Learning under Feature Drifts in Textual Streams

Damianos P. Melidis
Faculty of Electrical Engineering and Computer Science & L3S Research Center, Leibniz University Hannover.
melidis@l3s.de

Myra Spiliopoulou
Faculty of Computer Science, Otto-von-Guericke-University Magdeburg.
myra@ovgu.de

Eirini Ntoutsi
Faculty of Electrical Engineering and Computer Science & L3S Research Center, Leibniz University Hannover.
ntoutsi@l3s.de

ABSTRACT

Huge amounts of textual streams are generated nowadays, especially in social networks like Twitter and Facebook. As the discussion topics and user opinions on those topics change drastically with time, those streams undergo changes in data distribution, leading to changes in the concept to be learned, a phenomenon called concept drift. One particular type of drift, that has not yet attracted a lot of attention is feature drift, i.e., changes in the features that are relevant for the learning task at hand. In this work, we propose an approach for handling feature drifts in textual streams. Our approach integrates i) an ensemble-based mechanism to accurately predict the feature/word values for the next time-point by taking into account the different features might be subject to different temporal trends and ii) a sketch-based feature space maintenance mechanism that allows for a memory-bounded maintenance of the feature space over the stream. Experiments with textual streams from the sentiment analysis, email preference and spam detection demonstrate that our approach achieves significantly better or competitive performance compared to baselines.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**; • **Computing methodologies → Ensemble methods**; • **Information systems → Data stream mining**;

KEYWORDS

Concept Drifts, Feature Drifts, Ensemble Learning, Time Series, Textual Streams

ACM Reference Format:
Damianos P. Melidis, Myra Spiliopoulou, and Eirini Ntoutsi. 2018. Learning under Feature Drifts in Textual Streams. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271717

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271717

1 INTRODUCTION

Huge amounts of textual streams are generated nowadays, especially in social networks like Twitter and Facebook. A key characteristic of those streams is velocity, i.e., the content of the stream changes over time as new topics arise, old topics disappear and even for persistent topics, changes might occur. Such changes in the underlying data generation process might cause changes in the learned hypothesis, a phenomenon known as concept-drift [11]. Concept drift introduces new challenges for stream learners with the most important being the ability of the models to adapt to the underlying population changes. One particular type of change, that has not yet attracted a lot of interest [2] is feature drifts, i.e., changes in the features that are relevant for the learning task. The recent publication of [23] discusses such streams referring to them as *feature-evolving streams*, where new features may arrive over time and existing features may update their value.

Such changes are especially frequent in textual streams as the feature space is high-dimensional (e.g., when considering words as features) and sparse (i.e., words are not observed in all documents and over the whole course of the stream). To deal with the new incoming features, one has to maintain a valid feature space over time that reflects the (temporal) importance of the features for the learning task. We mostly interested in the second property of such streams, the update of value of features. We are especially interested in temporal feature drifts, i.e., changes in the importance of features for the different classes over time, we refer to this as *trends*, hereafter. For the sentiment analysis task, we show an example of words with temporal trends in 1.

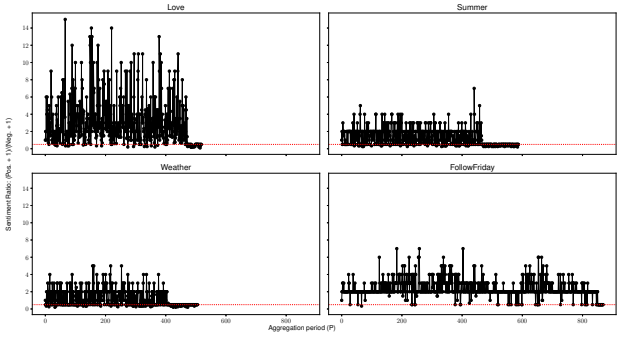


Figure 1: Sentiment Ratio ($\frac{\text{num. positives}+1}{\text{num. negatives}+1}$) for four words, where the red dashed line indicates the threshold for the change of sentiment.

