots\}$, while the 2nd set of words shows people’s reactions by common terms $\{earthquake, feel, lol, omg, scare, damn, \dots\}$. This comparison gives us an idea of how important is the data dictionary. For similar situations in the future, it is essential to have as many resources as possible available. Any assessment system should include for critical regions, for example countries located on edges of tectonic plates, dictionary of locations, list of first and last names, stations names, landmarks names, list of rivers, schools, hospitals, public offices, geopolitical entities, etc. Our system was unable to identify locations from tweets due the lack of features at used dictionaries. Complete dictionaries are essential for a larger scale event in future situations. However, in a more abstract level, random forest model has the potential to make an important contribution in a disaster response situation.

At a real emergency, after processing and averaging some few trees, we might have an approximation to assess the situation. Over the time, new trained trees are incorporated to improve the results.

Learned main features might be organized as rules; to detect rules, component trees of a random forest with $T = 100$ and deep $D = 4$ are combined into one single tree that shows relations among variables. Figure 5.4 displays for each node the most predominant five split features derived from:

$$Node(j) = \arg \max_{1 < k < d} \left(\sum_{i=1}^T F_{t_i}(k, j) \right) \quad (5.9)$$

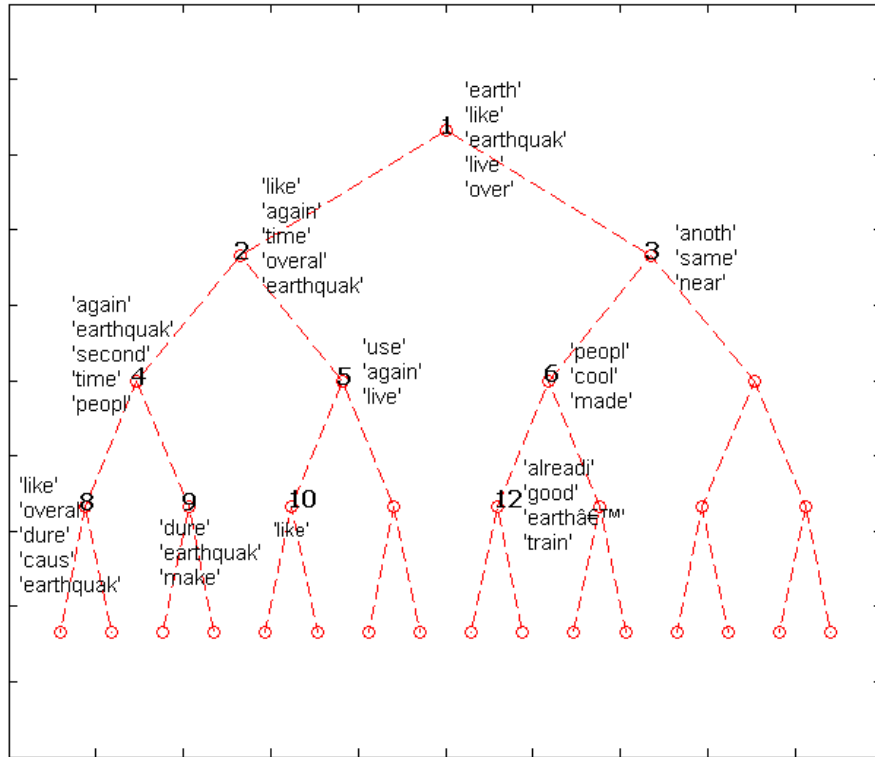
Branches to the right are positive evaluation results of split features, while branches to the left are negative results. For example rules discovered for 1st case using a dictionary of terms from risk assessment standards can be the sequence of nodes 1, 3, 6: $YES(earth, like, earthquake, live, over) \rightarrow NO(another, same, near) \rightarrow (people, cool, made)$, and the sequence of nodes 4, 9: $YES(again, earthquake, second, time, people) \rightarrow (dure, earthquake, make)$.

While for the 2nd case, using a dictionary of terms commonly used in Twitter, uncovered rules such as sequence 2, 5, 11: $YES(lol, quake, time, earthquake, like) \rightarrow YES(feel, use, scare, big, always) \rightarrow (come)$ and sequence 4, 9: $YES(feel, like, it, tweet, quake) \rightarrow (felt, think, really, mom, anyone)$ provide information oriented to people’s reactions.

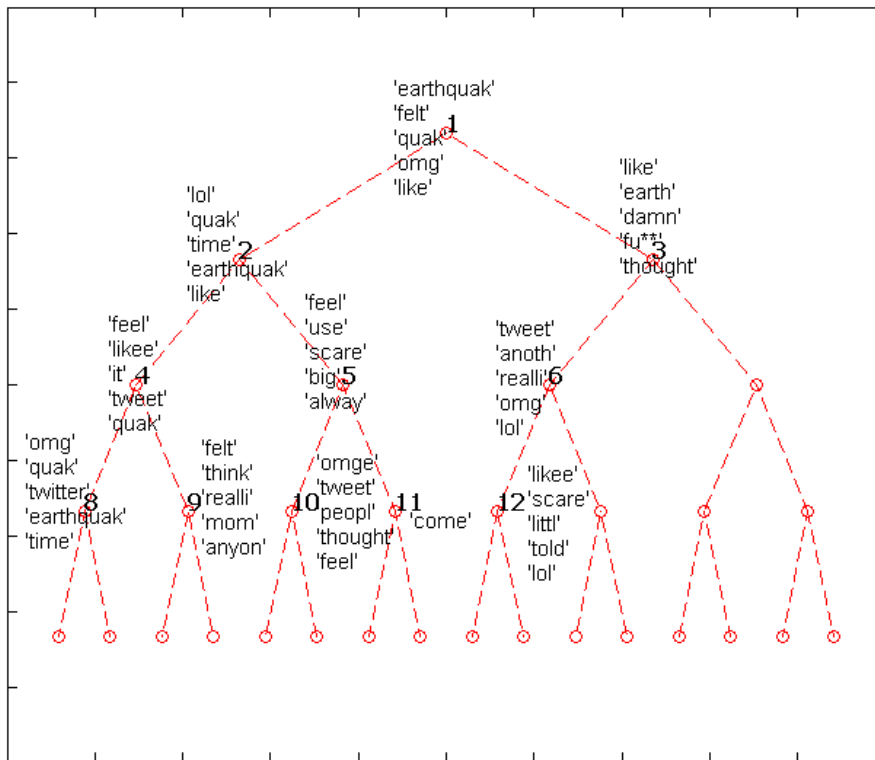
Due that data set in the experiments reported in this chapter corresponds to a small earthquake with no damage reported, the trees tend to organize the data to their left “negative” side. We believe that for real emergencies information will be spread mainly to the right side of the trees.

5.4 Conclusion

Random forest applied to uncover main features during an abnormal event produced a potential new evaluation tool. Random forest automatically and with no previous information organized predominant terms in a ranking list that indicates



(a) Results using a dictionary of risk assessment standard words



(b) Results using a dictionary of common terms used in Twitter

Figure 5.4: Hierarchy of main features for rules detection

their importance within the discovered set. Variables are the result of maximizing systematically the information gain, and their frequency shows and approximation of how information propagated during the emergency event. An averaged tree has been proposed showing the main test features for each node, from where rules can be inferred.

Results and the effect of using different dictionaries demonstrated how the emergency event can be described from different perspectives, being necessary the usage of multiple dictionaries of locations, names, and other entities for real situations. The empirical findings reported in this chapter present a relevant model that can be used to provide information to decision takers during difficult times, and will play an important role as large-scale data processing model for discovering patterns.

Chapter 6

Tracking News Topics by Particle Filtering

6.1 Introduction

As mentioned, society continuously communicate on social networks, blogs and news; the permanent generation of text data makes an attractive challenge to discover what people are talking about and track a non-linear status of a certain topic over the time. *Particle filtering* (PF) is a powerful methodology that might help on text non-linear tracking; it is mainly used for sequential signal processing with a wide scope of applications. Several non-linear problems such as localization and tracking of targets, recognition of objects in video or image, among others, include PF as part of their solution. The usage of PF for tracking a set of related words in a news stream is a novel application, the main idea is to track a predefined topic and its “relevance” (probability), estimating it over the time. PF uses a state-space model to perform estimations, on text the state-space model could be a probability model, like the *Unigram model*. This chapter present a straightforward model of a topic named as *nuclear issue* from a set of news generated during March 2011 earthquake in Japan. PF *sampling importance re-sampling (SIR) algorithm* is used to process data as it arrives, for rapid adaptation to changing signal characteristics. Every time a news is read, the algorithm computes the posterior probability density function (pdf) of the system’s state based on available information, tracking and estimating the predefined topic “nuclear issue”.

In addition, the conventional SIR algorithm is modified by adding a correction term every time SIR random samples also known as *particles* and their associated weights are generated, improving SIR’s estimation over the time. It computes the posterior *pdf* of the system’s state based on available information, adding an observation based correction term to the generated particles.

6.2 Particle Filtering

Particle filter is a sequential Monte Carlo method based on point mass or particle representations of probability densities, which can be applied to any statespace model. The basic idea is the recursive computation of relevant probability distributions using the concept of importance sampling and approximation of probability

distributions with discrete random measures [Djuric et al., 2003]. Sequential Monte Carlo methods mainly have high computational complexity and need important computing resources; these are the reasons why PF became a very active area of research since last decade. Many real problems require the estimation of the system's state that changes over the time using a set of measurements; some systems can be represented as a statespace model of the form:

$$\begin{cases} x_t = f_t(x_{t-1}, u_t) \\ y_t = g_t(x_t, v_t) \end{cases} \quad (6.1)$$

Where $t = 1, 2, 3, \dots, n$ is a time index, x_t is the state (or signal) that needs to be estimated, y_t is a vector of observations, u_t is a state noise vector, v_t is an observation noise vector, and f_t and g_t are nonlinear functions which might vary with time. The first equation in 6.1 is known as a state equation, and the second one, as measurement equation. The standard assumptions are that functions f_t and g_t , and the distribution of noises u_t and v_t are known. The basic idea of PF is to recursively estimate x_t from measurements y_t , in particular we seek filtered estimates of x_t based on the set of all available measurements y_t up to time t [Arulampalam et al., 2002].

