2018

Iterative Matrix Factorization Method for Social Media Data Location Prediction

Natchanon Suaysom
Harvey Mudd College

Iterative Matrix Factorization Method for Social Media Data Location Prediction

Natchanon Suaysom

Arthur Benjamin , Advisor

Puck Rombach, Reader

Department of Mathematics

May, 2017

Abstract

Since some of the location of where the users posted their tweets collected by social media company have varied accuracy, and some are missing. We want to use those tweets with highest accuracy to help fill in the data of those tweets with incomplete information. To test our algorithm, we used the sets of social media data from a city, we separated them into training sets, where we know all the information, and the testing sets, where we intentionally pretend to not know the location. One prediction method that was used in (Dukler, Han and Wang, 2016) requires appending one-hot encoding of the location to the bag of words matrix to do Location Oriented Nonnegative Matrix Factorization (LONMF). We improve further on this algorithm by introducing iterative LONMF. We found that when the threshold and number of iterations are chosen correctly, we can predict tweets location with higher accuracy than using LONMF.

Contents

List of Figures

List of Tables

Acknowledgments

I am grateful to my thesis advisor, Arthur Benjamin, and my mentor for the summer research and my second reader Puck Rombach, without whom my final report would not be finished.

Chapter 1

Background and Introduction

1.1 Location Prediction on Social Media Data

Social media data is a large collection of text posted online by people in a location in a specified time. Learning from the data allow us to understand many features about people in the location and how they use social media. Social media has been widely studied in different contexts, such as learning how one group of people posted can induce another group of people to post similar things (Meyer et al., 2016), or how twitter specific features such as hashtag can be used for the analysis (Otsuka et al., 2016). In many applications the GPS coordinate of the tweet is needed for the analysis. However, not all tweets location are accurate or known because the users chose not to disclose their location, or the strength of WiFi is not strong enough, which gives less accuracy to the geotags. The twitter data rank the accuracy of its geotags from 0-10 (Inc, 2016). We want to give location to those tweets with unknown location and low level of geotag accuracy. Work on this topic has been done in macrolocation (predicting the city or country of where the tweet is from) (Cheng et al., 2010) or using location from other social media to predict data from Twitter (Papadimitriou et al., 1998). We want to take a more specific approach into predicting microlocation of tweets (in range of less than 1 kilometer). This paper will follows the approach in (Dukler, Han and Wang, 2016) and aims to improve the results of the work in (Dukler, Han and Wang, 2016). To that end, we need to infer the location of tweets from other data which are definitely accurate, which are its content and timestamp. In order to do that, data mining algorithms have to be used because of the large volume of data.

Tweets in the Dataset			
Tweetnumber	Content	TimeStamp	Geotag
319644647033016320	I'm at Steve Nash Sports Club (Vancouver, BC) http://t.co/NNxJYqbr07	2013-04-04 02:56:37	(49.2834,-123.117)
514785031680983040	"See ya later #Vancouver you've been great! But now in the words of #Will-Smith it's welcome to #Miami"	2014-09-24 14:34:50	(40.1971,-123.175)
300641864887988225	An economy based on endless growth is unsustainable *drop the bass*	2013-02-10 16:26:20	(49.2636,-123.186)

Table 1.1 Example of tweets in the Vancouver dataset

1.2 Description of the Datasets

Let T be the set of all tweets in our dataset. An element $t \in T$ consists of the text, location, time, and date that the user posted the tweet. The goal of the project is to do location prediction of a tweet based only on other information. More precisely, the set of tweets T is divided randomly into a training set T_{train}, where the tweets have all possible information and a testing set T_{test}, where tweets have all information except its locations. The division is based on a fraction f, so that $|T_{train}| = f|T|$ and $|T_{test}| = (1-f)|T|$. The goal is to use the information from T_{train} to predict the latitude and longitude of each testing tweet. Let $l_t, \hat{l}_t$ be the real and predicted location of each tweet (as an $\mathbb{R}^2$ vector of latitude and longitude), respectively. Then we want to minimize

$$\|l_t - \hat{l}_t\|$$

for each tweet $t \in T$. To assess how well we do, we set a threshold d (for example, $d = 250$ meters) where we count the number of tweets t such that $\|l_t - \hat{l}_t\| < d$ and calculate the percentage of these tweets that the prediction falls within d, we call that number the **accuracy of prediction within distance** d.

In our datasets that we use to benchmark our algorithm, we have Vancouver datasets having about 4 million tweets and Barcelona datasets having about 1.5 million tweets. An example of tweet from Vancouver datasets are shown in the Table 1.1.

We can see from Table 1.1 that we have the content, time stamp, and

geotag that can be used for the prediction. The data has to be preprocessed into the form of matrix for us to be able to analyze them. A package sklearn is used for this approach because it contains an efficient way to turn the content into matrices. The details on possible approach and algorithms are discussed in the next section.

1.3 The approaches to location prediction

This section contains all the algorithms we have tried so far. Since the amount of data is large, a machine learning technique that requires data storage and runtime that is worse than linear is unlikely to be able to used. Some of those approaches and their limitations are discussed. All algorithms start with preprocessing to turn the content of the tweets into a matrix.

1.3.1 Location Comparison approach

This section involves the idea of trying to predict tweets without doing clustering. The most intuitive idea is to look at all possible positions from T_{train}, denoted by $l_1, l_2, \ldots, l_k$ (in the Vancouver data set, k was found to be about 120,000 positions). For each $1 \leq i \leq k$, we look at the p most popular words ($p = 5$ was used). For each testing tweet t_{test}, we see what words are in it and compare it to the popular words in all locations l_i, and says that the location l_i is the predicted location if it has highest similarity with test tweet t_{test}. This algorithm is summarized in Algorithm 1. After implementing this approach, we found that each test tweet t takes about 1.2 seconds to check, which means that the algorithm becomes intractable if we need to do it for milion of tweets. So we stopped developing this algorithm and looked at other approaches instead. Specifically we should use dimensionality reduction method such as clustering algorithm that would make data size more managable. This leads to the approaches we consider later in the report.

1.3.2 Using clustering for prediction

From the previous algorithm we can see that trying to do the prediction on the whole datasets can take too much time. This motivates the use of clustering techniques to divide the data into different parts before doing the prediction. An algorithm for training data and testing data can be written as follows

- **Training Data:** Suppose that we can divide the tweet set T into disjoint parts $T_1, T_2, \cdots, T_k$ where each T_i talks about the same topic, in other words, the bag of words of T_i contains a number of words that appear in most of the tweets in T_i. The list of those words indicate what each topic in T_i is about.

- **Testing Data:** For each testing tweet t_{test} we can find which T_i that t_{test} should belong to. Then the prediction $\hat{l}_{t_{test}}$ is the most frequent position that appears in T_i.

It can be seen that the prediction may not be accurate if we predict the testing tweet based on the topic, because although the tweets in each T_i have uniform content, they may not have uniform location. Obtaining the topics T_i with uniform location and content is the main problem we need to address. Several techniques are used to obtain that in later sections.

The clustering can be done in a number of ways. However, some of the clustering methods cannot be used because it requires large data storage or long runtime. The clustering techniques we investigated are discussed below.

Spectral Clustering approach

A popular clustering technique on text mining uses spectral clustering. Suppose G is a matrix and each entry G_{ij} is the similarity between the words that appear in two tweets t_i and t_j. Using eigenvalues, G can be decomposed into different clusters. Since these allows pairwise comparisons from all tweets, we expect that the cluster will be accurate. However, the implementation can't yet be done because G would be a matrix of size n^2 where n is the number of tweets, which is in the order of 10^{12}, and is still too large to store on a computer. Thus we consider other clustering algorithm instead.

Nonnegative Matrix Factorization

Let A be the matrix of size $M \times N$ where $M = |T|$ is the number of tweets, and N is the number of all words that appear in all the tweets. Then we define the entries A_{ij} to be the number of times that the tweet with index i contains the word with index j. Suppose we factor A into two matrices W, H with the approximation given in algorithm 2 with a specified number

of topics $k = 100$ (unless specified otherwise, we always use $k = 100$ topics). Then we get that

$$A_{m \times n} \approx W_{m \times k} H_{k \times n}$$

In order to do the clustering, for each tweet in T_{train} represented by row t we put t into the cluster T_i where

$$i = \arg\max_j W_{t,j}$$

This clustering technique will be used throughout the paper and the prediction algorithm is discussed in the next section.

Chapter 2

Nonnegative Matrix Factoring Algorithm for location prediction

In the previous section, we indicated that clusters of tweets are uniform in content, but not necessarily in location. The algorithms that we propose must be able to extract clusters that are uniform in location, so we must have a quantity to measure the uniformity of each cluster. For each topic T_i, we propose the function $D(T_i)$ to do this. Let N_{T_i} be the number of tweets in cluster T_i. And for a tweet t, let $\ell(t)$ be the vectors in $\mathbb{R}^2$ showing the latitude and longitude of tweet t.

- The following value is called Mean Square Distance and is used in (Dukler, Han and Wang, 2016). For topic T_i,

$$D(T_i) = 1 - \frac{1}{N_i^2} \sum_{t,t' \in T_i} \|\ell(t) - \ell(t')\|^2$$

- Let S be the set of pairs of tweets (t, t') where

$$\|\ell(t) - \ell(t')\| \leq d$$

  for some distance d, then let $D(T_i) = \frac{|S|}{N_{T_i}^2}$, denoting the fraction of pairs of points in the cluster T_i that are within distance d from each other.

Note that both values will be between 0 and 1 and both can be used for the improvement of the normal NMF algorithm, as we shall see in the next section.

