Improving Tweet Stream Classification by Detecting Changes in Word Probability

Kyosuke Nishida, Takahide Hoshide
NTT Service Evolution Laboratories, NTT Corporation, Kanagawa, Japan
{nishida.kyosuke, hoshide.takahide}@lab.ntt.co.jp

Ko Fujimura
Otsuma Women's University
Tour, Japan
fujimura@otsuma.ac.jp

ABSTRACT
We propose a classification model of tweet streams in Twitter, which are representative of document streams whose statistical properties will change over time. Our model solves several problems that hinder the classification of tweets; in particular, the problem that the probabilities of word occurrence change at different rates for different words. Our model switches between two probability estimates based on full and recent data for each word when detecting changes in word probability. This switching enables our model to achieve both accurate learning of stationary words and quick response to bursty words. We explain how to implement our model by using a word suffix array, which is a full-text search index. Using the word suffix array allows our model to handle the temporal attributes of word n-grams effectively. Experiments on three tweet data sets demonstrate that our model offers statistically significant higher topic-classification accuracy than conventional temporally-aware classification models.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications—Data mining; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms
Algorithms, Experimentation

Keywords
Twitter, Document classification, Data streams, Concept drift, Control charts, Suffix arrays

1. INTRODUCTION
Twitter, a micro-blogging service, has emerged as a new information infrastructure. Over 300 million status messages, called tweets, are posted per day by users all over the world. The demand for classifying tweets in order to organize the massive volume of tweets has become more urgent.

Classifying document streams is a challenging problem because their statistical properties will change over time. This characteristic is called concept drift, and it is one of the hottest topics in data mining [2, 13]. There are multiple types of changes, and the effectiveness of strategies for building classifiers depends on the types of changes.

For classifying tweet streams, the trade-off between quick response to bursty words and accurate learning of stationary words is a significant problem. Conventional methods conduct temporal selection or weighting on documents; so, they cannot handle the different time-scale changes expected for each word. We propose a probabilistic classification model, P-Switch, as an extension of the multinomial naive Bayes classifier. In order to resolve the trade-off, our model switches between the two probability estimates based on full and recent data for each word, by monitoring changes in the probability of word occurrence. It implements monitoring by using a control chart [20], which is a statistical process control tool.

We then describe an implementation method that uses a word suffix array [7]. Our classification model can be formalized as searching the positions of word n-grams of a new tweet, in the training documents that are concatenated chronologically. We, therefore, regard the construction of full-text search indexes as a learning procedure. This approach can effectively handle word n-grams, which are highly important information given the 140 character limit of tweets.

Experiments on topic classification were conducted on three real-world tweet data sets that have different characteristics in terms of topics, languages, and changes. Results show that our model offers statistically significant improvements over the standard multinomial naive Bayes classifier and conventional document weighting and selection methods in terms of accuracy and macro F1 measures.

Contributions: (i) analysis of the changes in tweet streams, (ii) proposal of a probabilistic classification model that adapts to and detects changes at word-level, (iii) an implementation approach that uses word suffix arrays for effectively learning temporal factors in word n-grams, and (iv) significantly improved topic classification accuracy for tweet streams.

Outline: §2 describes the problem definitions on document stream classification. §3 presents the results of our analysis and preliminary experiments on tweet streams. §4 explains our proposed model. §5 describes the implementation method. §6 presents the results of the experiments. §7
describes related work and §8 summarizes the contributions of this study.

2. CLASSIFYING DOCUMENT STREAMS

We address previous work on document stream classification and explain the scope of this study.

2.1 Problem Definitions

Problem definitions for document stream classification can be roughly classified into three types depending on how the training and test examples are presented:

[P1] Test-Then-Train (Incremental) [6, 35, 11, 38]
At each time step \( t = 1, 2, \ldots \), a classifier is given a new document \( d_t \), and outputs a class prediction, \( \hat{c}_t \), for \( d_t \). It then updates itself based on the true class label, \( c_t \).

[P2] Test-Then-Train (Chunk) [17, 16, 15, 30, 18, 12]
At each \( t = 1, 2, \ldots \), a classifier is given a new training document set, \( D_t = \{d_{t,1}, d_{t,2}, \ldots\} \), and outputs a class prediction, \( \hat{c}_{t,j} \), for each \( d_{t,j} \in D_t \). It then updates itself based on the true class label set, \( C_t = \{c_{t,1}, c_{t,2}, \ldots\} \).

