Event Summarization using Tweets

Deepayan Chakrabarti and **Kunal Punera**
Yahoo! Research
701 1st Avenue
Sunnyvale, CA 94089
{deepay,kpunera}@yahoo-inc.com

Abstract

Twitter has become exceedingly popular, with hundreds of millions of tweets being posted every day on a wide variety of topics. This has helped make real-time search applications possible with leading search engines routinely displaying relevant tweets in response to user queries. Recent research has shown that a considerable fraction of these tweets are about "events", and the detection of novel events in the tweet-stream has attracted a lot of research interest. However, very little research has focused on properly displaying this real-time information about events. For instance, the leading search engines simply display all tweets matching the queries in reverse chronological order. In this paper we argue that for some highly structured and recurring events, such as sports, it is better to use more sophisticated techniques to summarize the relevant tweets. We formalize the problem of summarizing event-tweets and give a solution based on learning the underlying hidden state representation of the event via Hidden Markov Models. In addition, through extensive experiments on real-world data we show that our model significantly outperforms some intuitive and competitive baselines.

Introduction

A key component of real-time search is the availability of real-time information. Such information has recently proliferated thanks to social media websites like Twitter and Facebook that enable their users to update, comment, and otherwise communicate continuously with others in their social circle. On Twitter, user compose and send short messages called "tweets", putting the medium to a wide array of uses. Recent research has shown that one of the big use-cases of Twitter is users reporting on events that they are experiencing: Sakaki et al. (2010) demonstrated that careful mining of tweets can be used to detect events such as earthquakes. Since then a lot of research works have focused on detection of novel events on Twitter (Petrović, Osborne, and Lavrenko 2010) and other social media websites (Becker, Naaman, and Gravano 2010) such as Flickr and Youtube.

However, the state of the art in real-time *usage* of this stream of event-tweets is still rather primitive. In response to searches for ongoing events, today's major search engines simply find tweets that match the query terms, and present the most recent ones. This approach has the advantage of leveraging existing query matching technologies, and for simple one-shot events such as earthquakes it works well. However, for events that have "structure" or are long-running, and where users are likely to want a summary of all occurrences so far, this approach is often unsatisfactory. Consider, for instance, a query about an ongoing game of American Football (we will use this as our running example). Just returning the most recent tweets about the game is problematic for two reasons: (a) the most recent tweets could be repeating the same information about the event (say, the most recent "touchdown"), and (b) most users would be interested in a summary of the occurrences in the game so far. This approach can clearly be improved upon using information about the structure of the event. In the case of Football, this corresponds to knowledge about the game play.

This motivates our goal of *summarizing* long-running structure-rich events. Our work complements recent work on detecting events in the twitter stream (Petrović, Osborne, and Lavrenko 2010). We assume that a new event has been detected; our goal is to extract a few tweets that best describe the chain of interesting occurrences in that event. In particular, we focus on repeated events, such as sports, where different games share a similar underlying structure (e.g., touchdowns, interceptions, fumbles, in American football) though each individual game is unique (different players, different sequences of touchdowns, fumbles, etc.). We want our approach to learn from previous games in order to better summarize a recent (perhaps even an ongoing) game; queries about this game can then present a full summary of the key occurrences in the game, as well as the latest updates.

Our proposed solution is a two-step process. We first design a modified Hidden Markov Model that can segment the event time-line, depending on both the burstiness of the tweet-stream and the word distribution used in tweets. Each such segment represents one distinct "sub-event", a semantically distinct portion of the full event. We then pick key tweets to describe each segment judged to be interesting enough, and combine them together to build the summary.

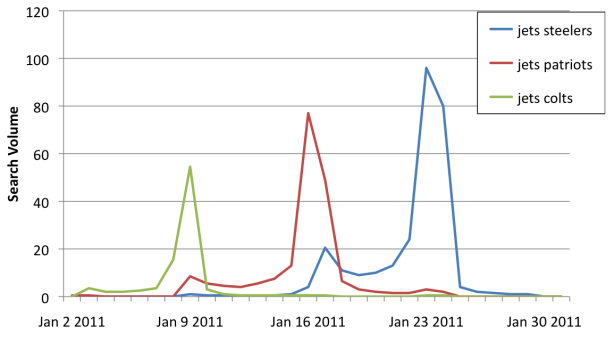

Figure 1: Volume of queries to the Google search engine containing names of both teams involved in NFL games. We see marked spikes on day of the games. The y-axis values only make relative sense.

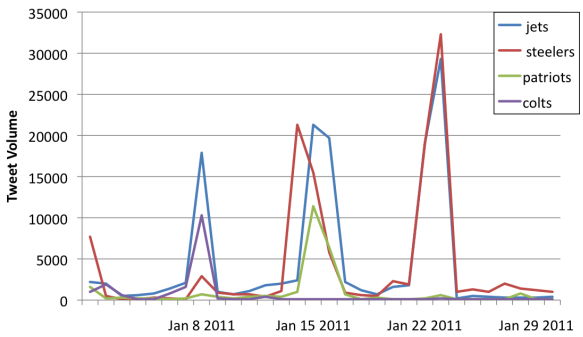

Figure 2: The absolute number of tweets referring to Football team names. There are marked spikes on the days that the teams compete in games.

OUR CONTRIBUTIONS.

We are, to the best of our knowledge, the first to study the problem of *summarizing* events on Twitter. To accomplish this, our algorithm tackles several challenges.

(1) *Events are typically "bursty".* Some types of sub-events generate far more tweets per unit time than others. Our algorithm ensures that the summary does not contain tweets from periods of low activity, yet does not over-represent the bursty periods either.

(2) *Separate sub-events may not be temporally far apart.* Our algorithm solves this problem by automatically learning language models for common types of sub-events. This allows it to separate different types of sub-events even when they are temporally close.

(3) *Previous instances of similar events are available.* Our algorithm can build offline models for the different sub-events using tweets from previous events – we do not want to re-learn the characteristics of a touchdown in every game.

(4) *Tweets are noisy.* Tweets are very short (less than 140 characters) and noisy, rife with spelling mistakes. Our algorithm achieves robust results in the face of these issues.

(5) *Strong empirical results.* We empirically compare our model against several baselines; in particular, we show that our algorithm achieves better recall over the set of important occurrences, and also gives a better sense of the context of these occurrences and the players associated with them.

ORGANIZATION.

