Automatic Summarization of Events from Social Media

Freddy Chong Tat Chua*
Living Analytics Research Centre
Singapore Management University
80 Stamford Road, Singapore
freddy.chua.2009@smu.edu.sg

Sitaram Asur
Social Computing Lab
visited Hewlett Packard Research Labs
Palo Alto, California, USA
sitaram.asur@hp.com

ABSTRACT
Social media services such as Twitter generate phenom-
ena volume of content for most real-world events on a daily
basis. Digging through the noise and redundancy to un-
derstand the important aspects of the content is a very
challenging task. We propose a search and summarization
framework to extract relevant representative tweets from an
unfiltered tweet stream in order to generate a coherent and
concise summary of an event. We introduce two topic mod-
els that take advantage of temporal correlation in the data
to extract relevant tweets for summarization. The summa-
ization framework has been evaluated using Twitter data
on four real-world events. Evaluations are performed using
Wikipedia articles on the events as well as using Amazon
Mechanical Turk (MTurk) with human readers (MTurkers).
Both experiments show that the proposed models outper-
form traditional LDA and lead to informative summaries.

Categories and Subject Descriptors
H.3.3 [Information Systems]: Information Search and Re-
trieval—Information filtering

General Terms
Experimentation, Measurement

Keywords
Social Media, Summarization, Twitter, Topic Models

1. INTRODUCTION
Social media services such as Twitter and Facebook pro-
vide rapid access to the dissemination of opinions and news.
When a news-worthy event occurs in the real world, Twitter
users instantaneously post numerous tweets detailing all as-
psects of the event. Due to these real-time updates, there is
a flood of information propagated through these networks.
By closely monitoring these streams of information, prior
research have shown that it is possible to detect real world
events from Twitter [21, 23, 24, 29, 30]. An event refers
to any concept of interest that gains the attention of the
populace. Examples of real-world events range from global
catastrophes such as earthquakes [23], political protests or
unrest [30], to launches of new consumer products.

The easiest way to extract tweets related to an event is
through a search query. However, for popular events, this
typically results in a significantly large stream of tweets,
which makes the task of understanding the aspects of the
event and the opinion of people, a difficult and mostly futile
task. It has been observed that, despite the high frequency,
the actual information content in the tweet stream is fairly
limited [7, 25]. This is due to the fact that several of the
tweets contain redundant information. Also, many of the
tweets that are returned by a search query are not relevant
to the event. This is due to ambiguity in the search keywords
and the noise prevalent in social media. In this paper, we
address this problem of summarizing a targeted event of
interest for a human reader by extracting the most repre-
sentative tweets from the unfiltered tweet stream for the
event.

Problem definition. Formally, we define our problem as
follows. Given an event of interest e and an unfiltered tweet
stream D, the task is to extract K number of tweets from D
to form a summary S_e, such that each of these tweets d ∈ S_e
adequately covers different aspects of the event e, where K
is a choice of parameter that the human reader may choose,
with larger values of K yielding longer summaries.

Challenges. The above problem definition leads to the
question of how we can measure different aspects of the event
e from D. By aspects, we mean the features of the event
that serve as the main discussion points on social media.
For example, in the case of a product launch, aspects might
include the date of the launch, the features of the product,
the price and initial reviews of its performance. Several chal-
enges arise when we attempt to perform basic text analysis
on tweets. 1) Words are often misspelled in tweets which
means that we cannot use a dictionary or knowledge-base
(Freebase, Wikipedia, etc.) to find words that are relevant
for e. 2) Many tweets in D are the result of noise and are ir-
relevant to e, causing unnecessary computation on majority
of the tweets. 3) Tweets, by their very nature, are short and
that causes poor performance when we apply unsupervised
learning techniques that have been developed for traditional
text analysis.

*This work was completed when the author visited HP Labs.
A naive solution for the problem of extracting relevant tweets would be to apply a standard topic model on the unfiltered tweet stream $D$ to obtain a set of topics $Z$ that can be inspected by the human reader. The event of interest $e$ may emerge as one of the topics $z_e \in Z$ found after topic modeling. Then using the words ranked highly in the topic $z_e$, we can obtain a set of words related to the event which addresses the first challenge. The tweets which are found to have low probability in the discovered topic can be discarded as irrelevant to $e$ which will address the second challenge. However, the problem of such an approach is that there is no guarantee we can find topic $z_e$ for event $e$ regardless of how many topics we use.

To overcome the problems of this naive approach and address the aforementioned challenges efficiently, we propose a framework that performs search and summarization in a bootstrapping manner.

**Figure 1: Framework for Search and Summarize**

Our Search and Summarize framework (shown in Figure 1) proceeds by first applying a simple keyword-based query $Q$ on the unfiltered tweet stream $D$ to obtain an initial subset of relevant tweets $D^1_e$ for the event $e$. We propose two topic models that can then be applied on $D^1_e$ to obtain topics that are highly relevant to the event. Using the discovered topics, we subsequently uncover other tweets $D^2_e$ from $D$ that are relevant to the event $e$ but may not contain the keywords given by the query $Q$. The combined set of tweets $D_e := D^1_e \cup D^2_e$ is then used to refine the model and find more aspects of the event $e$.

The main contributions of this paper are as follows:

1. Our analysis of twitter data revealed that the content of tweets for an event $e$ is typically strongly related to other tweets about the same event written around the same time. That is, given three tweets $d_1, d_2, d_3 \in D_e$, that are written respectively at $t_1, t_2, t_3$, suppose $t_1 < t_2 < t_3$, then the similarity between $d_1$ and $d_2$ will be higher than the similarity between $d_1$ and $d_3$.