[P3] Train-Then-Test [8]
At each \( t = 1, 2, \ldots \), a classifier is given a new training document set, \( D_t^* = \{d_{t,1}^*, c_{t,1}^*\}, \ldots\), and a new test document set, \( D_t = \{d_{t,1}, d_{t,2}, \ldots\} \). It updates itself based on \( D_t^* \) and then outputs \( \hat{c}_{t,j} \) for each \( d_{t,j} \in D_t \).

Temporal-aware classification methods for batch learning have also been proposed [21, 5, 28, 29]: a classifier is given \( D_t^* \) and \( D_t \) for all \( t = 1, 2, \ldots, T \). It outputs a class prediction, \( \hat{c}_{t,j} \), for each \( d_{t,j} \in D_t \), based on \( (\forall t) D_t^* \).

2.2 Concept Drift

Concept drift means that the statistical properties of data change over time [36, 34], and it creates the most difficult situation in document stream classification. There are three major types of concept drift: [C1] sudden shift, [C2] gradual drift, and [C3] recurring themes.

One example of concept drift is found in the spam mail filtering problem because mail is characterized by skewed and changing class distributions, users’ dynamic and growing interests, recurring and periodic themes (e.g., Christmas-related spam), and intelligent and adaptive adversaries.

2.3 Learning Strategies

Previous studies on handling concept drift in document streams employ instance selection, instance weighting, ensemble, and feature selection methods.

Instance selection methods select and adapt to previous training examples [16, 15, 11, 38]. Sliding window is a typical approach that is used to select the most recent previous examples. Klinkenberg et al. proposed a method for automatically deciding the size of window and demonstrated high classification performance in the topic classification of news articles [15].

Instance weighting methods assume that newer instances are more essential for classifying the current instance [15, 18, 5, 11, 28, 29]. Lebanon et al. proposed a naive Bayes classifier with a weighting scheme for handling temporal document streams [18].

These instance selection and weighting methods will adapt well to C2, but often fail to handle C1, because learning with old data worsens the classification of most recent data when sudden changes occur. There is a trade-off between response quickness, needed for C1, and learning accuracy, needed for C2. For example, using a short window improves response quickness but fails to offer good learning accuracy.

Ensemble methods use a set of classifiers adapted to the current concept [8, 35, 30, 12]. New models are added, and old or poor models are removed, in a fixed period of time or when a decrease in classification accuracy is observed. Ensemble methods are effective for C2 and C3; e.g., Katkis et al. proposed a general framework for tracking recurring contexts using ensemble classifiers and improved email filtering accuracy [12].

Feature selection methods dynamically select the most valuable feature (word) set in order to respond to concept drift, which is characterized by changes to the underlying feature space [6, 35, 11]. Feature selection methods are frequently combined with other methods; e.g., Wenerstrom et al. [35] proposed a feature-adaptive ensemble approach, which uses multiple classifiers with different features.

2.4 Scope of this Study

This study handles the incremental test-then-train problem definition P1, the most basic problem definition, and tackles the trade-off between responding to sudden shift (C1) and gradual drift (C2). We propose a classification model, a new weighting method at the word level, for tweet streams. While conventional instance weighting methods use temporal weighting functions on documents, our model is able to respond to changes in each word. Extending our model to suit ensemble and dynamic feature selection methods in order to handle recurring themes (C3) is our future work.

3. TWEET STREAMS

This section provides the results of analysis and preliminary experiments on the topic classification of tweets, where topics are defined by hashtags.

3.1 Data set

We collected three tweet data sets as shown in Table 1: tweets on 12 Nippon (Japanese) professional baseball teams (NPB), tweets on 7 Japanese television networks (TV), and tweets on 30 major league baseball teams (MLB), by searching hashtags via the Twitter Streaming API (statuses/filter), from 13 September 2011 to 26 September 2011.

These data sets consist of tweets that include a single hashtag; that is, a tweet belongs to a single class. Classes are hashtags on baseball teams for the NPB and MLB data sets, and hashtags on television networks for the TV data set. Retweets (including the tweets containing the string “RT” or “QT”), replies, mentions are not included in these data sets. In the problem definition P1, document \( d \) means a tweet text without any hashtags, and class label \( c \) means a hashtag. For the NPB and TV data sets, we selected noun, verb, and adjective words by using MeCab 0.98 with IPA dictionary\(^1\) for Japanese word segmentation and part of speech labeling. For the MLB data set, we removed stop words and treated punctuation and other non-alphabetic characters as separate tokens.

