Sentiment Knowledge Discovery in Twitter Streaming Data

Albert Bifet and Eibe Frank

University of Waikato, Hamilton, New Zealand
{abifet,eibe}@cs.waikato.ac.nz

Abstract. Micro-blogs are a challenging new source of information for data mining techniques. Twitter is a micro-blogging service built to discover what is happening at any moment in time, anywhere in the world. Twitter messages are short, and generated constantly, and well suited for knowledge discovery using data stream mining. We briefly discuss the challenges that Twitter data streams pose, focusing on classification problems, and then consider these streams for opinion mining and sentiment analysis. To deal with streaming unbalanced classes, we propose a sliding window Kappa statistic for evaluation in time-changing data streams. Using this statistic we perform a study on Twitter data using learning algorithms for data streams.

1 Introduction

Twitter is a "what's-happening-right-now" tool that enables interested parties to follow individual users' thoughts and commentary on events in their lives—in almost real-time [26]. It is a potentially valuable source of data that can be used to delve into the thoughts of millions of people as they are uttering them. Twitter makes these utterances immediately available in a data stream, which can be mined using appropriate stream mining techniques. In principle, this could make it possible to infer people's opinions, both at an individual level as well as in aggregate, regarding potentially any subject or event [26].

At the official Twitter Chirp developer conference in April 2010 [28], the company presented some statistics about its site and its users. In April 2010, Twitter had 106 million registered users, and 180 million unique visitors every month. The company revealed that 300,000 new users were signing up per day and that it received 600 million queries daily via its search engine, and a total of 3 billion requests per day based on its API. Interestingly, 37 percent of Twitter's active users used their phone to send messages.

Twitter data follows the data stream model. In this model, data arrive at high speed, and data mining algorithms must be able to predict in real time and under strict constraints of space and time. Data streams present serious challenges for algorithm design [3]. Algorithms must be able to operate with limited resources, regarding both time and memory. Moreover, they must be able to deal with data whose nature or distribution changes over time.

The main Twitter data stream that provides all messages from every user in real-time is called Firehose [16] and was made available to developers in 2010. To deal with this large amount of data, and to use it for sentiment analysis and opinion mining—the task considered in this paper—streaming techniques are needed. However, to the best of our knowledge, data stream algorithms, in conjunction with appropriate evaluation techniques, have so far not been considered for this task.

Evaluating data streams in real time is a challenging task. Most work in the literature considers only how to build a picture of accuracy over time. Two main approaches arise [2]:

- **Holdout**: Performance is measured using on single hold-out set.
- **Interleaved Test-Then-Train or Prequential**: Each individual example is used to test the model before it is used for training, and accuracy is incrementally updated.

A common problem is that for unbalanced data streams with, for example, 90% of the instances in one class, the simplest classifiers will have high accuracies of at least 90%. To deal with this type of data stream, we propose to use the Kappa statistic, based on a sliding window, as a measure for classifier performance in unbalanced class streams.

In Section 2 we discuss the challenges that Twitter streaming data poses and discuss related work. Twitter sentiment analysis is discussed is Section 3, and the new evaluation method for time-changing data streams based on the sliding window Kappa statistic is proposed in Section 4. We review text data stream learners in Section 5. Finally, in Section 6, we perform an experimental study on Twitter streams using data stream mining methods.

2 Mining Twitter Data: Challenges and Related Work

Twitter has its own conventions that renders it distinct from other textual data. Consider the following Twitter example message ("tweet"): `RT @toni has a cool #job`. It shows that users may reply to other users by indicating user names using the character @, as in, for example, `@toni`. Hashtags (#) are used to denote subjects or categories, as in, for example `#job`. `RT` is used at the beginning of the tweet to indicate that the message is a so-called "retweet", a repetition or reposting of a previous tweet.

In the knowledge discovery context, there are two fundamental data mining tasks that can be considered in conjunction with Twitter data: (a) graph mining based on analysis of the links amongst messages, and (b) text mining based on analysis of the messages' actual text.

Twitter graph mining has been used to tackle several interesting problems:

- **Measuring user influence and dynamics of popularity**. Direct links indicate the flow of information, and thus a user's influence on others. There are three measures of influence: indegree, retweets and mentions. Cha et

al. [5] show that popular users who have high indegree are not necessarily influential in terms of retweets or mentions, and that influence is gained through concerted effort such as limiting tweets to a single topic.