In case of text sources, the state-space approach could be used to model the dynamic of a topic in a news stream over the time. The state vector is designed to contain the information that describes the system, as well as in tracking system the state vector may contain position, velocity and/or acceleration of a target or object; in our proposed idea the model contains a measurement of a topic relevance estimated from the frequency of a set of related words. Imagine a certain topic as a “nanorobot” moving in a sea of text data generated from news and social networks, our goal is to estimate its *position*, considering it as topic's relevance, a metric of how much people are talking about it over the time. We believe that, in the same way as an object is tracked in a video, an object composed by text (a set of words in this case) can be automatically “localized” in a huge amount of information continuously generated over the time. We introduce a simple model to demonstrate it by using particle filtering.

PF implements a recursive *Bayesian filter* to estimate a state variable of a system by Monte Carlo simulations. PF represents the required posterior distribution function (pdf) by a set of random samples with associated weights and compute estimates based on these samples and weights [Arulampalam et al., 2002]. Let $X(t) = \left\{ x_{(t)}^m, w_{(t)}^m \right\}_{m=1}^M$ denote a random measure that characterizes the posterior pdf $p(x(t)|y(1:t))$, where $x_{(t)}^m$ is the m^{th} particle of the state vector at time instant t , $w_{(t)}^m$ is the weight of that particle, and M is the number of particles. This random measure $X(t)$ is obtained from $X(t-1)$ by using the observations $y(t)$. Then, the posterior pdf at time t can be approximated as:

$$p(x(t)|y(1:t)) \approx \sum_{i=1}^M w_{(t)}^i \delta(x(t) - x_{(t)}^i) \quad (6.2)$$

Where $\delta(\cdot)$ is the *Dirac delta function*. With this approximation, computations of expectations are simplified to summation, for example:

$$E(g(X)) \approx \sum_{i=1}^M w_{(t)}^i g(x_{(t)}^i) \quad (6.3)$$

Due that it is difficult to draw samples from $p(x(t))$, the particles and weights are generated using the principle of importance sampling, that relies on a proposal distribution $\pi(x(t))$ known as importance density function (IDF) from where samples $x_{(t)}^{(m)}$ are easily generated. Then, weights are defined as:

$$w_{(t)}^m \propto \frac{p(x_{(t)}^m | y(1:t))}{\pi(x_{(t)}^m | y(1:t))} \quad (6.4)$$

From 6.4 it is derived the following recursive equation:

$$w_{(t)}^m \approx w_{(t-1)}^m \frac{p(y(t) | x_{(t)}^m) p(x_{(t)}^m | x_{(t-1)}^m)}{\pi(x_{(t)}^m | x_{(t-1)}^m, y(t))} \quad (6.5)$$

Thus, PF consists in a recursive estimation of state $x(t)$ by a propagation of the weights and particles as each measurement is received sequentially.

6.3 Tracking a Topic

Information exchange by internet opened a door to research and applications focused on *NLP (natural language processing)*; NLP models are required for text processing; in case of topic's tracking the main idea is to adapt a NLP model to a state-space model as seen in equation 6.1. We assume the system's state as a certain topic's relevance denoted as R , and its estimated value as \hat{R} . By using a set of news related to natural disasters, it is desired to calculate the topic's relevance R every time a news is read, R is based on the frequencies of related words. In our experiments, the topic is fixed by words: $\{energy, nuclear, radiation, people\}$, therefore the tracking by PF to be described estimates and follows a "nuclear issue" topic.

The state-space approach is applied in the following way:

1. The state vector R is a vector with a single element whose value is equal to the join probability of words $\{energy, nuclear, radiation, people\}$, that means, the model assigns a probability to the topic "nuclear issue" for each news (6.8).
2. The measurement vector Y represents noisy observations that are related to the state vector; in this case we add a randomly generated Gaussian noise to R . Therefore, to analyze and make inferences about the system, we use vector R that describes the evolution of the state, and vector Y that represents noisy measurements of R .

R is calculated by using *unigram model* where the probability of appearance of a word wd_i in a news is calculated without considering any influence from other

related words within the topic. We define R as the probability of observing words wd_1, \dots, wd_4 in a news, and it is approximated as:

$$R = P(wd_1, \dots, wd_4) = \prod_{i=1}^4 P(wd_i | wd_1, \dots, wd_{i-1}) \quad (6.6)$$

Where $P(wd_i | wd_1, \dots, wd_{i-1})$ is the maximum likelihood estimation (MLE) which is approximated by unigram model as $P(wd_i)$, therefore the state is calculated as:

$$R = P(wd_1, \dots, wd_4) = \prod_{i=1}^4 P(wd_i) = \prod_{i=1}^4 \frac{c(wd_i)}{\sum_{\tilde{wd}} c(\tilde{wd})} \quad (6.7)$$

Where $c(wd_i)$ is the count (frequency) for wd_i within a news, and $\sum_{\tilde{wd}} c(\tilde{wd})$ is the total amount of words for that news. Equation 6.7 has a constrain, the value of R becomes zero if any wd_i doesn't exist, for that case MLE does not work. To solve that limitation it is assigned a small probability λ to the "unknown" words equation 6.7 is arranged as:

$$R = P(wd_1, \dots, wd_4) = \prod_{i=1}^4 \left((1 - \lambda)P(wd_i) + \frac{\lambda}{N} \right) \quad (6.8)$$

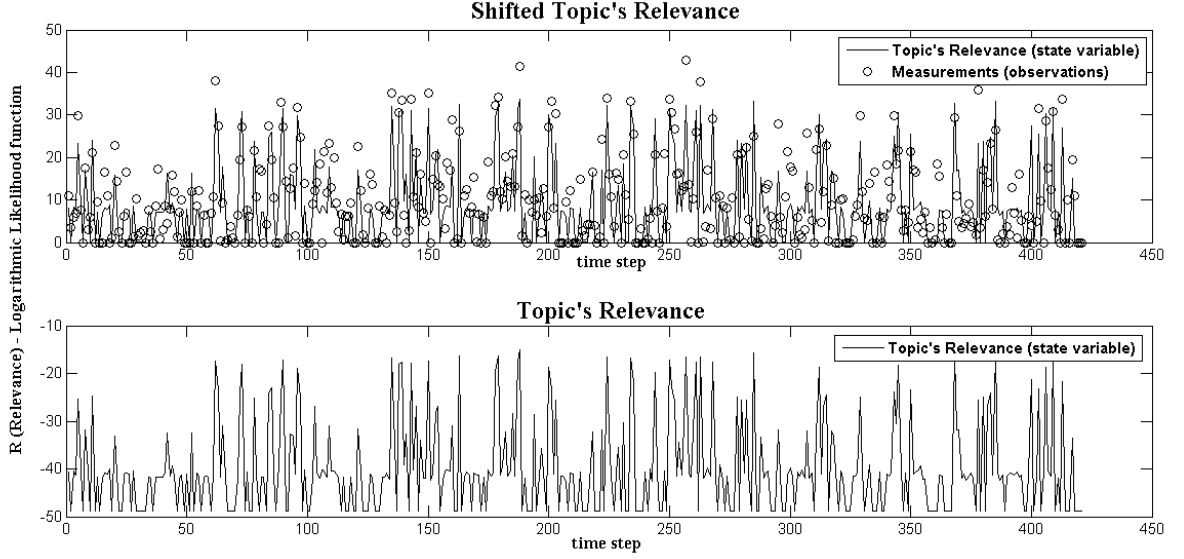
Where λ is a least value, for instance $\lambda = 0.05$, N is the total amount of words of the set of news, and $P(wd_i) = \frac{c(wd_i)}{\sum_{\tilde{wd}} c(\tilde{wd})}$. Computation of equation 6.8 gives very small numbers that may produce underflow during simulation. Underflow is a condition in a computer program where the result of a calculation is a smaller number than the program can actually manage in memory. In practice, to avoid underflow, it is often more convenient to work with the logarithm of the likelihood function, called log-likelihood, thus equation 6.8 is arranged as:

$$R = P(wd_1, \dots, wd_4) = \prod_{i=1}^4 \ln \left((1 - \lambda)P(wd_i) + \frac{\lambda}{N} \right) \quad (6.9)$$

This last expression returns negative values; therefore, as final step, equation 6.9 is shifted by adding a positive amount β to R in order to manipulate during PF positive values of R (see figure 6.1). Even though the dimension of the state vector R is a single one, it represents and compresses information of 4 different variables (words). We assume that when a person talks about something, a person mentions words related to it. In that sense, R represents how much a person talks about a "nuclear issue" in a news, representing a measurement of that topic's weight over the whole unknown topics in that news, over the time. With this model, we are trying to demonstrate whether it is possible to track and estimate, automatically, the status of a certain topic over the time or not.

6.4 Sampling Importance Re-sampling

Sampling-importance-re sampling (SIR) was proposed in [Gordon et al., 1993] and it is known as *bootstrap filter*. Considering the state-space model given by vectors R and Y (Figure 6.1), we denote $x_{(t)}$ and $y_{(t)}$ as the values of R and Y at time t


 Figure 6.1: State variable R and observations Y news set

respectively. The particles x_t^m are generated from an importance density function $\pi(x_t)$, for our case the prior density function $p(x_t|x_{t-1}^m)$ is used for drawing the particles, that means; the importance density function $\pi(x_t)$ is equal to the transition density function $p(x_t)$, therefore:

$$\pi(x_t|x_{t-1}^m, y(1:t)) = p(x_t|x_{t-1}^m) \quad (6.10)$$

And the weights, upon the reception of the measurement y_t , are calculated by:

$$\tilde{w}_t^m = w_{t-1}^m \frac{p(y_t|x_t^m)p(x_t^m|x_{t-1}^m)}{\pi(x_t^m|x_{t-1}^m, y_t)} \quad (6.11)$$

Where \tilde{w}_t^m is a non-normalized weight that can be expressed by using 6.10 as:

$$\tilde{w}_t^m = w_{t-1}^m p(y_t|x_t^m) \quad (6.12)$$

Equation 6.12 indicates that weights are proportional to likelihood of the observation given the drawn particles $p(y_t|x_t^m)$. Figure 6.2 shows the steps of SIR algorithm; in addition, the steps of its implementation are explained below:

1. Initialization, drawn M particles $x_{(0)}^m$ from a normal distribution $\mathcal{N}(x_{(0)}, 1)$ and set all weights as $w_{(0)}^m = \frac{1}{M}$, in this case, all particles have the same weights; the variance of the weights is zero. That means, the starting point is characterized by having all particles with the same probability (weights).
2. Draw particles from the importance density function. Particles are generated by a Gaussian distribution 6.13, where particles created in time $(t - 1)$, the parents, are used as means.
 $x_t^m \sim \pi(x_t|x_{t-1}^m)$, $m = 1, 2, \dots, M$, for our case it is used a normal Gaussian distribution with mean x_{t-1}^m and standard deviation σ . Thus,

$$\pi(x_t|x_{t-1}^m) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_t - x_{t-1}^m)^2}{2\sigma^2}} \quad (6.13)$$

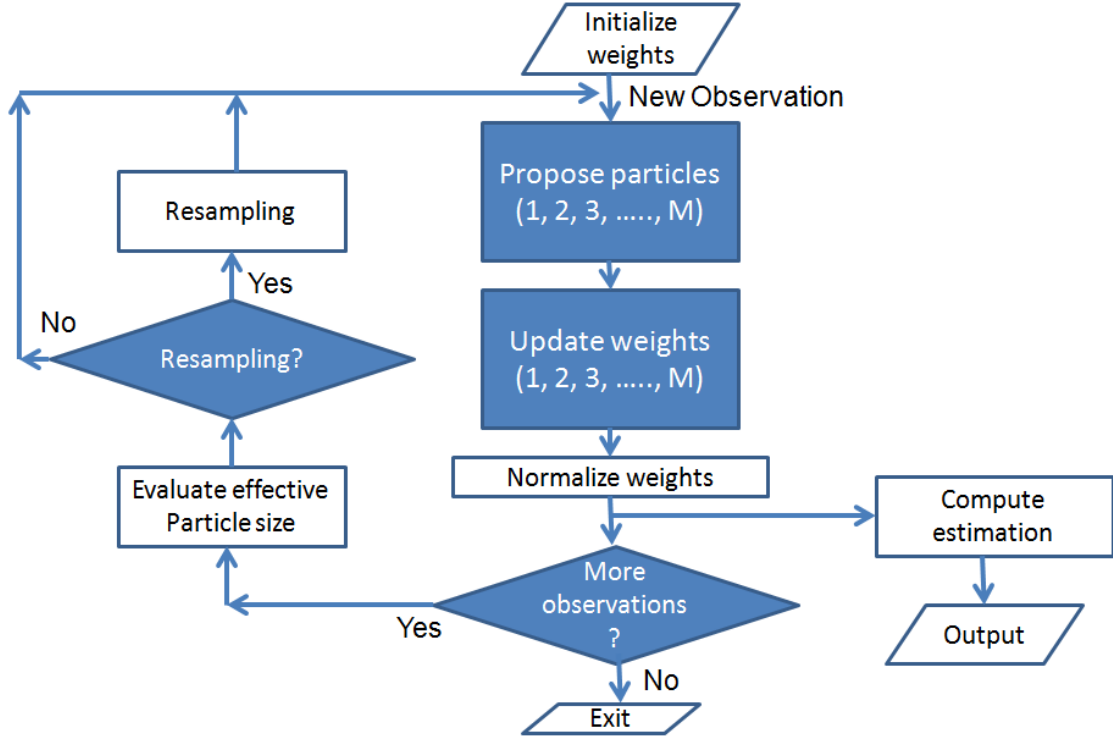


Figure 6.2: SIR algorithm, also known as bootstrap filter

Particles generated will be the parents of next time particles, and so on. Such sequence is called a particle stream, and it represents one possible evolution of the system's state over the time.

3. Compute the non-normalized weights of previously obtained particles by equation 6.12, since $\pi(x_t) = p(x_t)$, the weights are calculated by:

$$\tilde{w}_t^m = \tilde{w}_{t-1}^m \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_t - x_t^m)^2}{2\sigma^2}} \quad (6.14)$$

The weights are introduced to correct for the bias that arises due to sampling from a different function (step 2) than the one that is being approximated. Equation 6.14 shows that the weight of the m^{th} stream is obtained by updating its value at time $(t - 1)$.

4. Normalize the weights so that they sum up to one, that is:

$$w_t^m = \frac{\tilde{w}_t^m}{\sum_{j=1}^M \tilde{w}_t^j} \quad (6.15)$$

Due that weights represent probabilities that the particles are accurate representations of the system based on observations, they are arranged by equation 6.15.

5. Having the random measure $X(t) = \{x_t^m, w_t^m\}_{m=1}^M$ we are able to calculate an estimation \hat{R} of the state variable as follows:

$$\hat{R}(t) = \sum_{i=1}^M w_t^i x_t^i \quad (6.16)$$

In this step we are using the set of particles and weights to approximate the probability of the selected topic “nuclear issue” (state variable).

6. At each step the particles are propagated and assigned weights, as time evolves, almost all particles except for very few have negligible weights. The variance of the weights increases with time, which is an unwanted effect. This degradation leads to a deteriorated functioning of particle filtering. A measurement of this degeneracy is given by equation below known as *effective particle size*. In this step we evaluate this metric:

$$\hat{M}_{eff} = \frac{1}{\sum_{i=1}^M (w_{(t)}^i)^2} \quad (6.17)$$

7. If the effective particle size is below a predefined threshold (in our case threshold is equal to 50), re-sampling is carried out. Re-sampling will eliminate particles with small weights and replicate those with large weights. According to the multinomial probability given by the weights $w_{(t)}^m$ we draw indices k_m for $m = 1, 2, \dots, M$, where $p(k_m = i) = w_{(t)}^i$, once indices are available, we set the new particles as $x_{(t)}^m = x_{(t)}^{k_m}$, and the new weights as $w_{(t)}^m = \frac{1}{M}$. Any potential degradation of the process measured in step 6 is corrected by assigning the same probability (weights) to all particles (as in step 1), and then, the set of particles with same weights are propagated in the next time step.
8. We iterate from step 2 until the last available observation $y_{(t)}$. By applying above process, we ensure that Monte Carlo sampling is easily performed by a relatively simple importance density function and the weights are evaluated without difficulty at each step.

6.5 Model Evaluation

When defining the measurement vector Y it is assumed as noisy observations of the state. R always takes a positive value or zero, therefore all negative values of Y are changed to a very small value, close to zero; in that way we imagine our “nano-robot” having a sensor that measures always positive values of Y . This modification improved the estimation of R . It was implemented an evaluation at stage 2 of algorithm described above; before drawing particles, the state variable given by model is evaluated, if its value is close to zero, the particles are generated with fixed values close to zero. This modification avoided to create negative particles that lead to a negative value of the estimation. Figure 6.3 shows our results for the first 100 news arranged sequentially, we used $M = 400$ at each iteration with $\sigma = 11$ for $\pi(x_{(t)})$. Simulation shows that the estimated value follows the state variable giving suitable results in terms of tracking topic’s relevance over the time. Different strategies can be evaluated by analyzing the mean squared error (MSE) (Figure 6.4); similar outputs can be obtained with fewer particles ($M = 100, M = 200$) by increasing σ of $\pi(x_{(t)})$. The strategy to select will depend on the availability of computing resources that will indicates how many particles can be used, and the value of the standard deviation of $\pi(x_{(t)})$ that should produce particles with large

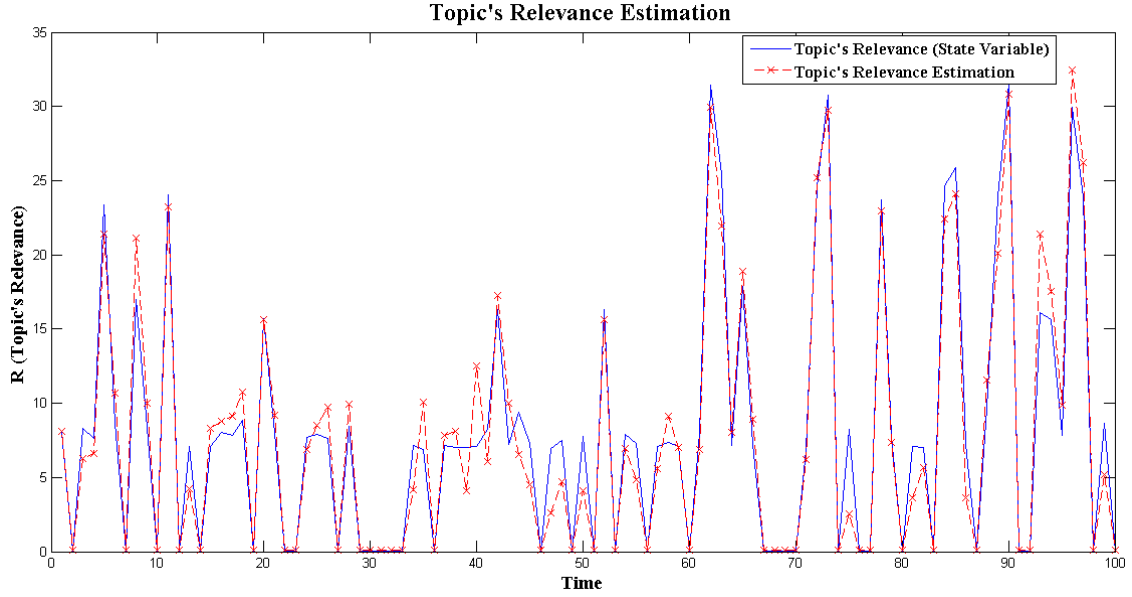


Figure 6.3: Topic's relevance and its estimations by SIR particle filtering

enough variance in order to avoid narrow regions for the estimation, but not too large to avoid too dispersed particles [Arasaratnam et al., 2007].