2. Based on this observation, we proposed a Decay Topic Model (DTM) which learns improved posterior knowledge about tweets $d \in D_e$, written at a later time, given the prior knowledge of earlier tweets. The importance of this prior knowledge with respect to each topic $z$ decays as an exponential decay function with two parameters - the actual difference in time between the tweets and a $\delta_z$ parameter for each topic $z \in Z$.

3. By assuming that the time associated with each topic $z$ is distributed with a Gaussian distribution $\mathcal{G}_z$, we infer the decay parameters $\delta_z$ using the variance of the Gaussian distributions. That is, suppose a topic $z$ has a large time variance, then it implies that the topic “sticks” around longer and should have a smaller decay, while topics with a smaller time variance lose their novelty quickly and should have a larger decay. By adding Gaussian components to the topic distribution, we obtain the Gaussian Decay Topic Model (GDTM).

4. Based on these two models, we propose a framework for solving the extraction and summarization problem for events from a social media stream.

5. We perform a qualitative and quantitative evaluation of these models on the summarization of four real-world events and demonstrate that the use of temporal correlation facilitates the generation of concise and relevant summaries. Both our methods were found to outperform traditional LDA for this purpose with GDTM providing the best overall performance.

The paper is organized as follows. In Section 2, we discuss some related work in the area of summarization and topic modeling. In Section 3, we describe our framework in detail. Section 4 details our comprehensive evaluation on four real-world events. In Section 5, we present the conclusions from our paper.

2. RELATED WORK

2.1 Text Summarization

Nenkova and McKeown have reviewed an extensive survey of text summarization techniques [19]. According to them, summarization systems perform three successive independent steps to summarize a given target text: 1) Using an intermediate representation for the target text which captures its key features, 2) Using the intermediate representation to assign scores for individual sentences within the text and 3) Selecting a set of sentences which maximizes the total score as the summary for the targeted text. They have summarized the list of intermediate representations into several broad categories such as the use of topic signatures [9, 15, 16], the use of word-counting frequency approaches [13, 20], latent space approaches using matrix factorization [12], or Bayesian approaches [6, 10, 14, 26].

Ganesan et al. have proposed the generation of abstract short text summaries from text [11]. They first obtain lists of n-grams (minimum of n is 2) from the raw text and generate a score for each n-gram based on its representativeness and readability. Subsequently, optimal n-grams are chosen for summarization. In our event summarization, we follow the traditional approach of finding an intermediate representation using topics and modeling n-grams using noun phrases in tweets. The distinctive feature in our work is the use of the temporal correlation between tweets which has not been considered in traditional text summarization.

2.2 Micro-Blog Event Summarization

Chakrabarti and Punera have proposed a variant of Hidden Markov Models to obtain an intermediate representation for a sequence of tweets relevant for an event [7]. Their approach does not use the continuous time stamps present in tweets and does not address the problem of obtaining the minimal set of tweets relevant to an event.
Meng et al. have summarized opinions for entities in Twitter by mining hash-tags to infer the presence of entities and inferring sentiments from tweets [18]. However, not all tweets contain hash-tags which makes it difficult to gain sufficient coverage for an event this way.

Sharifi et al. have proposed the Phrase Reinforcement Algorithm [25] to find the best tweet that matches a given phrase, such as trending keywords. They produce one tweet as a summary for one phrase while we propose to provide a set of tweets to summarize an event.

Yang et al. have also proposed a framework for summarizing the unfiltered tweet stream [31]. Their main focus is on creating a scalable approach by compressing the tweet stream to fit in limited memory, followed by the use of Non-negative Matrix Factorization (NMF) to find topics in the tweet stream. Since they do not filter the tweets for a specific event of interest, the topics discovered using their framework will only contain globally major events. Our proposed framework finds a summary for a targeted event of interest.

2.3 Dynamic Topic Models for Social Media

Ahmed et al. have used an exponential decay function to model the dynamic user behavior in search logs [5]. But they have assumed that the parameters of the decay function remains constant for all topics. We have taken a different approach by assuming that there is a decay parameter for each topic and we infer the parameters of the decay function using the variance of Gaussian distribution on the time of the written words.

Saha and Sindhwani have improved upon existing non-negative matrix factorization to provide an online version for finding emerging topics in social media [22]. But unlike our work, they do not address the problem of short sentences in social media.

Wang and McCallum have proposed a non-markovian approach to model the trend of topics evolution in text [27]. Their approach assumes that the time stamp on each word is generated by a Beta distribution because of the different shapes a Beta distribution can take. We have used a Gaussian distribution instead because the symmetric shape of the Gaussian curve allows us to use the variance for inferring the decay parameters of our Gaussian Decay Topic Model (GDTM).

Wang et al. have proposed a temporal topic model called (TM-LDA) that exploits the temporal correlation between the posts for each specific author [28]. They assume that a tweet topic distribution is related to the next tweet via a square matrix with dimensions equal to number of topics. But the algorithm solves for the matrix by minimizing the transition error in Euclidean space. Our approach describes the model as a generative process to preserve the probabilistic foundations of LDA. We have also explicitly used the time for each tweet to describe the amount of temporal correlation between consecutive tweets.

3. SUMMARIZATION FRAMEWORK

Figure 1 provides an overview of the framework we propose in this paper. To summarize for the event of interest $e$ from the unfiltered tweet stream $D$, we first begin by assuming that we have a set of queries $Q$, where each query $q \in Q$ is defined by a set of keywords. For example, the set of queries for the event “Facebook IPO” can be \{ { facebook, ipo }, { fb, ipo }, { facebook, initial, public, offer }, { fb, initial, public, offering } \}.

1. From $D$, we extract all tweets that match at least one of the queries $q \in Q$. A tweet matches a query $q$ if it contains all of the keywords in $q$. The set of tweets obtained is denoted by $D^1_q$.

2. Next, we apply a topic model on $D^1_q$, to find keywords that describe the main aspects of the event that are being discussed. We have developed two topic models that are designed to extract relevant tweets. We will describe these two topic models in Sections 3.4 and 3.5.