By the figure we observe that exist words which are always associated with the same class like the word "love" (positive). On

the other hand, there are words like "summer" that display a seasonal trend and are probably negative out of season. Or words, like "weather" that display an autocorrelated trend depending on the weather conditions of the last days. Finally, there are *sudden* words such as "followFriday" that display short peaks in a class. Such sort of temporal drifts might also occur at the class level, more positive tweets in weekends as shown by [31]. Such temporal trends are existing also in other textual streams such as in prediction of email preference or spam detection. For example an user may change her interest on a subject turning not interesting email to interesting ones as time evolves.

In this work, we propose an approach for handling feature drifts that discovers trends in the association of features with the different classes and uses those trends to predict the feature values for the next timepoint. As already shown via the aforementioned examples, there is a large variety of such trends from regular to seasonal and sudden occurrences; to capture this variety we propose an ensemble that takes into account different feature periodicities. The ensemble's prediction are seamlessly integrated in a Multinomial Naive Bayes (MNB) classifier whose model naturally depend on the class conditional feature probabilities. Although there are already approaches for adaptive MNB classifiers over streams, e.g., [29] and [32], our work is the first to *explicitly* tackle the feature drift problem. Our experimental findings show that such an explicit handling of feature drifts, result in improved performance compared to existing adaptive approaches.

The remainder of the paper is organized as follows: related work is discussed in Section 2. Basic concepts are discussed in Section 3, together with the problem definition. Our approach for handling feature drifts in textual streams by exploiting different feature/class trends is presented in Section 4. Experimental results are discussed in Section 5. Finally, we conclude in Section 6.

2 RELATED WORK

Data streams are generated from non-stationary distributions and therefore, the learned hypotheses might change over the stream, a phenomenon known as *concept drifts* [11]. A particular type of drifts, not adequately addressed thus far by the related work, is *feature drifts* referring to changes in the relation between a feature and the target class over time. The authors in [27] argue that concept drifts might lead into feature drifts and propose online feature selection to maintain a representative feature space over the stream, upon which a heterogeneous ensemble, of MNB and SVM experts, is trained. Our approach also maintains a representative feature space over the stream, however differently from online feature selection methods our goal is to use feature history over the stream and different feature trend detectors in order to predict future feature values that comprise the input to an final MNB model.

Feature drifts are frequent in textual streams; we focus hereafter on approaches based on Naive Bayes (NB) classifiers. In [17], the authors couple NBs with incremental feature selection based on information gain, for the email spam detection task. In [20], a local likelihood method was presented that extends NBs by considering feature/word information from a particular time-spanning window. In [29], the authors employ exponential weighted moving average to improve NBs in the presence of sudden words. In [32], a temporal extension of NBs is proposed that decays the likelihood of a feature/word for a class and also the class priors based on the recency of their occurrence in the stream. In contrast to the aforementioned methods, in our approach we explicitly model feature drifts and moreover, we consider four different types of feature drifts.

Regarding the application domain, feature drifts for textual streams have been investigated in the context of sentiment analysis e.g., [32] and spam detection, e.g., [21]. Besides, time series prediction methods have been employed for analyzing sentiment in the Twitter stream. For example, in [12], such methods are used to investigate particular sentiment characteristics like sentiment velocity for a given entity, whereas in [28] the goal is to predict the overall sentiment in Twitter stream. Finally in [24], authors improved lexicon-based sentiment analysis using the predicted sentiment of words based on simple time series methods. In our approach, we also rely on time-series prediction methods for predicting different trends for the features/words, however those predictions are integrated in an MNB model for predicting sentiment at the tweet level. Most similar to our work, is the taxi-demand prediction method of [26] which consists of an ensemble of three experts, each capturing different trends, namely, regular, seasonal and autocorrelated. In our approach, we moreover allow for the detection of sudden events and we average out the predictions of the experts assuring that the ensemble will predict as good as the best single expert or the best combination of experts in hindsight. Moreover, the application domain is completely different.