Another parameter we use is the confidence that a tweet will fall in a given topic. For a testing tweet t, let W_t be the row of tweet containing t, then the maximum entry in that row, which is defined to be

$$C(t) = \max_i(W[t, i])$$

For a prediction of a tweet t_{test} to be accurate, for a tweet t falling in a topic T_i, we want a function $D(T_i)$ and $C(t)$ to be large. These quantities are used as thresholds for the prediction. After using the threshold above, the prediction algorithms are based on of the following three main components.

- Preprocessing the data. This parses the data from the tweet set to the dictionary and matrices. The data structure for the matrix must be a sparse matrix, which saves a lot of amount of data. On average the dictionary has size roughly 30,000 and the tweet set is of size 1,000,000. Since tweets have an average of 10 words, the nonzero entries of the matrix are about $10/30,000 = 0.03\%$. This means that using a sparse matrix would save roughly 99.97% of data storage. Although this will pose a challenge because operations on sparse matrices are limited, it is necessary for us to use it to be able to run the algorithm. The dictionary must be in terms of a hash table, so that the look up time scales logarithmically with the size of the dictionary, which will allow us to do many operations in reasonable time. The full algorithm parses the data into a matrix A and a dictionary as shown in Algorithm 1

Algorithm 1 PREPROCESSING

Input: Set of tweets T
Output: A bag of words matrix A
1: Obtain all the words that appear in the tweets.
2: Remove all words not in the top words list and words that appear altogether less than 10 times. Suppose that the list of words has size w.
3: Create an n-by-w word count matrix A, where the entry A_{ij} is the number of times that tweet i contains the word j
4: Bag of words matrix A

- The next step is to do the actual nonnegative matrix factorization. The algorithm uses both NMF and NLS algorithm found in 2 and 3. We use the NMF Algorithm from sklearn packages, which is optimized for sparse matrices and allows us to specify tolerance on them. Specifically we want to minimize the function

$$f(W, H) = \|A - WH\|_2^2$$

The algorithm produces two matrices W and H, which is then used on the training and testing part.

Algorithm 2 Solving Non-Negative Matrix Factorization

Input: Non-negative matrices $A_{m\times n}$, rank k, tolerance ϵ, maximum iterations T
Output: Non-negative matrices $W_{m\times k}, H_{k\times n}$ so that $A \approx WH$.

$i = 0$ (Number of iterations), W_0 and H_0 are assigned randomly.
while $i \leq T$ and $|f(W_{i-1}, H_{i-1}) - f(W_i, H_i)| \leq \epsilon$ **do**
  Obtain the optimal solution using NLS for $W_{i+1} = \min_{W\geq 0} \|H_i^T W - A^T\|_F^2$
  Obtain the optimal solution using NLS for $H_{i+1} = \min_{H\geq 0} \|HW_i^T - A\|_F^2$
  Increment i
end while
Output W_i, H_i

Algorithm 3 Solving Non-Negative Least Square

Input: Non-negative matrix $W_{m\times k}$ and $a_{m\times 1}$
Output: Non-negative matrix $h_{k\times 1}$ which minimizes $\arg\min_{h\geq 0} \|Wh - a\|_2^2$

$h := 0$
$I := \{1, 2, \ldots, n\}$
$P := \phi$
$w := W^T(a - Wh)$
while $I \neq \phi$ and $w_j > 0$ for some j **do**
  Define $t = \arg\max_j \{w_j : j \in I\}$.
  Remove t from I and add it to P.
  Define $W_P = W$ and change the row of j of W_P to a zero row vector if $j \in I$.
  Let $z = \arg\min_{z_0} \|W_P z_0 - a\|_2$ with a known least square method. Set $z_j = 0$ if $j \in I$.
  while $z_j \leq 0$ for some $j \in S$ **do**
    Define $q = \arg\min_q \{\frac{h_q}{(h_q - z_q)} : z_q \leq q, q \in P\}$
    $\alpha := \frac{g_q}{(g_q - z_q)}$
    $h := h + \alpha(z - h)$
    Move all indices $j \in P$ such that $h_j = 0$ to I.
    Define W_P as in 5. and solve $z = \arg\min_{z_0} \|W_P z_0 - a\|_2$ with a known least square method.
  end while(now $z_j > 0$ for all $j \in P$)
  $h := z$
  $w := W^T(a - Wh)$
end while
Output h.

- The algorithm for training data involves breaking the set of tweets T into clusters $T_1, T_2, \ldots, T_k$. These clusters are then used on the testing

(prediction) part.

Algorithm 4 ALGORITHM FOR TRAINING DATA

Input: Set of training tweets T_{train}, parameter α, numberOfTopics = k
Output: Matrices H, statistics on each decomposed topic
Preprocessing and data decomposition.
1: Preprocessing data from algorithm above to obtain bag of words matrix A_{train}
2: Obtain the location part from the set of tweets.
3: Decompose A using NMF into two matrices W, H with k topics
From matrix W, separate the training tweets into k topics.
4: **for** i = 1:T **do**
5: Evaluate $\arg\max_j(W[i, j])$ to be the topics of tweet i
6: **end for**
7: Keep the result from the for loop in matrix T_{topic}
Calculate the statistics for each topic using data from $\mathbf{T_{topic}}$
8: **for** i = 1:k **do**
9: Calculate the MSD of each topic $\frac{1}{N_{T_i}^2} \sum_{p,q \in T_k} ((x_p - x_q)^2 + (y_p - y_q)^2)$
Where N_{T_i} is the number of tweets in topics i, (x_i, y_i) is the position coordinate in the tweet i.
10: Calculate the most popular position in the tweet in each topic.
11: Keep the results from above in list TopicStat
12: **end for**
13: Output matrix $H_{train} = H$ and list TopicStat

- For the prediction part, we use the results of W, H from the training, the algorithm 5 follows.

Algorithm 5 ALGORITHM FOR TESTING DATA

Input: Set of training tweets T_{test}, bagOfWords A_{test} numberOfTopics = k, matrix H_{train}
Output: Prediction coordinate for each testing tweet and statistics on the prediction
1: Project the text distribution on each topic obtained from H_{train} into the testing tweet by calculating $W_{test} = A_{test} H_{train}^T$, which is the matrix of size $n_{test} \times$ topic
2: Use the peak location in topicStat and assign them to all tweets in each topic.
3: Compare the resulting prediction and report the percentage of prediction within 250m,500m,1KM
Output: The prediction accuracy

2.1 Improving on NMF

We can see from the general approach mentioned earlier that the problem with NMF is that the clusters we got are not necessarily location specific. For example, there can be many location hotspots in topic, which makes the

result less accurate. In (Dukler, Han and Wang, 2016) we fixed the problem by appending the location part to the matrix before doing NMF, which makes the result better, they can be established by the following. Suppose Vancouver is divided into square grid of size 100×100 meter. This gives a total of $r \times c = 180 \times 100$ grid dimension for Vancouver. Define

$$b_{i,j} = \begin{cases} 1 & t \text{ is in grid } i, j \\ 0 & \text{Otherwise} \end{cases}$$

For each tweet t define the following location vector $\mathbf{v}_t$ of size $1 \times rc$

$$\mathbf{v}_t = [b_{1,1}, b_{1,2}, \ldots, b_{r,c-2}, b_{r,c-1}, b_{r,c}]$$

to be the enumerations of $b_{i,j}$ for all i, j. Note that this vector has 1 in one coordinate, and 0 in all other coordinates. Next define the location matrix of size $m \times rc$

$$\mathbf{L} = \begin{bmatrix} \mathbf{v_{t_1}} \\ \mathbf{v_{t_2}} \\ \vdots \\ \mathbf{v_{t_m}} \end{bmatrix}$$

In order to allow us to take into account the location into the algorithm, we modify the input bag of words matrix A to V in the following ways, let $V = [A|L]$, we do NMF on V to get approximations $V \approx WH$ instead to make each cluster from NMF more location sensitive.

One modification that should be made is to adjust the fact that L and A should be taken into account equally, because if $\|L\|_F >> \|A\|_F$ then only location will be taken into account in clustering, or if $\|A\| >> \|L\|$ only text information will be used. This means that we need both matrices to have similar norm. Let α be the adjusting parameter. Define

$$V = [A|\mathbf{L}'] \tag{2.1}$$

where

$$\mathbf{L}' = \alpha \frac{\|\mathbf{A}\|_F}{\|\mathbf{L}\|_F} \times \mathbf{L}$$

, we have that $\|\mathbf{A}\| = \|\mathbf{L}'\|$, so we take into account both text and location in clustering. Note that V is of size $(m + rc) \times n$. In order to modify this input and use this in the prediction, suppose we use k topics, then when we factored V to be approximately WH where W has size $m \times k$, which is the

same, so we don't need to modify it. However now H has size $k \times (rc + n)$, where w is the number of words in bag of words matrix. In order to offer predicted A in the testing set we need to remove the location part, so we only use the first m rows of H. The results of this are shown in (Dukler, Han and Wang, 2016) to be better than regular NMF.

This requires additional modification to the training algorithm. It is shown in Algorithm 6 and 8 and it means that we need to obtain the location matrix and append it to the original bag of words matrix with adjusted parameter. For the testing algorithm, the only modification is for us to remove the location part from H by removing the last rc rows of H.

2.2 Iterative NMF

Here we used LONMF and improve on it. The following algorithm makes use of the advantage of the NMF in doing unsupervised tasks (which means that we don't know what the clusters are; an example of a supervised task would be classifying pictures into digits, where we know what each clusters should be). This algorithm should keep breaking the topic into smaller ones. Given more selection of topics to the testing part, we hope to get higher accuracy in the testing algorithm. We have two metrics on how many iterations we should do the algorithm

- Specifying the number of iterations beforehand, note that if almost all tweets are removed then the algorithm should stop.
- Keep doing the iteration and take those topics that pass the threshold, this means that all the topic we get will be all possible good topics (those that $D(T)$ is large enough for our purpose) from the data set.

