Learning to Detect Information Outbreaks in Social Networks

Jiayuan Ma
jiayuanm@stanford.edu
Stanford University

Xincheng Zhang
xinchen2@stanford.edu
Stanford University

1. INTRODUCTION

This is the information age. Everyday everyone in the world is receiving tons of information on an unparalleled scale, and no one could afford to consume all the incoming information. However, almost everyone is eager to know every ongoing hot topic in the world in order not to be left behind in the age of information explosion. One simple way to resolve this dilemma is to keep one's eyes on some trusted influential nodes in the information diffusion networks, so that one can keep himself/herself updated with the information from these sources. The general problem of detecting outbreaks in networks asks how to select a set of nodes to detect some dynamic diffusion/spreading process over a network.

Many real-world problems can be modeled under this setting. In the domain of weblogs (such as Twitter), bloggers publish (tweet) posts and use hyper-links (retweet) to refer other bloggers' posts and contents on the web. Under this circumstances, we want to select a set of blogs to read so that we can keep ourselves updated with most of the stories (whether it can be linked or direct posts) that propagate over the blogger's network. This is also an example of a outbreak detection.

Previous researches focuses on modeling the information cascades directly on the network structure (see section 1.1). In this family of models, higher-degree nodes are more likely to participate in a cascade, while lower-degree nodes are less likely to do so. Yet in reality, content information also matters. The probability of nodes participating in a cascade is relevant to the type of information they receives. In the blogger's example, an individual blogger may be a professional sports fan who only cares about sports stories/news. In this case, he/she may prefer to reading and spreading (re-tweeting) sports news. And these people are only likely to participate in sports-related information cascades, and this makes them unlikely to be involved in a politics-related information outbreak.

This project presents learning to detect outbreaks, which studies how to incorporate content-related information into normal outbreak detection. We propose a learning framework that allows for modeling outbreak detection with content information, and evaluate our model on real world collaboration network and Twitter network. Furthermore, because we take into consideration the content information flowing on the network, we can use some standardized machine learning techniques to predict if a given specific piece of content information will cause an information outbreak. We will also show some quantitative results on this outbreak prediction task as well.

This paper is organized as follows. In section 1.1, we give the definitions of basic diffusion models where the rest of paper depends upon. In section 2, we give the formulation of our problems. We provide our approach to solving the problem in section 3. We report our empirical results in section 4 and wrap up with section 5 on what we plan to do in future. Due to the page limit, we omit the related work section in our final report. We refer interested readers to our milestone report for related work.

1.1 Basic Diffusion Models

In Kempe et al. [4]'s highly-cited paper, two basic information diffusion models have been generalized: Independent Cascades (IC) model and Linear Threshold (LT) model. Both of them are the most widely-studied diffusion models.

Given a graph $G = (V, E, w)$, where V is a set of n nodes, $E \subseteq V \times V$ is a set of m directed edges, and $w : V \times V \longrightarrow [0, 1]$ is a weight function such that $w_{u,v} = 0$ if and only if $(u, v) \notin E$. We start the diffusion process with an initial set of active nodes S_0, and the process cascades in discrete steps $i = 0, 1, 2, \ldots$. Let S_i denote the set of vertices activated at step i. The whole process stops at t when $S_t = \phi$. IC and LT only differ in how every individual node is activated, and we will explain their differences.

1. **Independent Cascades** (IC): If a node u first become active in step i, it is given a *single* chance to activate each of its inactive neighbor v, with the probability of success being $w_{u,v}$. Each successfully-activated node v will become active in step $i + 1$. Notice that this process is *unrepeatable*: we cannot make any further attempts on the same edge.

2. **Linear Threshold** (LT): An LT influence graph further assumes that $\sum_{v \in V} w_{u,v} \leq 1$ for every u. The dynamics of LT proceed as follows.

   Each node u has a threshold θ_u which is uniformly distributed in the interval $[0, 1]$, which models the uncertainty of individual's conversion threshold.