3 BASIC CONCEPTS AND PROBLEM STATEMENT

We observe a textual stream D of documents arriving at distinct time-points $t_1, \cdots, t_i$ with t_i being the current time-point. Depending on the application, at each time-point t_i, a batch of documents instead of a single document might arrive. A document $d \in D$ is represented by the bag-of-words model as $d = \{v_1, v_2, \cdots v_k\}$. Given a predefined set of classes C, the goal of a stream classifier is to maintain a valid classification model over the stream. Without loss of generality, we assume the classification problem is a binary classification problem, i.e., $C = \{+, -\}$. Our learning setting is fully supervised, that is, class labels are available for all documents. However, the label of a document d is available shortly after its arrival. Therefore, the goal is to make a prediction for d using the current classifier and then, upon the arrival of its label to use the labeled instance $(d, c), c \in C$, for training. This setup is known as first-test-then-train or prequential evaluation [10].

Assuming a (hidden) probability distribution P generating the instances of D, the characteristics of P might change with time, i.e., for two time-points $t_i \neq t_j$ it might hold that $P_i \neq P_j$, a phenomenon called *concept drift* [11]. There are different reasons for concept drifts in a stream environment, which can be explained based on the Bayes theorem [14]. In particular, according to the Bayes theorem, the classification of a new document d from the stream depends on the class priors, i.e., $P(c), c \in C$, and the likelihood of observing d under the different classes, i.e., $P(d|c)$:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (1)$$

We assume that the document probability $P(d)$ is the same for all documents $d \in D$. Consequently, concept drift $P_i \neq P_j$ between two different time-points $t_i \neq t_j$ might occur due to changes in the following variables of Equation 1:

[C1] changes in the class priors, i.e., $P_i(c) \neq P_j(c)$.

[C2] changes in the likelihood of a document for a class $c \in C$, i.e., $P_i(d|c) \neq P_j(d|c)$.

[C3] changes in the posterior of a document for a class $c \in C$, i.e., $P_i(c|d) \neq P_j(c|d)$.

A specific type of drifts, not directly captured by the $C1 - C3$ types above, is *feature drifts* [2]. A feature drift occurs when there are changes in the relevance of the features for the learning task over time. We distinguish between two types of feature drifts that might occur between two different time-points $t_i \neq t_j$, namely:

[F1] the likelihood of a feature v for a class c may change with time, i.e., $P_i(v|c) \neq P_j(v|c)$. For example, for the sentiment classification task the feature "summer" displays a seasonal trend being positive in-season and negative off-season.

[F2] the feature space might change, i.e., $F_i \neq F_j$, where F_i (F_j) is the feature space for time-point t_i (t_j, respectively). In literature this is referred as dynamic feature space, e.g., [27].

Our goal is to build a classification model over the stream that besides dealing with concept drifts (types $C1 - C3$), it also explicitly targets feature drifts (types $F1$ and $F2$). An MNB classifier comprises a natural model choice for dealing with feature drifts as it decomposes the problem of estimating the likelihood of a document for a class to the problem of estimating the likelihood of observing the features/words of the document under that class (the so-called class-conditional feature independence assumption). In particular, according to a MNB classifier, the class of a document d is the class that maximizes the posterior, namely:

$$\hat{c}_d = \arg\max_{c \in C} P(c)P(d|c) = \arg\max_{c \in C} P(c) \prod_{v \in d} P(v|c) \quad (2)$$

Based on this formula, MNB captures $F1$ type drifts (variable $P(v|c)$) and $C2$ type drifts (by combining $F1$ type drifts as: $P(d|c) = \prod_{v \in d} P(v|c)$). Moreover, the classifier explicitly models the class priors, so it tackles $C1$ type drifts. Finally, it tackles $F2$ type drifts by maintaining a feature space of the top most frequent features over the stream (c.f. Section 4.2). Since, $C3$ type drift is the result of $C1$ and $C2$ type drifts, we show how to integrate $C1 - C3$ and $F1$ type drifts to an MNB classifier in Section 4.3. The seamless integration of feature drifts in the model is not the only reason for choosing MNB as the learning model. The MNB classifier has proven to perform modestly over streams [7] and in particular, over high dimensional streams like textual streams [30]. Moreover, such models can be efficiently maintained over streams as they rely on simple statistics.