From the first metric, we develop the Algorithm 6 and 14.

Algorithm 6 2ND APPROACH TRAINING

Input: T_{train}, and I, number of iteration, α, a percentage threshold for $D(T)$, set of topics L

1: **for** i = 1:I **do**
2: Let V_i in iteration i be defined as in Equation 2.1 from the remaining tweets.
3: Call training algorithm on V_i (NMF and analysis on W_i, H_i), where we only use the first n column of H_i.
4: Remove topics T such that $D(T) \geq \alpha$ from T_{train},
5: Add those removed topics into L.
6: Stop when less than 10% of tweets are left.
7: **end for**

Output: H_is and topic statistics.

Algorithm 7 2ND APPROACH TESTING

Input: T_{train}, T_{test},H_is, L and I, number of iteration

1: **for** i = 1:I **do**
2: Call testing algorithm to each H_i
3: Keep the statistics of the projection
4: **end for**
5: Out of at most $100I$ topics, select T with largest $D(T)$ that the tweet belongs to.
6: Only predict those tweet such that the value $C(t) \geq \beta$

Output: Prediction coordinate and accuracy.

From the second metric, we develop the Algorithm 8 and 9.

Algorithm 8 3RD APPROACH TRAINING

Input: T_{train}, and $I = 0$, number of iteration, α, a percentage threshold, L a set of passing topics.

1: **while** Some topic T_k has $D(T_k) \geq \alpha$ **do**
2: Let V_i in iteration i be defined as in Equation 2.1 from the remaining tweets.
3: Call training algorithm on V_i (NMF and analysis on W_i, H_i), where we only use the first n column of H_i.
4: Remove topics T such that $D(T) \geq \alpha$ from T_{train}.
5: Add those removed topics into L.
6: Set $I = I + 1$
7: **end while**

Output: H_is and topic statistics, I, and L.

Algorithm 9 3RD APPROACH TESTING

Input: T_{train}, T_{test},H_is, L and I, number of iteration, β, a confidence threshold

1: **for** i = 1:I **do**
2: Call testing algorithm to each H_i
3: Keep the statistics of the projection
4: **end for**
5: Out of $100I$ topics, select T with largest $D(T)$ that the tweet belongs to.
6: Only predict those tweet such that the value $C(t) \geq \beta$

Output: Prediction coordinate and accuracy.

Chapter 3

Result from the iterative NMF

3.1 Heatmap Construction

For the analysis to be done on how well the clustering of T_{train} is, we need to be able to look at the distribution of tweets visually on the map to categorize them. Using Google Maps package available in Python, we are able to generate the distribution on Vancouver map. We can also plot only some dates or some time interval if we filter the tweets accordingly. Algorithm 10 shows how this is done. To see what the topic is about, the 5 most popular words are drawn from the topic by calculating the largest 5 entries of row i of H_{train} because H is of size $topic \times words$, so that $H_{i,j}$ represents the probability that topic i contains word j. Then we infer the topic name from those words and its location.

The heatmap shows different clustering behavior of tweets in each topic in the city. From the heatmap, we can roughly divide each topic into following 3 categories, clear, mixed, and random. The expectation of what each iteration in Algorithm will do is also discussed below, which will be the motivation for Algorithms 6 and 8.

Algorithm 10 HEATMAP AND TOPIC ANALYSIS

Input: T_{train}, list of used words V, W, number of topics T, number of words needed w
Draw Heatmap
1: Obtain topics for each tweet from $\arg\max_j W[i, j]$
2: Pick the boundary latitude and longitude for the city.
3: **for** i=1:T **do**
4: Obtain all order pairs of latitude and longitude for tweets in topics i.
5: Use heatmap function of Google Map to plot all latitude and longitude, within the
6: **end for**
Obtain words indicating topic
7: **for** i=1:T **do**
8: Find the indices $j_1, j_2, \ldots, j_w$ of largest w entries of row i of H.
9: From those indices $j_1, j_2, \ldots, j_w$, look at the words in those indices from V.
10: Manually process the words and the heatmap location to get the best description of the topics, if possible.
11: **end for**
Output: Heatmaps and words representing each topic.

1. Clear location, which means that the topic clustered clearly in one particular spot, we expect clear location topics to be removed in each iteration, so that it doesn't get clustered again in later iterations. This is because for each iteration we removed those topics with low $D(T)$, such as topic 1 and 2 in Figure 3.1.

2. Mixed location, which means that the topic can be clearly divided into subtopics, thus showing a few hotspots in the heatmap such as topic 2 in Figure 3.2. We expect the subtopics of them to appear individually in the subsequent iteration, because as the iteration goes on, less tweets are clustered, so it is likely that tweets in mixed topics are broken down into its subtopics.

3. Random Location, which means that the hotspots are distributed throughout the area with no clear pattern such as topic 3 in Figure 3.4. Since this is likely to be from those tweets that are not location sensitive anyway, we don't expect the algorithm to solve this problem. These topics will keep appearing in each iteration until the algorithm terminates. For this type of topic, the prediction can only says that the distribution of locations of tweets having similar content will be similar, but cannot predict accurately in microscale.

We run Algorithm 6 on the whole dataset with 100 topics and in order to assess how well we do, some topics are tracked through each iteration.

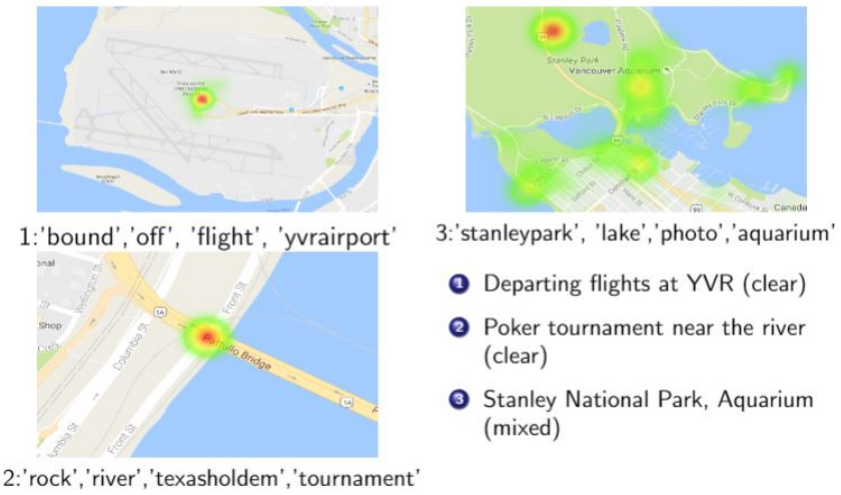

Figure 3.1 The heatmap showing 3 topics from iteration 1. Their categories and changes from previous iterations are shown.

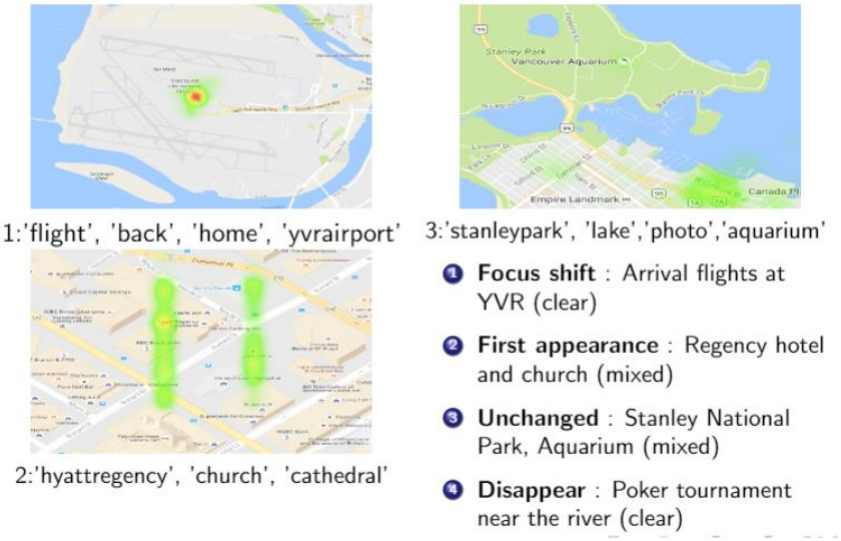

Figure 3.2 The heatmap showing 3 topics from iteration 2. Their categories and changes from previous iterations are shown.

Figure 3.3 The heatmap showing 3 topics from iteration 3. Their categories and changes from previous iterations are shown.

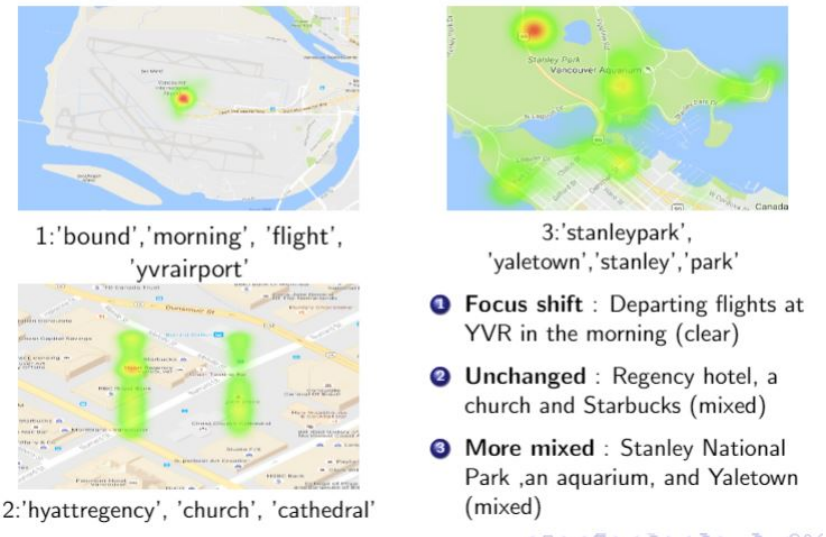

Figure 3.4 The heatmap showing 3 topics from iteration 4. Their categories and changes from previous iterations are shown.