\(^1\)http://mecab.sourceforge.net
Table 1: Tweet data set. We collected tweets that included a single hashtag from 13 to 26 September 2011 by using Twitter Streaming API. Classes are hashtags on “baseball teams” for NPB and MLB data sets, and hashtags on “television networks” for TV data set. Each hashtag corresponds to a class label.

<table>
<thead>
<tr>
<th>Data</th>
<th># Tweets</th>
<th># Classes</th>
<th>Language</th>
<th>Track Keywords (Hashtags)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB</td>
<td>314,210</td>
<td>12</td>
<td>Japanese</td>
<td>#dragons, #hanshin, #giants, #swallows, #baystars, #sbhawks, #seibulions, #chibalotte, #lovefighters, #orix, #buffaloes, #rakuteneagles</td>
</tr>
<tr>
<td>TV</td>
<td>245,080</td>
<td>7</td>
<td>Japanese</td>
<td>#nhk, #etv, #ntv, #tvasahi, #tbs, #tvtokyo, #fujitv</td>
</tr>
<tr>
<td>MLB</td>
<td>200,521</td>
<td>30</td>
<td>English</td>
<td>#angels, #astros, #athletics, #bluejays, #braves, #brewers, #cubs, #dbacks, #dodgers, #indians, #mariners, #marlins, #mets, #orioles, #padres, #phillies, #pirates, #rangers, #rays, #reds, #redsox, #rockies, #royals, #4giants, #stlcardinals, #tigers, #twins, #whitesox, #yankees</td>
</tr>
</tbody>
</table>

3.2 Analysis Results

We analyzed tweet streams in terms of the changes in class distribution and word probability, the occurrence of new words, and the effectiveness of considering word n-grams.

3.2.1 Changes in Class Distribution

Fig. 1 shows (a) the cosine similarities between class distributions on 13 September 2011 and all other days and (b) the total number of tweets for each day. We can see that the class distribution on the NPB and MLB data sets fluctuate more than that on the TV data set. The TV data set also fluctuates dramatically as shown in Fig. 2, which shows the class distribution over ten minute periods.

In summary, the class distributions in tweet streams fluctuate constantly and dramatically at micro and macro time scales. Classification models have to respond to these abrupt and gradual changes in class distribution.

3.2.2 Changes in Word Probability

Fig. 3 shows (a) the hourly number of tweets containing “home-run” and (b) the daily number of tweets containing “yokohama” from the NPB data set. “Home-run” is a bursty word since its relevant classes change rapidly. For example, a player of the team #chibalotte hit a home-run at 4PM and the word probability for “home-run” increased suddenly (Fig. 3a). On the other hand, “yokohama” is a relatively stationary word. The most relevant class #baystars, referring to the baseball team that has its home ground in Yokohama, does not change (Fig. 3b).

In summary, the statistical properties of tweet topics change at different rates for different words. Conventional instance weighting and selection methods cannot handle these differences.

3.2.3 Occurrence of New Words

In order to analyze the occurrence and change of effective words for classifying tweets, we calculated the chi-squared statistics $\chi^2_c$ (see details in [39]) of all words for each class $c$ in each day, and got the daily top 100 words in terms of $\chi^2_c$ for each class.

Fig. 4 shows (a) the cosine similarities between the daily top 100 words on 13 September 2011 and all other days and (b) the cumulative number of distinct daily top 100 words. The results of Fig. 4 are averaged over classes.

We can see from Fig. 4b that effective words for classifying
These results are averaged over days. Chi-squared statistics from the MLB data set, "white sox" is more suitable for classifying tweets or not. We first calculated the influence of the day of the week, as shown in Fig. 4a. The temporally-aware multinomial naive Bayes classifier (Eqs. (1)–(5)) for identifying the class #whitesox than "white". Moreover, Fig. 5b shows that the distribution on top 200 word n-grams in terms of max, contains word bigrams and trigrams (37.6%, 52.0%, 28.1% for the NPB, TV, and MLB data sets).

In summary, considering word sequences can be effective for classifying tweets because tweets are short and limited to 140 characters.

### 3.3 Preliminary Experiments

We applied a temporally-aware multinomial naive Bayes classifier, which was proposed by Salles et al. [29], and independently, by Lebanon et al. [18], to problem definition P1.