In the next section, we present a detailed case-study on the need for real-time summarization of coverage of American Football games on Twitter. This also provides a concrete motivation for our work. Then, we discuss several baselines and present our proposed solution based on a modification of Hidden Markov Models. These solutions are compared in the experiments section, where the accuracy and utility of our algorithm is demonstrated. We defer the discussion of related work to the end of the paper as presenting the details of our approach first helps us better contrast it with existing works. Finally, we conclude the paper listing directions for future research.

Sports Coverage on Twitter

We want to summarize events so that the summaries can be intelligently consumed by downstream applications like search engines answering user queries. But is there an actual demand for such summaries? In particular: (1) Are web users interested in real-time information about these events? (2) Is such real-time information available? (3) Are existing methods to exploit this real-time information sufficient?

THE DEMAND FOR REAL-TIME COVERAGE OF SPORTS EVENTS.

For the first question, we consider the example of users looking for information on professional American Football games. In Figure 1 we plot the number of occurrences of three bigrams in the search query logs of the Google search engine. The bigrams – "jets steelers", "jets patriots", and "jets colts" – refer to the names of teams that played each other on three different Sundays in January 2011. [1] Even though Google Trends [2] does not reveal absolute numbers, the graphs clearly indicate a large spike in user interest concurrent with the games; this interest also extends into the next day (when users typically access news stories). We observe a similar surge in user interest for soccer matches during the World Cup 2010. Clearly, an automated method to answer user queries about events using real-time information would have significant utility.

THE SUPPLY OF REAL-TIME INFORMATION ON SPORTS EVENTS.

To answer the second question, we consider the problem of obtaining real-time information about these events. One readily available source is Twitter, and previous research has shown that some events such as earthquakes can be successfully identified by mining user tweets (Sakaki, Okazaki, and Matsuo 2010). We want to take this one step further and mine user tweets to construct real-time summaries of events. Here we again consider the example of American Football games. For the purposes of this experiment we identify

[1] Plotting the frequency of occurrences of the names of both teams in a game focuses in on the interest in the game as opposed to just general information about the teams.

[2] www.trends.google.com

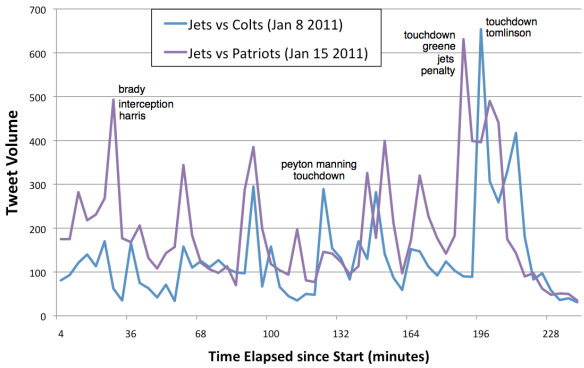

Figure 3: The absolute number of tweets referring to Football teams plotted over the duration of the game. It is clear that plays of interest are marked by spikes in tweet volume. The top frequent words during some spikes are marked showing that words in the tweets describe the plays.

tweets referring to teams by looking for certain hash-tags [3]. In particular, in Figure 2 we plot the number of occurrences of tweets containing the hash-tags "#jets", "#steelers", "#patriots", and "#colts". From the Figure, it is clear that the tweets referring to the teams spike during the times when the events take place. The three games mentioned in Figure 1 show up clearly in the tweets in addition to other games the teams are involved in.

In order to check whether the tweets that contain references to the team names are about the game at all we show in Figure 3 the frequency of such tweets for two games over the duration of the games. It is clear that the ebb and flow of the tweets is correlated to the happenings of the game. In addition to the frequency we have tagged some of the frequency spikes with the common words in the tweets. As we can see the words in the tweets associated with the spikes do document the happenings at these events.

CHARACTERISTICS OF SPORTS COVERAGE IN TWEETS.

We have seen that tweets referring to Football teams involved in games tend to provide details about specific plays of interest. In particular, we define as a "sub-event" a discrete chunk of the full event that can stand on its own: e.g. touchdown, interception, etc. in case of a Football game. Hence, our goal in this work is to construct a event-summary by selecting tweets, from the set of relevant tweets, that describe these sub-events. Here we list some characteristics and issues of this data that our proposed approach will have to deal with.

First, notice from Figure 3 that sub-events are marked by increased frequency of tweets. More precisely, it is the change in tweet-frequency as opposed to absolute levels that demarcate sub-events. Second, boundaries of sub-events also result in a change in vocabulary of tweets. The vocabulary specific to a sub-event tends to come from two sources: one part of it is sport specific comprising words like touchdown, interception, etc., while another part is game-specific, such as player names. Finally, we consider recurring sport events with every repeating event sharing at least the sport-specific vocabulary, and hence our approach should be able to learn from past games to do a better job of summarizing future ones. Some or all of these issues also pertain to other events such music festivals or award shows.

So the final question becomes whether existing methods address these issues. At all major search engines, the current methods for surfacing the content from twitter in response to search queries involve little more than returning the matching tweets in the reverse chronological order. This technique works well for many use-cases of twitter, but not for summarizing events, as it does not take into account the many of the properties of event coverage on twitter described above. We evaluated this technique as a baseline in our experiments in Section but do not report the numbers as it proved to be extremely uncompetitive. In the next section we propose a series of approaches that are designed to exploit the properties of coverage of events on Twitter.

Algorithms

Our goal is to summarize tweets about events in real-time, allowing for intelligent integration of tweets into search results. In this section, we will discuss three algorithms to accomplish this. In order to aid understanding of the issues involved and the design choices we make, we start by presenting the simplest approach and progressively repair its shortcomings using more sophisticated methods. Also, while the algorithms are given for structured long-running events in general, we continue to use the running example of sporting events to elucidate ideas.

Baseline: SUMMALLTEXT

The straightforward approach to summarizing tweets is to simply consider each tweet as a document, and then apply a summarization method on this corpus. In particular, we associate with each tweet a vector of the TF-logIDF of its constituent words (Baeza-Yates and Ribeiro-Neto 1999). Defining the distance between two tweets to be the cosine distance between them, we select those tweets which are closest to all other tweets from the event. This algorithm - we call it SUMMALLTEXT- is formally described in Algorithm 1.