   The node u is activated when the weighted sum of its activated neighbors v is no less than the threshold θ_u. Mathematically, node u will become active if

$$\sum_{v \in \cup_{0 \leq j \leq i-1} S_j} w_{u,v} \geq \theta_u \tag{1}$$

2. PROBLEM FORMULATION

Given a social network structure $G = (V, E)$, we want to select a subset $\mathcal{A} \in \mathcal{P}(V)$ of nodes in the network such that $\mathcal{A}$ is solution for

$$\begin{aligned} \max_{\mathcal{A} \subseteq V} : \quad & R(\mathcal{A}) \\ \text{subject to} : \quad & c(\mathcal{A}) \leq B \end{aligned} \tag{2}$$

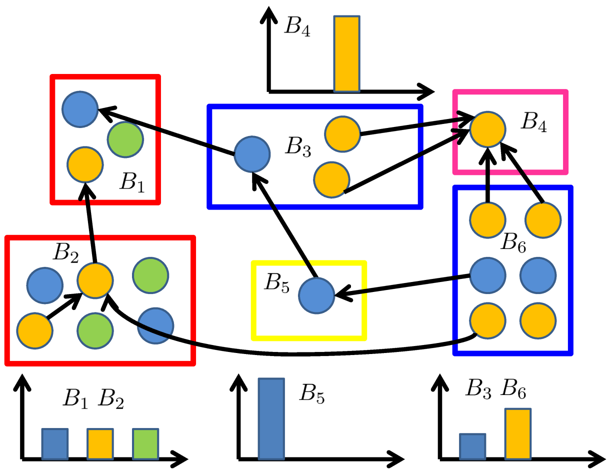


Figure 1: An illustration of blogger network. Each blogger is denoted by a rectangle, and each post tweeted/posted by bloggers are represented by colored circles. All the circles enclosed by the blogger rectangle are the tweets/posts published/referred by this blogger. The links between circles denotes the reference relationships. The color information on each post circle denotes the type of content/topic the post belongs to. The color of the bounding rectangles in this picture means the latent topic distribution for each individual bloggers. The topic distribution of each bloggers are also displayed in this diagram.

where R is a submodular set function defined on $R : \mathcal{P}(V) \longrightarrow \mathbb{R}$ according to some measure (in terms of the number of nodes affected or the reduction of time spent on detecting outbreaks) and $c(x)$ is a non-negative cost function defined on each vertex $c : V \longrightarrow \mathbb{R}$. $c(\mathcal{A})$ is further defined as $c(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s)$, and B is a budget we can spend for selecting the nodes. In this project, we use the cardinality of the seed node set to define of $c(\cdot)$ such that $c(\mathcal{A}) = |\mathcal{A}|$. The above is the original outbreak detection formulation, and we incorporate content information $\mathcal{C}$ into the formulation below. Given a social network structure $G = (V, E)$ where each node $v \in V$ is attached with content information $c_v \in \mathcal{C}$.[1] The probability of information will propagate from node u to node v through edge uv is given by

$$p_{u,v} = Sim(c_u, c_v) \tag{3}$$

where Sim is defined as a function $Sim : \mathcal{C} \times \mathcal{C} \longrightarrow [0, 1]$ which measures the similarity of content information c_u and c_v corresponding to node u and v, and normalizes into a valid probability value. As we can see from the definition, the more similar contents c_u and c_v are, the more likely information will spread between u and v.

The problem boils down to finding a reasonable representation of c_u for each $u \in V$ quantitatively. In our project, we use the content history of a node for quantization. In the case of collaboration network, the content history of a node is all the papers the author node has published before. In the case of Twitter network, the content history of a node is all the posts the node has published/retweeted in the past. The rationale of this formulation is based on the assumption that nodes that share a similar content history are more likely to propagate similar types of information. An illustration is given in Figure 1.

[1]In section 3, we will see that we characterize the content space $\mathcal{C}$ using n-dimensional vector space $\mathbb{R}^n$.

Comparison to previous approaches: In the previous researches, the probability of $p_{u,v}$ is only relevant to the degree information of node u and v. For example, in independent cascade models, the whole graph $G = (V, E)$ is parameterized by a single uniform probability p to each edge of the graph (usually from 1% to 10%) in separate trials. In linear threshold models, each edge from u to v is assigned probability as $p_{u,v} = 1/d_v$ where d_v is the degree of node v. Unlike these traditional approaches, we bias the assignment of $p_{u,v}$ based on the content correlation of node u and v. In our project, $p_{u,v}$'s are no longer simply determined by the network structure, and they are related to the proximity in the content space.

3. MODEL AND ALGORITHM

In this section, we describe our approach of incorporating content information in the task of outbreak detection. Section 3.1 introduces two unsupervised approaches of modeling content space, which are used to aggregate information content into different high-level topics. Section 3.2 describes submodular optimization, which is used in our project as the infrastructure of outbreak detection. Section 3.3 give a description of our approach of predicting the outbreak of a given specific piece of information.