- **Community discovery and formation**. Java et al. [15] found communities using HyperText Induced Topic Search (HITS) [17], and the Clique Percolation Method [8]. Romero and Kleinberg [25] analyze the formation of links in Twitter via the directed closure process.
- **Social information diffusion**. De Choudhury et al. [7] study how data sampling strategies impact the discovery of information diffusion.

There are also a number of interesting tasks that have been tackled using Twitter text mining: sentiment analysis, which is the application we consider in this paper, classification of tweets into categories, clustering of tweets and trending topic detection.

Considering sentiment analysis [18, 21], O'Connor et al. [19] found that surveys of consumer confidence and political opinion correlate with sentiment word frequencies in tweets, and propose text stream mining as a substitute for traditional polling. Jansen et al. [14] discuss the implications for organizations of using micro-blogging as part of their marketing strategy. Pak et al. [20] used classification based on the multinomial naïve Bayes classifier for sentiment analysis. Go et al. [12] compared multinomial naïve Bayes, a maximum entropy classifier, and a linear support vector machine; they all exhibited broadly comparable accuracy on their test data, but small differences could be observed depending on the features used.

2.1 The Twitter Streaming API

The Twitter Application Programming Interface (API) [1] currently provides a Streaming API and two discrete REST APIs. Through the Streaming API [16] users can obtain real-time access to tweets in sampled and filtered form. The API is HTTP based, and GET, POST, and DELETE requests can be used to access the data.

In Twitter terminology, individual messages describe the "status" of a user. Based on the Streaming API users can access subsets of public status descriptions in almost real time, including replies and mentions created by public accounts. Status descriptions created by protected accounts and all direct messages cannot be accessed. An interesting property of the streaming API is that it can filter status descriptions using quality metrics, which are influenced by frequent and repetitious status updates, etc.

The API uses basic HTTP authentication and requires a valid Twitter account. Data can be retrieved as XML or the more succinct JSON format. The format of the JSON data is very simple and it can be parsed very easily because every line, terminated by a carriage return, contains one object.

3 Twitter Sentiment Analysis

Sentiment analysis can be cast as a classification problem where the task is to classify messages into two categories depending on whether they convey positive or negative feelings.

Twitter sentiment analysis is not an easy task because a tweet can contain a significant amount of information in very compressed form, and simultaneously carry positive and negative feelings. Consider the following example:

```
I currently use the Nikon D90 and love it, but not as much as
the Canon 40D/50D. I chose the D90 for the video feature. My
mistake.
```

Also, some tweets may contain sarcasm or irony [4] as in the following example:

```
After a whole 5 hours away from work, I get to go back again,
I'm so lucky!
```

To build classifiers for sentiment analysis, we need to collect training data so that we can apply appropriate learning algorithms. Labeling tweets manually as positive or negative is a laborious and expensive, if not impossible, task. However, a significant advantage of Twitter data is that many tweets have author-provided sentiment indicators: changing sentiment is implicit in the use of various types of emoticons. Hence we may use these to label our training data.

Smileys or *emoticons* are visual cues that are associated with emotional states [24, 4]. They are constructed by approximating a facial expression of emotion based on the characters available on a standard keyboard. When the author of a tweet uses an emoticon, they are annotating their own text with an emotional state. Annotated tweets can be used to train a sentiment classifier.

4 Streaming Data Evaluation with Unbalanced Classes

In data stream mining, the most frequently used measure for evaluating predictive accuracy of a classifier is prequential accuracy [10]. We argue that this measure is only appropriate when all classes are balanced, and have (approximately) the same number of examples. In this section, we propose the Kappa statistic as a more sensitive measure for quantifying the predictive performance of streaming classifiers. For example, considering the particular target domain in this paper, the rate in which the Twitter Streaming API delivers positive or negative tweets may vary over time; we cannot expect it to be 50% all the time. Hence, a measure that automatically compensates for changes in the class distribution should be preferable.