Algorithm 1 SUMMALLTEXT

INPUT: Tweet corpus Z, tweet word vocabulary V, desired number of tweets n
OUTPUT: Set of key tweets T

for $i \in Z, w \in V$ **do**
  $\mathbf{z}_i(w) = \text{tfidf}(w, i, Z)$
end for
for $i \in Z$ **do**
  $\text{score}(i) = \sum_{j \in Z} \text{cosine}(\mathbf{z}_i, \mathbf{z}_j)$
end for
T = top n tweets with maximum score

[3] When Twitter users want to categorize their tweets they often use hash-tags; syntactically, these are words prefixed with "#" embedded in the tweet

ISSUES. While this algorithm has the advantage of simplicity, it has several defects. First, $O(|Z|^2)$ computations are required to compute the scores (though this could be reduced in practice if pruning techniques and approximations such as LSH are used). More importantly, the result set T is likely to be heavily biased towards the most popular sub-event, to the complete exclusion of other sub-events. The usual method used in such cases is to sequentially add those tweets to T that (a) have high scores, while (b) are diverse enough from the the tweets already added to T. However, simply using a text-based diversity function will not help in our case, since we want T to contains tweets for every major sub-event, even if those sub-events are of the same class — for example, in football, we want to get tweets for *all* touchdowns, even though the tweets describing each of these touchdowns are presumably quite similarly worded. In essence, a *time-based* diversity function is necessary.

Baseline: SUMMTIMEINT

In order to pick tweets from the entire duration of the event, we need to combine summarization with a segmentation strategy. The simplest idea is to split up the duration into equal-sized time intervals (say, 2 minutes for football), and then select the key tweets from each interval. However, clearly, not all intervals will contain useful sub-events. We detect such intervals by their low tweet volume relative to the average, and do not select any key tweets from such intervals.

The SUMMTIMEINT algorithm is described formally in Algorithm 2. It has two extra parameters: (a) a segmentation TS of the duration of the event into equal-time windows, and (b) the minimum activity threshold ℓ, whereby time segments where tweet volume is less than $\ell\%$ of all tweets are ignored. Both the window length and the threshold ℓ are to be picked experimentally with the goal of having no more than one complete sub-event in each time segment.

Algorithm 2 SUMMTIMEINT

INPUT: Tweet corpus Z, tweet word vocabulary V, desired number of tweets n, minimum activity threshold ℓ, time segments TS
OUTPUT: Set of key tweets T

$TS' = \{s \in TS \mid \text{tweet volume in segment } s > \ell\% \text{ of } |Z|\}$
for each segment $s \in TS'$ **do**
  $Z[s] = Z$ restricted to the same time as s
  $T_s = \text{SUMMALLTEXT}(Z[s], V, n/|TS'|)$
end for
$T = \bigcup T_s$

ISSUES. Algorithm ensures that the selected tweets are spread across the entire duration of the event. Diversity within any given time segment now becomes less useful, because only a few tweets are picked from each segment. However, the segmentation based on equal-sized time windows is far too simplistic, for three reasons discussed below.

Burstiness of tweet volume: Tweets come in bursts, and the durations of these bursts can vary. If the event is split into constant-time stages, one single long burst can be split into multiple stages, and the key tweets from each stage are likely to be near-duplicates. Conversely, if each stage is too long, it might cover several sub-events in addition to the bursty sub-event; since only a few tweets can selected from each time segment, some sub-events are likely to be missing from the final set of key tweets. The problem remains even if the event is split into stages with constant number of tweets in each stage. Clearly, none of these trivial solutions is satisfactory.

Multiple sub-events in the same burst: Even if bursts in tweet volume can be detected accurately, segmenting by burst volume can be misleading. For instance, if two sub-events occur close together in time (e.g., an "interception" followed soon after by a "touchdown") and both generate significant tweets, then their respective tweets might get smeared into one seemingly continuous burst in tweet volume. However, a careful examination of the word distribution of the tweets in this burst would reveal the presence of the two different sub-events. Thus, a good segmentation should consider changes in language model along with changes in tweet volume.

"Cold Start": At the beginning of the game, when the tweet language models are unknown and the thresholds defining "bursty" behavior are unclear, the segmentation algorithm could be very inaccurate. This is especially the case if the thresholds for bursts or changes in language model have to be learnt automatically. Thus, there is a significant risk of missing important sub-events that happen early.

We need a segmentation method that can overcome all these obstacles. A good segmentation will isolate at most one sub-event in each time segment, which can then be used by the summarizer to output summaries (or choose to ignore segments where little seems to be happening). In the next section, we propose a variant of the Hidden Markov Model (HMM) to solve this problem.

Our Approach: SUMMHMM

We have seen that there are two parts to event summarization: detecting stages or segments of an event, and summarizing the tweets in each stage. For instance, periods of significant tweeting activity (e.g, after a soccer "goal" or a football "touchdown") might be interleaved with lulls in tweet volume, or the predominant personality being discussed in the tweet-stream of an event could change from one proper name to another, and so on. Accurate knowledge of the boundaries between stages is crucial in finding the best tweets about the most important sub-events.

To segment an event, we turn to a model that has worked very well for many such problems: the Hidden Markov Model (HMM). The HMM is able to learn differences in language models of sub-events completely automatically, so that the model parameters are tuned to the type of event. The parameters of the HMM are easily interpretable as well. These advantages make it easy for the HMM to be applied to a wide variety of events, and make it our tool of choice for event segmentation. However, the standard HMM requires some modifications to be applicable to our problem, so we first present a short background on HMMs before discussing SUMMHMM.

BACKGROUND ON HMMS. The standard HMM posits the existence of N states labeled $S_1, \ldots, S_N$, a set of observation symbols $v_1, \ldots, v_M$, the probabilities $b_i(k)$ of observing symbol k while in state i, the probabilities a_{ij} of transitioning from state i to state j, and the initial state distribution π_i (Rabiner 1989). Starting from an initial state, the HMM outputs a symbol picked according to the symbol probabilities for that state, and then transitions to another state based on the transition probabilities (self-transitions are allowed). Now, given several sequences of symbols, one aims to learn the symbol and transition probabilities of the HMM that best fit the observed sequences. In our case, each state could correspond to one class of sub-events (e.g., "touchdown", "interception", "fumble", etc.), and the symbols are the words used in tweets. Thus, our event HMM models each event as a sequence of states, with tweets being the observations generated by the states. The variation in symbol probabilities across different states account for the different "language models" used by the Twitter users to describe different classes of sub-events, and the transitions between states models the chain of sub-events over time that together make up any given event.

Our Modifications We enhance the standard HMM with several modifications to devise SUMMHMM which is more relevant to event summarization.

OUTPUTS PER TIME-STEP. One difference between SUMMHMM and the standard HMM is immediately clear – the observation from a given state of SUMMHMM consists of all tweets for that time period (i.e., a multiset of symbols) instead of just one symbol, as in the standard HMM. This requires a simple extension of the standard model.