The temporally-aware multinomial naive Bayes classifier is given by:

\[
\hat{c}_t \leftarrow \arg \max_c \left\{ p_K(c|t) \prod_{w_i \in W(d_t)} p_K(w_i|c, t) \right\},
\]

\[
p_K(c|t) = \frac{\sum_{\tau=1}^{t} K_h(t-\tau)[c_{\tau} = c]}{\sum_{\tau=1}^{t} K_h(t-\tau)},
\]

\[
p_K(w_i|c, t) = \frac{\sum_{\tau=1}^{t} K_h(t-\tau)f_{c, \tau}(w_i)}{\sum_{\tau=1}^{t} \sum_{w \in d_{\tau}} K_h(t-\tau)f_{c, \tau}(w)},
\]

where \([\cdot]\) is 1 if the predicate is true and 0 otherwise, \(W(d_t)\) is the set of selected words in document \(d_t\), and \(f_{c, \tau}(w)\) is the frequency of word \(w\) in \(d_{\tau}\). \(K_h(t-\tau)\) is a kernel function such that:

\[
K_h(t-\tau) = \| t - \tau < h \|
\]

\[
K_h(t-\tau) = (1 - (t - \tau)/h) \cdot \| t - \tau < h \|
\]

where \(h\) is the width of the kernel. Using Eq. (4) means adopting an instance selection method and using Eq. (5) means adopting an instance weighting method. The proposal of Salles et al. [29] learns a kernel function with a training data set; however, its learning cost for P1 is extremely high. Accordingly, we used the above kernel functions in this preliminary experiment. We note that Lebanon et al. reported that Eq. (5) was the best kernel function in their experiments [18].
We evaluated the macro $F_1$ measure of the instance selection and weighting methods while changing the value of $h$ (Fig. 6). Feature selection was conducted by using the chi-squared statistics $\chi^2$ based on documents $\{d_t \mid t-h < \tau < t\}$; that is, $W(d_t) = \{w \mid \max_i(\chi^2_w) > 30.0\}$.

For the TV data set, decreasing kernel width $h$ improved classification accuracy. This data set exhibited significant changes in class and word probabilities, and new and important words emerged constantly (see Figs. 2 and 4); therefore, learning from newer tweets was important. The instance weighting method achieved better classification accuracy than the instance selection method in which all data within the kernel are equivalent. We note that macro $F_1$ is not a monotonic function of kernel width because of the existence of recurrent themes, as shown in Fig. 6b.

For the NPB and MLB data sets, decreasing the kernel width worsened classification accuracy. The instance selection and weighting methods cannot learn stationary words. This switching of the two probability estimates for recurrent themes, as shown in Fig. 6b, responding to such shifts is important for achieving further improvements in classification accuracy.

### 3.4 Data Characteristics Considered

From the results of data analysis and preliminary experiments, an ideal classification model for handling concept drift in tweet streams should be able to: [A1] respond to changes in class distribution, [A2] respond to changes in word probability, [A3] handle the emergence of new words effectively, [A4] resolve the trade-off between quick response to bursty words and accurate learning of stationary words, [A5] consider word n-grams, and [A6] treat recurring themes.

We propose a method that realizes the first five abilities, A1 to A5. A6 lies outside the scope of this study.

### 4. PROPOSED MODEL

We propose a classification model of tweet streams in which statistical properties will change over time. Our model, called P-Switch, is based on the analysis results shown in §3.

#### 4.1 Outline

At each time step $t = 1, 2, \ldots$, our proposed model outputs a class prediction, $\hat{c}_t$, for a new given document, $d_t$, as follows:

$$\hat{c}_t \leftarrow \arg \max_c \left( p(c|t) \prod_{w_i \in W(d_t)} p(w_i|c, w_{i-1:t-1}) \right), \quad (6)$$

where $w_i$ is the $i$-th word in $d_t$, $w_{i-1:t-1}$ is the word sequence from the $(i-n+1)$-th to $(i-1)$-th word in $d_t$, and $W(d_t)$ is the set of selected words in $d_t$. Our model then updates itself based on the true class label $c_t$. Our model, Eq. (6), is a temporally-aware version of the multinomial naive Bayes classifier. The following sections describe three essential components: class distribution, word probability, and word n-gram probability.

#### 4.2 Class Distribution

To acquire ability A1, our model incrementally updates the class distribution, $p(c|t)$, based on $c_t$:

$$p(c|t) = (1 - \gamma) p(c|t-1) + \gamma[c_t = c], \quad (7)$$

where $[\cdot]$ is 1 if the predicate is true and 0 otherwise.

Eq. (7) is the class distribution estimated using an exponentially weighted moving average (EWMA), where $\gamma$ is a smoothing parameter that represents the degree of weighting decrease ($0 < \gamma < 1$). The weighting for each older data point decreases exponentially, it never reaches zero.