Just like accuracy, Kappa needs to be estimated using some sampling procedure. Standard estimation procedures for small datasets, such as cross-validation, do not apply. In the case of very large datasets or data streams, there are two basic evaluation procedures: holdout evaluation and prequential evaluation. Only the latter provides a picture of performance over time. In prequential evaluation

	Predicted Class+	Predicted Class-	Total
Correct Class+	75	8	83
Correct Class-	7	10	17
Total	82	18	100

Table 1. Simple confusion matrix example

	Predicted Class+	Predicted Class-	Total
Correct Class+	68.06	14.94	83
Correct Class-	13.94	3.06	17
Total	82	18	100

Table 2. Confusion matrix for chance predictor based on example in Table 1

(also known as interleaved test-then-train evaluation), each example in a data stream is used for testing before it is used for training.

We argue that prequential accuracy is not well-suited for data streams with unbalanced data, and that a prequential estimate of Kappa should be used instead. Let p_0 be the classifier's prequential accuracy, and p_c the probability that a chance classifier—one that assigns the same number of examples to each class as the classifier under consideration—makes a correct prediction. Consider the simple confusion matrix shown in Table 1. From this table, we see that Class+ is predicted correctly 75 out of 100 times, and Class- is predicted correctly 10 times. So accuracy p_0 is 85%. However a classifier predicting solely by chance—in the given proportions—will predict Class+ and Class- correctly in 68.06% and 3.06% of cases respectively. Hence, it will have an accuracy p_c of 71.12% as shown in Table 2.

Comparing the classifier's observed accuracy to that of a chance predictor renders its performance far less impressive than it first seems. The problem is that one class is much more frequent than the other in this example and plain accuracy does not compensate for this. The Kappa statistic, which normalizes a classifier's accuracy by that of a chance predictor, is more appropriate in scenarios such as this one.

The Kappa statistic κ was introduced by Cohen [6]. We argue that it is particularly appropriate in data stream mining due to potential changes in the class distribution. Consider a classifier h, a data set containing m examples and L classes, and a contingency table where cell C_{ij} contains the number of examples for which $h(x) = i$ and the class is j. If $h(x)$ correctly predicts all the data, then all non-zero counts will appear along the diagonal. If h misclassifies some examples, then some off-diagonal elements will be non-zero.

We define

$$p_0 = \frac{\sum_{i=1}^{L} C_{ii}}{m}$$

$$p_c = \sum_{i=1}^{L} \left(\sum_{j=1}^{L} \frac{C_{ij}}{m} \cdot \sum_{j=1}^{L} \frac{C_{ji}}{m} \right)$$

In problems where one class is much more common than the others, any classifier can easily yield a correct prediction by chance, and it will hence obtain a high value for p_0. To correct for this, the κ statistic is defined as follows:

$$\kappa = \frac{p_0 - p_c}{1 - p_c}$$

If the classifier is always correct then $\kappa = 1$. If its predictions coincide with the correct ones as often as those of the chance classifier, then $\kappa = 0$.

The question remains as to how exactly to compute the relevant counts for the contingency table: using all examples seen so far is not useful in time-changing data streams. Gama et al. [10] propose to use a forgetting mechanism for estimating prequential accuracy: a sliding window of size w with the most recent observations, or fading factors that weigh observations using a decay factor α. As the output of the two mechanisms is very similar (every window of size w_0 may be approximated by some decay factor α_0), we propose to use the Kappa statistic measured using a sliding window. Note that, to calculate the statistic for an n_c class problem, we need to maintain only $2n_c + 1$ estimators. We store the sum of all rows and columns in the confusion matrix ($2n_c$ values) to compute p_c, and we store the prequential accuracy p_0. The ability to calculate it efficiently is an important reason why the Kappa statistic is more appropriate for data streams than a measure such as the area under the ROC curve.

5 Data Stream Mining Methods

We experimented with three fast incremental methods that are well-suited to deal with data streams: multinomial naïve Bayes, stochastic gradient descent, and the Hoeffding tree.

Multinomial Naïve Bayes The multinomial naïve Bayes classifier is a popular classifier for document classification that often yields good performance. It can be trivially applied to data streams because it is straightforward to update the counts required to estimate conditional probabilities..

Multinomial naive Bayes considers a document as a bag-of-words. For each class c, $P(w|c)$, the probability of observing word w given this class, is estimated from the training data, simply by computing the relative frequency of each word in the collection of training documents of that class. The classifier also requires the prior probability $P(c)$, which is straightforward to estimate.