3. Once we have obtained the set of topics $Z$ from the topic models, the top ranked words in each topic $z \in Z$ are the keywords that describe various aspects of the event $e$. We may obtain the additional set of tweets $D^2_e$ by finding tweets $d \in D$ that are not present in $D^1_q$ by selecting those with high perplexity score with respect to the topics. Section 3.6 will elaborate on this.

4. $D^1_q$ and $D^2_e$ can be merged to refine the model and improve upon the topics for the event $e$.

5. Using the final set of topics $z \in Z$, we can summarize the event $e$ by selecting the tweets $d$ from each topic $z$ that give the best (lowest) perplexity. This is described in Section 3.7.

The whole process can be performed for several iterations to improve the quality of the summary.

3.1 NP+LDA

Due to the noisy nature of tweets, it is typical to find that many of the words in a tweet contribute little or no information to the aspects of the target event. In order to avoid processing the unnecessary words in tweets, we remove stopwords and only consider noun phrases by applying a Part-of-Speech Tagger to extract noun phrases using the following regular expressions

$\text{Base}_{NP} := \text{determiner? adjectives + nouns +}$
$\text{Conj}_{NP} := \text{Base}_{NP}(\text{of Base}_{NP})*$

We then model the noun phrases in tweets using the NP+LDA model as described in Chua et al. [8]. Instead of generating a topic for every word, we only generate a topic for each noun phrase which may consist of several words [8]. The subsequent topic models we propose in the rest of the paper extends from NP+LDA.

3.2 The Problem of Short Documents

As we mentioned earlier, one of the inherent difficulties of modeling tweets is the short length of the tweets, most of which consist of typically 20 to 30 words. In order to understand why this is difficult, let us examine how topic modeling works on traditional documents using Figure 2.

Figure 2(a) shows three documents, $d_1$, $d_2$ and $d_3$ containing certain words $w_{1:8}$. The document $d$ contains the word $w$ if there is an edge connecting $d$ and $w$. Topic models exploit the co-occurrences of words between documents to find relations between words. Given that $d_1$ and $d_2$ share the common words $\{ w_1, w_2, w_3, w_4 \}$, we can infer that this
3.3 Temporal Correlation of Twitter Content

Given this problem, the question is whether we can exploit other features to make up for the weakness and sparsity of Twitter. An additional feature that Twitter provides is the time-stamp on each of the tweets, showing when the tweet was published. Figures 3 and 4 show the trend of words written by Twitter users for the event “Facebook IPO”. In these figures, the x-axis represents the time-stamps with each vertical bin representing a day while the y-axis represents the frequency of the words written for the respective day (bin). In Figure 3, the words {“date”, “17”, “may”, “18”} represent the topic of Twitter users discussing the launch date of “Facebook IPO”. Figure 3(a) for “date” and 3(c) for “may” show two spikes around the same period of time. Figure 3(b) shows that “17” has temporal co-occurrence with “date” and “may” in the first spike while Figure 3(d) shows that “18” has temporal co-occurrence with the second spike. Based on such temporal co-occurrence relationships, it leads us to infer that these words {“date”, “17”, “may”, “18”} possibly belong to the same topic.

In Figure 4, the words represent the topic of Twitter users discussing the launch price of “Facebook IPO”. Similar to the previous analysis of the launch date, the first spike in Figure 4(a) shows that the word “price” co-occurs with “28” in Figure 4(b) and “35” in Figure 4(c). Figure 4(d) shows that the word “38” co-occurs with the word “price” in the second spike. Using such temporal co-occurrences, we can infer that these words {“price”, “28”, “35”, “38”} are likely to belong to the same topic.

By assuming that tweets written around the same time for the same event are similar in content, we could arrange the set of tweets in a sorted order such that tweets written around the same time can “share” words from other tweets to compensate for their short length. Figure 5 shows an illustrated example of this idea. Assuming that tweet $d_2$ is written after tweet $d_1$, we could imagine $d_2$ as inheriting some of the words in $d_1$ as shown by the blue - - lines. Similarly, $d_3$ could also inherit some of the words written by $d_2$ as shown by the red - - lines. The inheritance need not...
be strictly binary, instead it can be weighted according to the time difference between the two consecutive tweets. The next section will explain how we model the inheritance using an exponential decay function. As a result of this inheritance between tweets, the sparse Twitter data becomes denser and will improve the inference of topics from tweets.

### 3.4 Decay Topic Model (DTM)

Given that we want to allow tweets to inherit the content of previous ones, we need to define a model such that each tweet inherits not only the words of the immediately preceding tweet but also earlier tweets, subjected to an increasing decay as we increase the time difference between tweets. However there are several computational issues that we have to cope with. 1) Suppose we duplicate the existence of these words in later tweets for their inheritance, the size of the corpus will be inflated due to the duplication. The inflated corpus causes unnecessary repeated computation for inference of the duplicated words. 2) Suppose the duplication proceeds for every subsequent tweet, this accumulation of words will result in a snowball effect that eventually causes tweets with newer time-stamps to inherit the content of all previous tweets. The snowballed tweets of later time-stamps will have less diverse variations in their topic because of the baggage incurred from the inheritance.

We need to define our model such that 1) it avoids repetitive computation and 2) it decays the inheritance of the baggage incurred from the inheritance.