6.6 A Correction Term Based SIR algorithm

A modified particle filter sampling importance re-sampling algorithm might improve SIR results by adding a correction term every time SIR particles and their associated weights are generated. It computes the posterior probability density function of the system's state based on available information, adding an observation based correction term to the generated particles. As explained previously, distributions that are easy to sample from and whose shape is supposed to be close to the true posterior distribution are chosen. The wanted effect is to generate particles from regions of the support of state value X_t , in figure 6.5 are displayed an importance density function (a Gaussian function) as well as the particles and weights that form a random measure to approximate the posterior distribution of x_t . It is important to remember that, based on equation 6.12, the particle weights are proportional to the likelihood of the observation given the drawn particles, therefore the most important particles, those with higher weights, will be located in the region of the observation value (figure 6.5). At the same time, the generation of particles by equation 6.13 ignores the current observation y_t , which makes the algorithm inefficient in some estimated values. A better solution could be obtained by pushing the most important particles to the high likelihood region, located in the area where the true state value is. We incorporate an intermediate step in SIR algorithm immediately after generating the particles. In order to move some particles to the significant region of state space, we will add to them a correction term $c_{(t)}$ defined below:

$$c_{(t)} = \begin{cases} +|y_{(t)} - x_{(t)}^m| + s_t, & \text{if } x_{(t)}^m \leq y_{(t)} \\ -|y_{(t)} - x_{(t)}^m| + s_t, & \text{if } x_{(t)}^m > y_{(t)} \end{cases} \quad (6.18)$$

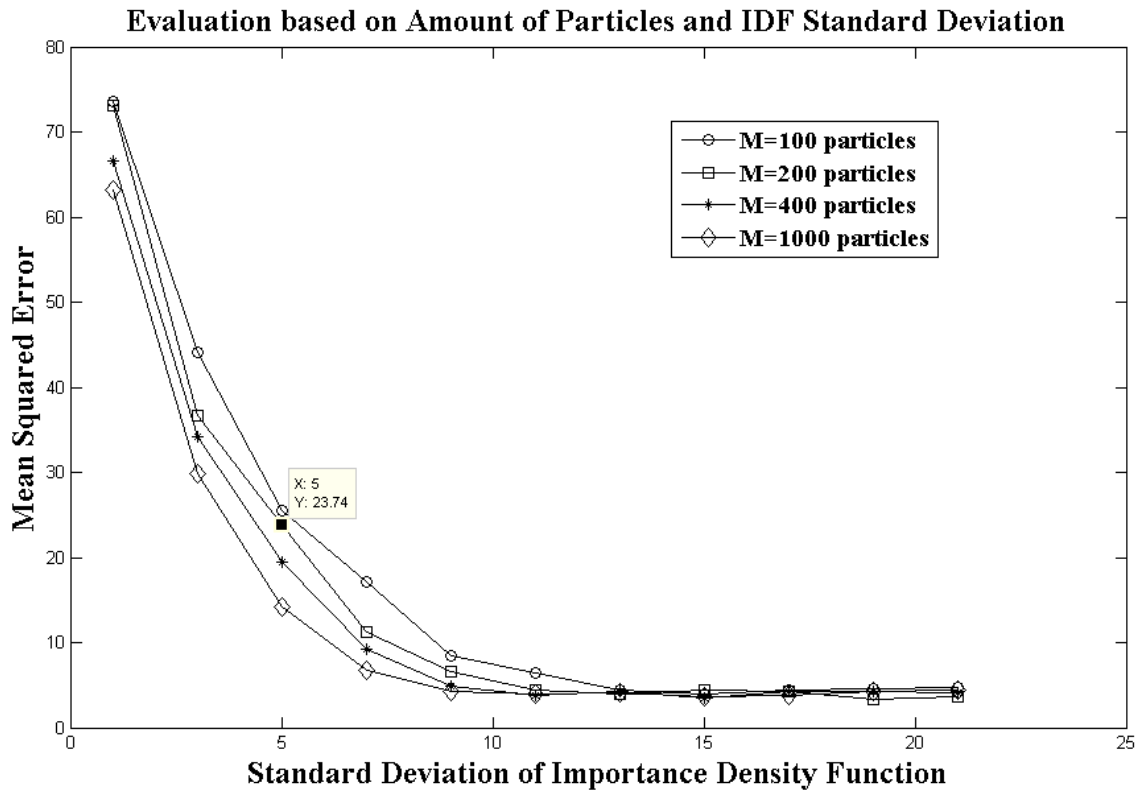


Figure 6.4: Evaluation by amount of particles and standard deviation value

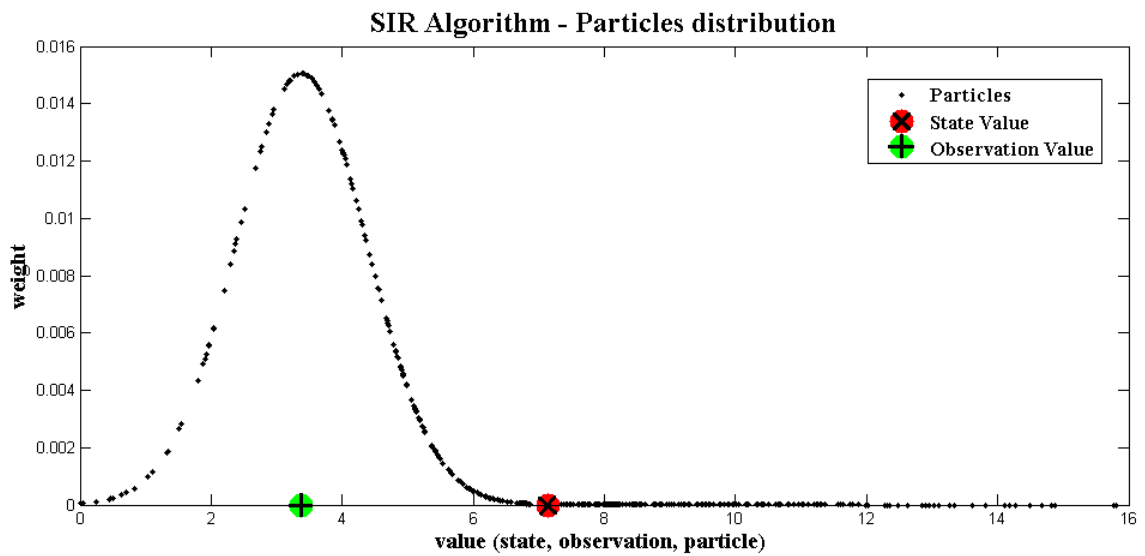


Figure 6.5: SIR algorithm single iteration showing particles generated by a Gaussian function with their weights, and state and observation values

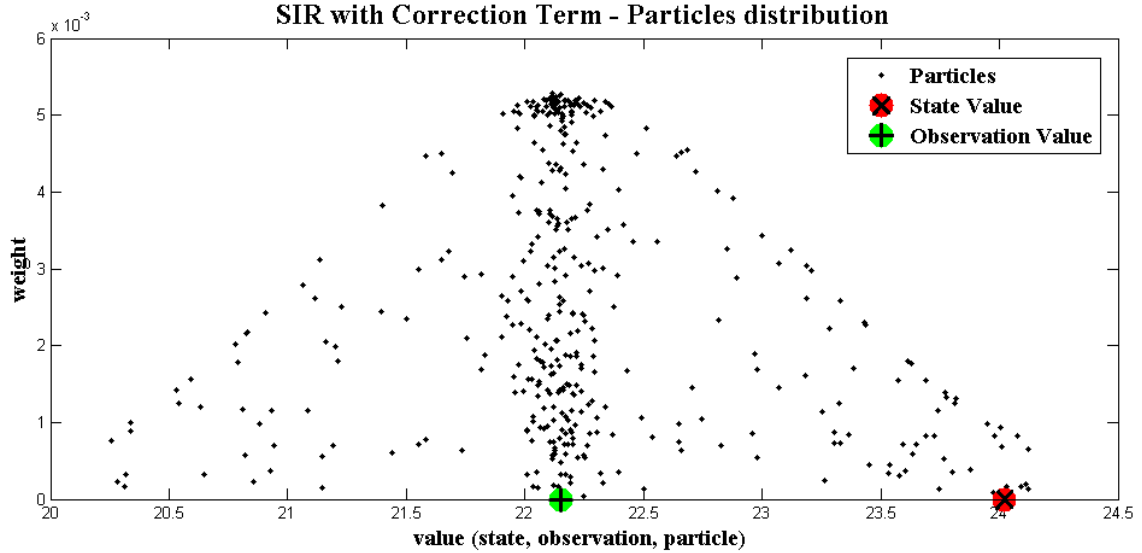


Figure 6.6: Particles values modified by a correction term

Where $s_{(t)}$ is a correction noise value. The criteria is to identify those particles far from the region of the observation, and subsequently add (or subtract) $c_{(t)}$ from them. The particles affected are defined by following rule:

$$\text{ADD } c_{(t)} \text{ IF } |y_{(t)} - x_{(t)}^m| > \sigma \quad (6.19)$$

Rule 6.19 exposes that if the distance between a particle and the observation value is more than the standard deviation of the importance density function, the particle will be “corrected” by $c_{(t)}$. Figure 6.6 shows a result of this additional step applied after particles are generated. To demonstrate the performance of the proposed filter we ran SIR conventional filter and our proposal. Figure 6.7 displays results for a window of time from 1 to 100, with $M = 400$ particles and $\sigma = 11$ for $\pi(x_{(t)})$. Simulation demonstrates that the estimated state value by using correction term is more accurate, in most of the cases, than conventional SIR algorithm. A second measurement to evaluate proposed algorithm is by analyzing the mean squared error (MSE) (Figure 6.8). Although it improves the results of SIR, similar outputs are obtained with fewer particles, for example ($M = 100$, $M = 200$). The results improve a negligible value by adding more particles. The strategy to select will depend on the availability of computing resources that will indicates how many particles can be used, and the value of the standard deviation of $\pi(x_{(t)})$ that should produce particles with large variance but not too dispersed [Gordon et al., 1993].

6.7 Conclusion

It has been demonstrated that a topic can be modeled, tracked and estimated in a text stream by particle filtering over the time. The main advantage of choosing the importance density function equal to the prior pdf is the ease in the computation of weights. Importance density function has been assumed to be a normal Gaussian distribution which leads to an easy generation of particles.