Assuming n_{wd} is the number of times word w occurs in document d, the probability of class c given a test document is calculated as follows:

$$P(c|d) = \frac{P(c) \prod_{w \in d} P(w|c)^{n_{wd}}}{P(d)},$$

where $P(d)$ is a normalization factor. To avoid the zero-frequency problem, it is common to use the Laplace correction for all conditional probabilities involved, which means all counts are initialized to value one instead of zero.

Stochastic Gradient Descent Stochastic gradient descent (SGD) has experienced a revival since it has been discovered that it provides an efficient means to learn some classifiers even if they are based on non-differentiable loss functions, such as the hinge loss used in support vector machines. In our experiments we use an implementation of vanilla stochastic gradient descent with a fixed learning rate, optimizing the hinge loss with an L_2 penalty that is commonly applied to learn support vector machines. With a linear machine, which is frequently applied for document classification, the loss function we optimize is:

$$\frac{\lambda}{2}||\mathbf{w}||^2 + \sum [1 - (y\mathbf{x}\mathbf{w} + b)]_+,$$

where w is the weight vector, b the bias, λ the regularization parameter, and the class labels y are assumed to be in $\{+1, -1\}$.

We compared the performance of our vanilla implementation to that of the Pegasos method [27], which does not require specification of an explicit learning rate, but did not observe a gain in performance using the latter. On the contrary, the ability to specify an explicit learning rate turned out to be crucial to deal with time-changing Twitter data streams : setting the learning rate to a value that was too small meant the classifier adapted too slowly to local changes in the distribution. In our experiments, we used $\lambda = 0.0001$ and set the learning rate for the per-example updates to the classifier's parameters to 0.1.

Hoeffding Tree The most well-known tree decision tree learner for data streams is the *Hoeffding tree* algorithm [9]. It employs a pre-pruning strategy based on the Hoeffding bound to incrementally grow a decision tree. A node is expanded by splitting as soon as there is sufficient statistical evidence, based on the data seen so far, to support the split and this decision is based on the distribution-independent Hoeffding bound.

Decision tree learners are not commonly applied to document classification due to the high-dimensional feature vectors involved. Simple linear classifiers generally yield higher accuracy. Nevertheless, we include Hoeffding trees in our experiments on Twitter data streams to verify that this observation also holds in this particular context. Moreover, decision trees can potentially yield valuable insight into interactions between variables.

6 Experimental Evaluation

Massive **O**nline **A**nalysis (MOA) [2] is a system for online learning from examples, such as data streams. All algorithms evaluated in this paper were implemented in the Java programming language by using WEKA [13] and the MOA software.

In our experiments, we used the Twitter training datasets to extract features using text filters in WEKA. Each tweet was represented as a set of words. We extracted 10,000 unigrams using the default stop word list in WEKA. We used term presence instead of frequency, as Pang et al. [22] reported that term presence achieves better results than frequency on sentiment analysis classification. The resulting vectors are stored in sparse format.

6.1 The `twittersentiment.appspot.com` and Edinburgh Corpora

Twitter Sentiment (`twittersentiment.appspot.com`) is a website that enables visitors to research and track the sentiment for a brand, product, or topic. It was created by Alec Go, Richa Bhayani, Karthik Raghunathan, and Lei Huang from Stanford University. The website enables a visitor to track queries over time. Sentiment classification is based on a linear model generated using the maximum entropy method.[1] The Twitter Sentiment website provides an API to use the maximum entropy classifier: one can use it to determine the polarity of arbitrary text, retrieve sentiment counts over time, and retrieve tweets along with their classification.

The developers of the website collected two datasets: a training set and a test one, which were also used for sentiment classification in [12]. The training dataset was obtained by querying the (non-streaming) Twitter API for messages between April 2009 and June 25, 2009 and contains the first 800,000 tweets with positive emoticons, and the first 800,000 tweets with negative emoticons. The list of positive emoticons used was: :), :-), :), :D, and =). The negative emoticons used were: :(, :-(, and : (. The test dataset was manually annotated with class labels and consists of 177 negative tweets and 182 positive ones. Test tweets were collected by looking for messages that contained a sentiment, regardless of the presence of emoticons. Each tweet contains the following information: its polarity (indicating the sentiment), the date, the query used, the user, and the actual text.