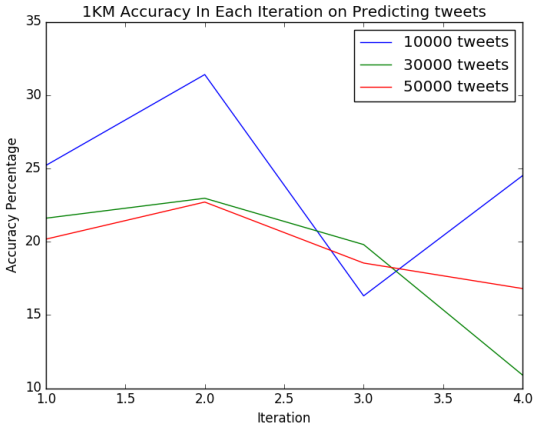

Figure 3.5 The results for predicting tweets location within 1 KM with 4 iterations. The number of tweets we are predicting is shown.

From Figure 3.1, 3.2, 3.3 and 3.4, we can see that during the iterations, some topics changed their focus, disappear, or pick up more subtopics. The change throughout each iteration has more variety than we expected. The fact that the airport topics appear throughout the iterations mean that NMF doesn't capture all the stories about airport in its first iteration. The disappeared topic such as the casino topic is what we expected to see because its heatmap present very clear location. However, the fact that the topic on Stanley Park doesn't get broken is surprising. In this case the algorithm doesn't do what we expected.

Since this only has a few topics, it doesn't represent the whole performance of the algorithm. The next section looks at the tweet as a whole and presents the result in terms of accuracy of prediction.

3.1.1 Accuracy Prediction

After the analysis of heatmap, we did the algorithm for the testing part (14 and 9). The results are shown below.

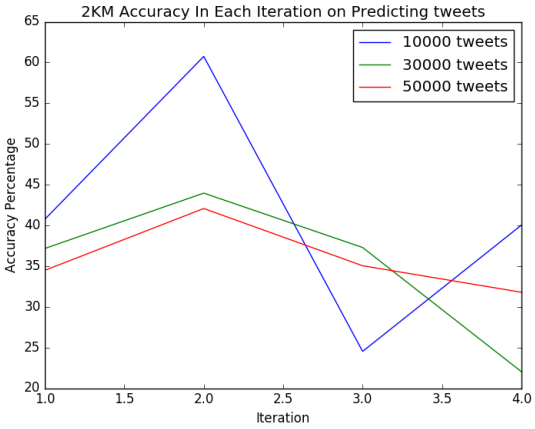

Figure 3.6 The results for predicting tweets location within 2 KM with 4 iterations. The number of tweets we are predicting is shown.

We can see that the second iteration always does better than the first. This is reflected in the heatmap analysis, because a lots of changes in the second iterations are those we expected. The fact that later iterations don't give better results may be from overfitting, and also from the fact that we should be more confident in later iterations, but the algorithm doesn't take that into account. We can adjust the algorithm to take into account the confidence level by increasing the prediction threshold. We investigate the effect of increasing the threshold in different ways in the next section.

3.2 Result from an improved iterative approach

From the second approach, we can see that higher iterations don't necessarily give higher predictions accuracy. After considering the conditions of the algorithm, we can see that the threshold for $D(T)$ and the confidence $C(t)$ are constant in each iteration. This is counterintuitive because as the iteration goes, we should get result that are more clustered and we should have more confidence in them. These suggest that the thresholds should both increase in each iteration. For simplicity, we started by considering linearly increasing thresholds. For this experiment we consider increasing them in three different linear ways for comparison.

- For comparison, a fixed threshold of $D(T) = 0.7$ and $\arg\max_i = 0.9$.

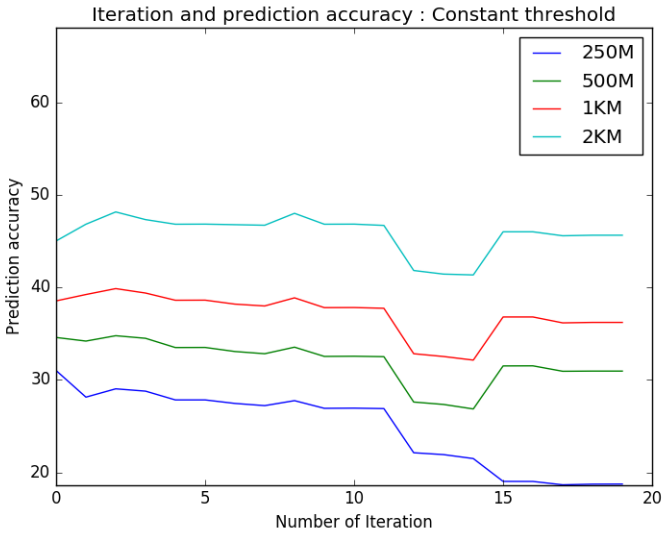

Figure 3.7 The results for predicting tweets location with 20 iterations.

- **slowly increasing threshold**, we use the threshold $D(T) = 0.7 + \frac{i}{200}$ and $C(t) = 0.9 + \frac{i}{200}$.
- **increasing threshold** we use the threshold $D(T) = 0.7 + \frac{i}{60}$ and $C(t) = 0.9 + \frac{i}{100}$.

The results for these thresholds are shown in the figures 3.7, 3.8, 3.9.

It can be seen that when the threshold is constant, the accuracy never significantly increases. For both the slowly increasing threshold and for increasing threshold, the accuracy increases rapidly up to some point then drops off. It can be seen that in either cases, increasing the threshold actually increases the accuracy. It remains to investigate what way of increasing threshold is best for this accuracy prediction. The number of iterations also plays an important role in location prediction. As the number of iterations become too large, at that point only few tweets remain. It is likely that those remaining tweets are in random topics, because those that are not random will not pass the threshold in previous iterations. So that in later iterations only random topics are considered, which explains why the algorithm could have lower accuracy. One way to solve this is to not consider those random topics at all, but that needs to be done with carefully chosen thresholds to make sure that we don't remove tweets that give valuable information.

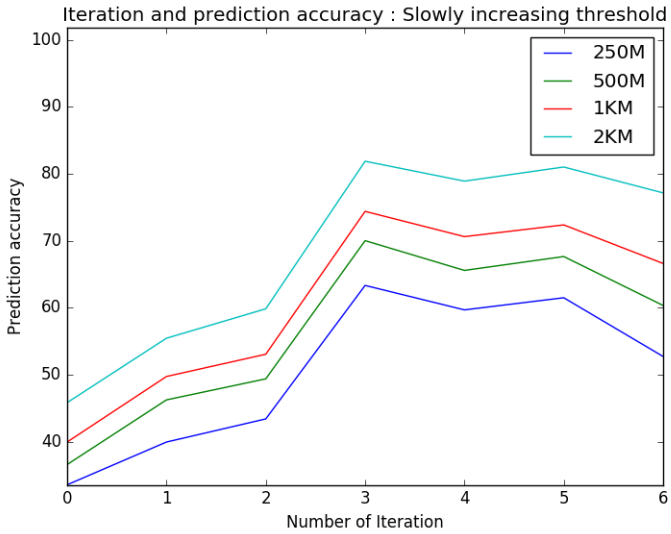

Figure 3.8 The results for predicting tweets location with 6 iterations.

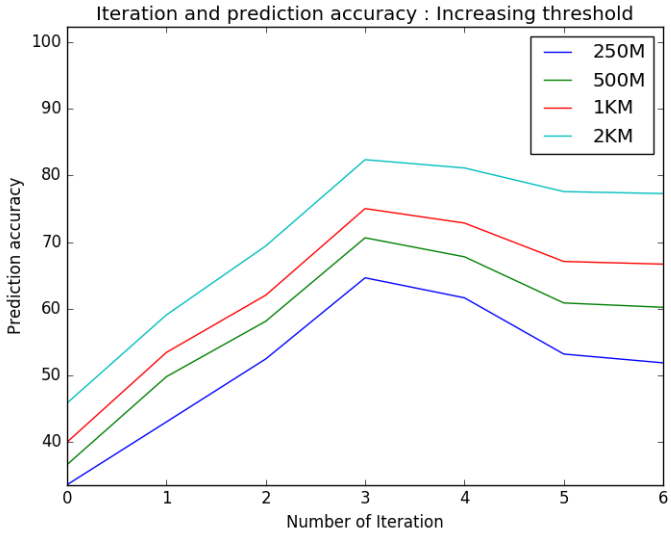

Figure 3.9 The results for predicting tweets location with 6 iterations.

Chapter 4

Additional features

Let f be a feature that can be grouped into N types, $f_1, f_2, \ldots, f_N$. For example, using time would group this into 24 hours, or using days in the week would group this into 7 days. If we found that the tweets in group f_i, defined by $T(f_i)$, happens to be location sensitive or has other important characteristic, then we can make separate prediction on each $T(f_i)$ to get higher accuracy.