3.1 Content Modeling

In our project, we used two popular unsupervised techniques in natural language processing to model the property of information content that propagates over the network.

3.1.1 Latent Semantic Analysis

For the papers in collaboration network, we use Latent Semantic Analysis (LSA) [3] to carry out content modeling on the this network data. Each node in this network represents an author in the research community, and the content attached to this node is the paper information the author published in the past. To simplify the process, we only use the title of the published papers as the content information in our project.

We filter out a list of stop words (like "a", "the", "of" etc.) from the titles of those papers and treat the rest words in the titles as terms. From above, we know we are just using the titles of those papers as the content information, and we will just build corpus from each document (title) and its terms. Before we actually using the Latent Semantic Analysis, we did the term frequency times inverse document frequency (*tf-idf*) weighting on corpus to decrease the influence of common words appearing in every documents. The *term frequency* $tf(t, d)$ of a term t with regard to document d is simply defined as the raw frequency of t in document d. The *inverse document frequency* is obtained by dividing the total number of documents by the number of documents containing the term t, and then taking the logarithm of that quotient. Therefore, the idf of term t with regard to the set of all the documents D is defined as

$$\mathit{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{4}$$

The next step of LSA is to perform singular vector decomposition (SVD) on the term-document matrix X, where X_{td} is given by $tf(t, d) \times \mathit{idf}(t, D)$. By using SVD, we have

$$X = U \Sigma V^T \tag{5}$$

By only retaining the k largest singular values and their corresponding columns in the U and V matrices, we have a low-rank approximation of the matrix X

$$X \approx \hat{X} = U_k \Sigma_k V_k^T \tag{6}$$

By doing this, we are mapping features in high-dimensional space into a much lower-dimension space (k in this example) while preserving as much content information as possible. When projecting to the lower dimension space, we can represent each title with vectors in $\mathbb{R}^k$ space. In practice, we choose the lower dimension k as 10 to balance between the computational feasibility and information preserving quality. After acquiring the content vector, we employed spectral clustering on those vectors to label the latent topic for each paper title. Therefore, we can quantize each node (author) with a distribution on a latent topic space which we mined from the content information.

3.1.2 Latent Dirichlet Allocation

Though LSA is powerful and informative, but it requires much effort and computing resources to run and experiment with. When very large and noisy data come in, we need a more robust model for modeling the content information. Latent Dirichlet Allocation (LDA) [1] is such a probabilistic model for uncovering the underlying semantic structure of a document collection based on hierarchical Bayesian analysis. The original idea of LDA is to model documents as if they arise from multiple topics, where each topic is defined to be a distribution over a fixed vocabulary of terms. Let us take Twitter network as an example. In Twitter network where each node represents a Twitter user, the content information attached to each node is the tweets (microblogs) the user published or retweeted before. The tweets may have different latent topics, such as sports, finance, politics and entertainment. We want to model Twitter users and their microblogs as a distribution over the latent topics.

LDA can be naturally applied as an enhancement of LSA, and it assigns a document with a distribution of multiple topics. The basic pipeline of LDA is linearly scanning through each word in the document, and initializes by randomly assigning some topics to each word. Afterwards, LDA goes through a iterative improvement process (like Expectation Maximization) until it converges. For each word, it computes the probability of topic given document and word given topic. Then we reassign each word to a new topic based on the maximum likelihood estimation from the previous iteration. Finally, this process converges and each document is assigned a distribution over the latent topic space.

For Twitter data, we follow the same step as in the collaboration network. First, we filter out stop words and http hyperlinks[2]. Secondly, we build corpus using only term frequency of the documents. We utilize a parallel version of LDA [6] implementation from PLDA package to speed up the computation latency.

3.2 Submodular Optimization

Following the same approach in [2], we approximate the submodular optimization problem (2) using a greedy hill climbing algorithm. In outbreak detection, the main operation when deciding whether or not to add a node to the "outbreaker" set is to retrieve a random cascade emanating from that node. For IC models, we toss the biased random coins and remove all edges not for propagation from G to obtain a new graph G' in advance. With this approach, the random cascade from a node v is simply the set of nodes that are reachable from v in graph G', denoted as $R_{G'}(\{v\})$. This can be achieved with a linear scan of the graph G' by Depth-First Search (DFS) where we can obtain $R_{G'}(\{v\})$ for all vertices $v \in V$. In our implementation, we use the lazy forwarding techniques to select remaining seeds. The reason why we can use a priority queue

[2]In this project, we regard the hyperlinks in the tweets as irrelevant information.