In our first experience we did not use the current observation $y_{(t)}$ to improve sampling. Once the particles are generated the only thing we do to steer the particles

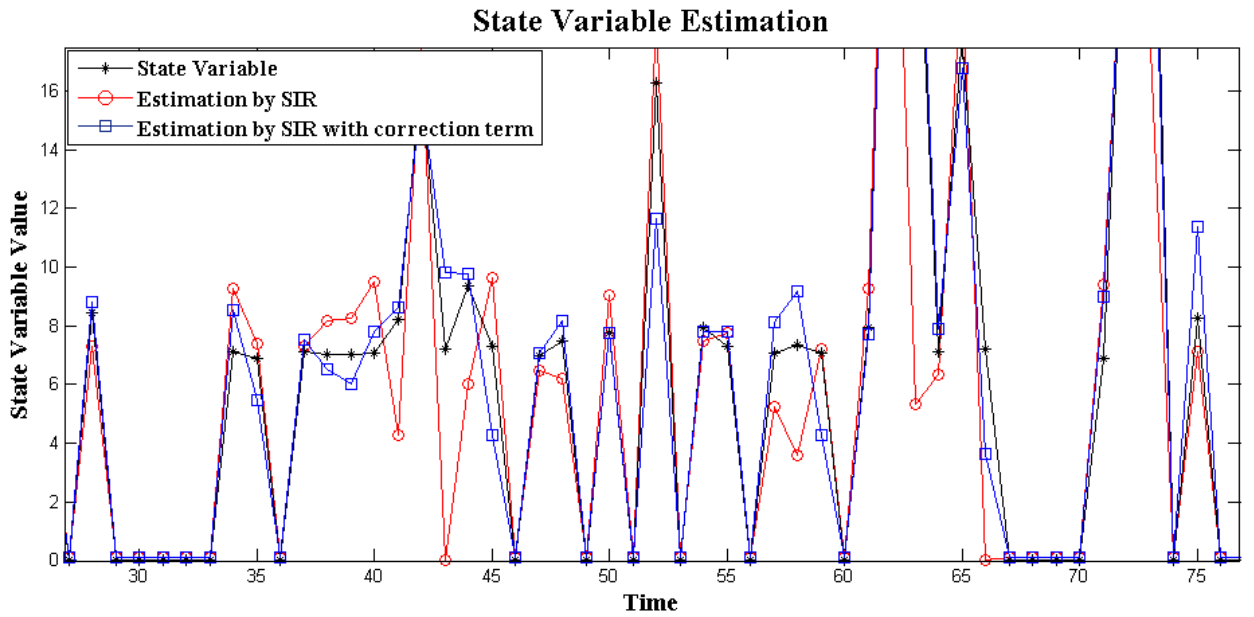


Figure 6.7: Comparison of estimated values between SIR and SIR with correction term

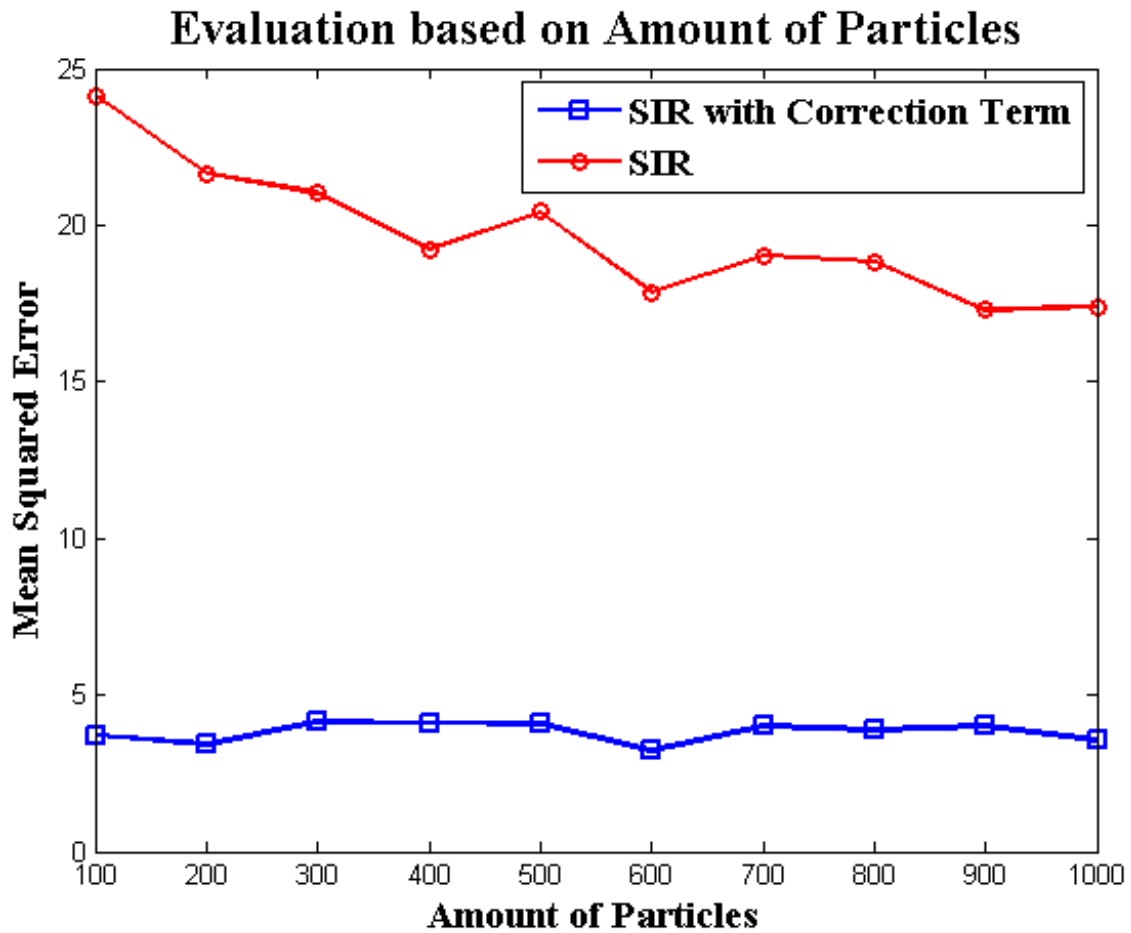


Figure 6.8: Comparison SIR and SIR with correction term by mean squared error

towards the region of the state space with large probability density is by re-sampling. SIR algorithm has been implemented and modified to avoid generation of negative estimations for an always positive state variable. NLP unigram model has been matched to a state-space approach needed for PF, and emulated the observations by adding noise to the state variable, although a more advance topic's model should be developed.

Secondly, a better filter is proposed to improve estimations by modifying some particles with the addition of a correction term. Once the particles are generated our proposal improves the values of particles located far from observation value, obtaining an effect similar to re-sampling. It implies a good improvement as seen in figure 6.8, a big gap exists by comparing both mean squared errors.

Chapter 7

Natural Disasters Topics Detection Over Twitter Data Stream

7.1 Introduction

Twitter has become a worldwide used social network, capable to receive hundreds of comments per minute about several topics of interest. Even more, amounts of data increase when emergencies happen, such as natural disasters. During those critical moments people's natural reactions include fear, worries, the action of escape from danger, make telephone calls to family or friends, and lately, it has become very common to share experiences immediately after the event by using social networks. We recorded posts from Twitter during a minor earthquake in California on May 8th 2014; people felt the earthquake and immediately reported brief text updates, at 06:41 UTC time zone (Coordinated Universal Time) the number of tweets per minute increases rapidly, returning after 25 minutes approximately to an average number. A huge amount of tweets were created as an instantaneous reaction to an emergency, in real-time. Tweets per minute can be monitored by a detection system that recognizes peaks exceeding a predefined threshold; those tweets can be entered in a Bayesian machine learning model that infers topics and subjects previously learned from a huge amount of tweets training data. It could show immediately the content and context of a potential emergency, without the delay that would imply to read all of them in sequence. A model towards the mentioned detection system, that incorporates Latent Dirichlet Allocation (LDA) model, also known as topics model, is explored in this chapter. The key idea behind the LDA model for text is to assume that words in each document were generated by a mixture of topics, where a topic is represented as a multinomial probability distribution over words. The mixing coefficients for each document and the word topic distributions are hidden and learned from data using unsupervised learning methods.

Next, we explore the adaptation of Bayesian topic model LDA to discover topics on Twitter stream text data related to natural disasters. By uncovering topics over the time, the inferences would expose the concept and context of the most significant issues during a potential emergency.

7.2 Latent Dirichlet Allocation

LDA model was introduced at [Blei et al., 2003] based on a general Bayesian framework, the method has been broadly applied in machine learning and data mining, particularly in text analysis and computer vision, with the Gibbs sampling algorithm as most used for approximation of the posterior distribution. In general, there is a general motivation to migrate the application of topics model from batch to on-line mode to implement real time systems. It implies to speed-up the learning of topic models, as well as improves the speed and accuracy of the inference for the posterior distribution.

In statistical natural language processing, a document has multiple topics and its words reflect the particular set of topics it addresses [Griffiths and Steyvers, 2004]. LDA treat each topic as a probability distribution over words, and a document as a probability mixture of these topics. If we have k topics, the probability of the i^{th} word in a given document is:

$$p(w_i) = \sum_{j=1}^k p(w_i|z_i = j)p(z_i = j) \quad (7.1)$$

Where z_i is a latent variable indicating the topic from which the i^{th} word was drawn and $p(w_i|z_i = j)$ is the probability of the word w_i under the j^{th} topic. $p(z_i = j)$ is the probability of choosing a word from topic j . Intuitively $p(w_i|z_i = j)$ indicates which words are important to a topic, whereas $p(z_i = j)$ is the predominance of those topics within a document. Given D documents containing k topics expressed over W unique words, it is possible to represent $p(w|z)$ with a set of k multinomial distributions ϕ over the W words such as $p(w|z = j) = \phi_w^{(j)}$, and $p(z)$ with a set of D multinomial distributions θ over the k topics, such that for a word in a document d :

$$p(z = j) = \theta_j^{(d)} \quad (7.2)$$

Latent Dirichlet Allocation [Blei et al., 2003] is a model that combines equation 7.1 with a prior probability distribution on θ to provide a complete generative model for documents. Figure 7.1 shows the graphical model representation of LDA where each piece is a random variable. For each word in document d LDA performs:

1. Sample a topic $z_{i,d}$ from a multinomial distribution with parameter θ_d having Dirichlet prior.
2. Sample a word $w_{i,d}$ from a multinomial distribution with parameter $\theta_{z_{i,d}}$ having Dirichlet prior.