4.1 Separate Clustering Approach

One way to do the prediction separately on each f_i is to divide T_{train} into $T_{train}(f_1), T_{train}(f_2), \ldots, T_{train}(f_N)$. We can divide the input to the algorithm into different parts in the beginning. This will change the training and the testing algorithm in the following ways.

Algorithm 11 2ND APPROACH TRAINING

Input: T_{train}, and I, number of iteration, α, a percentage threshold for $D(T)$, set of topics L. Divide T_{train} into $T_{train}(f_1), T_{train}(f_2), \ldots, T_{train}(f_N)$ where tweets in $T_{train,j}$ has property f_j.

1: **for** j = 1:k **do**
2: **for** i = 1:I **do**
3: Call training algorithm on $T_{train}f(j)$ (NMF and analysis on $W_{i,j}, H_{i,j}$)
4: Remove topics T such that $D(T) \geq \alpha$ from $T_{train,j}$,
5: Add those removed topics into L.
6: Stop when less than 10% of tweets are left.
7: **end for**
8: **end for**

Output: $H_{i,j}$s and topic statistics, and L.

Algorithm 12 2ND APPROACH TESTING

Input: T_{train},T_{test},$H_{i,j}$s for all i, j, L and I, number of iteration, Divide T_{test} into $T_{test}(f_1), T_{test}(f_2), \ldots, T_{test}(f_N)$ where tweets in $T_{test}(f_j)$ has property f_j.

```
for j = 1:k do
    for i = 1:I do
        Call testing algorithm to each H_{i,j}
        Keep the statistics of the projection
    end for
end for
Out of at most 100I topics, select T with largest D(T) that the tweet belongs to.
Only predict those tweet such that the value C(t) ≥ β
```

Output: Prediction coordinate and accuracy.

4.1.1 Advantage and Disadvantage

This algorithm is easily implemented and offer more topics for us to consider, thus giving higher accuracy. However, this algorithm has the disadvantage that for an already existing clustering we cannot do the analysis without redoing the clustering. Another approach allows us to do the prediction seperately after clustering, but requires different condition on the features to be met.

4.2 Separate Feature Predictions Approach

This algorithm will fix the disadvantage of approach 1 and also allow multiple features to be used at once. Suppose that the anomaly is very strongly seen in the training set, which means that there exists i such that

$$T(f_i) >> T(f_j)$$

for all $j \neq i$. For example, when people tweets about national holiday on a specific day of a year, then the topic about that holiday will mainly consist of events on that day. Note that this approach will not work well with time, because in general the time in the afternoon will always be more popular than time at night, so even there exists such f_i satisfying the equation above, we still cannot get better prediction.

Algorithm 13 2ND APPROACH TRAINING

Input: T_{train}, and I, number of iteration, α, a percentage threshold for $D(T)$, set of topics L

1: **for** i = 1:I **do**
2: Call training algorithm on T_{train} (NMF and analysis on W_i, H_i)
3: Remove topics T such that $D(T) \geq \alpha$ from T_{train},
4: Add those removed topics into L.
5: Find whether there exists any f_p such that $|T_{train}(f_p)| >> |T_{train}(f_q)|$ for all other $q \neq p$ add f_p into set of anomaly features P.
6: Stop when less than 10% of tweets are left.
7: **end for**

Output: H_is and topic statistics.

Algorithm 14 2ND APPROACH TESTING

Input: T_{train}, T_{test},H_is, L and I, number of iteration, P, set of anomaly features.

1: **for** i = 1:I **do**
2: Call testing algorithm to each H_i
3: Keep the statistics of the projection
4: **end for**
5: Out of at most $100I$ topics, select T with largest $D(T)$ that the tweet belongs to.
6: If there exists any $f_p \in P$ for any topic, predict the testing tweet with property f_p with the data from training tweets having f_p. Otherwise do the prediction normally.
7: Only predict those tweet such that the value $C(t) \geq \beta$

Output: Prediction coordinate and accuracy.

4.3 Date, time, and language as features

Another feature that is available for the prediction is date and time. We expect that when people mention an event or a location, they will do it at the same date or the same time. This motivates us to include that in our prediction.

4.4 Time

It is not surprising that the topics' and users' spatial distribution varies with time. If we can take time into account when predicting locations, we expect that it can improve the results. We used algorithm 11 because it makes more sense to consider tweets in each time interval separately. An example of one of the 100 topics of an iteration shows the effect of time in heatmap distribution. This topic contains topic terms about night markets

and clubs, and thus is expected to be active during night. We observe that the distributions in the daytime and during the night are different.

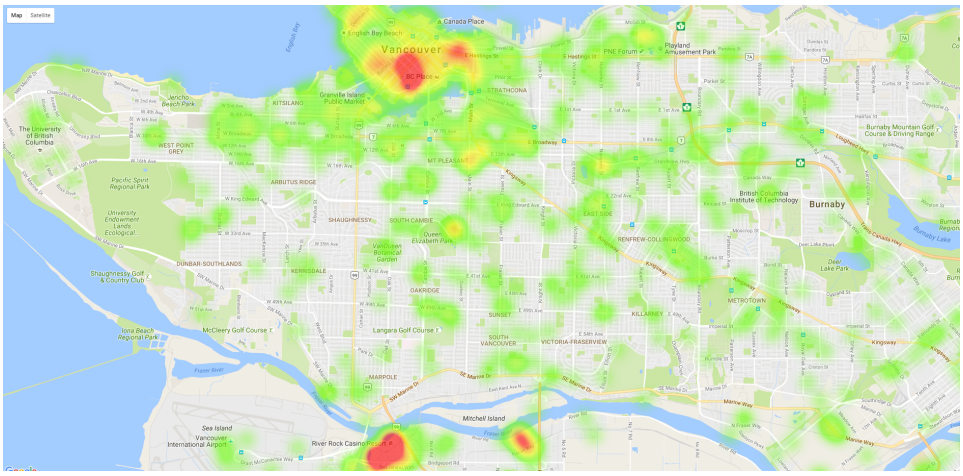

Figure 4.1 Heatmap of Topic 67, Daytime

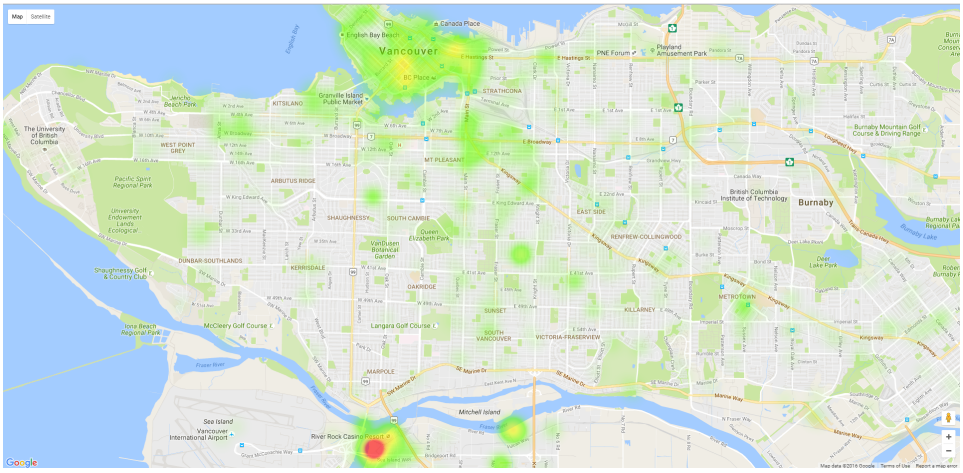

Figure 4.2 Heatmap of Topic 67, Nighttime

Another interesting topic is also shown in Figure 4.3. The topic is about theatres. It appears that Vogue theatre is more popular than Scotiabank theatre during the day, and the opposite happens during the night. The topic, its top words, and its heat map during the day and night are shown in Figure 4.3.

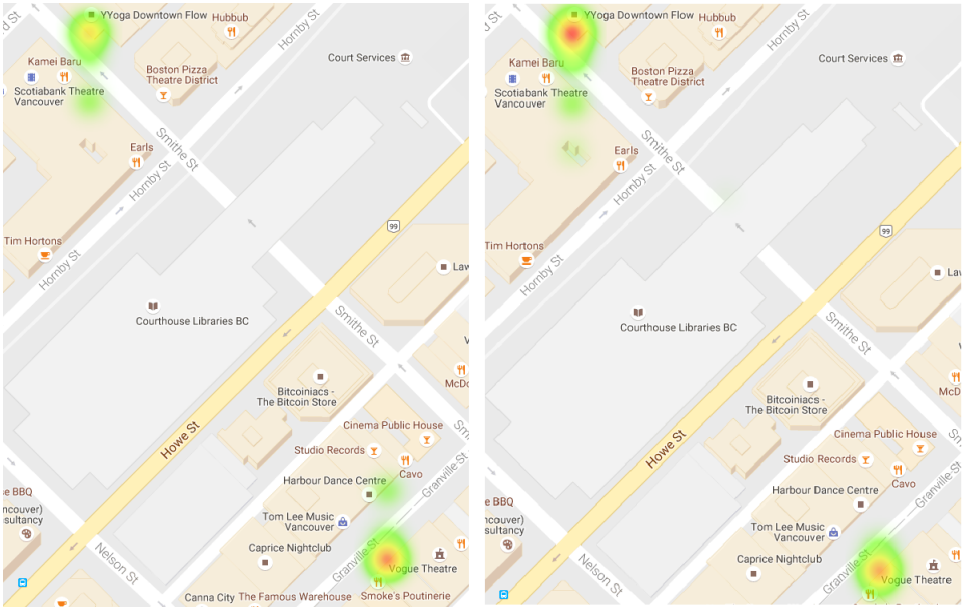

Figure 4.3 Heat Maps Comparison between Day and Night of Movie Topics

This motivates us to use time as a factor in location prediction. An approach we used is let $f_0 = (t_0, t_1)$, $f_1 = (t_1, t_2), \ldots, f_{n-1} = (t_{n-1}, t_n)$ be the feature that each tweet is in that time interval. We calculate the distribution separately for these time intervals. Then, suppose that there is a tweet t that we want to predict its location, we find the topic that it belongs to by finding the maximum probability that the tweet is in a topic (the maximum value of row t in the matrix W from the LONMF), and find what time interval it is in, then compare the results with the actual location of each tweet in the testing data. It can be seen that there is an improvement in the accuracy of location prediction when time effect is taken into account. This approach follows Algorithm 13.