Algorithm 1 Submodular Optimization(G, k, p)

```
 1: // $T$ is the number of simulation iterations.
 2: Initialize $S = \phi$, $T = 10000$.
 3: // Build graphs in advance and initialize $T$ heaps.
 4: for $i = 1$ to $T$ do
 5:     Compute $G'_i$ by removing each edge from $G$ with probability $1 - p$.
 6:     Run DFS on $G'_i$ and compute $R_{G'_i}(\{v\})$ for all $v \in V$.
 7:     $H_i$ = MakeHeap($v$, $|R_{G'_i}(\{v\})|$) for all $v \in V$.
 8:     for every $v \in V$ do
 9:         $cur_v^i \leftarrow$ false
10:     end for
11: end for
12: // Lazy forwarding
13: for $i = 1$ to $k$ do
14:     Set $s_v = 0$ for all $v \in V/S$.
15:     for $iter = 1$ to $T$ do
16:         while true do
17:             $v \leftarrow$ DeleteMin($H_i$)
18:             if $cur_v^i$ then
19:                 break
20:             else
21:                 DecreaseKey($H_i$, $v$, $|R_{G'_i}(\{v\}) \cap \overline{R_{G'_i}}(S)|$)
22:                 $cur_v^i \leftarrow$ true
23:             end if
24:         end while
25:         $s_v$ += $|R_{G'_i}(\{v\}) \cap \overline{R_{G'_i}}(S)|$
26:     end for
27:     $S = S \cup \text{argmax}_{v \in V/S}\{s_v\}$
28: end for
29: Output $S$.
```

for lazy forwarding is due to the diminishing return properties of submodular functions. For all sensor placements $\mathcal{A} \subseteq \mathcal{B} \subseteq V$ and node $s \in V \backslash \mathcal{B}$, we have

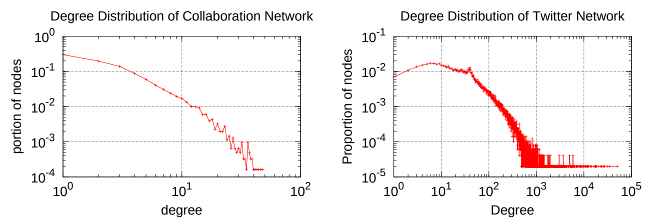
$$R(\mathcal{A} \cup \{s\}) - R(\mathcal{A}) \geq R(\mathcal{B} \cup \{s\}) - R(\mathcal{B}) \qquad (7)$$

Therefore, the $R(\mathcal{S})$ computed in the previous iteration can be used to bound $R(\mathcal{S} \cup \{u\})$ in the current iteration. For more details, we refer the readers to the pseudocode in Algorithm 1. The time complexity of the submodular optimization is $O(|E||V|)$, where $|E|$ is the number of edges and $|V|$ is the number of nodes in G.

3.3 Outbreak Prediction

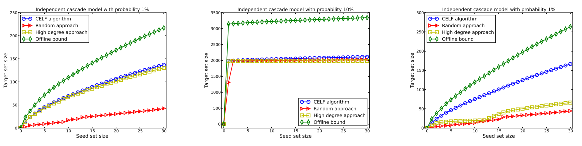
With the content information being modeled, we can predict if a specific given tweet will become popular in the information diffusion network. After running submodular optimization, we have chosen a set of seed nodes (or sensor nodes) as the potential candidate of supernodes. Because information emanating from these seed nodes will result in a large-scale cascade, these nodes are the key to understanding the information diffusion process.

After we finished modeling the content in section 3.1, we have obtained the latent topic distribution for each of the node in the graph. Therefore, for each of these seed nodes we know exactly their content distribution. Consider a scenario where one piece of specific information (e.g. a piece of tweet) come into the network, how likely is it going to trigger a cascade of information diffusions so that this information can reach a large portion of the nodes in the network? We try to answer this question by evaluating the probability of these seed nodes' accepting this piece of twitter information. Because we have modeled the tweets in the same latent topic s-