4 LEARNING UNDER FEATURE DRIFTS WITH MNB CLASSIFIERS

We propose an MNB classifier that deals with concept drifts and feature drifts. Our approach is based on the observation that different features/words follow different trends with respect to their association to the class attribute. That is, in a sentiment classification task the word "love" has a *regular* trend in positive class, the word "summer" has a *seasonal* trend, the word "weather" may exhibit *autocorrelated* trend based on the atmospheric conditions of previous days and sudden words, in a Twitter stream, such as "followFriday" may exhibit short (*sudden*) peak trends. Therefore, our idea is to learn those trends for each feature/word independently and combine their predictions via an MNB classifier. An overview of our framework is shown in Figure 2. As shown in this figure,

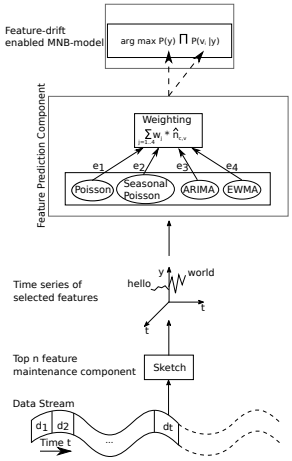


Figure 2: Overview of our approach.

our approach combines three components:

- a *feature prediction component* that uses an ensemble of experts each capturing distinct feature/word trends to tackle the $C1 - C3$ and $F1$ type drifts (Section 4.1),
- a *feature space maintenance component* that maintains a valid fixed size feature space over the stream to fulfill the memory-bounded requirement of streams but also to tackle $F2$ type drifts (Section 4.2), and
- a *feature-drift enabled MNB model* that utilizes the aforementioned components for document classification (Section 4.3).

Consequently, the proposed the feature-drift enabled MNB model can accommodate all types of drift $C1, C2, C3, F1$, and $F2$.

4.1 An ensemble for learning different feature trends

The goal of the ensemble is to predict for each feature/word its value at the next time-point. To this end, we rely on a feature's history from the stream. In particular, let us assume the textual stream $\mathbf{D}$ arriving over time and let $T = \{t_1, .., t_i\}$ be the ordered set of observed time-points up to point t_i. We observe feature occurrences over the stream and in particular, class priors and word class-conditional counts. For simplicity, hereafter we refer to words, however we follow the same methodology for class priors.

We denote $n_{v,c}$ the conditional count of a word v in a class $c \in C$. Assuming an aggregation period of P time units, the sampling values form a discrete univariate time series, defined as follows:

$$N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\} \quad (3)$$

where $n_{v,c}(lP)$ is the observed conditional count of v in class c at the l-th aggregation period. We will refer to the history of the whole time series as *word history*, H_v. Each of the following models will use the whole or a sliding window of H_v, which is referred to *model history*, hereafter.

In order to solve the problem statement, we will present an ensemble of experts each modeling a distinct trend. We present each expert in Section 4.1.1 and the aggregation of their prediction by the ensemble in Section 4.1.2.

4.1.1 Feature prediction by experts. Thus far we discuss four distinct trends, *regular*, *seasonal regular*, *autocorrelated* and *sudden*. In this section we present four models, each to capture one of these distinct trends.

Poisson Model: We employ a *Poisson* model to capture *regular* trends. This model uses the *Poisson* distribution to relate the probability of having n as $n_{v,c}((l+1)P)$, $(P(n_{v,c}((l+1)P) = n))$ defined by the following equation:

$$P(n;\lambda) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{4}$$

where λ is the average number of $n_{v,c}(P)$ over the *model history*. The goal is to learn regular trends over the *whole* history of the word, therefore the *Poisson* model utilizes the whole *word history*. That is $H_{\text{Poisson}} \equiv H_v$.