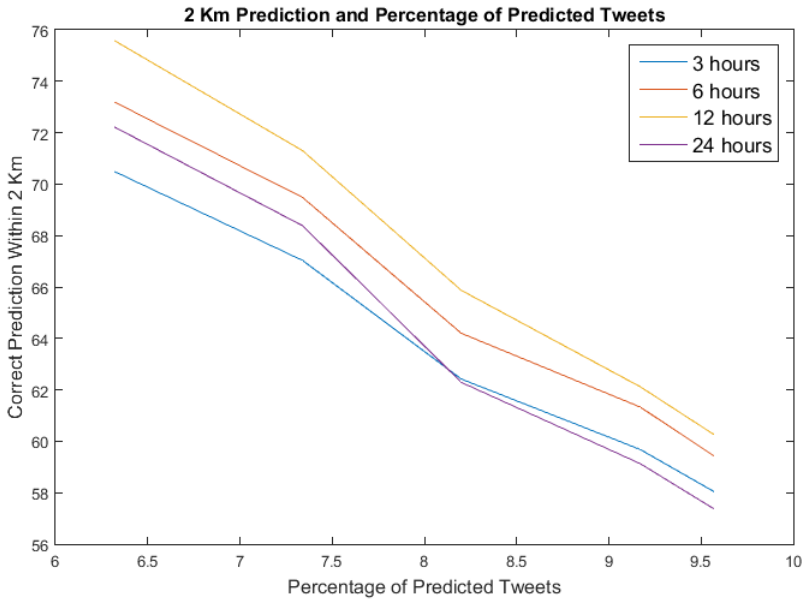

Figure 4.4 Graph showing the percentage of correctly predicted tweets within 2 kilometers in a given time frame

It appears that dividing days into two parts (day and night) gives the best result. However, dividing the days into finer intervals doesn't give a more accurate result. Finer scale can be misleading because few tweets can become a peak for the model. For example, in a three hours interval division, the time from 3 A.M. to 6 A.M. has very few tweets, which means that few tweets from some location could be considered in the model as a peak. This makes the model less effective in finer time intervals. A way to avoid this is to divide the time based on the density of tweets in that interval, so that each interval (t_i, t_{i+1}) contains the same number of tweets. This would avoid sparse time interval to mislead the model. It can be seen that some topics are sensitive to both location and times, which is what gave us some improvement in the accuracy.

4.5 Date

For date we use Algorithm 13 to find the anomaly, which would be the day that most tweets occur to detect some events. The dataset span across 1100 dates from 2012-2015. We expect that when people talk about an event, they

will talk about it at the same date. Suppose that for each topic T, there exists a date d_T such that there are many more tweets in d_T than in other dates. We use constant threshold and exact prediction for comparison between regular LONMF and the modified algorithm shown in Figure 4.5. It appears that almost no improvement to the result. This is possibly because any topic that already has good prediction accuracy doesn't need date to distinguish them.

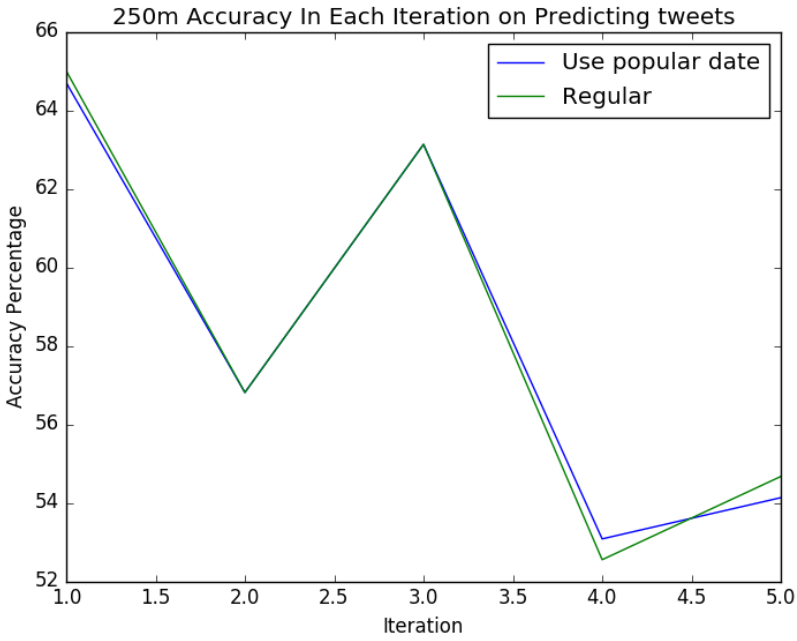

Figure 4.5 Graph showing the percentage of correctly predicted tweets in most popular date

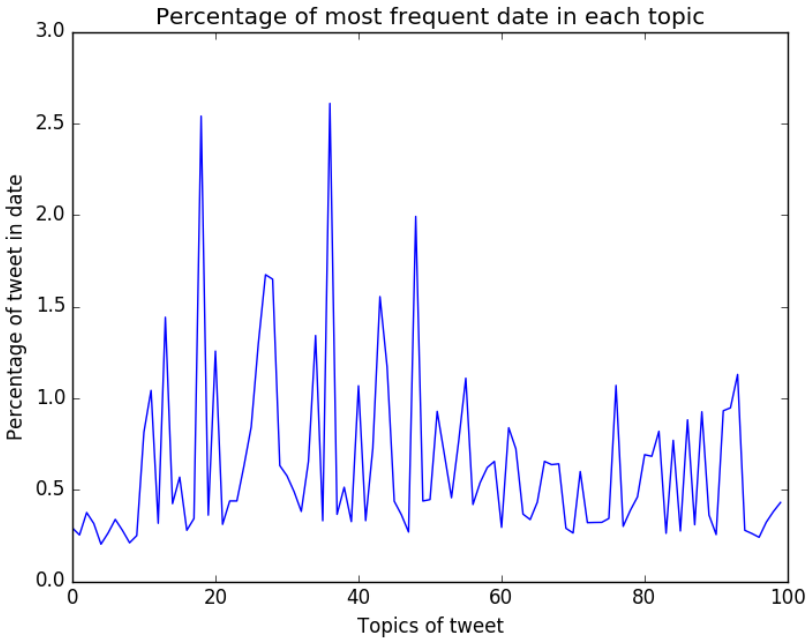

Figure 4.6 Graph showing the percentage of tweets in most popular date

Upon further inspection, we plotted the following graph on Figure 4.6 showing the percentage of the tweets on the most popular date on each topic. It can be seen that the value is very low, which means that if we consider the most popular date separately, very few tweets will change its value.

4.6 Language

Using language for the prediction is similar to using time. We cannot use Algorithm 13 because when we try to detect the most popular feature, in this case the algorithm will give out English, which will not be very useful. We expect to use Algorithm 11 instead, which will cluster all features separately, because we expect that tweets with the same foreign language could have similar location distribution. We use the Python library that allows language detection. However, we found that this could be difficult to do. Since each tweet has only 10-15 words, it is possible that when it mentioned a foreign product or restaurant name, the algorithm will decide that the whole tweet is in another language, making it hard for us to define what language a tweet

should belong too. Moreover, some tweets contain URL and hashtag that is not necessarily meaningful, and cannot be put into a certain language. These obstacles make the results showing that very few tweets are in English, which is incorrect. More appropriate natural language processing for short sentence should be used in the future for language processing.

Chapter 5

Algorithm Review

5.1 Algorithm Optimization

It appears that with iterative method, the time it takes increases linearly with number of iteration. the following graphs show the amount of time taken in each iteration of the algorithm. Since the training part relies on NMF which has a fixed runtime, we will now focus on optimizing the testing part. The time taken for algorithm withour optimization is shown in Figure 5.1, 5.2.

It is known that using **for** loop in the program takes a long time, and for some of them we cannot avoid. This means we need to find the alternative that is faster for them. This entails using matrix operation whenever possible. Another part that requires optimization is the data structure we used. Since our machine is large enough to hold the datasets.