DETECTING BURSTS IN TWEET VOLUME. Another difference is that the standard HMM does not account for different rates of tweets over time, since it only outputs one symbol per time-step. Thus, it is unable to model bursts in tweet volume. Instead, we allow each state to have its own "tweet rate" which models the expected fraction of tweets in a game which come from that state. This allows for differentiation between states on the basis of the burstiness of the tweet-stream, which complements the differentiation based on the "language model" of state-specific symbol probabilities.

COMBINING INFORMATION FROM MULTIPLE EVENTS. Note that in order to learn the parameters of SUMMHMM, we require several observation sequences generated by it. In fact, if we build a SUMMHMM for just the current event, using only the tweets seen until now, then it will be very similar to an algorithm that simply detects "change-points" in a data stream. While change-point detection systems are very useful in practice, they suffer from the problem of cold-start: every time there is a change-point, a new model of the data must be learnt, so there is a time lag before the next change-point can be detected. Also, since the change-point system can only model the tweets it has seen so far, it can be slow to trigger when a new class of sub-event occurs, explaining away the first wave of tweets from the new sub-event as just the variability of the current model. Clearly, modeling each event by itself has disadvantages. This motivates learning the HMM parameters by training on *all* available events of a certain type (e.g., all football games in a season). Since all football games share the same classes of sub-events ("touchdown", "interception", etc.), combining the data from multiple events allows us to (a) better learn SUMMHMM parameters, and (b) better detect state transitions in a new game, thus solving the cold-start problem.

However, this also has the disadvantage that SUMMHMM has to account for tweet words that only occur in some of the events, but not in others. The most common example of this is proper names. For instance, tweets about two different football games will almost never share player names or team names. However, such proper names could be very important in distinguishing between states (e.g., certain players only play in defense or offense), and so ignoring names is not a solution. To account for this, we maintain *three* sets of symbol probabilities: (1) $\theta^{(s)}$, which is specific to each state but is the same for all events, (2) $\theta^{(sg)}$, which is specific to a particular state for a particular game, and (3) $\theta^{(bg)}$, which is a background distribution of symbols over all states and games. Thus, $\theta^{(s)}$ encapsulates different classes of sub-events, while $\theta^{(sg)}$ captures proper names and other symbols that can vary across games but still give some signal regarding the correct state; finally, $\theta^{(bg)}$ handles all noisy or irrelevant tweets. The standard HMM uses only $\theta^{(s)}$. This differentiation of symbol probabilities across specific events is another aspect of SUMMHMM that distinguishes it from the standard HMM.

Mathematical Formulation As in the standard HMM, let S represent the set of states, V the set of observation symbols (i.e., the tweet vocabulary), and a_{ij} the probability of transitioning from state i to state j. We shall use $i, j \in S$ to represent states, $x, w \in V$ to represent a symbol, g to represent one particular event, and t to represent a time-step. We assume that the fraction of tweet words in the corpus that are generated by state i is given by a Geometric distribution with parameter κ_i. Let $\phi_i^{(s)}$, $\phi_i^{(sg)}$, and $\phi_i^{(bg)}$ represent the probabilities that a symbol generated by state i is picked according to $\theta^{(s)}$, $\theta^{(sg)}$, or $\theta^{(bg)}$ respectively. Let $\theta_{iw}^{(s)}$, $\theta_{igw}^{(sg)}$, and $\theta_w^{(bg)}$ represent the probability of w being generated by the state-specific distribution for state i, the distribution for state i and game g pair, or the background distribution respectively. Together, $\Theta = \{a, \kappa_i, \phi_i^{(s)}, \phi_i^{(sg)}, \phi_i^{(bg)}, \theta_{iw}^{(s)}, \theta_{igw}^{(sg)}, \theta_w^{(bg)}\}$ is the set of unknown parameters of SUMMHMM.

Let Z be the sequence of observations from all the events in the corpus, X the sequence of states for each of those events, and W the sequence of symbol distributions ($\theta^{(s)}$, $\theta^{(sg)}$, or $\theta^{(bg)}$) from which all the symbols in Z are generated. Thus, the pair (X, W) is the set of hidden variables.

We also define some auxiliary variables. Let $N_{igw}^{(s)}$, $N_{igw}^{(sg)}$, and $N_{igw}^{(bg)}$ represent the number of instances in which w is generated by $\theta^{(s)}$, $\theta^{(sg)}$, and $\theta^{(bg)}$; these are multinomial random variables whose values are known once X and W are fixed. Let N_{ij} represent the number of transitions from state i to j; this too can be computed given X and W. Finally, let N_{gt} be the number of symbols in game g at time-step t, and let $N_g = \sum_t N_{gt}$; both of these can be computed

from the data Z.

Now, the complete data likelihood is given by:

$$P(Z,X,W|\Theta) = \left(\prod_{i,j} a_{ij}^{N_{ij}}\right) \times \left(\prod_{i,g} \pi_i^{I(X_{g1}=i)}\right)$$

$$\times \left[\prod_{i,g}\left((1-\kappa_i)^{\sum_w N_{igw}^{(s)}+N_{igw}^{(sg)}+N_{igw}^{(bg)}} \cdot \kappa_i\right)^{\frac{1}{N_g}}\right]$$

$$\times \left[\prod_{i,g,w}\left(\phi_i^{(s)}\cdot\theta_{iw}^{(s)}\right)^{N_{igw}^{(s)}} \cdot \left(\phi_i^{(sg)}\cdot\theta_{igw}^{(sg)}\right)^{N_{igw}^{(sg)}} \cdot \left(\phi_i^{(bg)}\cdot\theta_w^{(bg)}\right)^{N_{igw}^{(bg)}}\right]$$

where $I(.)$ is the indicator function.

Our goal is to find the best parameter setting Θ^* that maximizes $P(Z|\Theta)$. We do this by extending the EM algorithm to SUMMHMM. Define $\xi_{gt}(i,j)$ to be the probability that there is a transition from state i to state j at time t in game g. The computation of this value via recursion is a standard step in the Baum-Welch algorithm and for ease of exposition, we assume that it has already been computed (the detailed algorithm to compute it can be found in (Rabiner 1989)). Now, starting with an arbitrary setting of model parameters Θ, the EM algorithm iterates the sequence of Equations in Table 1 until convergence:

These steps give a locally optimal parameter setting for SUMMHMM. Typically, 15 iterations are enough for convergence. The number of states in SUMMHMM has to be picked by the user; we found that 10 states yield good results.

Algorithm Summary

SUMMHMM takes multiple events of the same type as input, and learns the model parameters Θ that best fit the data. Given Θ, the optimal segmentation of the events can be quickly found by the standard Viterbi algorithm (Rabiner 1989), which we do not describe here. Each segment can then be summarized, yielding the final set of top tweets for each event.