The Edinburgh corpus [23] was collected over a period of two months using the Twitter streaming API. It contains 97 million tweets and requires 14 GB of disk space when stored in uncompressed form.[2] Each tweet has the following information: the timestamp of the tweet, an anonymized user name, the tweet's text, and the posting method that was used.

The corpus was collected between November 11th 2009 and February 1st 2010, using Twitter's streaming API. It is thus a representative sample of the entire stream. The data contains over 2 billion words and there is no distinction between English and non-English tweets. We only considered tweets in English and only those that contained emoticons.

[1] The software is available at `http://nlp.stanford/software/classifier.shtml`.
[2] The corpus can be obtained at http://demeter.inf.ed.ac.uk/.

	Accuracy	Kappa	Time
Multinomial Naïve Bayes	75.05%	50.10%	116.62 sec.
SGD	82.80%	62.60%	219.54 sec.
Hoeffding Tree	73.11%	46.23%	5525.51 sec.

Table 3. Total prequential accuracy and Kappa measured on the `twittersentiment.appspot.com` data stream

	Accuracy	Kappa
Multinomial Naïve Bayes	82.45%	64.89%
SGD	78.55%	57.23%
Hoeffding Tree	69.36%	38.73%

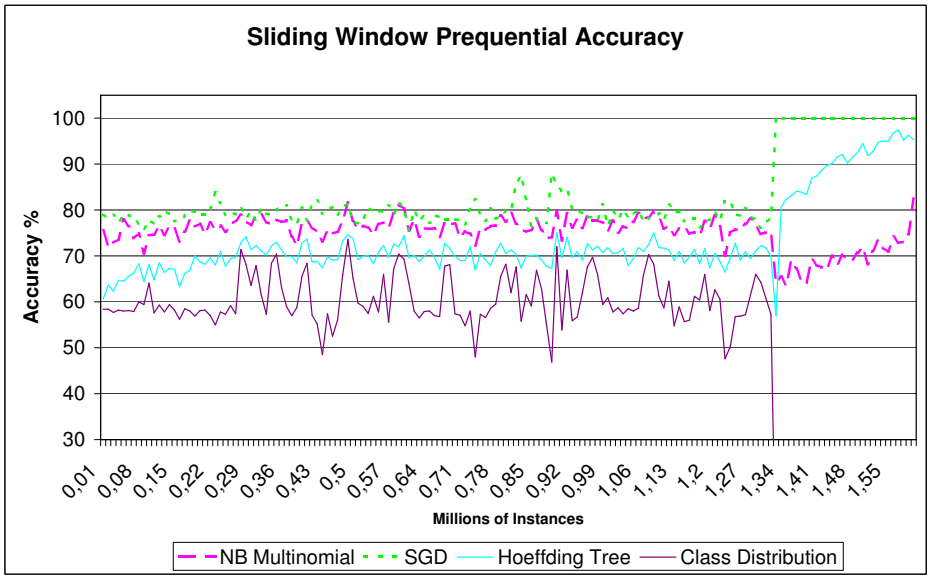
Table 4. Accuracy and Kappa for the test dataset obtained from `twittersentiment.appspot.com`

6.2 Results and Discussion

We performed two data stream experiments: one using the training dataset from `twittersentiment.appspot.com`, and another one with the Edinburgh Corpus. We also performed a classic train/test experiment based on each training set and the test set from `twittersentiment.appspot.com`.

First, we consider the data from `twittersentiment.appspot.com`. We performed a prequential evaluation, testing and then training, using the training stream of $1,600,000$ instances, half positives and half negatives. Figure 1 shows the learning curve for this stream measuring prequential accuracy and Kappa using a sliding window of size $1,000$. Table 3 reports the total prequential accuracy and Kappa. In this data stream the last $200,000$ instances are positive, as the data was collected to have the same number of positive and negative tweets: the rate of tweets using positive emoticons is usually higher than that of negative ones. We see at the end of the learning curve in Figure 1 that prequential accuracy still presents (apparently) good results, but that the value of Kappa is zero or below. This is an extreme example of a change in class distribution (one class disappears completely from the stream), which shows very clearly why Kappa is useful when the distribution of classes evolves over time. We see that the worst method in accuracy, Kappa, and time for this dataset is the Hoeffding Tree, supporting our hypothesis that tree learners are not appropriate in this context.