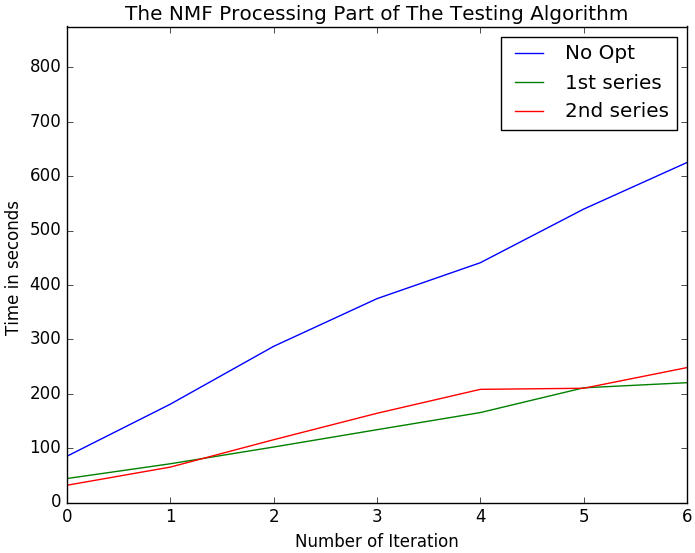

Figure 5.1 Graph showing the increase in speed for each series of optimization in the first part of testing algorithm

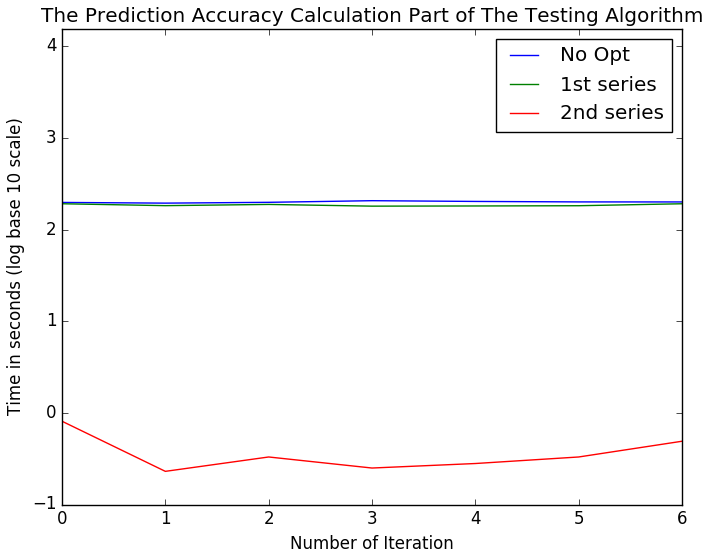

Figure 5.2 Graph showing the increase in speed for each series of optimization in the second part of testing algorithm

1. In series 1, instead of calculating the pairwise distance from the predicted grid to the actual grid using for loop, we did them using matrices operations, and matrix filtering is used so we can calculate which intervals the distance from the previous calculations fall into. In addition, the calculation of location matrix is also vectorized.

2. In series 2, initially we wrote a for loop to find the best topic for each tweet for each iteration, which entails double for loop. We instead store the data into 3-dimensional matrix, A, where the dimensions are iteration, projections and prediction coordinates, and tweets. It takes advantage of the built-in matrix index slicing to get better algorithm performance.

5.2 Algorithm Alternative

5.2.1 Other approaches in location prediction

NMF on more than one feature, in this case location and text, is also studied in other context. Yoo, et.al. studied Nonnegative Matrix Partial Co-Factorization (NMPCF). Yoo et.al. used this technique to separate the music into drum part and the harmonic part. Specifically, for a matrix X, we want to find matrices U_D, V_D for the drum part and U_H, V_H for the harmonic part such that

$$X \approx U_D V_D^T + U_H V_H^T$$

We can apply this approach to our algorithm by using W_T, H_T and W_L, H_L as the text and location part respectively. We can solve this with multiplicative update rule. First we find the gradient of the error function

$$\mathcal{L} = \frac{1}{2}\|A - W_T H_T - W_L H_L\|_F^2 + \frac{\lambda}{2}\|Y - U_L W_L\|_F^2$$

to get that

$$\frac{\partial \mathcal{L}}{\partial W_T} = -AH_T + W_T W_L^T H_L + W_L H_L^T H_T$$

$$\frac{\partial \mathcal{L}}{\partial H_T} = -A^T W_T + H_T W_T^T W_T + H_L W_L^T W_T$$

$$\frac{\partial \mathcal{L}}{\partial W_L} = -AH_L + W_L W_T H_T + W_T H_T H_L$$

$$\frac{\partial \mathcal{L}}{\partial H_L} = -A^T W_L + H_L W_L^T W_L + H_T W_T^T W_L$$

Note that if we write, for any matrix U

$$\frac{\partial \mathcal{L}}{\partial U} = [\frac{\partial \mathcal{L}}{\partial U}]^+ - [\frac{\partial \mathcal{L}}{\partial U}]^-$$

where both $[\frac{\partial \mathcal{L}}{\partial U}]^+$ and $[\frac{\partial \mathcal{L}}{\partial U}]^-]$ are positive.

Then we can use the multiplicative update as in (Yoo et al., 2010) to get the rules

$$U \leftarrow U \odot \frac{[\frac{\partial \mathcal{L}}{\partial U}]^+}{[\frac{\partial \mathcal{L}}{\partial U}]^-}$$

where for any two matrices A, B, $A \odot B$ is an entrywise product of A, B, and $\frac{A}{B}$ is an entrywise division of A, B. The convergence of this is proven in (Yoo et al., 2010).With this result, we can use W_L, W_T to find topics that are both sensitive in text and location, and proceed similarly to the existing algorithm. Unfortunately, our data is large and the above multiplicative rule above require $O(mn)$, where m and n are the dimensions of matrix, which is nonlinear, thus we cannot use our machine to calculate this quickly.

Algorithm 15 NMPCF Training

Input: T_{train}, and I, number of iteration, L_i a set of passing tweets for round i.

1: Set L_1 to be all tweets, which is T_{train}
2: **for** i=1:I **do**
3: Use NMPCF on the appended bag of words and location $[A|L']$ of L_i to get $W_{i,T}, H_{i,T}, W_{i,L}, H_{i,L}$.
4: Calculate most popular location for each topic from $\arg\max(W_{i,T})$ and $\arg\max(W_{i,L})$.
5: Remove those tweets that its location is not in the most popular location for each topic.
6: Add those tweets into L_i.
7: **end for**

Output: H_is ,topic statistics, and all L_i.

In order to apply NMPCF to our algorithm, we proposed the modified training algorithm for location prediction shown in Algorithm 15. Note that we don't need to modify the testing algorithm because we used LONMF only in the training part. The algorithm look at both the factored matrix in location part and in text part, then it calculates the most popular location for each topic. If a tweet has location that is not the most popular ones, it should be reclustered in the next iteration. This makes sure that in the topics

of each iteration, the tweets in the same topic have the same location (or within an acceptable error). This approach has an advantage that it doesn't require specifying the percentage threshold α the same way Algorithm 8 does. With further study on optimized NMPCF on sparse matrix, we could use this algorithm to do location predition.

Chapter 6

Conclusion

6.1 Results and approaches summary

The approaches developed here are improvements to the approaches in (Dukler, Han and Wang, 2016). NMF gives us a good way to cluster the text based on the content, but not necessarily location. LONMF improves this by properly modifying the input matrices. We took a step further and improve LONMF with iterative approach to further investigate subtopics. The algorithm takes advantage of the fact that NMF is a semisupervised task, and repeatedly cluster the tweets to increase the prediction accuracy. The results show that when the threshold is set to be increasing, which follows from the fact that we are more confident as each iteration goes because we have fewer tweets, the accuracy increase up to a number of iterations. This is because the confident threshold has certain limits. Within the iterations, some topics become more accurate, get mixed with other topics, gets removed since it passes the MSD thresholds, or become unchanged. It remains for us to see how many iterations and what values of the parameters should be used for best prediction accuracy, and what patterns the behavior of each topic follows as it moves along in each iteration.

Although we studied different features of the text, such as languages, and the date and time of tweets, we found that only time can be used effectively. Using time works fairly well because many activities of people tweeting in each topic are done only at night or during the day, but more specific time interval division doesn't necessarily yield better accuracy since at late night very few people tweets.

When we used language, we expect that in some parts of Vancouver, a

certain language can be more popular than the others, helping the location prediction. Since most tweet is only about 10-15 words long, and could contain incomplete or incorrect words, the algorithm is not accurate enough to conclusively detect one language. For example, a tweet can be in English but mention a product or restaurant name with other languages, making it harder to confirm one language to the tweets. The algorithm also takes hours to run to classify the languages, because it needs to compare the tweets against large dictionary of foreign words and some words are used in multiple western languages. Another issue is that certain language may not support Ascii or UTF-8 encoding, so we had to filter them out in the algorithm. In future work more flexible natural language processing should be used.

When we used date, we expected that many topics could be centered in some day of the years. However this is not the case because we found that only approximately $0.5 - 1\%$ of tweets in each topic has the most frequent date, so that the increase in accuracy is not guaranteed.

6.2 Further work

Although optimization steps have been taken to make the algorithm faster, more modification to the algorithm itself can be done to reduce computational time. The algorithm will take time proportional to the number of iterations. This can be computationally intensive and we either need to find an algorithm that is equivalent to iterative NMF or try to see if each iteration can use the information from the previous iteration to reduce runtime. For example, when iteration i uses $|T_i|$ tweets with factored matrices W_i, H_i, and iteration $i + 1$ would use tweets $T_{i+1} \subseteq T_i$ with factored matrices W_{i+1}, H_{i+1}. We could uses rows associated with T_{i+1} of W_i and H_i to approximate W_{i+1}, H_{i+1} without doing the entire NMF steps.

The approach in (Yoo et al., 2010) shows that Nonnegative Matrix Partial Co-Factorization (NMPCF) could be used to cluster the tweets both in text and location simultenously. We expected that factored matrices used in Algorithm 15 will be more accurate than the factored matrices in LONMF because it is well studied to work with multiple features matrix factorization. However, the algorithm for solving NMPCF presented in (Yoo et al., 2010) requires dense matrix multiplications, which is too computational intensive for us to use on matrix of large size such as our tweet bag of words matrix. In future work if we can develop NMPCF algorithm that is as effective in

using sparse matrix as LONMF, then we can use the algorithm to do location prediction.