Note that only the computation of Θ is time-consuming, and this can be done periodically and offline. Then, for a new event, the segments can be computed online, using an old Θ, as new tweets are generated. Figure 3 gives the pseudo-code for our approach.

Algorithm 3 SUMMHMM

INPUT: Tweet corpus Z, tweet word vocabulary V, desired number of tweets n, minimum activity threshold ℓ
OUTPUT: Set of key tweets T

Learn Θ by iterating the equations in Table 1 until convergence
Infer time segments TS by the Viterbi algorithm (Rabiner 1989)
$TS' = \{s \in TS \mid$ tweet volume in segment $s > \ell\%$ of $|Z|\}$
for each segment $s \in TS'$ **do**
  $Z[s] = Z$ restricted to time s
  $T_s = \text{SUMMALLTEXT}(Z[s], V, n/|TS'|)$
end for
$T = \bigcup T_s$

$$\begin{aligned}
\gamma_{gt}(i) &= \sum_j \xi_{gt}(i,j) \\
\text{sum}_{igw} &= \theta_{iw}^{(s)}\cdot\phi_i^{(s)} + \theta_{igw}^{(sg)}\cdot\phi_i^{(sg)} + \theta_w^{(bg)}\cdot\phi_i^{(bg)} \\
\zeta_{iw}^{(s)} &= \sum_{g,t} I(Z_{gt}=w)\gamma_{gt}(i)\cdot\frac{\theta_{iw}^{(s)}\cdot\phi_i^{(s)}}{\text{sum}_{igw}} \\
\zeta_{igw}^{(sg)} &= \sum_{t} I(Z_{gt}=w)\gamma_{gt}(i)\cdot\frac{\theta_{igw}^{(sg)}\cdot\phi_i^{(sg)}}{\text{sum}_{igw}} \\
\zeta_{w}^{(bg)} &= \sum_{i,g,t} I(Z_{gt}=w)\gamma_{gt}(i)\cdot\frac{\theta_{w}^{(bg)}\cdot\phi_i^{(bg)}}{\text{sum}_{igw}} \\
\theta_{iw}^{(s)} &= \frac{\zeta_{iw}^{(s)}}{\sum_x \zeta_{ix}^{(s)}} \\
\theta_{igw}^{(sg)} &= \frac{\zeta_{igw}^{(sg)}}{\sum_x \zeta_{ixg}^{(sg)}} \\
\theta_{w}^{(bg)} &= \frac{\zeta_{w}^{(bg)}}{\sum_x \zeta_{x}^{(bg)}} \\
\phi_i^{(s)} &= \frac{\sum_w \zeta_{iw}^{(s)}}{\sum_w\left(\zeta_{iw}^{(s)}+\sum_g \zeta_{igw}^{(sg)}+\zeta_w^{(bg)}\right)} \\
\phi_i^{(sg)} &= \frac{\sum_{g,w} \zeta_{igw}^{(sg)}}{\sum_w\left(\zeta_{iw}^{(s)}+\sum_g \zeta_{igw}^{(sg)}+\zeta_w^{(bg)}\right)} \\
\phi_i^{(bg)} &= \frac{\sum_w \zeta_{w}^{(bg)}}{\sum_w\left(\zeta_{iw}^{(s)}+\sum_g \zeta_{igw}^{(sg)}+\zeta_w^{(bg)}\right)} \\
a_{ij} &= \frac{\sum_{g,t}\xi_{gt}(i,j)}{\sum_{g,t}\gamma_{gt}(i)} \\
\pi_i &= \frac{\sum_g \gamma_{g1}(i)}{\sum_{gj}\gamma_{g1}(j)} \\
\kappa_i &= \frac{\sum_{g,t}\gamma_{gt}(i)}{\sum_{g,t}\gamma_{gt}(i)\cdot(N_{gt}+1)}
\end{aligned}$$

Table 1: One single iteration of the EM algorithm.

Experiments

We first describe the experimental setup, and then in successive sections evaluate various aspects of our proposed algorithms.

Experimental Setup

In this section we describe the dataset used, the process of ground truth construction, and the baselines considered in this evaluation.

FINDING TWEETS RELEVANT TO SPORTS EVENTS.

Our goal in this paper was to summarize structured recurring events. For our experiments, we selected the sport of professional American Football. Football teams enjoy enormous popularity in the USA (see Figures 1 and 2) and play a large number of games with each other over a year, giving us an ideal test-bed to evaluate our proposed model.

Since our emphasis is on summarization, all algorithms assume that the problem of searching the tweets relevant to a user query about a sports event has already been solved. To simulate the perfect search process, we scanned Twitter feeds for the period of Sep 12th, 2010 to Jan 24th, 2011 for tweets containing the names of NFL teams: we noticed that on Twitter users often appended their posts about sport events with hash-tags containing the names of the teams involved. Hence, we collect the tweet-corpus relevant to the game between Green Bay Packers and Chicago Bears on Jan 23rd 2011 by finding all tweets during game-time that contain either *#packers* or *#bears*. We understand that our tweet-corpus is a subset of all the tweets pertinent to the sports event, and that a stronger search system might find other relevant tweets; however, constructing such a system is beyond the scope this work. Even with these constraints we obtained over 440K tweets over 150 games for an average of around 1760 tweets per game.

CLEANING THE TWEETS.

The event-tweets stream obtained by the process described above is very noisy. The two chief sources of noise are spam and tweets unrelated to the game. Spam-tweets can be easily removed since they almost always have a URL in them. We also do not consider users that have less than 2 or greater than 100 tweets for any one football game. Finally, we place similar thresholds on number of occurrences of words: we remove (Porter-stemmed) words that occur fewer than 5 times or in more than 90% of the tweets for the football game. While this removes most of the noise, we are still left with some tweets that do not strictly refer to the game under consideration: typically, user rants about their favorite teams, the game of football, or the world in general. After this cleaning, we only consider games with greater than 1500 tweets from at least 100 independent users: we are left with 53 of them. All experiments henceforth are conducted on this corpus.

OUR APPROACH AND BASELINES.

The technical details of our approach and baselines are given in the Algorithms section earlier. Here we give any implementation details and parameter settings.

SUMMALLTEXT: This approach constructs the game summary by finding the set of tweets that are close to all others in the corpus. We implemented this baseline by representing tweets via a TF-logIDF representation and used Cosine similarity as the comparison measure.