### 3.4.1 Decaying Inheritance

In order to address the issue of inheritance of the previous tweets, we introduce a decay factor $\delta_{zt}$ such that the second issue can be addressed by using an exponential decay function as follows:

$$p_{zt} = \exp(-\delta_{zt})$$

where $p_{zt}$ is the probability of topic $z$ in tweet $t$, and $\delta_{zt}$ is the decay factor associated with topic $z$. The larger the value of $\delta_{zt}$, the faster the topic $z$ loses its novelty. $t_i$ is the time that tweet $d_i$ was written. The summation is over all the tweets $[1, n-1]$ that was written before tweet $d_n$. Each $p_{zt}$ is decayed according to the time difference between tweet $d_i$ and tweet $d_1$. Although the summation seems to involve an $O(n)$ operation, the task can be made $O(1)$ via memoization, a programming technique.

4. For each noun phrase $np$ in tweet $d$, sample a topic variable $z_{d, np}$ from a multinomial distribution using $\theta_d$ as parameters.

$$z_{d, np} \sim Mult(\theta_d)$$

5. For each noun phrase $np$, sample words $w_{d, np, v}$ for the tweet $d$ using topic variable $z_{d, np}$ and the topic word distribution $\phi_z$.

$$P(w_{d, np} | z_{d, np} = k, \phi) = \prod_{v \in np} P(w_{d, np, v} | z_{d, np} = k, \phi_k)$$

The inference algorithm for DTM is given by,

$$P(z_{d, np} = k | w_{d, np}, \alpha, \beta, \delta_k) \propto \left[ \Gamma(\beta + q_{k,v} + |v \in np|) \right]^{\alpha} \left[ \Gamma(V\beta + q_k) \right]^{\frac{\alpha + \sum_{i=1}^{n-1} p_{i,k} \cdot \exp(-\delta_{z}(t_n - t_i))}{\Gamma(V\beta + q_k)}}$$

where $V$ is the total size of vocabulary, $|np|$ is the number of words in the noun phrase, $|v \in np|$ is the number of times $v$ appear in $np$, $q_{k,v}$ is the number of times $v$ is inferred as topic $k$ and $q_k$ is the number of words that are in topic $k$.

### 3.4.2 Decay T opic Model (DTM)

Given that we want to allow tweets to inherit the content of previous ones, we need to define a model such that each tweet inherits not only the words of the immediately preceding tweet but also earlier tweets, subjected to an increasing decay as we increase the time difference between tweets. However there are several computational issues that we have to cope with. 1) Suppose we duplicate the existence of these words in later tweets for their inheritance, the size of the corpus will be inflated due to the duplication. The inflated corpus causes unnecessary repeated computation for inference of the duplicated words. 2) Suppose the duplication proceeds for every subsequent tweet, this accumulation of words will result in a snowball effect that eventually causes tweets with newer time-stamps to inherit the content of all previous tweets. The snowballed tweets of later time-stamps will have less diverse variations in their topic because of the baggage incurred from the inheritance.

We need to define our model such that 1) it avoids repetitive computation and 2) it decays the inheritance of the baggage incurred from the inheritance.

### 3.4.2.1 Decaying Inheritance

In order to address the issue of inheritance of the previous tweets, we introduce a decay factor $\delta_{zt}$ such that the second issue can be addressed by using an exponential decay function as follows:

$$p_{zt} = \exp(-\delta_{zt})$$

where $p_{zt}$ is the probability of topic $z$ in tweet $t$, and $\delta_{zt}$ is the decay factor associated with topic $z$. The larger the value of $\delta_{zt}$, the faster the topic $z$ loses its novelty. $t_i$ is the time that tweet $d_i$ was written. The summation is over all the tweets $[1, n-1]$ that was written before tweet $d_n$. Each $p_{zt}$ is decayed according to the time difference between tweet $d_i$ and tweet $d_1$. Although the summation seems to involve an $O(n)$ operation, the task can be made $O(1)$ via memoization, a programming technique.

4. For each noun phrase $np$ in tweet $d$, sample a topic variable $z_{d, np}$ from a multinomial distribution using $\theta_d$ as parameters.

$$z_{d, np} \sim Mult(\theta_d)$$

5. For each noun phrase $np$, sample words $w_{d, np, v}$ for the tweet $d$ using topic variable $z_{d, np}$ and the topic word distribution $\phi_z$.

$$P(w_{d, np} | z_{d, np} = k, \phi) = \prod_{v \in np} P(w_{d, np, v} | z_{d, np} = k, \phi_k)$$

The inference algorithm for DTM is given by,

$$P(z_{d, np} = k | w_{d, np}, \alpha, \beta, \delta_k) \propto \left[ \Gamma(\beta + q_{k,v} + |v \in np|) \right]^{\alpha} \left[ \Gamma(V\beta + q_k) \right]^{\frac{\alpha + \sum_{i=1}^{n-1} p_{i,k} \cdot \exp(-\delta_{z}(t_n - t_i))}{\Gamma(V\beta + q_k)}}$$

where $V$ is the total size of vocabulary, $|np|$ is the number of words in the noun phrase, $|v \in np|$ is the number of times $v$ appear in $np$, $q_{k,v}$ is the number of times $v$ is inferred as topic $k$ and $q_k$ is the number of words that are in topic $k$.

### 3.4.2.2 Decay T opic Model (DTM)

Given that we want to allow tweets to inherit the content of previous ones, we need to define a model such that each tweet inherits not only the words of the immediately preceding tweet but also earlier tweets, subjected to an increasing decay as we increase the time difference between tweets. However there are several computational issues that we have to cope with. 1) Suppose we duplicate the existence of these words in later tweets for their inheritance, the size of the corpus will be inflated due to the duplication. The inflated corpus causes unnecessary repeated computation for inference of the duplicated words. 2) Suppose the duplication proceeds for every subsequent tweet, this accumulation of words will result in a snowball effect that eventually causes tweets with newer time-stamps to inherit the content of all previous tweets. The snowballed tweets of later time-stamps will have less diverse variations in their topic because of the baggage incurred from the inheritance.

We need to define our model such that 1) it avoids repetitive computation and 2) it decays the inheritance of the baggage incurred from the inheritance.