In using additional features, it may be necessary to use multiple features at once. This is the idea that is suggested but not fully implemented in our algorithm. It should also be noted that our algorithm is designed to only use feature division or looking at subset of tweets with most popular feature. However, some features could be more continuous and cannot be analyzed in these ways. For example, when we use date, we may expect certain distribution of tweets in day of the year to happen, not only the most popular day. If we think of tweets distribution on date or time as function, it could follow analytic patterns that is useful to us.

Bibliography

Adomavicius, Gediminas, and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17(6):734–749.

Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Languages in Social Media*, 30–38. Association for Computational Linguistics.

Akaike, Hirotogu. 1998. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, 199–213. Springer.

Anderson, Reid, Fan Chung, and Kevin Lung. 2006. Local graph partitioning using PageRank vectors. *Computer Society, IEEE* 475–486.

Ashbrook, Daniel, and Thad Starner. 2003. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing* 7(5):275–286.

Beale, Mark, Martin Hagan, and Howard Demuth. 2015. *Neural Network Toolbox^{TM}: Getting Started Guide*. The Mathworks, Inc., Natick, Massachusetts.

Bellotti, Victoria, Bo Begole, Ed H Chi, Nicolas Ducheneaut, Ji Fang, Ellen Isaacs, Tracy King, Mark W Newman, Kurt Partridge, and Bob Price. 2008. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1157–1166. ACM.

Bo, Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. *Proceedings of COLING 2012: Technical Papers* 1045–1062.

Burbey, Ingrid. 2011. *Predicting future locations and arrival times of individuals*. Ph.D. thesis, Virginia Tech.

Candès, Emmanuel J, and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics* 9(6):717–772.

Carter, Simon, Manos Tsagkias, and Wouter Weerkamp. 2011. Twitter hashtags: Joint translation and clustering. *Webscienceorg* .

Chang, Hau-wen, Dongwon Lee, Mohammed Eltaher, and Jeongkyu Lee. 2012. @ Phillies tweeting from Philly? Predicting Twitter user locations with spatial word usage. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, 111–118. IEEE Computer Society.

Cheng, Chen, Haiqin Yang, Irwin King, and Michael R Lyu. 2012. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, vol. 12, 17–23.

Cheng, Zhiyuan, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating Twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 759–768. ACM.

Compton, Ryan, David Jurgens, and David Allen. 2014. Geotagging one hundred million Twitter accounts with total variation minimization. In *Big Data (Big Data), 2014 IEEE International Conference on*, 393–401. IEEE.

Dalvi, Nilesh, Ravi Kumar, and Bo Pang. 2012. Object matching in tweets with spatial models. *WSDM 12 Proceedings of the Fifth ACM International Conference on Web Search and Data Mining* 43–52.

Dodds, Peter Sheridan, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *PLoS ONE* 6(12):e26,752.

Dukler, Lu Maples Suaysom, Han, and Wang. 2016. Social media data analysis.

Eisenstein, Jacob, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1277–1287. Association for Computational Linguistics.

Flatow, David, Mor Naaman, Ke Eddie Xie, Yana Volkovich, and Yaron Kanza. 2015. On the accuracy of hyper-local geotagging of social media content. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 127–136. ACM.

Godin, Fréderic, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for Twitter hashtag recommendation. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, 593–596. International World Wide Web Conferences Steering Committee.

Hamerly, Greg, and Charles Elkan. 2002. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, 600–607. ACM.

Han, Bo, Paul Cook, and Timothy Baldwin. 2013. A stacking-based approach to Twitter user geolocation prediction. In *ACL (Conference System Demonstrations)*, 7–12.

———. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research* 49:451–500.

Hecht, Brent, Lichan Hong, Bongwon Suh, and Ed H Chi. 2011. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 237–246. ACM.

Hofmann, Thomas. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 50–57. ACM.

Hong, Liangjie, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the Twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*, 769–778. ACM.

Hong, Liangjie, and Brian D Davison. 2010. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, 80–88. ACM.

Hossjer, Ola, and Christophe Croux. 1995. Generalizing univariate signed rank statistics for testing and estimating a multivariate location parameter. *Non-Parametric Statistics* 4:293–308.

Hu, Bo, and Martin Ester. 2013. Spatial topic modeling in online social media for location recommendation. In *Proceedings of the 7th ACM conference on Recommender Systems*, 25–32. ACM.

Hu, Bo, Mohsen Jamali, and Martin Ester. 2013. Spatio-temporal topic modeling in mobile social media for location recommendation. In *2013 IEEE 13th International Conference on Data Mining*, 1073–1078. IEEE.

Inc, Twitter. 2016. Faqs about adding location to your tweets. URL https://support.twitter.com/articles/78525.

Jurgens, David. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships. *ICWSM* 13:273–282.

Kannan, Ramakrishnan, Grey Ballard, and Haesun Park. 2015. A high-performance parallel algorithm for nonnegative matrix factorization. *arXiv preprint arXiv:150909313* .

Kim, Hyunsoo, and Haesun Park. 2008. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications* 30(2):713–730.

Kim, Jingu, Yunlong He, and Haesun Park. 2014. Algorithms for non-negative matrix and tensor factorizations: a unified view based on block coordinate descent framework. *Journal of Global Optimization* 58(2):285–319.

Koren, Yehuda, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Kuang, Da, Jaegul Choo, and Haesun Park. 2015. Nonnegative matrix factorization for interactive topic modeling and document clustering. In *Partitional Clustering Algorithms*, 215–243. Springer.

Kurashima, Takeshi, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. 2013. Geo topic model: joint modeling of user's activity

area and interests for location recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 375–384. ACM.

Lai, Eric, Daniel Moyer, Baichuan Yuan, Eric Fox, Blake Hunter, Andrea L Bertozzi, and Jeffrey Brantingham. 2014. Topic time series analysis of microblogs. Tech. Rep. CAM14-76, University of California, Los Angeles.

Lawrence, Page, Brin Sergey, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University.

Lee, Kisung, Raghu K Ganti, Mudhakar Srivatsa, and Ling Liu. 2014. When Twitter meets Foursquare: tweet location prediction using Foursquare. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 198–207. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Li, Hao, Baichuan Yuan, Cassidy Mentus, and Michelle Feng. 2016. *Model the development of themes in Twitter*. 285J Report, University of California, Los Angeles.

Li, Wen, Pavel Serdyukov, Arjen P de Vries, Carsten Eickhoff, and Martha Larson. 2011. The where in the tweet. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2473–2476. ACM.

Lu, Hsin-Min, and Chien-Hua Lee. 2015. A Twitter hashtag recommendation model that accommodates for temporal clustering effects. *Intelligent Systems, IEEE* 30(3):18–25.

Mahmud, Jalal, Jeffrey Nichols, and Clemens Drews. 2014. Home location identification of Twitter users. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5(3):47.

Mall, Raghvendra, Rocco Langone, and Johan AK Suykens. 2013. Kernel spectral clustering for big data networks. *Entropy* 15(5):1567–1586.

Mei, Qiaozhu, Chao Liu, Hang Su, and ChengXiang Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of the 15th International Conference on World Wide Web*, 533–542. ACM.

Meyer, Travis, Daniel Balague, Miguel Camacho-Collades, Hao Li, Katie Khuu, P. Jeffrey Brantingham, and Andrea Bertozzi. 2016. A year in Madrid as described through the analysis of geotagged Twitter data.

Otsuka, Eriko, Scott A. Wallace, and David Chiu. 2016. A hashtag recommendation system for Twitter data streams. In *Computer Social Networks*, 1–26. University of PugetSound.

Papadimitriou, Christos H, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. 1998. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 159–168. ACM.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Priedhorsky, Reid, Aron Culotta, and Sara Y Del Valle. 2014. Inferring the origin locations of tweets with quantitative confidence. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & Social Computing*, 1523–1536. ACM.

Rombach, Puck. 2011. Largest component. URL https://www.mathworks.com/matlabcentral/fileexchange/30926-largest-component.

Sadilek, Adam, Henry Kautz, and Jeffrey P Bigham. 2012. Finding your friends and following them to where you are. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, 723–732. ACM.

Schwarz, Gideon. 1978. Estimating the dimension of a model. *The Annals of Statistics* 6(2):461–464.

Vardi, Yehuda, and Cun-Hui Zhang. 2000. The multivariate l1-median and associated data depth. *Proceedings of the National Academy of Sciences* 97(4):1423–1426.

Von Luxburg, Ulrike. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.

Wallach, Hanna M. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, 977–984. ACM.

Wang, Xiaolong, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in Twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 1031–1040. ACM.

Wit, Ernst, Edwin van den Heuvel, and Jan-Willem Romeijn. 2012. âĂŸAll models are wrong...âĂŹ: an introduction to model uncertainty. *Statistica Neerlandica* 66(3):217–236.

Yang, Shuang-Hong, Alek Kolcz, Andy Schlaikjer, and Pankaj Gupta. 2014. Large-scale high-precision topic modeling on twitter. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1907–1916. ACM.

Yin, Zhijun, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. 2011. Geographical topic discovery and comparison. In *Proceedings of the 20th International Conference on World Wide Web*, 247–256. ACM.

Yoo, Jiho, Minje Kim, Kyeongok Kang, and Seungjin Choi. 2010. Nonnegative matrix partial co-factorization for drum source separation. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1942–1945. IEEE.

Zachariah, Dave, Martin Sundin, Magnus Jansson, and Saikat Chatterjee. 2012. Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Processing Letters* 19(4):231–234.