### 4.3 Word Probability

To acquire abilities A2, A3, and A4, our model estimates the temporally-aware word probability for each word.

First, our model chronologically concatenates documents given until $t$ for each class $c, D_{c,t}$.

A standard multinomial naive Bayes classifier estimates the word probability of $w_i$ in class $c$ with the maximum likelihood (ML) method,

$$p_{\text{ML}}(w_i|c, t) = \frac{f_c(w_i)}{\sum_{w \in D_{c,t}} f_c(w)}, \quad (8)$$

where $f_c(w)$ is the frequency of $w$ in $D_{c,t}$.

Additionally, our model estimates the word probability with the EWMA method,

$$p_{\text{EWMA}}(w_i|c, t) = \sum_{j \in J_c(w_i)} (1 - \lambda)^{|D_{c,t}| - j}\lambda, \quad (9)$$

where $J_c(w_i)$ is the position of word $w_i$ in $D_{c,t}$, $J_c(w_i) = \{j \mid D_{c,t}[j] = w_i, 1 \leq j \leq |D_{c,t}|\}$, and $\lambda$ is a smoothing parameter of EWMA ($0 < \lambda < 1$). Eq. (9) is a closed-form expression of EWMA.

Our model monitors the two probability estimates for each word $w_i$ in class $c$, and switches between them as follows:

$$p(w_i|c, t) = \begin{cases} p_{\text{EWMA}} & \text{if } p_{\text{EWMA}} > p_{\text{ML}} + L_{c,t} \\ p_{\text{ML}} & \text{otherwise} \end{cases}, \quad (10)$$

where $L_{c,t} = \sqrt{p_{\text{ML}}(1 - p_{\text{ML}}) / (2 - \lambda)}$. Note that Eqs. (10) and (11) mean monitoring the statistical process of $w_i$ occurring in $D_{c,t}$ with a Bernoulli EWMA control chart [31]. EWMA charts are generally used for detecting small shifts in the process mean [20]. The value of $L$ in Eq. (10) determines the upper control limit of the chart, $p_{\text{ML}} + L_{c,t}$. As long as $p_{\text{EWMA}}$ is within the control limit, the word occurrence process is assumed to be currently stable (i.e., our model uses $p_{\text{EWMA}}$)

Fig. 7 clarifies that Eq. (9) preserves the probabilistic interpretation. We can see that the EWMA value, $p_{\text{EWMA}}$, tracks the changes in true probability. The EWMA method is able to respond to sudden shifts quickly. On the other hand, the ML method is able to estimate the probability for stationary words more accurately. The ML method, however, isapt to underestimate the probabilities of bursty words. This switching of the two probability estimates for each word enables our model to resolve the trade-off between quick response to bursty words and the accurate learning of stationary words.

### 4.4 Word n-gram Probability

To acquire ability A5, our model extends the bag-of-words model to the n-gram language model [26]. It uses the absolute discounting method [22, 4] as the smoothing method
used in estimating the word \( n \)-gram probability:

\[
p(w_i|c, w_{i-n+1}^{-1}, t) = \frac{\max\{f_c(w_i^{i-1} - \kappa, 0) + \kappa \cdot r_c(w_i^{i-1}/w_i^{i-1+1}) f_c(w_i|c, w_{i+n+2}^{-1}, t), \}
\]

where \( r_c(w_i^{i-1}) \) is the number of distinct words following word sequence \( w_i^{i-1} \). The lowest order model in this linear interpolation is \( p(w_i|c, t) \); therefore, Eq. (12) retains the temporal effects.

The smoothing of Eq. (8) is also calculated according to the absolute discounting method:

\[
p_{\text{ML}}(w_i|c, t) = \frac{\sum_{w \in D_{c,t}} f_c(w)}{\sum_{w \in D_{c,t}} f_c(w)}, \text{ if } f_c(w_i) = 0
\]

\[
\text{otherwise}
\]

and Eq. (13) is used instead of Eq. (8).

5. IMPLEMENTATION

We can implement our model by constructing full text search indexes (e.g., suffix arrays [19]) for each \( D_{c,t} \). WSA enables our model to efficiently get \( J_c(w_i^{i-1} + 1) \), the positions of \( w_i^{i-1} \) in \( D_{c,t} \), for each word \( n \)-gram \( w_i^{i-1} \) in each class \( c \).