### 3.4.2.3 Exploiting Temporal Correlations

In order to exploit temporal correlations in the Twitter data, we introduce a decay factor $\delta_{zt}$ such that the second issue can be addressed by using an exponential decay function as follows:

$$p_{zt} = \exp(-\delta_{zt})$$

where $p_{zt}$ is the probability of topic $z$ in tweet $t$, and $\delta_{zt}$ is the decay factor associated with topic $z$. The larger the value of $\delta_{zt}$, the faster the topic $z$ loses its novelty. $t_i$ is the time that tweet $d_i$ was written. The summation is over all the tweets $[1, n-1]$ that was written before tweet $d_n$. Each $p_{zt}$ is decayed according to the time difference between tweet $d_i$ and tweet $d_1$. Although the summation seems to involve an $O(n)$ operation, the task can be made $O(1)$ via memoization, a programming technique.
not show that the resulting topics are well differentiated by time. This motivates us to address a deficiency in the Decay Topic Model. Since we have already modeled the temporal correlation of tweets by adding the exponential decay function between the tweets’ topic distributions, we could also add additional parameters to the topic word distributions to model the assumption that words specific to certain topics has higher chance of appearing at specific times. This leads us to introduce the Gaussian Decay Topic Model.

### 3.5 Gaussian Decay Topic Model (GDTM)

The generative process for the Gaussian Decay Topic Model (GDTM) follows that of DTM with the addition of the time stamp generation for each noun phrase,

1. In addition to topic word distribution \( \phi_z \), each topic \( z \) has an additional topic time distribution \( G_z \) approximated by the Gaussian distribution with mean \( \mu_z \) and variance \( \sigma_z^2 \).

\[
G_z \sim N(\mu_z, \sigma_z^2)
\]

2. Then the time \( t \) for a noun phrase \( np \) is given by the following,

\[
P(t_{np} | G_z) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{(t_{np} - \mu_z)^2}{2\sigma_z^2}\right)
\]

Since every topic \( z \) is now associated with a Gaussian distribution \( G_z \), we can use the shape of the distribution curve to determine the decay factors \( \delta_z \), \( \forall z \in Z \). The \( \delta_z \), which was previously used for transferring the topic distribution from previous tweets to the subsequent tweets can depend on the variances of the Gaussian distributions. Topics having small values of variance \( \sigma_z^2 \) indicate aspects that have short lifespans and should decay quickly (larger \( \delta_z \)), while topics with large variance represent aspects that should decay slowly giving it a smaller \( \delta_z \). The inference algorithm for GDTM is as follows,

\[
P(z_{d,np} = k | w_{d,np}, \alpha, \beta, \delta_k, \mu_k, \Sigma_k^2) \propto \frac{\Gamma(\beta + q_k)}{\Gamma(\beta + |np| + q_k)} \left[ \prod_{v \in np} \frac{\Gamma(\beta + q_k, v + [v \in np])}{\Gamma(\beta + q_k, w)} \right] \left[ \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(t_{np} - \mu_k)^2}{2\sigma_k^2}\right) \right]^{|np|} \left[ \alpha + \sum_{i=1}^{n} p_{i,k} \cdot \exp(-\delta_k(t_n - t_i)) \right]
\]

where \( V \) is the total size of vocabulary, \( |np| \) is the number of words in the noun phrase, \( |v \in np| \) is the number of times \( v \) appear in \( np \), \( q_{k,v} \) is the number of times \( v \) is inferred as topic \( k \) and \( q_k \) is the number of words that are in topic \( k \).

We use the concept of half-life to estimate the value of \( \delta_z \). Given that we want to find the \( \delta_z \) value that causes a tweet to discard half of the topic from previous tweet,

\[
\exp(-\delta \cdot (t_n - t_{n-1})) = 0.5
\]

\[
\delta \cdot \Delta T = \log 2
\]

\[
\delta = \log 2 / \Delta T
\]

Figure 7 shows a Gaussian distribution with an arbitrary mean and variance. The value of \( \Delta T \) is affected by the variance (width) of the distribution. To estimate \( \Delta T \), let \( \Delta T = \tau \Delta t \) where \( \tau \) is a parameter and \( \Delta t \) is estimated as follows,

\[
\frac{P(0)}{P(\Delta t)} = \frac{2p}{p} = \frac{\exp(0)}{\exp(-\Delta t)} = 2
\]

\[
\frac{(\Delta t)^2}{2\sigma^2} = \log 2
\]

\[
\Delta t = \sqrt{2\sigma^2 \log 2}
\]

Finally, \( \delta \) is given by,

\[
\delta = \frac{\log 2}{\tau \sqrt{2\sigma^2 \log 2}}
\]

As shown in Equation 1, the larger the variance \( \sigma^2 \) is, the smaller the \( \delta \) (decay) and vice versa.
the exponential of the log likelihood normalized by
the inferred probabilities. The perplexity of tweet
differentiate between the importance of different words using
e being more relevant to event e
are then ranked in ascending order with lower perplexity
e related method of achieving this is to perform query expan-
Fig. 8: Intensity and the Gaussian Components
Figure 8(a) shows the the intensity (y-axis) of all the topics
each differentiated with different colors with respect to
time (x-axis) found by the Gaussian Decay Topic Model (GDTM). Figure 8(b) shows the corresponding Gaussian components for each topic.
While some other probability distributions can also be used to describe the time distribution of words in an event, we choose the Gaussian distribution because of the ease of computing sufficient statistics for inference using Gibbs Sampling. We also exploit the symmetry of Gaussian distribution in estimating δ. This symmetric property cannot be observed in most continuous distributions.
3.6 Additional Tweets From Unfiltered Tweet Stream
After finding the topics from the initial set of relevant tweets \(D^1_e\), the next step is to find additional tweets \(D^2_e\) from the unfiltered tweet stream \(D\) using the trained model. A related method of achieving this is to perform query expansion by using the top words in a topic for keyword search. Instead of applying a threshold on selecting the top-k keywords for query expansion, we compute a perplexity score for each tweet \(d \in D, d \notin D^1_e\). Tweets relevant to the event e are then ranked in ascending order with lower perplexity being more relevant to event e. Using the perplexity score instead of keyword search from each topic allows us to differentiate between the importance of different words using the inferred probabilities. The perplexity of tweet \(d\) is given by the exponential of the log likelihood normalized by the number of words in a tweet.