The second experiment uses the data from `twittersentiment.appspot.com` in a classic train/test set-up. Table 4 reports accuracy and Kappa for the test set. The results for accuracy for naïve Bayes are comparable to those in [12]. As SGD is very sensitive to change in the data distribution, we trained it on a randomized version of the training dataset for this particular test. Doing this improves its accuracy on the test set, but naïve Bayes is somewhat better. Note that the



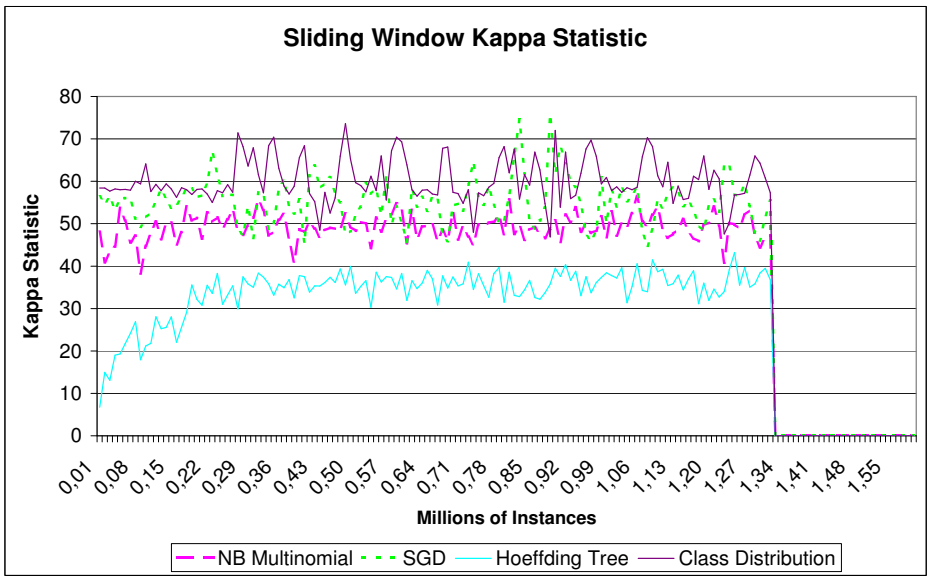


Fig. 1. Sliding window prequential accuracy and Kappa measured on the `twittersentiment.appspot.com` data stream. (Note: solid line shows accuracy in both graphs.)

	Accuracy	Kappa	Time
Multinomial Naïve Bayes	86.11%	36.15%	173.28, sec.
SGD	86.26%	31.88%	293.98 sec.
Hoeffding Tree	84.76%	20.40%	6151.51 sec.

Table 5. Total prequential accuracy and Kappa obtained on the Edinburgh corpus data stream.

Hoeffding tree is the slowest of the three methods, as the current implementation does not use sparse instances as multinomial naïve Bayes and SGD do.

The `twittersentiment.appspot.com` data does not constitute a representative sample of the real Twitter stream due to the fact that the data was augmented to be balanced. Hence we now turn to the Edinburgh corpus. We converted the raw data following the same methodology as Go et al. [11, 12]:

- Feature Reduction. Twitter users may use the @ symbol before a name to direct the message to a certain recipient. We replaced words starting with the @ symbol with the generic token USER, and any URLs by the token URL. We also replaced repeated letters: e.g., *huuuuuungry* was converted to *huungry* to distinguish it from *hungry*.
- Emoticons. Once they had been used to generate class labels, all emoticons were deleted from the input text so that they could not be used as predictive features.