SUMMTIMEINT: This approach summarizes tweets as SUMMALLTEXT, but in each time window separately. It takes two parameters: we set $t = 120$ `secs` and $\ell = 1\%$. These parameters were set via evaluation on a held-out validation set (10% of ground truth).

SUMMHMM: This is our proposed HMM-based approach. We learn the underlying latent space of tweets by running 15 iterations with $K = 10$ states. Later in this section we will analyze the structure of these learnt hidden states. After finding the segments, the summary tweets are generated by calling SUMMTIMEINT with parameters $\ell = 1.5\%$ (tuned through the use of a held-out 10% validation set).

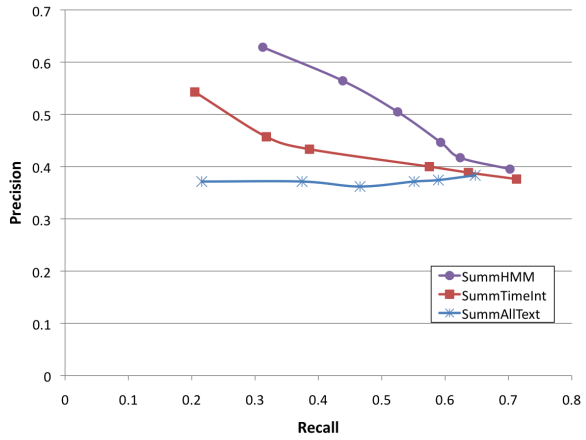

Figure 4: PRECISION-RECALL curves of our approach and baselines. To produce the curve the number of tweets was varied from $10 \ldots 70$ in increments of 10. Around the operating point of 30 tweets, SUMMHMM has a PRECISION@30 $= 0.5$ and a RECALL@30 $= 0.52$.

MANUAL GROUND TRUTH CONSTRUCTION.

Our proposed approach and baselines all output a set of tweets they consider as a good summary of the game. In order to evaluate these approaches we had human editors manually label every tweet output by them. Each output tweet was matched with the happenings in the game and labeled as *Comment-Play*, *Comment-Game*, or *Comment-General*. To be labeled as *Comment-Play* the tweet had to explicitly describe and occur with a few minutes of the play in question. The specific type of play (touchdown, field-goal, interception, fumble, etc.), was also noted in the label. These tweets were also assigned additional labels if they gave extra information, like current score, number of yards, and other contextual information (*Comment-Play-Details*), or names of the players involved (*Comment-Play-Names*). Tweets labeled *Comment-Game* typically described the state of the game at that point in time, such as the current score or reports of player injuries etc., and can be considered highly useful in a game summary. Tweets labeled *Comment-General* were statements that were not related to the football game under consideration. A total of 2175 tweets were manually labeled this way.

Constructing Play-by-Play Summary

Here we evaluate our approach (SUMMHMM) and baselines, SUMMTIMEINT and SUMMALLTEXT, on the task of obtaining a useful play-by-play summary of football games. For these results we use the set of manually labeled tweets described above. To report performance we resort to measures that are standard in research in information retrieval. We want an ideal game summary to contain all *important plays* in football, like touchdowns, field goals, interceptions, and fumbles. Hence, we define RECALL of an approach as the fraction of such important plays in a game that are found by it. Further, we want the ideal game summary to include as few irrelevant tweets as possible: the screen real-estate to

display the summary and the user's attention are both limited. Hence, we define the PRECISION of an approach as the fraction of its output tweets that are labeled *Comment-Play* or *Comment-Game*. Note that, as an approach outputs more tweets typically the PRECISION will become smaller and the RECALL will increase. Hence, we report PRECISION and RECALL by varying the number of output tweets from $10 \ldots 70$ in increments of 10.

EVALUATION AT OPERATING POINT.

In Figure 4 we plot the PRECISION-RECALL curves of SUMMHMM and the baseline methods. First thing to notice is that performance of SUMMHMM dominates the performance of the SUMMTIMEINT and SUMMALLTEXT over the whole set of operating points. As it is difficult to show the full set of 70 output tweets to users, owing to limited user attention and screen real-estate, the operating point of a deployed system is likely to be lower. In the more realistic operating ranges $10 - 30$ the performance of SUMMHMM is significantly higher than both SUMMTIMEINT and SUMMALLTEXT. In the middle of that range in terms of PRECISION@20 and RECALL@20, SUMMHMM is 25% and 16% better, respectively, than the nearest competitor.

RECALL OF DETAILS OF PLAYS.

Here we compare the performance of SUMMHMM and baselines in terms of whether the output tweets give context and details around the play. In Figure 5 we plot the measure RECALL@30 on the task of retrieving *Comment-Play*, *Comment-Play-Details*, and *Comment-Play-Names* labeled tweets. To be clear, RECALL@30 for *Comment-Play-Details* is the fraction of important plays in a game that are matched by some top-30 tweet that has been labeled *Comment-Play-Details* by the editors. If an important play is matched with some tweet in the top-30 that is only labeled *Comment-Play* or *Comment-Play-Names*, then it does not count towards RECALL@30 for *Comment-Play-Details*. From the figure, first note that SUMMHMM outperforms the baselines in all three tasks. In fact, as the tasks become harder the performance difference between SUMMHMM and the nearest competitor is higher as well: SUMMHMM beats others by more than 33% on finding *Comment-Play-Details* and by 26% on finding *Comment-Play-Names*.

Second, note that the performance of SUMMHMM on finding tweets labeled *Comment-Play-Details* and *Comment-Play-Names* is lower than finding tweets that are labeled *Comment-Play*. This is due to the fact that while twitter users are remarkably consistent on how they refer to plays such as touchdowns and field-goals, there is a much larger variability when they refer to details such number of yards gained or names of players involved. Player names in particular lend themselves to misspellings and abbreviations as well as ambiguities; most twitter users are posting through mobile devices. As an extreme example, apart from his correct name, the Ravens player "T J Houshmandzadeh" is referred to as "tj", "tjh", "houzhma", "houshma", "houshmoundzadeh" in the tweets. Also, there are 19 players in 14 teams with the name "Jackson". This extremely variability and ambiguity in the names is the reason the RECALL performance on retrieving the name is low. As future work we will consider special algorithms for resolving some of these issues surrounding names before deploying a system like this.