\[
perplexity(d) = \exp\left(-\frac{\log P(d|\theta, \phi, G)}{N_d}\right)
\]  

where \(N_d\) is the number of words in tweet \(d\). Since tweets with fewer words tend to have higher probabilities and therefore lower perplexity, we normalized by \(N_d\) in order to favour tweets with more words.

3.7 Summarization
Our goal is to use the extracted topics to summarize the event \(e\). Summarizing the event is a multi-objective problem. On one hand we want to select tweets such that we maximize the total perplexity using as few tweets as possible. But we also want the topic overlap between the selected tweets to be as minimum as possible.
The models described earlier are designed to provide us diverse topics representing the various different aspects of the event that are being discussed on twitter. Using the topics learned from the set of relevant tweets \(D_e\), we can obtain the most representative tweet from each topic to summarize the target event \(e\).
To choose the most representative tweet for topic \(z\), we compute the perplexity with respect only to topic \(z\) for all tweets \(d \in D_e\) and choose the tweet that has lowest perplexity with respect to \(z\).

\[
perplexity(d, z) = \exp\left(-\frac{\log P(d, z|\theta, \phi, G_z)}{N_d}\right)
\]

The list of representative tweets for each topic forms the summary of the event \(e\). Note that, depending on the size of the summary required, we could extract additional representative tweets for each topic, based on the perplexity scores.

4. EXPERIMENTS
To validate our choice of using the temporal correlation between tweets to extract topics, we evaluate the two models DTM and GDTM with respect to the LDA baseline. Unlike our DTM and GDTM models, the LDA baseline does not consider the use of noun phrases and assumes that every tweet has no temporal relation to other tweets. One possible way to evaluate the temporal correlation is to compare the convergence log-likelihoods of these models and assume that the model with the highest log-likelihood during convergence is a better model. Alternatively, we can also compute the perplexity score of a held-out test set.

However these approaches have the following problems, 1) The models make different assumptions on the generative process of the data, especially LDA which considers tweets as a bag-of-words while DTM and GDTM consider noun phrases. 2) Tweets contain a great deal of noise in them. Many of the tweets containing keywords such as “Facebook” and “IPO” are found to be spam instead. These tweets try to gain attention and visibility by riding on the popularity of these trending keywords during the occurrence of these events. A model that optimizes for the log-likelihood or perplexity score risks over-fitting the parameters to these noisy tweets.

We therefore evaluate the temporal correlation and the two derivative models by comparing 1) the quality of the summaries generated from these models and 2) their utility towards finding additional tweets from the unfiltered tweet stream that are related to the event and yet do not contain the keywords from the original queries.

4.1 Events and Data Set
We perform our experiments for four real-world events, selected to cover natural disasters, politics and company events. For each event, we apply a set of queries on the unfiltered tweet stream \(D\) to obtain the relevant set of tweets, \(D^1_e\). The events used in this study are:

1. Facebook IPO: The Initial Public Offer (IPO) of Facebook Inc. [3]. We use \{ \{ Facebook | FB\}, IPO \}, \{ \{ Facebook | FB\}, Initial, Public, (Offer | Offering) \} as queries to obtain a set of 9,570 tweets.

2. Obamacare: The Patient Protection and Affordable Care Act [4]. We use \{ \{ Obamacare \}, \{ PPACA \}, \{ Obama, Health, Care \}, \{ Obama (Healthcare | Health-care) \} \} to obtain a set of 136,761 of tweets.
3. **Japan Earthquake**: The earthquake that occurred near Tokyo, Japan in 2011 [1]. We use \{ \{ Fukushima \}, \{ (Japan | Tokyo), (Earthquake | Quake | Tsunami) \} \} to obtain a set of 251,802 tweets.

4. **BP Oil Spill**: The oil spill resulted from British Petroleum (BP) drilling in the Gulf of Mexico [2]. We use \{ \{ BP, Oil, Spill \}, \{ Gulf, Mexico, Oil, Spill \} \} to obtain a set of 79,676 tweets.

Note that the number of tweets for these events ranges from a small 9,570 tweets for Facebook IPO to a mammoth 251,802 tweets for the Japan Earthquake.

### 4.2 Summarization Results

Fair evaluations of our summaries require both a quantitative comparison with simulated true summaries and qualitative assessment from human readers. Due to the difficulty of obtaining human generated summaries from our data sets, we construct the true summaries by using the headlines of news articles found in the reference section of the events’ Wikipedia articles. The human readers are crowdsourced from Amazon Mechanical Turk (MTurk).

#### 4.2.1 Quantitative Comparison with Wikipedia

Wikipedia forms a comprehensive resource for all manner of real-world content including the events that we consider in this paper. Each Wikipedia article for an event contains a section that references the relevant news articles which contributed to the article. These news articles thus can be considered as proxies for each of the important pieces of news about the event. Since Wikipedia articles are edited and discussed by the general public, the news articles that are referenced represent the popular choices of the internet public. For each of the Wikipedia references for our events, we extract the headline text which gives a one line summary of the corresponding news article. The headline also has an advantage of resembling the language style used in tweets. To construct a true summary for each event from its corresponding Wikipedia article, we aggregate the one-line summaries of all the news articles referenced in the Wikipedia article. We then compare the true summary with the summaries we generated from each model using a similarity metric.