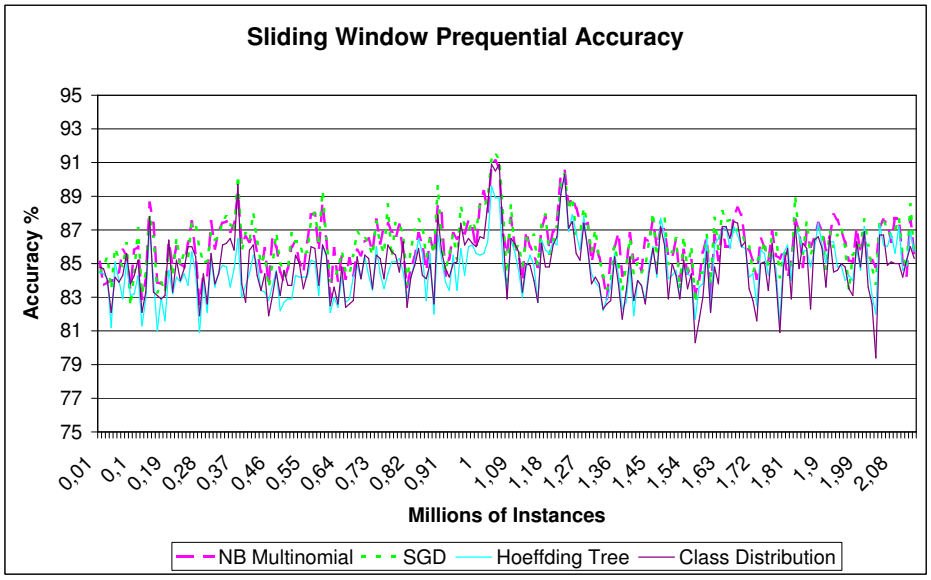
Once this steps had been performed WEKA's text filter was used to convert the data into vector format.

The resulting data stream contains $324,917$ negative tweets and $1,813,705$ positive ones. We observe that negative tweets constitute 15% of the labeled data and positive ones 85%. It appears that people tend to use more positive emoticons than negative ones.

Figure 2 shows the learning curve measuring prequential accuracy and Kappa using a sliding window of $1,000$, and Table 5 reports the total prequential accuracy and value of Kappa. We see in the learning curve of Figure 2 that accuracy is similar for the three methods, but this is not the case when one considers the Kappa statistic. Again, Kappa provides us with a better picture of relative predictive performance. In this stream, we see that multinomial naïve Bayes and SGD perform comparably.

Finally, we test the classifiers learned with the Edinburgh corpus using the test set from `twittersentiment.appspot.com`. Again, the training data was randomized for SGD as in the case of the `twittersentiment.appspot.com` data. Table 6 shows the results. The value of Kappa shows that multinomial naïve Bayes is the most accurate method on this particular test set.

An advantage of the SGD-based model is that changes in its weights can be inspected to gain insight into changing properties of the data stream. Table 7 shows the change in coefficients for some words along the stream obtained from the Edinburgh corpus. The coefficients correspond to December 26th 2009, and



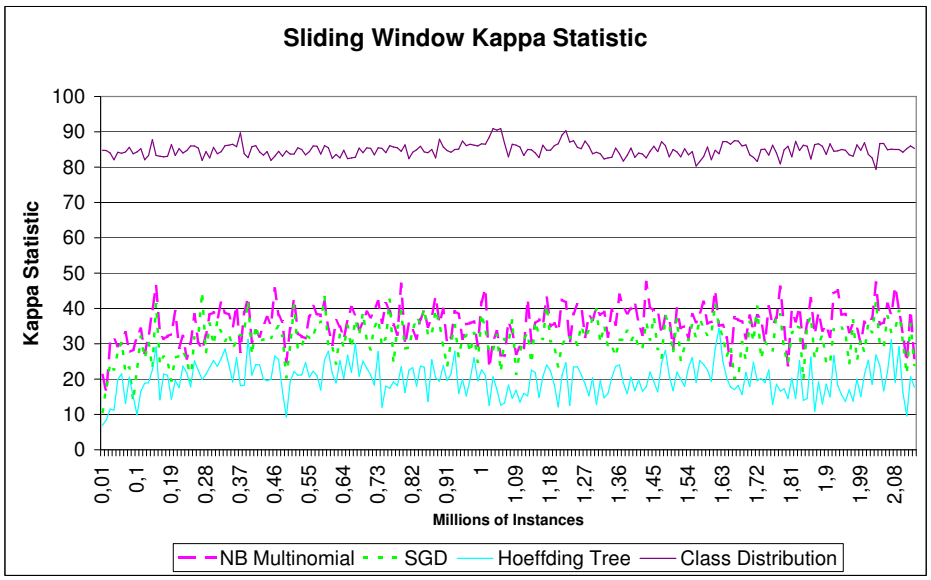


Fig. 2. Sliding window prequential accuracy and Kappa measured on data stream obtained from the Edinburgh corpus. (Note: solid line shows accuracy in both graphs.)