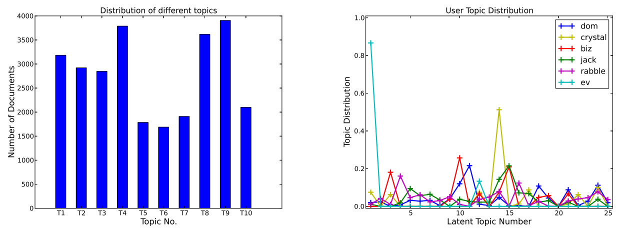
(a) Collaboration network dataset (b) Twitter dataset

Figure 2: Degree distribution of two datasets



(a) IC $p = 0.01$ (b) IC $p = 0.1$ (c) Topic-sensitive

Figure 3: Sensor node placement on the collaboration network



(a) Distribution of 10 topics (b) 6 Different users profiles

Figure 4: Topic distribution

pace as the nodes in the network, we can measure the proximity between the content of tweets and the topic distribution of nodes in the graph. We can use these distances (like anchor distances) to predict if this piece of tweet will be accepted by the influential seed nodes and later forwarded to other nodes in the network.
With the help of machine learning approaches, we use the distances between influential seed nodes and the topic distribution of tweets to train classifiers which predict if a specifically-given tweets will trigger a large cascade of impact (or outbreaks) in the information diffusion network.

4. EMPIRICAL RESULTS

In this section, we showcase some of the experimental results we obtained on two datasets we used in our project. We also summarize the major findings when running various experiments. Furthermore, we will show the prediction accuracies for outbreak prediction tasks.

4.1 Dataset

4.1.1 Collaboration network data

By using the citation network in KDDCup 2003, we wrote a python crawler that requests the arxiv API and downloads all the paper titles, abstract and authors into our local database. Part of these information is later used as the content information. We treat all the names with the same first letter of given name and same last name as the same author. As the paper[4] pointed out, this approach won't affect the result. We show a plot of degree distribution for this collaboration network in Figure 2(a). There are $8,179$ nodes and $26,140$ edges in this network.

4.1.2 Twitter network

We are using the Twitter tweets data from SNAP and the Twitter social graph data from [5]. Instead of using the whole Twitter social graph, which is prohibitively large for computation, we downsample a subset of the original network for our experiments. The sampling process is simply a 1-step breadth first search (BFS) starting from a list of celebrity nodes provided in [5]. We use the subgraph induced by all the traversed node in the BFS. There are $50,011$ nodes and $3,954,516$ edges in this network, and the corresponding degree distribution is in Figure 2(b).

4.2 Results on the Collaboration Network

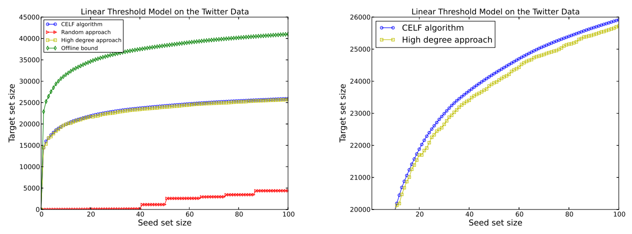
In this section, we are showing results on the collaboration networks. Figure 3 shows the results of running our implementation of submodular optimization on the collaboration network. Here we use the Information Cascade (IC) model with probability $p = 0.01$ and 0.1 respectively. We run the simulation 1000 times to report the results, and we compare the results with the strategies of selecting random nodes (Random Approach) and high degree nodes (High Degree Approach). We also included in our plot the curve of the maximum possible theoretical upper-bounds (Offline Bound), which is simply $|R(S)|/(1 - 1/e)$. As can be shown in the plots, our results are very similar to the findings in [4]. In Figure 4(a), we show the topic clustering results on the collaboration network on high energy physics theory. The topic distribution is quite well-balanced through all the papers we crawled from the arxiv. In Table 1, we also demonstrate some interesting clustering results on the paper titles. The key words part are extracted manually from the titles that are aggregated under the same topic label. In Figure 3(c), we give the result after adding content information into outbreak detection. Interestingly, we observe that compared to Figure 3, the outbreak detection is more efficient in a sense that we have more activated nodes with the same size of seed nodes.