The similarity metric we use for the comparison of summaries is adapted from the metric proposed by Lin and Hovy [17]. The metric counts the total number of matching n-grams (excluding stop-words) between the true summary \(S^{tr}\) and the summary \(S^m\) generated from model \(m\). We let \(NG^m_n\) denote the set of n-grams from the true summary and \(NG^m_n\) denote the n-grams from summaries generated by the model \(m\).

\[
g_n = \sum_{ng \in NG^m_n} \min(|ng|, |ng|) \quad (3)
\]

\[
Sim(S^{tr}, S^m) = 0.2 \cdot g_1 + 0.3 \cdot g_2 + 0.5 \cdot g_3 \quad (4)
\]

Equation 3 first calculates the number of n-grams common to both \(S^{tr}\) and \(S^m\). In order not to let a few frequent n-gram dominate the counts, each n-gram is limited to the minimum number of counts between the true summary and the generated summary. Equation 4 calculates the final similarity score between the summaries by aggregating the number of matched 1, 2 and 3 grams. The weights allocated are meant to give a higher importance to 3-grams and lower importance to 1-grams.

![Figure 9: Results of Sim(S^{tr}, S^m) Score](image)

Figures 9(a) and 9(b) show two sets of results for different number of topics. The y-axis gives the similarity score between the model-generated summaries and the true summary, while the x-axis differentiates between the various events. Figures 9(a) and 9(b) show that GDTM consistently gives better performance than LDA. DTM is shown to be better than GDTM for “Obamacare” at \(K=8\) in Figure 9(a) and “Facebook IPO” at \(K=10\) in Figure 9(b). Overall, GDTM shows better performance over DTM and LDA with DTM showing inconsistent performance. DTM is sometimes better than LDA and sometimes slightly worse-off than LDA. This suggests that estimating the appropriate decay parameters is important for using the temporal correlation features.

Since we extract the most representative tweet for each topic, the use of \(K\) topics gives \(K\) tweets as the summary for each event. In our experiments, we use \(K = 8\) and \(K = 10\), to obtain 8-tweet summaries and 10-tweet summaries for each event. We choose these values of \(K\) to avoid generating long summaries for the events so that the human evaluation task in Mechanical Turk will be easier for our mechanical turk workers.
4.2.2 Qualitative Evaluation on Mechanical Turk

We used Amazon’s Mechanical Turk to find human evaluators for this task. Each mechanical turk worker (mturker) was presented the generated summaries of the four events from each model. Since each event has three summaries from the three models, mturkers were instructed to choose 1 or 2 out of the 3 summaries as the best representations of the event. We also provided a feature for mturkers to leave comments. To avoid bias to any one model, we did not show which model generated the summaries and we randomized the order of presented summaries. We required mturkers to fulfill three criteria before they could participate in our experiment: 1) have approval rating of ≥ 95%, 2) have completed more than 1000 tasks, and 3) are located in United States (USA) because the events “Facebook IPO”, “Obamacare” and “BP Oil Spill” are more relevant to the population of USA. Although “Japan Earthquake” did not occur in USA, the event can be considered to be of interest to everyone worldwide. We employed a total of 100 different mturkers. Each mturker spent an average of 7 minutes and 10 seconds to complete the task which translated to an hourly wage of approximately USD$4.2.

Table 1: List of summaries from the three models

<table>
<thead>
<tr>
<th>Summary from LDA</th>
<th>Summary from DTM</th>
<th>Summary from GDTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook IPO on for May 18</td>
<td>Facebook IPO</td>
<td>Facebook IPO on for May 18</td>
</tr>
<tr>
<td>Facebook co-founder Saverin renounces U.S. citizenship before IPO</td>
<td>Social media shares down ahead of Facebook IPO</td>
<td>GDM boosts Facebook IPO price</td>
</tr>
<tr>
<td>Not sure I agree, but it sure is fun to cover RT @jbruin: FB IPO day is better than Christmas</td>
<td>Facebook announces IPO price is 38</td>
<td>Mark Zuckerberg Made out Nicely in the Corporate World</td>
</tr>
<tr>
<td>Facebook Goes Public With Initial Public Offering on NASDAQ</td>
<td>Facebook Goes Public With Initial Public Offering on NASDAQ</td>
<td>Facebook IPO Roadshow <a href="http://t.co/1m0L1f1S">http://t.co/1m0L1f1S</a></td>
</tr>
<tr>
<td>Facebook Co-Founder Saverin Gives Up U.S. Citizenship Before IPO</td>
<td>Facebook co-founder Saverin renounces U.S. citizenship ahead of IPO</td>
<td>Facebook IPO Roadshow</td>
</tr>
<tr>
<td>Summary from GDTM</td>
<td>Summary from DTM</td>
<td>Summary from GDTM</td>
</tr>
<tr>
<td>Facebook reportedly looking at May 17 for IPO</td>
<td>Facebook sets $28-30 price range for IPO</td>
<td>Facebook IPO</td>
</tr>
<tr>
<td>Facebook sets $28-30 price range for IPO</td>
<td>Facebook goes public</td>
<td>Facebook Co-Founder Saverin Gave Up U.S. Citizenship Before IPO</td>
</tr>
<tr>
<td>Facebook goes public</td>
<td>Facebook Co-Founder Saverin Gave Up U.S. Citizenship Before IPO</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
<tr>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
<tr>
<td>Woot! it’s Facebook ipo day!</td>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
<tr>
<td>EXCLUSIVE: Here’s The Inside Story Of What Happened On The Facebook IPO</td>
<td>EXCLUSIVE: Here’s The Inside Story Of What Happened On The Facebook IPO</td>
<td>Facebook IPO</td>
</tr>
<tr>
<td>As IPO Approaches, Facebook Addresses Weaknesses</td>
<td>As IPO Approaches, Facebook Addresses Weaknesses</td>
<td>NASDAQ CEO blames CFO over Facebook IPO</td>
</tr>
<tr>
<td>#1Mby1M - via @business</td>
<td>#1Mby1M - via @business</td>
<td>Twitter IPO Roadshow</td>
</tr>
<tr>
<td>Summary from DTM</td>
<td>Summary from GDTM</td>
<td>Summary from GDTM</td>
</tr>
<tr>
<td>Facebook IPO</td>
<td>Facebook IPO</td>
<td>Facebook IPO</td>
</tr>
<tr>
<td>Facebook IPO on for May 18</td>
<td>Facebook IPO on for May 18</td>
<td>Facebook IPO on for May 18</td>
</tr>
<tr>
<td>Social media shares down ahead of Facebook IPO</td>
<td>Social media shares down ahead of Facebook IPO</td>
<td>Social media shares down ahead of Facebook IPO</td>
</tr>
<tr>
<td>Facebook announces IPO price is 38</td>
<td>Facebook announces IPO price is 38</td>
<td>Facebook announces IPO price is 38</td>
</tr>
<tr>
<td>Facebook Goes Public With Initial Public Offering on NASDAQ</td>
<td>Facebook Goes Public With Initial Public Offering on NASDAQ</td>
<td>Facebook Goes Public With Initial Public Offering on NASDAQ</td>
</tr>
<tr>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
<tr>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
<tr>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
<td>Facebook raises IPO price as offering nears</td>
</tr>
</tbody>
</table>