Seasonal Poisson Model: This model can capture time series with a *seasonal regular* trend. Using the exponential smoothing algorithm [15], the expert will find the average over each group of aggregation periods and then it will aggregate all averages using weighted average with fading weights α_s based on the recency of the group. For example if the aggregation period is $P = 1$ hour and the model group these periods in weeks, storing the last γ_s weeks, it will keep $7 \cdot 24$ aggregation periods for each group (week). Then it will aggregate the average of each week using the exponential smoothing algorithm. These weights increase as we move from the current week to the previous ones. By this, the averages observed two weeks before are less relevant to the average observed in *this* week. We calculate the weights set ω using the exponential smoothing algorithm [15]. Thus we can compute the weighted average over the weeks of the history.

Thus we define ω_s as follows:

$$\omega_s = \alpha_s \cdot \{1, (1-\alpha_s), (1-\alpha_s)^2, \ldots, (1-\alpha_s)^{\gamma_s - 1}\}, \gamma_s \in \mathbb{N} \tag{5}$$

where γ_s is the number of groups of aggregation periods which the model considers and $0 < a_s < 1$ is the smoothing factor. Both variables are user-defined. Having computed the average for each group, $\lambda_k, k = \{1, \ldots, \gamma_s\}$, we can now define the weighted average μ_s as follows:

$$\mu_s = \sum_{k=1}^{\gamma_s} \frac{\lambda_k \cdot \omega_{s_k}}{\Omega_s}, \Omega_s = \sum_{k=1}^{\gamma_s} \omega_{s_k} \tag{6}$$

Then substituting the smoothed average, μ_s, into 4 we get the seasonal Poisson model. The expert keeps the newest γ_s groups containing g aggregation periods (P), consequently its history is sliding and equals to $|H_{\text{Seasonal}}| = \gamma_s\ g\ P$ and it holds $H_{\text{Seasonal}} \subset H_v$.

Autoregressive Integrated Moving Average Model: To model autocorrelated trends, we use the AutoRegressive Integrated Moving Average technique (ARIMA) [8]. The model is defined by three parameters $ARIMA(p, d, q)$, where p and q are the order of the autoregressive and moving average submodels, respectively and d is the lag of the series. The model stores a *sliding window* of size γ_a aggregation periods (P). We denote the model's history by $H_{\text{ARIMA}} \subset H_v$ and it holds $|H_{\text{ARIMA}}| = \gamma_a\ P$. A detailed description of ARIMA model can be found at [9] (Chapter 4 and 5).

Exponential Weighted Moving Average Model: The last model intends to capture regularity only on the short-term history and it is inspired by [29]. The model aggregates the observations in the history $H_{\text{EWMA}} \subset H_v$ using the exponential moving average with fading weights α_e reflecting the recency of the observations. More formally, we use the exponential smoothing [15] to weight each observed value $(N_{v,c}(jP))$ of the short history, containing the last γ_e aggregation periods P of the series $(H_{\text{EWMA}} \subset H_v)$. The model keeps a sliding window over the last γ_e P, $|H_{\text{EWMA}}| = \gamma_e\ P$. Then predicts the future value as the weighted average of the values in H_{EWMA}, using the same logic as the second model. We define the set of weights w_e for each observed value inside the sliding window of size γ_e:

$$\omega_e = \alpha_e \cdot \{1, (1-\alpha_e), (1-\alpha_e)^2, \ldots, (1-\alpha_e)^{\gamma_e - 1}\}, \gamma_e \in \mathbb{N} \tag{7}$$

We combine the predictions using the ω_e of 7 as their weighted average μ_e as follows:

$$\mu_e = \sum_{k=1}^{\gamma_e} \frac{X_{i,t_k} \cdot \omega_{e_k}}{\Omega_e}, \Omega_e = \sum_{k=1}^{\gamma_e} \omega_{e_k} \tag{8}$$