Given observed words $w = \{w_{i,d}\}$, LDA computes the posterior distribution over the latent topics $z = \{z_{i,d}\}$, the mixing proportions θ_d and topics ϕ_4 . Mainly, LDA uses Gibbs sampling [Griffiths and Steyvers, 2004] as inference method, where θ and ϕ are marginalized out, and only the latent variables z are sampled. After several iterations of Gibbs sampler, the estimations of θ and ϕ are calculated. Given the current state of all but one variable $z_{i,d}$ the conditional probability is:

$$p(z_{i,d} = j|z^{-i,d}, w, \alpha, \beta) = \frac{1}{C} a_{j,d} b_{w,j} \quad (7.3)$$

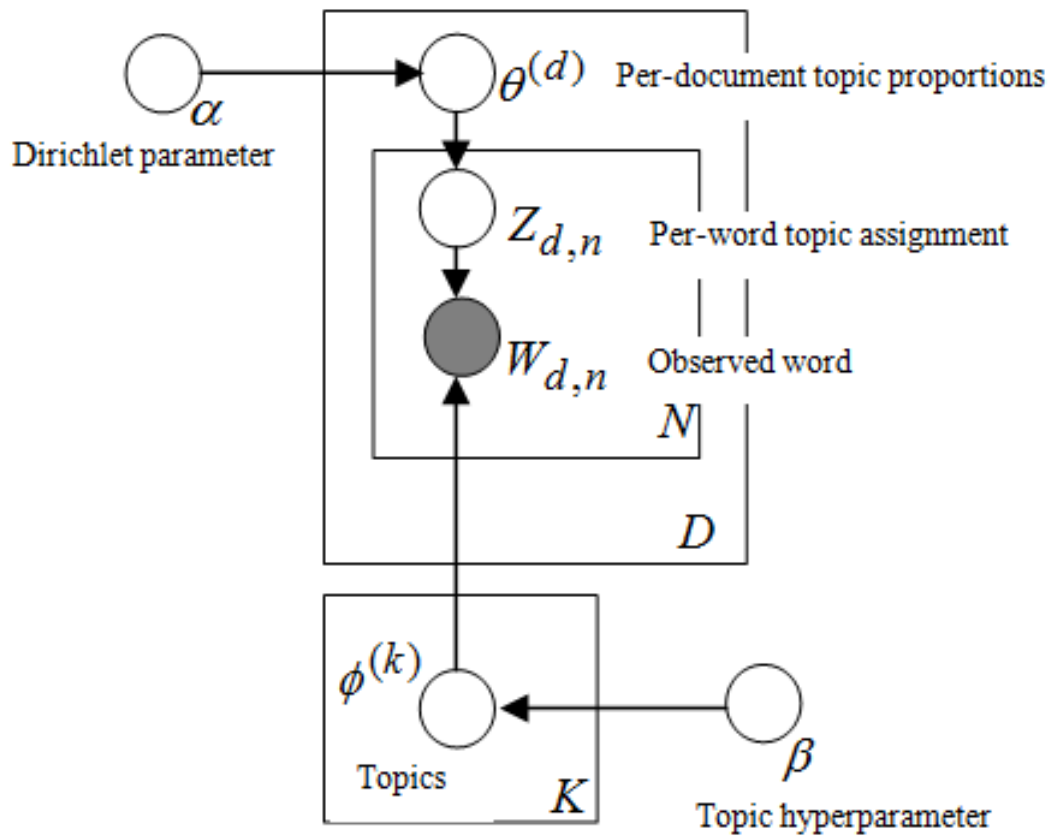


Figure 7.1: LDA graphical model

With $a_{j,d} = N_{j,d}^{-i,d} + \alpha$, where $N_{j,d}$ is the number of times a word in document d has been assigned to topic j .

And $b_{w,j} = \frac{N_{w,j}^{-i,d} + \beta}{N_j^{-i,d} + W\beta}$, where $N_{w,j}$ is the number of times a word w is assigned to topic j , N_j is the number of times a word has been assigned to topic j , and W is the amount of unique words.

The corresponding $-i, d$ indicates that the corresponding $z_{i,d}$ has been excluded from the counters N . The normalization constant of equation 7.3 is defined as:

$$C = \sum_j a_{j,d} b_{w,j} \quad (7.4)$$

7.3 LDA training

We collected a set of tweets with keywords related to natural disasters, such as {natural disaster, tsunami, earthquake, quake, flood, cyclone, avalanche, blizzard, landslide, typhoon, etc.}. The collection was intermittent, however, each time we gathered continuous tweets over the time. That approach allowed to register several *peaks*, a peak is an amount of tweets per minute that exceeds a threshold value set as 200. If there are more that 200 tweets at any time (minute) a potential emergency would be happening. Several peaks were recorded as a training set to detect topics, each peak is considered as a document, and every tweet within a peak is a sentence of that document. The training start with one random initialization of $z_{i,d}$ topic assigned to word i at document d , and adding counts N , algorithm below shows the initialization needed previous to learning.

Algorithm 1: Initialization algorithm:

1. For each peak
 - (a) For each tweet
 - i. Split tweet into words
 - ii. For each word
 - A. Assign $z_{i,d} = rand[0, k]$
 - B. Add counts $N_{j,d}, N_{w,j}, N_j$
 - iii. End
 - (b) End
2. End

The collapsed Gibbs sampling algorithm involves repeatedly sampling a topic assignment for each word in the corpus; each sampled topic assignment is generated from a conditional multinomial distribution over the k topics, which in turn requires the computation of k conditional probabilities given by equation 7.3. Gibbs sampler implementation is shown at algorithm 2 below. The training is a problem with time equal to $k \times w \times d \times n$ iterations where k is the number of topics, w is amount of words, d is amount of documents and n the iterations needed for convergence. k is defined by user, in our case we used $k = 30$, Dirichlet hyper-parameters α and β are set to 0.1.

Algorithm 2: Collapsed Gibbs Sampling

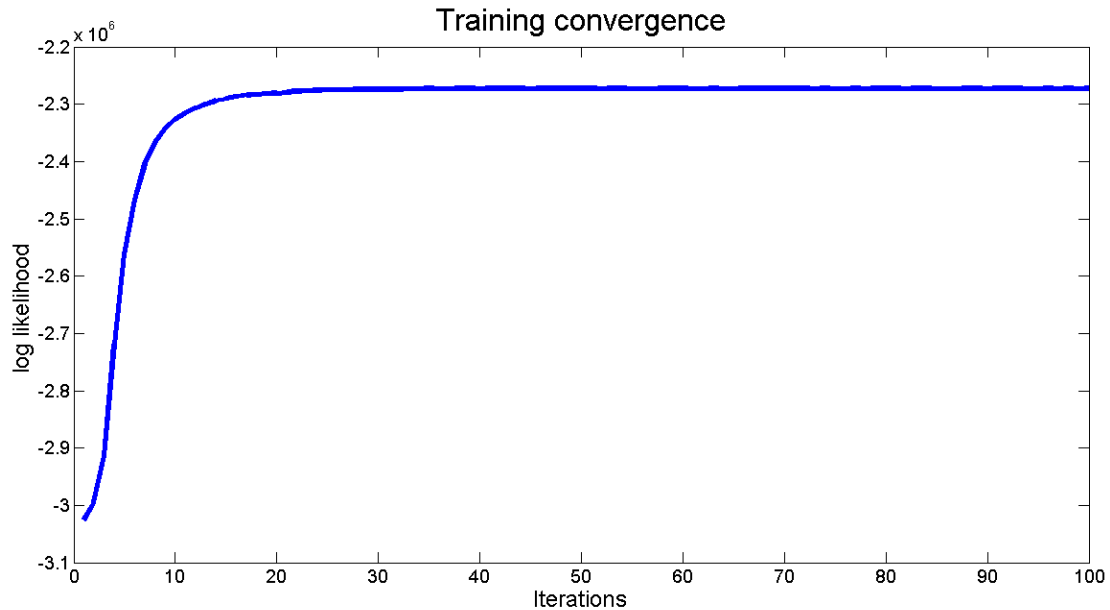


Figure 7.2: Training evaluation by log-likelihood

1. For many iterations (until convergence)
 - (a) For each word $w_{i,d}$ with an assigned topic $z_{i,d}$
 - i. Subtract 1 from counts related to $z_{i,d}$: $N_{j,d}, N_{w,j}, N_j$
 - ii. For $m = 0$ to k (number of topics)

$$p[m] = \frac{1}{C} a_{j,d} b_{w,j}$$
 for $w_{i,d}$ and $z_{i,d}$
 - iii. End
 - iv. Add 1 to counts related to $z_{i,d}$: $N_{j,d}, N_{w,j}, N_j$
 - v. Sample 1 from $p[m]$ (higher values have more chances of being selected)
 - vi. $z_{i,d} = \text{index}(\text{sample})$
 - (b) End
2. End

Training is an unsupervised learning where topics are assigned randomly at the beginning, and gradually updated by sampled topics with an improved probability. The log likelihood function is calculated at each iteration of the sampler, it is generally used to find the maximum likelihood estimate as it is often easier to find using the log likelihood function than it is using the likelihood function. In our case we use it to evaluate the training. Figure 7.2 shows the log likelihood value, after iteration # 15, minor changes occur and training can be stopped at any time. Tweets normally contain undefined words and characters; they are not a clean source of data such as scientific papers, in addition there have been detected many tweets generated by malicious and non-malicious bots operating on the web. For example alerts systems that generate hundreds of tweets with the same text, or marketing systems operating in the same way and publishing a product link repeatedly. The good news is those cases are frequent; therefore a same topic is assigned to them, recognizing those peaks easily. In addition, tweets in several languages were recorded

Topic: 0	Topic: 1	Topic: 2	Topic: 3	Topic: 4	Topic: 5	Topic: 6
earthquake	volcano	landslide	tsunami	flood	tsunami	alat
feel	after	victory	japan	flash	jadiel	untuk
felt	eruption	music	make	warning	unete	membunuh
omg	under	chinese	difference	out	muerte	itu
did	watch	wins	lessons	shit	imagenes	drone
first	careful	no1	fire	scared	fuertes	tim
ako	concerns	program	volunteer	hurricane	@teampgv	pemenangan
there	alaskan	exom	flood	alert	man	juru
nag	raises	landslide"	tornado	phone	inventor	prabowohatta
bed	miles	"it's	@operation	about	bolivian	bicara
			safe			

Table 7.1: Main topics learned

adding more complexity. Among all those cases, real emergencies events exist in the training data. Topics discovered and their main words associated show the result of learning process in table 7.1. The model is robust, raw data with hundreds of unusual characters is entered and coherent topics are discovered. Table 7.1 shows for example natural disasters topics 0, 1, 2, 3, 4 related to earthquake, volcano eruption, landslide, tsunami, and flood respectively. Topics 5 and 6 are non-English topics discovered.