4.3 Results on the Twitter Network

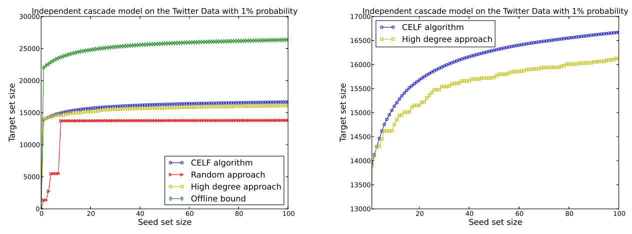
We preprocessed the tweets data from SNAP with LDA, which assigns each documents (tweets) a topic distribution. After that, we built user profile base on the document they authored. We chose the social sensors (users) based on the submodular optimization framework, but with an enhanced-by-topic-similarity weighted probabil-

Keywords	**Paper Titles**
Gravity	The Shape of Gravity Gravity in the Randall-Sundrum Brane World
String	Four Dimensional String Triality Enhanced Gauge Symmetry in String Theory
Geometry	Matter from Toric Geometry Compactification, Geometry and Duality: N=2
Symmetry	Second-Quantized Mirror Symmetry Mirror Symmetry is T-Duality

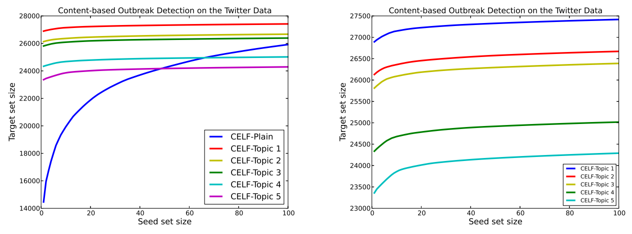
Table 1: Examples of topic modeling



(a) LT Model on Twitter



(c) IC Model on Twitter



(e) Topic-sensitive results

Figure 5: Sensor node placement on the Twitter data

ity graph and a plain linear threshold model graph. We fetched tweets which are retweeted by tweetmeme account, which is outbreak as positive example and manually select some tweets which are not trending as negative example. Finally, we built a discriminative SVM classifier by using those sensors as feature vector. We trained the classifier and tested if the classifier can predict which tweets are to outbreak. In Figure 5, we plot the results for running outbreak detection on the Twitter network using independent cascade model and linear threshold model. We enhance the edge probability of the graph and give it as an input to the submodular optimization algorithm. We calculate the intersections between users under one topic and we reward that overlap. The user pairs without interest similarities will have a default edge probability 0.01 if they have a follow relation. For each topic, we generate a different weighted graph and feed it as input to optimize. We noticed that we actually choose different sets of users for different topics since each topic will enhance the edge probability of different user pairs. The plot of this set of results can be found in Figure 5. We then train an SVM classifier based on the distance of user profile to a tweets to see if we can distinguish those tweets with higher probability to outbreak. We achieve an approximately 70% successful rate and we didn't observe a big improvement when we add content information to choose the different set of sensors. The results are summarized in Table 2. An example of topic distribution for different users is shown in Figure 4(b).

	Linear Kernel			
Seed Node Number	10	20	30	40
Plain Placement (Baseline)	51.8%	69.8%	72.2%	72.8%
Content-based Placement 1	63.6%	68.5%	69.0%	69.3%
Content-based Placement 2	59.4%	67.7%	71.3%	69.2%
Content-based Placement 3	53.8%	59.9%	70.9%	68.8%
Content-based Placement 4	52.7%	58.9%	70.4%	**74.4**%
Content-based Placement 5	52.9%	66.5%	71.3%	75.0%
	RBF Kernel			
Seed Node Number	10	20	30	40
Plain Placement (Baseline)	64.5%	71.3%	73.1%	72.4%
Content-based Placement 1	66.9%	64.7%	70.6%	74.1%
Content-based Placement 2	62.7%	68.4%	71.6%	72.4%
Content-based Placement 3	62.9%	68.1%	72.6%	74.0%
Content-based Placement 4	57.5%	64.4%	71.0%	72.5%
Content-based Placement 5	54.8%	72.0%	74.0%	**75.1**%

Table 2: Accuracy of predicting the outbreak of individual tweets

5. FUTURE WORK

In future, we are going to run experiments on the larger dataset, which is much harder to tackle in our current framework. Also, we'd like to research on an automatic way of tracking the popular tweets. Since twitter data does not contain the retweet amount and comment amount information, it is hard to determine if a tweet is trended. Finally, we'd like to extend our current model to incorporate more interesting features (like diffusion time, user preferences, etc).

6. REFERENCES

[1] D. Blei and J. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

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

[3] S. Dumais, G. Furnas, T. Landauer, S. Deerwester, S. Deerwester, et al. Latent semantic indexing. In *Proceedings of the Text Retrieval Conference*, 1995.

[4] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[5] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[6] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology, special issue on Large Scale Machine Learning*, 2011.