Figure 10: Mechanical Turk Results

Figure 10 shows the results from Mechanical Turk with the x-axis differentiating between the four events and the y-axis showing the number of votes for the respective models. Aggregating the total votes for all events, GDTM has the most votes with 207 votes followed by DTM with 146 votes and finally LDA with 130 votes.

We show the summaries we presented to our mturkers for the “Facebook IPO” event. From Table 1, we observe that the LDA summary does not contain the tweet about the co-founder renouncing his citizenship and more importantly, does not feature the aftermath of the Facebook IPO event. The DTM summary shows that the tweet about the co-founder renouncing his citizenship and the negative aftermath of Facebook IPO.

Some comments given by our mturkers were: 1) “I think these (GDTM) best summarize Facebook IPO because it shows a broad range of information related to the event.” 2) “Well I learned that their co-founder renounced his US citizenship just now!” 3) “I believe that other summaries (non-GDTM) had a large amount of personal opinion and not fact.” 4) “If I wanted to find information, Summaries (DTM) and (GDTM) had the most.” 5) “There is too much garbage posts in the other summaries (non-GDTM), and not true news.” These comments show that readers appreciated the GDTM summaries and felt that it was a good representation of the event.

4.3 Search Results

Next we evaluate the models on their ability to retrieve additional tweets that are relevant to the event but do not contain the keywords in the queries. The relevance of a tweet is measured based on perplexity, as given by Equation 2. We calculate the perplexity score for each of the tweets with respect to each of the models in the unfiltered tweet stream and then rank it accordingly. This way, we obtain three ranked lists from the three models.

Traditional Information Retrieval (IR) evaluation is done by going through each list from top to bottom to compute
the precision and recall curve at each \( k \). In our evaluation here, we do not make a binary decision as to whether each tweet is relevant to the event. Instead, we compute the number of n-grams that matched between the top-\( k \) tweets and the true summary. Because we vary \( n \) from 1 to 3, we obtain three sets of precision \( P_R^n \) and recall \( R_C^n \) values. The precision and recall are calculated as follows,

\[
g_n = \min(\{|ng| \in NG^r_n, \, |ng| \leq NG^{top-k}_n\})
\]

\[
PR_n^k = \frac{g_n}{|D^{top-k}|}
\]

\[
R_C_n^k = \frac{g_n}{|NG^r_n|}
\]

\[
PR^k = 0.2 \cdot PR_1^k + 0.3 \cdot PR_2^k + 0.5 \cdot PR_3^k
\]

\[
R_C^k = 0.2 \cdot R_C_1^k + 0.3 \cdot R_C_2^k + 0.5 \cdot R_C_3^k
\]

Varying \( k \) from 1 to the size of the unfiltered tweet stream gives us the Precision-Recall curve (PR-curve) as shown in Figure 11.

![Figure 11: Precision Recall Curves of \( D^2_e \)](image)

Figure 11 shows us that the results given by GDTM is significantly better than DTM and DTM is in turn better than LDA.

5. CONCLUSION

We have presented our framework for summarizing events from the unfiltered tweet stream of Twitter. We have developed two topic models, the Decay Topic Model (DTM) and Gaussian Decay Topic Model (GDTM) that leverage the temporal correlation that exists among tweets written around the same time, to extract meaningful topics that capture different aspects of the underlying event. We have shown how representative tweets with low perplexity can be selected from the extracted topics to generate a concise and information-rich summary of the events. Our experiments evaluating the summaries using Wikipedia links as well as the qualitative evaluation using Mechanical Turk have demonstrated that both our topic models generated summaries that outperformed traditional LDA in almost all cases with GDGM having the highest performance overall and also receiving the highest overall votes from the Turk workers. The Search and Summarize framework also proceeds in an iterative loop, with newer search queries being generated from the extracted topic models and the resultant tweets used to refine the topic model and the summary.

We believe that the area of social media summarization has lots of scope for future work. To that end, we have received insightful feedback from Mechanical Turk workers. Some workers preferred summaries that fit their own beliefs and opinions. Thus personalized summaries could be extracted that are tailored to suit particular sentiments or beliefs. And conversely, factual tweets could be weighted higher to generate objective summaries. We wish to extend our work in these directions.

6. REFERENCES