7.4 Inference

After training, the model counts tweets per minute on real time, as shown in figure 7.3. The tweets being counted are not every tweet, but those with keywords related to natural disasters, same as training data set. When a peak of tweets is detected in a certain minute, all the component tweets are recorded and entered in Gibbs sampler in order to perform one iteration. In that iteration, the system selects the best topic for each word based on the learned topics. The distribution of the detected peak over the learned topics is calculated by:

$$\hat{\theta}_{j,d} = \frac{N_{j,d} + \alpha}{N_j + k\alpha} \quad (7.5)$$

In the reduced example at figure 7.3, the system was set to learn only 5 topics, a peak having 475 tweets is detected at 10 : 24 UTC and immediately entered into Gibbs sampler in real -time. As result, the strongest topic (topic 0) and its main words are shown. Topics and words have a probability associated, which helps to analyze the data. After inference, our system continues monitoring new potential peaks.

To verify inference results of our model, a query was performed after the detection using keyword “earthquake”, showing the recent occurrence of a M4.6 earthquake in Indonesia (figure 7.4).

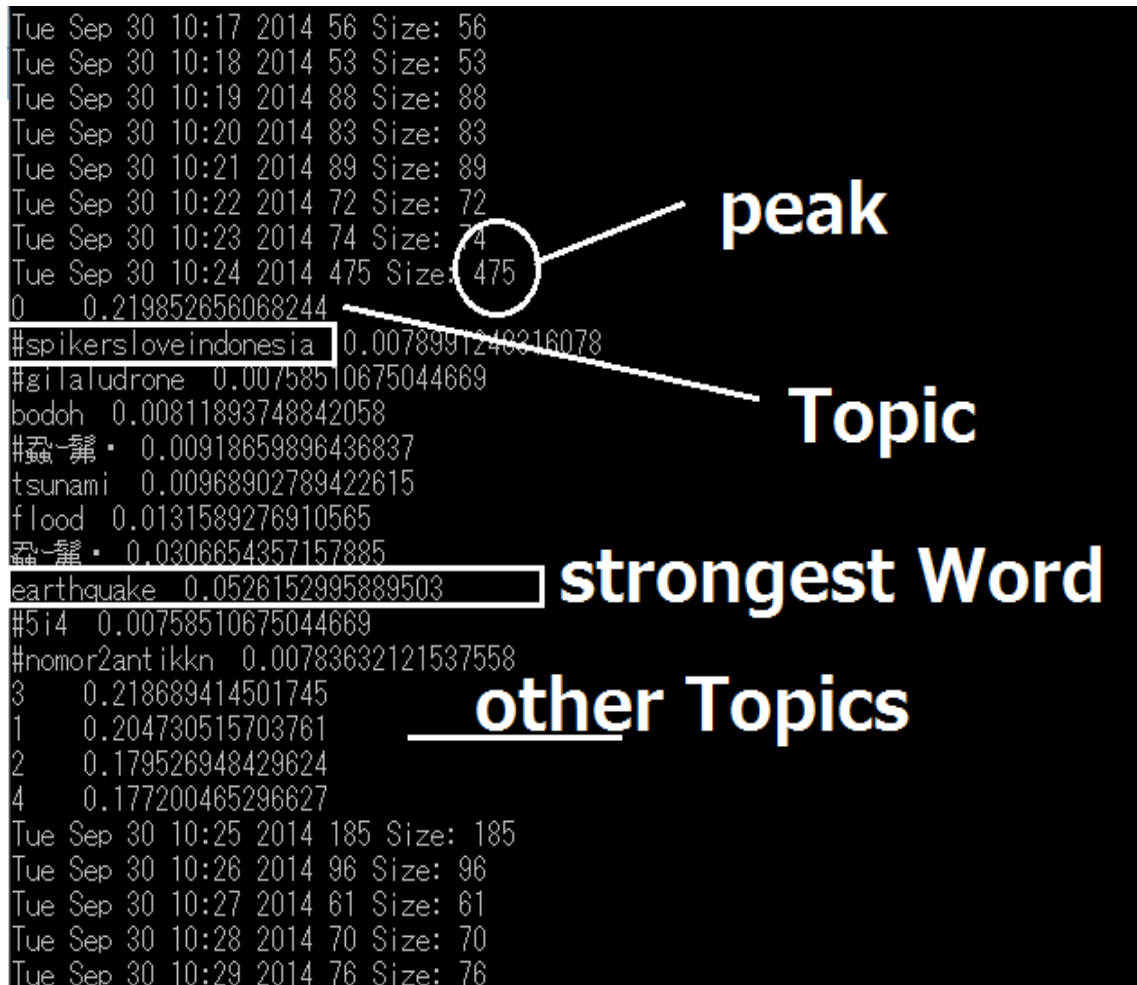





Figure 7.3: Natural disaster topics monitoring system

Results for **earthquake**
Top / All

 **Eko Kurnianto** @ekokurnianto · 39s
Let's pray for Padang **earthquake** and tsunami. September 30, 2009 (with @egakoguci) [pic] — path.com/p/2MPvhh
[View photo](#) [Reply](#) [Retweet](#) [Favorite](#) [...](#)

 **Earthquake Today** @EarthquakesMap · 59s
Mag:4.6 **earthquake** strikes Indonesia. #EarthquakeTrack Images: goo.gl/Hv
[View photo](#) [Reply](#) [Retweet](#) [Favorite](#) [...](#)

 **NC seismic observ.** @NCseismicobserv · 1m
#**earthquake** #지진 #地震 ird2014tdtx (tmp) Tonga Islands mb 5.0 2014/09/30 10:39:10 20.92S 175.58W 10km goo.gl/U8JQk2
Expand [Reply](#) [Retweet](#) [Favorite](#) [...](#)

 **Tabo(雲家・地震警戒)** @taborix · 1m
【海外地震・USGS・M4.5+】 M 4.6, 137km ENE of Amahai, Indonesia dlvr.it/73g2Zb (tabot)
Expand [Reply](#) [Retweet](#) [Favorite](#) [...](#)

 **宏観レンジャー** @kokanranger · 2m
【USGS M4.5+】 M 4.6, 137km ENE of Amahai, Indonesia dlvr.it/73g28d
[Reply](#) [Retweet](#) [Favorite](#) [...](#)

Figure 7.4: Query on Twitter for keyword “earthquake”

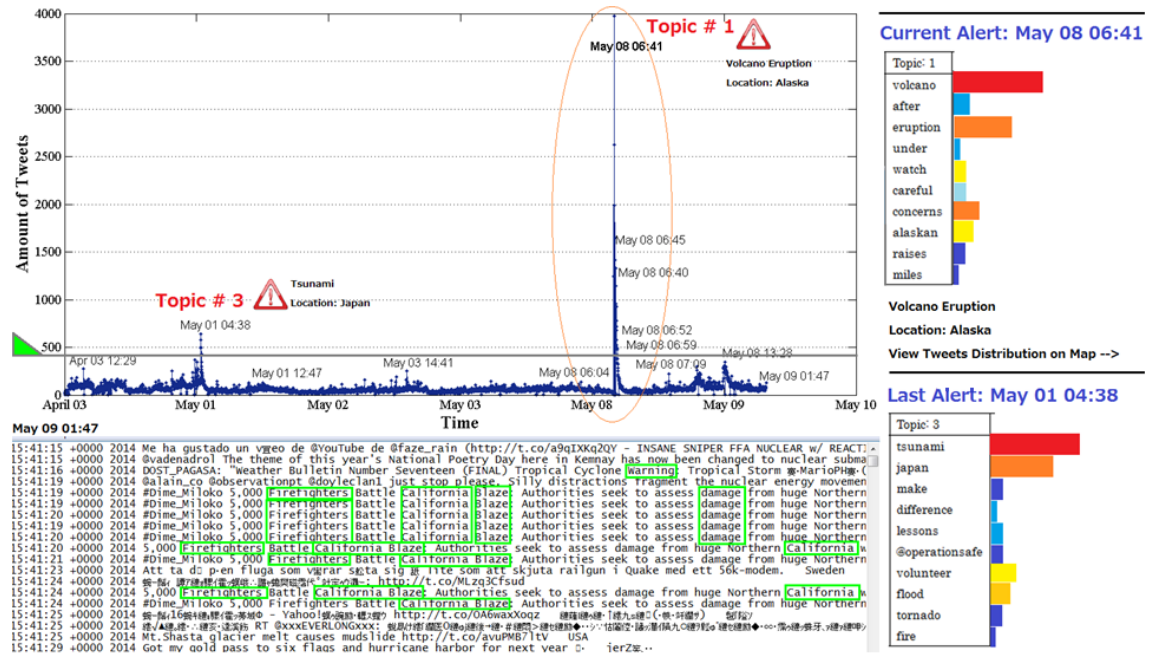


Figure 7.5: Web monitoring system of emergency topics

7.5 Conclusion

It was explored the application of Latent Dirichlet Allocation model in a dynamic environment like Twitter stream. Training is performed as a batch process, but later its results are the core for a real time inference. The model does not consume much memory; it has a unique counter which is initialized every minute. In a certain time, the counter with a value greater than a predefined threshold triggers the inference process. The inference is quick, the time complexity is $k \times w$, k topics times w words within the peak. After showing the inference results, the monitoring system continues tracking new tweets per minute. It is needed to improve the training data set and generate much more topics, in order to cover a wider spectrum of situations that could happen during an emergency. If the system previously has seen an issue, it is highly probable it recognizes the same situation for future events. Also, the inference process should be improved, with a more efficient sampler or by using particle filtering explained in previous chapter. This model can be a starting point for a future web implementation of a monitoring system (figure 7.5).