4.1.2 Averaging predictions of experts. Our assumption is that the trend of each word can be modeled in the best way by one expert or by a (weighted) combination of them. More formally, given the time series $N_{v,c}$ up to aggregation period $l-1$, the M experts will predict for the next aggregation period l and input their values to the ensemble, which will aggregate their prediction to result in the final prediction. In this work, we have four experts ($M = 4$). For them we denote their predictions as $\hat{n}_{v,c}(l)^{\text{Poisson}}, \hat{n}_{v,c}(l)^{\text{Seasonal}}, \hat{n}_{v,c}(l)^{\text{ARIMA}}, \hat{n}_{v,c}(l)^{\text{EWMA}}$ and of the ensemble as $\hat{n}_{v,c}(l)^{\text{Ensemble}}$. Let $n_{v,c}(l)$ be the observed value of the conditional count of v in c, at the next aggregation period l. We assume each of the predicted values and the observed one are in the range of $[0, B]$, where $B \in \mathbb{N}$ is the upper limit of the conditional count of such word in the total stream and it is user-defined.

To measure the performance of the ensemble we use a *loss function* L which outputs a non-negative quantity $L(n_{v,c}(l), \hat{n}_{v,c}(l))$ indicating the difference between the observed (true) outcome and the predicted one. Assuming an expert that can best model the input then we can measure the difference of the ensemble performance compared to this learner using the regret measure, r, which is defined as:

$$r = \sum_l L(n_{v,c}(l), \hat{n}_{v,c}(l)^{Ensemble}) - \min_{\substack{e=Poisson, Seasonal,\\ ARIMA, EWMA}} \sum_l L(n_{v,c}(l), \hat{n}_{v,c}(l)^e) \tag{9}$$

In order to predict with guaranteed regret we follow the results of [19]. To this end we use the weighted average algorithm, WAA, as formulated in [19]. We show the whole method in algorithm 1, which follows the prequential process. We use the square loss function, $L(n_{v,c}, \hat{n}_{v,c}) = (n_{v,c} - \hat{n}_{v,c})^2$. By the equation 8 in [19], for $n_{v,c} \in [0, B]$ we compute the $\tilde{c}_L$ equal to $2B^2$. This weighting scheme gives us constant upper bound on the regret of the ensemble predictions, as stated by the following theorem:

Theorem 4.1. *[Kivinen & Warmuth [19]] Let L be a monotone convex twice differentiable loss function the WAA as in algorithm 1*

with uniform initial weights, $w_{l=1,i} = 1$ and with $c \geq \tilde{c}_L$. Then for any sequence of $N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\}$ we have

$$Loss_{WAA}(N_{v,c}) \leq (min_e Loss_e(N_{v,c})) + c ln M. \quad (10)$$

As stated in [19], having a bound for the regret compared to the combination of learners that models best the input is better than regrets compared to the best single model (4.1). Using the same setting, we can state the following theorem that bounds the regret compared to the best combination as follows:

Theorem 4.2. *[Kivinen & Warmuth [19]] Let L be a monotone convex twice differentiable loss function in WAA as in algorithm 1 using arbitrary initial weights, $\mathbf{q}_{l=1}$ and parameter $c = \tilde{c}_L$. Then for any sequence of $N_{v,c} = \{n_{v,c}(P), \ldots, n_{v,c}(lP)\}$ and for all probability vectors $\mathbf{p}$ we have*

$$Loss_{WAA}(S) \leq Loss_{\mathbf{p}}^{\text{avg}} + \tilde{c}_L d_{re}(\mathbf{p}, \mathbf{q}). \quad (11)$$

Where $\mathbf{q}_t$ is the weight vector to combine each expert. As the weight vector is normalized and sums up to 1 is also a probability vector. Let p be such a probability vector, being the best combination of averaging the experts. To measure the difference of the two probability vectors, we use the *relative entropy* defined as $d_{\text{re}}(\mathbf{p}, \mathbf{q}) = \sum_{b=1}^{n} p_i \ln(p_i/q_i)$.