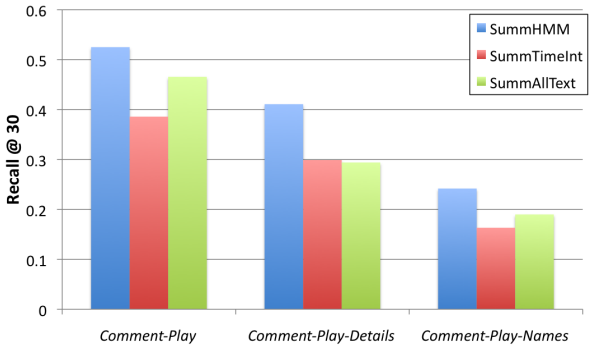

Figure 5: RECALL performance of our approach and baselines on finding tweets with context and details. As we can see the task of finding tweets labeled *Comment-Play-Details* and *Comment-Play-Names* is more difficult and the performance improvement due to SUMMHMM is higher.

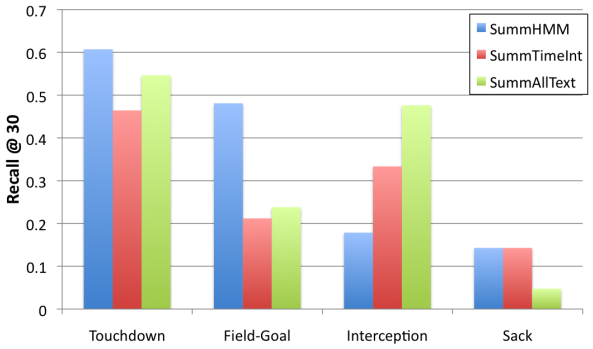

Figure 6: RECALL performance of our approach and baselines on finding tweets about different types of plays. As expected, scoring plays such as touchdowns and field-goals are easier to find than others.

RECALL OF TYPES OF PLAYS.

Here we discuss the ability of the various approaches in finding tweets that match different types of plays in the game. In Figure 6 we plot the RECALL@30 measure when restricted to finding just the plays of the types listed on the x-axis. As we can see all methods are strong when retrieving key touchdown-plays since these are the primary scoring mechanisms in American Football and precisely the plays which generate most user tweets. However, a second critical scoring mechanism, field-goal-plays, proves much harder for baselines while SUMMHMM detects 50% of them in the top-30 output tweets. The fraction of touchdown-plays detected in the top-30 tweets is around 60% while the number rises to almost 90% in the top-70 tweets.

An observation about a key result in Figure 6 gives insight into a strength and a weakness of our approach. As we

State label	Top words in θ^s	Top words in θ^{sg}
TOUCHDOWN	stop, catch, drive, great, pass	mark_sanchez
FIELD-GOAL	miss, kick, kicker, no_good, score	nick_folk
INTERCEPTION	int, throw, pick, touchdown, defense	mark_sanchez
DEFENSE & FUMBLE	sack, good, punt, stop, block	darrelle_revis
PENALTY & FUMBLE	challenge, run, hold, punt, call	darrelle_revis

Table 2: The label and top words from 5 hidden states learnt by SUMMHMM. The state label in column 1 is picked from among the top state-specific words listed in column 2. Due to paucity of space we only give the top (state, game)-specific word for each hidden state averaged over all games of the Football team Jets.

can see, SUMMHMM performs significantly worse on the task of finding interception-plays than the baselines. This is because our approach makes a critical assumption that after segmentation via HMM there is but one key play within each segment. This is often a strength since this lets us use the segment specific models to retrieve the play, but can sometimes turn into a weakness when the assumption is violated. In the case of interception-plays, they are oftentimes followed immediately by other scoring plays: in our data around 45% of interception-plays were followed within minutes by touchdown-plays or field-goal-plays. Hence, the assumption made by SUMMHMM is violated in the case of interception-plays. One way to tackle this issue is ensure diversity amongst tweets when selecting them from a segment; we plan to attempt this in future work.

REASONS FOR PRECISION@30 =0.5.

As we pointed out earlier in the deployment operating region of returning around 30 tweets, the PRECISION $= 0.5$. While this seems a little low as a number, upon inspection of the tweets output by the various approaches it is clear that the editors enforced a very strict standard in the judgments. A tweet was deemed relevant only if it referred to a specific identifiable play in the game. For example, tweets encouraging the teams such as "let's convert on 3rd down #packers" and giving personal opinions on the game such as "i hate that was the right call by the refs #packers" were deemed irrelevant. We believe that including these tweets into the summary adds color and drama to it, and hence these tweets should be scored relevant. However, the true worth of any of these presentations can only be determined once we observe how users interact with them; we hope to run this amended evaluation as part of the future work.

Anecdotal Evidence and Discussion

We have seen that SUMMHMM is very effective in generating play-by-play summaries of American Football games. In this section we provide anecdotal evidence of the system's performance as well as discussions of issues in summarizing events using Twitter updates.

HIDDEN STATES OF THE LEARNT HMM.

In Table 2 we display 5 of the 10 hidden states used in learning SUMMHMM and give their top state-specific as well as state-game-specific words. The rest of the states either acted as duplicates of these states, or had uninterpretable term distributions; this is common in learning hidden state models where the number of underlying hidden states has to be guessed. First, notice that these states do correspond to the different types of plays in American Football. Second, while the state-specific words capture the words that tweets use to describe the play in general, the words specific to the (state, game) pairs are typically player names that change from game to game. Column three of Table 2 shows the top (state, game) words averaged over all games of the Jets team. The results make complete sense as Mark Sanchez is the team's quarterback (or main offensive player) and Darrelle Revis seems to be their main defensive player. Nick Folk, the team's kicker is the top word for the state FIELD-GOAL. Finally, some of related play-types are represented by the same underlying hidden state. This happens because either these plays are referred to using similar words, or the same players participate in them, or they often occur very close together in time. Hence, we see that SUMMHMM finds a cohesive underlying hidden structure of Football games.

GENERALIZATION TO OTHER TYPES OF EVENTS.

SUMMHMM has been designed and implemented to generate good summaries for structured and recurring sports events, but its functionality is a strict super-set of those needed to work on other types of events. American Football has a very discrete nature with significant events well separated in clock-time, while other sports like soccer have a much more continuous game-flow; this can be challenging for the baselines described in this paper. We have tested SUMMHMM on soccer matches from World Cup 2010 and obtained results similar to American Football games. Other types of events such as music festivals and award-shows, or even sudden news events such as the Tunisian revolution, can benefit from a Twitter-based summary though such events are not repeated often. We expect that SUMMHMM will be able to handle these events, however, some of its features will be under-utilized. As future work we would like to apply our approach to these other types of events.

RELIANCE ON EFFECTIVE SEARCH.