Chapter 8

Conclusions

In this thesis we have presented unsupervised classification methods for main features extraction and characteristics from natural disaster text sources. Figure 8.1 displays the favorable and unfavorable characteristics we found after their application. We have used PCA to build an application that compresses, transforms and arranges news to uncover data patterns. We have discovered components meanings which serve as summary for the data set. For each news, the application detects the most related component showing its meaning and value through the time. This last characteristic shows how a pattern evolves every time a news is entered into the application, displaying as a result a general view based on main components change. A Self Organizing Map model has been proposed to discover spatial and temporal features from a news data set. SOM has been trained obtaining quantization points and frequent words have been mapped to create a semantic structure. Time dimension has been considered, groups of SOM's units were discovered having a high time dependency. Temporal clusters detection has been possible by the utilization of a time-dependent matrix that stores the transitions from a SOM unit to another; this matrix is the model's perception of frequent events over the time. Although our data set was relatively small, the proposed model was able to discover temporal dependencies. Used matrix also can be modified assigning a memory to it, in a way that it does not remember only the last fired unit, but the last k units fired, expanding its ability. Our application emphasizes the spatial-temporal arrangement of the units and the segregation of the information into separate areas. We believe that results are improved and determined largely by what model is exposed to. Enough input must change and flow continuously through time for a suitable learning. Also, we have obtained a manageable algorithm to select automatically the most important features from a high dimensional data set. The algorithm is straightforward, and is a derivation of the shared application between a linear prediction model besides ridge and lasso shrink methods. It has been demonstrated that different compression ratios can be obtained by the modification of a shrink parameter, high linear prediction accuracy is correlated with low compression ratios, and on the contrary, strong compression harmonizes with a degradation of the prediction accuracy. The same algorithm has been modified to uncover trend topics from a data stream; as far as we know it is the first time a linear regression prediction model is used for detecting main topics on data stream. Coefficients and their evolution over time represents the "lifetime" of main topics detected. Although the amount of features to detect is bounded to a dictionary of 10,000 variables, our algorithm

ALGORITHM	TRAINING	INFERENCE	COMPUTATIONAL CHALLENGES	VISUALIZATION	ISSUES
PCA	BATCH	STRONGER ELEMENTS WITHIN THE TRANSFORMATIONS	COVARIANCE MATRIX	RICH	NO SEMANTIC PROPERTIES
SOM	BATCH	EUCLIDEAN DISTANCE	HUGE AMOUNT OF NEURONS	RICH	WINNER TAKE ALL APPROACH
LINEAR MODEL	ON-LINE	MINIMIZING A PREDICTION ERROR	INVERSE OF A MATRIX	MEDIUM	SIMULATION OF THE OUTPUT
RANDOM FOREST	ON-LINE	MAXIMIZING THE INFORMATION GAIN	DETERMINANT OF COVARIANCE MATRIX, DECISION TREES	MEDIUM	TREES NEED COMPUTATIONAL RESOURCES
LDA	BATCH	MAXIMIZING THE POSTERIOR PDF	TOPICS DETECTIONS TRAINING	MEDIUM	COMPLEX TO DEBUG

Figure 8.1: Unsupervised methods summary of advantages and disadvantages

does not need special characters, like #, to recognize a trend topic. The algorithm is not based on probability neither counting of variables occurrences. It is a pure application of a linear prediction model.

Besides that, we have studied random forest and applied it to uncover main features during an anomalous event. Random forest automatically and with no previous information has organized predominant terms in a ranking list. Variables are the result of maximizing systematically the information gain, and their frequency shows and approximation of how information propagated during the emergency event. An averaged tree has been proposed showing the main test features for each node, from where rules have been detected. We have compared results and the effect of using different dictionaries; the analysis of the words have shown how the emergency event can be described from different perspectives, being necessary the usage of multiple dictionaries of locations, names, an other entities for real situations.

Furthermore, a topic has been modeled, tracked and estimated in a text stream by particle filtering over the time, by matching uni-gram model to a state-space approach. It has been used an importance density function (IDF) equals to the prior probability density function, to make easy in the computation of weights. This IDF has been assumed to be a normal Gaussian distribution which leads to an easy generation of particles. A modified sampling-importance-re-sampling algorithm has been proposed by the addition of a correction term. Better filter estimated values has been obtained by improving the values of particles located far from observation value, obtaining an effect similar to re-sampling.

Finally, we have modeled a real time natural disaster topics detector using Latent Dirichlet Allocation model on Twitter. Training is unsupervised and performed as a batch process, and later topics discovered are the core of a real time inference. The

system does not consume much memory; it has a unique tweets counter which is initialized every minute. In a certain time, the counter with a value greater than a predefined threshold triggers the inference process. The inference is quick, the time complexity is short. After showing the inference results, the monitoring system continues tracking new tweets per minute.

We believe that the findings reported in this paper present relevant models that can be used to provide information to decision takers during difficult times, and can play an important role as large-scale data processing model for discovering patterns in emergencies.

Bibliography

- Ienkaran Arasaratnam, Simon Haykin, and Robert J Elliott. Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007.
- M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.
- David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. 2007.
- Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
- William J Corvey, Sudha Verma, Sarah Vieweg, Martha Palmer, and James H Martin. Foundations of a multilayer annotation framework for twitter communications during crisis events. In *LREC*, page 2012. Citeseer, 2012.
- Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2–3):81–227, 2012.
- Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *Signal Processing Magazine, IEEE*, 20(5):19–38, 2003.
- Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.
- Christopher B Field. *Managing the risks of extreme events and disasters to advance climate change adaptation: Special report of the intergovernmental panel on climate change*. Cambridge University Press, 2012.
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

- Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.
- Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1): 5228–5235, 2004.
- Carlos Enrique Gutierrez, Mohamad Reza Alsharif, H Cuiwei, Mahdi Khosravy, Rafael Villa, Katsumi Yamashita, and Hayao Miyagi. Uncover news dynamic by principal component analysis. *Shanghai, China, ICIC Express Letters*, 7(4): 1245–1250, 2013.
- Carlos Enrique Gutierrez, Mohammad Reza Alsharif, and Katsumi Yamashita. Uncover trending topics on data stream by linear prediction modeling. *IJCSNS*, 14(5):1, 2014a.
- Carlos Enrique Gutierrez, Mohammad Reza Alsharif, Katsumi Yamashita, and Mahdi Khosravy. A tweets mining approach to detection of critical events characteristics using random forest. *International Journal of Next-Generation Computing*, 5(2), 2014b.
- CE Gutierrez, MR Alsharif, R Villa, K Yamashita, and H Miyagi. Data pattern discovery on natural disaster news. sapporo, japan. *ITC-CSCC, ISBN*, pages 978–4.
- CE Gutierrez, MR Alsharif, H Cuiwei, R Villa, K Yamashita, H Miyagi, and K Kurata. Natural disaster online news clustering by self-organizing maps. ishigaki, japan. In *27th SIP symposium*, 2012.
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.
- Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.
- Google Inc. Google person finder <http://google.org/personfinder/global/home.html>. 2011.
- UN ISDR. Unisdr terminology on disaster risk reduction. *Geneva, Switzerland, May*, 2009.
- Andreas M Kaplan and Michael Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business horizons*, 53(1):59–68, 2010.
- Teuvo Kohonen. Self-organization and associative memory. *Self-Organization and Associative Memory, 100 figs. XV, 312 pages.. Springer-Verlag Berlin Heidelberg New York. Also Springer Series in Information Sciences, volume 8, 1*, 1988.
- Marvin Marcus. *Introduction to linear algebra*. Courier Dover Publications, 1988.
-

- Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. Safety information mining-what can nlp do in a disaster-. In *IJCNLP*, pages 965–973, 2011.
- Sebastian Nowozin, Carsten Rother, Shai Bagon, Toby Sharp, Bangpeng Yao, and Pushmeet Kohli. Decision tree fields. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1668–1675. IEEE, 2011.
- Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM, 2008.
- Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.
- Raúl Rojas. *Neural networks: a systematic introduction*. Springer, 1996.
- Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE, 2009.
- Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.
- Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014. URL <http://arxiv.org/abs/1404.1100>.
- Lindsay I Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- Ryan Joseph Tibshirani. *The solution path of the generalized lasso*. Stanford University, 2011.
- Om Vikas, Akhil K Meshram, Girraj Meena, and Amit Gupta. Multiple document summarization using principal component analysis incorporating semantic vector space model. *Computational Linguistics and Chinese Language Processing*, 13(2): 141–156, 2008.
- Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.
- Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 433–443. Association for Computational Linguistics, 2011.
-

Youwei Zhang and Laurent E Ghaoui. Large-scale sparse principal component analysis with application to text data. In *Advances in Neural Information Processing Systems*, pages 532–539, 2011.

Appendix

Unsupervised Learning

In machine learning, the problem of unsupervised learning involves trying to find hidden structure in unlabeled data. Since the input data given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning. Unsupervised learning is closely related to the problem of density estimation in statistics, however, unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Many methods employed in unsupervised learning are based on data mining methods used to pre-process data.

Approaches to unsupervised learning

Clustering

It is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other clusters. It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem. Besides the term clustering, there are a number of terms with similar meanings, including automatic classification, numerical taxonomy, and typological analysis.

Hidden Markov Model

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. In a simple model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.

Blind Signal Separation using principal component analysis (PCA) or other similar method

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

Neural Networks, SOM

SOM is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples, called a map. SOM is useful for visualizing low-dimensional views of high-dimensional data, and it operates in two modes: training and mapping. Training builds the map using input examples (a competitive process, also called vector quantization), while mapping automatically classifies a new input vector. The procedure for placing a vector from data space onto the map is to find the node with the closest (smallest distance metric) weight vector to the data space vector.

Supervised learning turned to unsupervised in our work

Linear Regression

In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable and one or more explanatory independent variables. In linear regression, data are modeled using linear predictor functions, and unknown model parameters are estimated from the data. Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. Linear models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing a penalized version of the least squares loss function as in ridge regression.

Random Forest

Random forests are an ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. The method combines "bagging" idea and the random selection of features, in order to construct a collection of decision trees with controlled variance.

Graphical methods (i.e LDA)

A graphical model is a probabilistic model for which a graph denotes the conditional dependence structure between random variables. They are commonly used in probability theory, Bayesian statistics and machine learning. Generally, probabilistic graphical models use a graph-based representation as the foundation for encoding a complete distribution over a multi-dimensional space and a graph that is a compact or factorized representation of a set of independences that hold in the specific distribution. Two branches of graphical representations of distributions are commonly used, namely, Bayesian networks and Markov networks. Both families encompass the properties of factorization and independences, but they differ in the set of independences they can encode and the factorization of the distribution that they induce. An example of a graphical model is LDA (Latent Dirichlet allocation), in natural language processing, LDA is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.