	Accuracy	Kappa
Multinomial Naïve Bayes	73.81%	47.28%
SGD	67.41%	34.23%
Hoeffding Tree	60.72%	20.59%

Table 6. Accuracy and Kappa for the test dataset obtained from `twittersentiment.appspot.com` using the Edinburgh corpus as training data stream.

	Middle of Stream	End of Stream	
Tags	Coefficient	Coefficient	Variation
apple	0.3	0.7	0.4
microsoft	-0.4	-0.1	0.3
facebook	-0.3	0.4	0.7
mcdonalds	0.5	0.1	-0.4
google	0.3	0.6	0.3
disney	0.0	0.0	0.0
bmw	0.0	-0.2	-0.2
pepsi	0.1	-0.6	-0.7
dell	0.2	0.0	-0.2
gucci	-0.4	0.6	1.0
amazon	-0.1	-0.4	-0.3

Table 7. SGD coefficient variations on the Edinburgh corpus

February 1st 2010, respectively. Monitoring these coefficients, which determine how strongly absence/presence of the corresponding word influences the model's prediction of negative or positive sentiment, may be an efficient way to detect changes in the population's opinion regarding a particular topic or brand.

7 Conclusions

Twitter streaming data can potentially enable any user to discover what is happening in the world at any given moment in time. Because the Twitter Streaming API delivers a large quantity of tweets in real time, data stream mining and evaluation techniques are the best fit for the task at hand, but have not been considered previously. We discussed the challenges that Twitter streaming data poses, focusing on sentiment analysis, and proposed the sliding window Kappa statistic as an evaluation metric for data streams. Considering all tests performed and ease of interpretability, the SGD-based model, used with an appropriate learning rate, can be recommended for this data.

In future work, we would like to extend the results presented here by evaluating our methods in real time and using other features available in Twitter data streams, such as geographical place, the number of followers or the number of friends.

Acknowledgments

We would like to thank Alec Go, Lei Huang, and Richa Bhayani for very generously sharing their Twitter dataset with us. We would also like to thank Sasa Petrovic, Miles Osborne, and Victor Lavrenko for making their Twitter dataset publicly available.

References

1. Twitter API. http://apiwiki.twitter.com/, 2010.
2. A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis http://moa.cs.waikato.ac.nz/. *Journal of Machine Learning Research (JMLR)*, 2010.
3. A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 299–310, 2010.
4. P. Carvalho, L. Sarmento, M. J. Silva, and E. de Oliveira. Clues for detecting irony in user-generated contents: oh...!! it's "so easy" ;-). In *Proceeding of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, pages 53–56, 2009.
5. M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 10–17, 2010.
6. J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960.
7. M. De Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher. How does the data sampling strategy impact the discovery of information diffusion in social media? In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 34–41, 2010.
8. I. Derenyi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical Review Letters*, 94(16), 2005.
9. P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.
10. J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–338, 2009.
11. A. Go, R. Bhayani, K. Raghunathan, and L. Huangi. http://twittersentiment.appspot.com/, 2009.
12. A. Go, L. Huang, and R. Bhayani. Twitter sentiment classification using distant supervision. In *CS224N Project Report, Stanford*, 2009.
13. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.
14. B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, pages 3859–3864, 2009.

15. A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, pages 56–65, 2007.
16. J. Kalucki. Twitter streaming API. http://apiwiki.twitter.com/Streaming-API-Documentation, 2010.
17. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
18. B. Liu. *Web data mining; Exploring hyperlinks, contents, and usage data*. Springer, 2006.
19. B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 122–129, 2010.
20. A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, pages 1320–1326, 2010.
21. B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.
22. B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86, 2002.
23. S. Petrovic, M. Osborne, and V. Lavrenko. The Edinburgh Twitter corpus. In *#SocialMedia Workshop: Computational Linguistics in a World of Social Media*, pages 25–26, 2010.
24. J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48, 2005.
25. D. M. Romero and J. Kleinberg. The directed closure process in hybrid social-information networks, with an analysis of link formation on Twitter. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 138–145, 2010.
26. E. Schonfeld. Mining the thought stream. TechCrunch Weblog Article, http://techcrunch.com/2009/02/15/mining-the-thought-stream/, 2009.
27. S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine learning*, pages 807–814, 2007.
28. J. Yarow. Twitter finally reveals all its secret stats. BusinessInsider Weblog Article, http://www.businessinsider.com/twitter-stats-2010-4/, 2010.