5.1 Summary of WSA

Let \( T[1, \ldots, l] \) be a text of length \( l \) over a constant-sized alphabet \( \Sigma \), tokenized into \( k \) words by word-delimiters, and let \( I \) be the set of alphabet positions at which new words start. The word suffix array \( A[1, \ldots, k] \) is a permutation of \( I \) such that \( T[A[i-1], \ldots, l] < T[A[i], \ldots, l] \) for all \( 1 < i \leq k \); i.e., the array represents the lexicographic order of all suffixes, as shown in Fig. 8 (see details in [7]). The set of word positions, \( B \), is obtained as a by-product of constructing WSA. Our model uses \( B \) to find \( J_c(w_i^{i-1} + 1) \) in \( D_{c,t} \).

WSA can be constructed in \( O(l) \) time and \( O(k) \) space when using the linear-time construction algorithms for suffix arrays (e.g., [23]). It can search \( w_i^{i-1} \) alphabet length of \( m \) quickly with a binary search in \( O(m \log k + f_c(w_i^{i-1} + 1)) \), as in suffix arrays.

\[
T = \text{“ab#aa#aa#ab#baa#aab#a#aa#baa#”}
\]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( A )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>6</td>
</tr>
</tbody>
</table>

5.2 Learning and Classification Procedures

This section shows the learning and classification procedures for word \( n \)-gram probability (Eqs. (9)–(13)).

Learning constructs WSAs for each class. Algorithm 1 shows the learning algorithm for a document stream, \( \{d_t, c_t\} \).

The sequential update of WSA is not supported, so, it reconstructs the WSA of class \( c_t \) when given a new tweet \( d_t \) (lines 2–3).

For reducing the learning time, it is effective to divide tweets into periods and reconstruct the newest WSA on the most recent tweets, while retaining old WSAs. We note that it is able to obtain \( J_c(w_i^{i-1} + 1) \) from the divided WSAs by storing the offset position of each WSA and adding it to the positions of \( w_i^{i-1} \) found by each WSA.

Classification needs the \( f_c(w_i) \), \( J_c(w_i) \), and \( r_c(w_i) \) values (0 \( \leq i-j < n \)) to calculate the word \( n \)-gram probability for each word \( n \)-gram, \( w_i^{i-1} \), in class \( c \). Algorithm 2 shows the pseudo-code that acquires the above values from a WSA.

Our model uses three heuristics for speed enhancement: (1) caching search results (lines 6–11, 13), (2) caching the initial search interval for the first alphabets [19] (line 3), and (3) reducing the number of character comparisons by remembering the number of matching characters [19] (line 9). Caching the search results of \((n-1)\)-grams enables our model to narrow the initial interval when searching \( n \)-grams. The worst case of our approach is \( O(m \log k + f_c(w_i^{i-1} + 1)) \); however, Ferragina et al. reported in [7] that the \( O(m \log k + f_c(w_i^{i-1} + 1)) \) algorithm, which includes heuristics 2 and 3, is faster than the \( O(m \log k + f_c(w_i^{i-1} + 1)) \) [19] and \( O(m \Sigma + f_c(w_i^{i-1} + 1)) \) [1] algorithms.

5.3 Implementation Details

We used the original implementation of the WSA construction algorithm by Fischer [7] and the sais library by Fischer [7].

\[\text{http://sites.google.com/site/yuta256/sais}\]

\[\text{http://ab.inf.uni-tuebingen.de/people/fischer/wordSA.tgz}\]
Algorithm 2: Acquiring classification parameters for class $c$.

Require: $d$ = "w$_1$#w$_2$#⋯#w$_d$#" = w$_d^*$, new document
Require: $R(a)$, search range of WSA for each alphabet $a$
Require: $W(d)$, set of selected words in $d$

1: for $i = 1$ to $z$ do
2:    next if $w_i \notin W(d)$
3:    $R \leftarrow R(w_i[1])$
4:    for $j = i$ to $i + n - 1$ do
5:        $s \leftarrow w_i^j$/word ($i - j + 1$)-gram.
6:        if $is\_cached?(s) =$ true then
7:            $(f_c(s), J_c(s), r_c(s), R(s)) \leftarrow get\_cache(s)$
8:        else
9:            $(f_c(s), J_c(s), r_c(s), R(s)) \leftarrow search\_by\_WSA(s, R)$
10:       set_cache$(s, (f_c(s), J_c(s), r_c(s), R(s)))$
11:    end if
12:    break if $f_c(s) < 0$
13:    $R \leftarrow R(s)$ // narrow next search range.
14: end for
15: return $\{f_c(w_j^i), J_c(w_j^i), r_c(w_j^i)\}$ $1 \leq i \leq z, 0 \leq j < n$