In this work we assume that the initial process of event detection and retrieval of a relevant set of tweets is accomplished: SUMMHMM works on this set of relevant tweets. There exist many works on event detection on Twitter (Becker, Naaman, and Gravano 2010; Sakaki, Okazaki, and Matsuo 2010; Chen and Roy 2009) and search engines currently do a decent job of retrieving relevant tweets; hence, we believe this is a reasonable assumption. However, even if the retrieved set of tweets is noisy, we believe that SUMMHMM will be able to construct clean summaries. This is because SUMMHMM constructs summaries by forming a underlying model of the event. This model is most heavily influenced by the major trends in the tweet stream; any outlier tweet (erroneously retrieved) that does not correspond to one of the larger tweet clusters is ignored. We see this in the data used for experiments in Section . The data collection process is very simple and just looks for tweets containing "#team-name"; this results in the collection of many irrelevant tweets that are not removed even by our cleaning approach. However, these tweets end up being ignored by SUMMHMM when learning the model and constructing the summaries.

Related Work

Here we describe some of the related existing research work and discuss how our work differs from it.

MICROBLOGGING AND TWITTER.

There has been much recent interest on identifying and then tracking the evolution of events on Twitter and other social media websites, e.g., discussions about an earthquake on twitter (Sakaki, Okazaki, and Matsuo 2010), detecting new events (also called first stories) in the tweet-stream (Petrović, Osborne, and Lavrenko 2010), visualizing the evolution of tags (Dubinko et al. 2006), and other events on Flickr, Youtube, and Facebook (Becker, Naaman, and Gravano 2010; Chen and Roy 2009). The problem has also been approached from the point of view of efficiency: (Luo, Tang, and Yu 2007) propose indexing and compression techniques to speed up event detection without sacrificing detection accuracy. However, to the best of our knowledge, we are the first to study the *summarization* of events using user updates on Twitter. We assume that the event detection has already been performed, possibly using one of the aforementioned techniques; our goal is to collate all the information in the tweets and present a summarized timeline of the event.

SUMMARIZATION.

Summarization of text documents has been well studied in the IR community. A common method is based on computing relevance scores for each sentence in the document and then picking from among the best (Gong and Liu 2001). More complex methods are based on latent semantic analysis, hidden markov models, deep natural language analysis, among others (Gong and Liu 2001; Das and Martins 2007). However, these are primarily aimed at standalone webpages and documents. While the summarization of a sequence of tweets can be seen as a form of text summarization, there are nuances that are not considered by most prior algorithms. For instance, each tweet is very short (at most 140 characters), spelling and grammatical mistakes abound, sentences are often just phrases, and some tweets include snippets of others ("re-tweets"). In addition, the sequence of tweets for an event is created by the activity of a multitude of users instead of a single author, and hence there is little continuity in the tweet-stream. Facing such conditions, researchers have turned to simpler heuristics, such as by using the words before and after known topic-specific phrases to expand the set of phrases relevant for summarization (Sharifi, Hutton, and Kalita 2010). We also empirically found the simplest methods to work best, and in our experiments we used such a summarization method. The main contribution of our work is in modeling the underlying hidden structure of events; most any off-the-shelf summarization method can then be then used to extract the important tweets to construct the summary.

SEQUENTIAL MODELING.

There are a variety of methods to model sequence information. Change-point models try to detect instants of time when there is some marked change in the behavior of the sequence, e.g., the intensity with which observations arrive suddenly changes (Mannila and Salmenkivi 2001; Kleinberg 2002). Thus, change-points give a segmentation of the sequence into chunks of different intensities. Another way to segment the sequence is on the basis of differences in distribution of the observations themselves. A classical technique for this is the Hidden Markov Model (HMM) (Rabiner 1989; Wang et al. 2010), which posits the existence of an underlying process that transitions through a sequence of latent states, with observations being generated independently by each state in the sequence. HMMs have been extremely successful in areas as varied as speech recognition, gesture recognition, and bioinformatics.

As we showed earlier in this paper, the tweet-streams for sports events are very bursty, and segmentation into bursty episodes is necessary before accurate summarization can be performed. Segmentation based on the words and phrases appearing in the tweet-stream is just as critical, since a single high-intensity burst can actually be composed of many "sub-events," each with its own vocabulary and word distribution. In addition, for repeating events (such as sports), modeling the tweets from all the events can lead to better accuracy than modeling each event in isolation. None of the aforementioned techniques achieves all of these goals. Hence, we developed a modified HMM that can account for (a) shifts in intensity, and (b) shifts in the language model, both over time in a given event, and also across events.

Summary and Future Work

In this work we tackled the problem of constructing real-time summaries of events from twitter tweets. We proposed an approach based on learning an underlying hidden state representation of an event. Through experiments on large scale data on American Football games we showed that SUMMHMM clearly outperforms strong baselines on the play-by-play summary construction task, and learns a underlying structure of the sport that makes intuitive sense. As future directions of research we would like to test SUMMHMM on long-running one-shot events such as festivals and award shows, and to provide summaries for important but unpredictable events such as revolutions and natural disasters. Finally, we have not yet evaluated the summaries generated by our approach in real-time on search engine users; this is something we hope to do in the future.

References

Baeza-Yates, R. A., and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Boston, MA: Addison-Wesley Longman.

Becker, H.; Naaman, M.; and Gravano, L. 2010. Learning similarity metrics for event identification in social media. In *WSDM*.

Chen, L., and Roy, A. 2009. Event detection from flickr data through wavelet-based spatial analysis. In *CIKM*.

Das, D., and Martins, A. 2007. A survey on automatic text summarization.

Dubinko, M.; Kumar, R.; Magnani, J.; Novak, J.; Raghavan, P.; and Tomkins, A. 2006. Visualizing tags over time. In *WWW*.

Gong, Y., and Liu, X. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *24th SIGIR*.

Kleinberg, J. 2002. Bursty and hierarchical structure in streams. In *KDD*.

Luo, G.; Tang, C.; and Yu, P. 2007. Resource-adaptive real-time new event detection. In *SIGMOD*.

Mannila, H., and Salmenkivi, M. 2001. Finding simple intensity descriptions from event sequence data. In *KDD*.

Petrović, S.; Osborne, M.; and Lavrenko, V. 2010. Streaming first story detection with application to twitter. In *HLT*.

Rabiner, L. 1989. A tutorial on hmm and selected applications in speech recognition. *Proc. of the IEEE* 77(2):257–286.

Sakaki, T.; Okazaki, M.; and Matsuo, Y. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *WWW*.

Sharifi, B.; Hutton, M.-A.; and Kalita, J. 2010. Summarizing microblogs automatically. In *HLT*.

Wang, P.; Wang, H.; Liu, M.; and Wang, W. 2010. An algorithmic approach to event summarization. In *SIGMOD*.