Figure 9: Construction time of word suffix array on tweets of class #dragons on NPB data set. Total 53,000 tweets, 399,825 words, 2,701,985 alphabets. CPU: Xeon X5680 3.33GHz, Memory: 192GB.

which implements Nong et al.’s induced sorting algorithm [23], as the suffix array construction algorithm used by WSA. We also used Kasai et al.’s algorithm [10] for calculating the longest common prefixes. The character encoding of tweets was UTF-8, and the number of alphabets was 256; i.e., a three-byte Japanese character consisted of three alphabets. Fig. 9 shows the results of constructing WSA on the tweets in class #dragons for the NPB data set: the construction time of WSA was linear to the number of alphabets.

6. EXPERIMENTS

In order to evaluate the impact that our proposed model has on the incremental test-then-train problem definition P1 (see [2]), we first evaluated our model components and then compared our model against three conventional versions of multinomial naive Bayes classifiers (standard, instance weighting, and instance selection), by using the three tweet data sets shown in Table 1.

The classifiers used the chi-squared statistics feature selection method [39]: that is, $W(d_t) = \{w| \max_c\{\chi^2_c(w)\} > 30.0\}$, and conducted the smoothing of Eq. (13): $\kappa = 0.9$.

6.1 Effects of Individual Contributions

Our model, P-Switch, is also a temporally-aware version of multinomial naive Bayes. The following sections demonstrate the effects of our model’s components.

6.1.1 Class Distribution

In order to evaluate the effects by introducing the temporally-aware class distribution described in Eq. (7), we evaluated the Proposal1 model:

$$\tilde{c}_t \leftarrow \arg \max_c \{ p(c|t) \prod_{w_i \in W(d_t)} p_{ML}(w_i|c, t) \},$$

which extends the standard multinomial naive Bayes classifier, MNB:

$$\tilde{c}_t \leftarrow \arg \max_c \{ p(c) \prod_{w_i \in W(d_t)} p_{ML}(w_i|c, t) \},$$

while changing smoothing parameter $\gamma$ from 0.001 to 0.5. We can see from Fig. 10 that the Proposal1 model, where $\gamma = 0.01$, performed statistically significantly better than MNB (McNemar’s test: $p < .001$). We note that the multinomial naive Bayes classifier that assumes a uniform class distribution performed poorly.

6.1.2 Word Probability

In order to evaluate the effects of switching between the two probability estimates described in Eq. (10), we evaluated the Proposal2 model:

$$\tilde{c}_t \leftarrow \arg \max_c \{ p(c|t) \prod_{w_i \in W(d_t)} p_{EWMA}(w_i|c, t) \},$$

which extends the Proposal1 model, while changing smoothing parameter $\lambda$ from 0.0001 to 0.01. Other parameter settings were: $\gamma = 0.01$ and $L = 0.5$ for all data sets.

Fig. 11 shows that the Proposal2 model, when $\lambda = 0.002$, performed statistically significantly better than the Proposal1 model for all data sets (McNemar’s test: $p < .001$). The switching of the two probability estimates: $p_{EWMA}$, which is based on recent data, and $p_{ML}$, which is based on data that cover long-term period, enables the Proposal2 model to resolve the trade-off between quick response to bursty words and the accurate learning of stationary words. Moreover, treating class distribution and word probability at different time scales ($\gamma$ and $\lambda$), unlike instance selection and weighting methods, is effective in classifying tweet streams.

We also note that the best value for $\lambda$ is the same for all three data sets, which have different change characteristics as shown in Figs. 1 and 4.

6.1.3 Word n-gram Probability

In order to evaluate the effects of introducing the temporally-aware word n-gram probability described in Eq. (12), we evaluated our P-Switch model defined in Eq. (6), while changing the value of $n$ from 1 (corresponds to the Proposal2 model) to 5. Other parameters settings were: $\gamma = 0.01$, $\lambda = 0.002$, $L = 0.5$ for all data sets.

We can see from Fig. 12 that our model performed best when $n = 2$, and there were statistical differences (McNemar’s test: $p < .001$) between $n = 1$ and $n = 2$ on NPB and
are effective in handling concept drift in tweet streams.

... weighting method (conventional methods for handling concept drift: a temporally-aware multinomial naive Bayes classifier with the instance selection method (MNB-S; Eqs. (1) and (4)) (see details in §3.3 and [29, 18]). Parameter settings were: \(\gamma = 0.01, \lambda = 0.002, L = 0.5, \) and \(n = 2\). The kernel width of MNB-W and MNB-S was \(h = 10,000\), which was the best value for the TV data set. Note that the best value of \(h\) for the NPB and MLB data sets was \(\infty\) (corresponds to MNB) as shown in Fig. 6.

Fig. 13 plots macro F\(_1\) of the classifiers vs. the training number of tweets (time steps). Table 2 is the overall accuracy and macro F\(_1\) of the classifiers after learning all data. Although the NPB and MLB data sets contain various changes, MNB performed better than MNB-W and MNB-S; that is, learning stationary words with sufficient data is more important than responding to bursty words for these data sets. Our model resolved the trade-off between quick response and accurate learning by switching between the two probability estimates for each word. It offered a statistically significant improvement over MNB, up to 11\% in terms of macro F\(_1\) and 7.56\% in terms of overall accuracy, and up to 9.072\% and 7.480\% in terms of macro F\(_1\), for the NPB and MLB data sets, respectively (McNemar's test; \(p < .001\)).

Considering temporal effects enabled MNB-W and MNB-S to perform better than MNB on the TV data set; however, increasing the number of training tweets is not effective for achieving their higher classification accuracy, as shown in Fig. 13b. This is because using the small kernel width yields rapid dropping of old data. MNB, which learns all data, performed worst because the TV data set contained sudden and significant shifts as shown in Fig. 2. Our P-Switch model was able to respond to both the sudden shift and gradual drift; therefore, it achieved statistically signifi-
3, 25], they also do not consider temporal effects. Some studies that use suffix arrays (SAs) for document classification have been proposed. Teo and Vishwanathan proposed fast and space efficient string kernels based on SAs and used the kernel with the support vector machine [33]. Okanohara and Tsujii proposed a logistic regression model with all-effective substrings, which are found with SAs [24]. These studies target the finding of effective feature spaces for batch learning; our model, on the other hand, uses lazy learning for streaming documents—generalization beyond the training data is delayed until a new document is given.

Salles et al. proposed temporally-aware versions of Rocchio and k-nearest neighbors classifiers that conduct temporal weighting on documents [29], in addition to the multinomial naive Bayes classifier described in Eq. (1). Their experiments showed that there were no significant differences among the classifiers. We find that responding to changes at the word level is more important than model selection for classifying document streams.

7. RELATED WORK

As related work on classification of tweets, Sriram et al. devised a naive Bayes classifier for identifying tweet types such as news, events, opinions, deals, and private messages based on author information and tweet texts [32]. Sakaki et al. devised a support vector machine for finding tweets related to an event (e.g., earthquake) [27]. Kinsella et al. uses metadata from hyperlinked objects to improve topic classification [14]. The above studies, however, employ batch learning and do not consider concept drift. While sentiment classification methods have been proposed most recently [9, 3, 25], they also do not consider temporal effects.

Figure 13: Number of tweets vs. cumulative macro F1 on (a) NPB, (b) TV, and (c) MLB data sets. γ = 0.01, λ = 0.002, L = 0.5, n = 2, and h = 10000.

8. CONCLUSIONS

We proposed a classification model of tweet streams, which are representative of document streams whose statistical properties change over time.

We analyzed tweet streams by examining the changes in class distribution and word probability, the occurrence of new words and recurring themes, and the effectiveness of considering word n-grams. In particular, we demonstrated that tweet streams change significantly at the word level—the word occurrence probability changes at different rates for different words.

Our model solves this problem by detecting changes in word occurrence probability for each word. It switches between two probability estimates, the maximum likelihood and exponentially weighted moving average estimates, by using a control chart and is very effective in achieving both accurate learning of stationary words and quick response to bursty words. Also, it handles the changes in class distributions and extends the bag-of-words model to cover the n-gram language model. It preserves the probabilistic interpretation of the multinomial naive Bayes classifier.

We then presented an implementation method that uses a word suffix array. This approach enables our model to effectively handle word n-grams, which are highly important information given the 140 character limit of tweets. This new idea, searching the positions of words in documents concatenated chronologically, allows consideration of the temporal effects on changing word streams.

Experiments on topic classification were conducted using three real-world tweet data sets: tweets about Japanese baseball teams, television networks in Japan, and major league baseball teams. Results demonstrated that our model performed statistically significantly better than conventional instance weighting and selection models. In particular, our model was superior to the multinomial naive Bayes classifier with an instance weighting method by up to 16.33% and 16.35% in terms of accuracy and macro F1, for the television network data set.

9. REFERENCES