4.2 A sketch-based approach for feature space maintenance

To fulfill the bounded memory requirement of data streams [7], we maintain a fixed-size feature space over the course of the stream. We have used three versions of the algorithm. The `baselineSketch` uses the original algorithm by [25]. In our setting the sketch saves pairs of (word, occurence count). To account for the importance of frequent words we use the `fadingSketch` version. The last version is the `adwinSketch` which uses the ADWIN algorithm [4] as change detector to remove a saved word when it is significantly not used anymore in the stream. All different sketches allow us to maintain a fixed size feature space over the stream. Depending on the sketch type though, the update of the feature space varies. That is the sketch decides to remove not frequent words to accommodate new words coming from the stream. We devote Section 5.2 on the experiments to evaluate the impact of the sketch.

4.3 Incorporating feature drifts in MNB

This last section combines the previous two components, sketch and ensemble, to introduce a MNB classifier that can tackle the stated problem. We refer to this model as `temporalMNB` . As shown we present by its algorithm 1, for each arriving document d_i, firstly its words are saved in the sketch. Then assuming the $l-1$ as the current index of periods for the priors of each class c and the conditional counts of each word v. The ensemble predicts the $\hat{n}_c(lP)$ and $\hat{n}_{v,c}(lP)$, for the l-th (next) aggregation period using the weighted average of the experts' prediction (EnsemblePredict()). Then we define $P_{\text{temporal}}(c)$ as the predicted class prior probability and $P_{\text{temporal}}(w_i|c)$ as the respective predicted likelihood for the word. The `temporalMNB` model predicts using the following equation:

Algorithm 1 Temporal MNB

diffTime, t^{previous}, $l \leftarrow \{0_c, 0_1, \ldots, 0_{|V|}\}$
$x_{\text{previous}}, x_{\text{period}} \leftarrow \{[0,0]_c, [0,0]_1, \ldots, [0,0]_{|V|}\}$
$V \leftarrow \{\}$
$W \leftarrow \{\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_c, \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_1, \ldots, \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}_{|V|}\}$
$N \leftarrow \{\{\{\}, \{\}\}_c, \{\{\}, \{\}\}_1, \ldots, \{\{\}, \{\}\}_{|V|}\}$

function ensembleTrain(x_v, W_v, $\hat{x}_v$, t)
  diffTime[v] $\leftarrow$ diffTime[v] + ($t - t^{\text{previous}}[v]$)
  if diffTime[v] $\leq P$ **then** ▷ Aggregate for P time units
    for all $c_j \in C$ **do**
      $x_{\text{previous}}[v][c] \leftarrow x_{\text{previous}}[v][c] + x_v[c_j]$
      $N_v[c_j][l[v]] \leftarrow x_{\text{previous}}[v][c_j]$
    end for
  else ▷ New aggregation period
    for all $c_j \in C$ **do**
      $x_{\text{period}}[v][c_j] \leftarrow x_{\text{previous}}[v][c_j]$ ▷ Observe real value
      $N_v[c_j][l[v]] \leftarrow x_{\text{period}}[v][c_j]$
      for $e = 1$ to 4 **do** ▷ Update experts' weights
        $W_v^{l[v]+1}[c_j, e] \leftarrow \frac{W_v^{l[v]}[c_j,e] exp(-(x_{\text{period}}[v][c_j] - \hat{x}_v[c_j])^2/c)}{\sum_{e=1}^{4} W_v^{l[v]}[c_j,e] exp(-(x_{\text{period}}[v][c_j] - \hat{x}_v[c_j])^2/c)}$
      end for
    end for
    $k[v] \leftarrow l[v] + 1$, diffTime[v] $\leftarrow 0$, $t^{\text{previous}}[v] \leftarrow t$ ▷ Reset aggregation
    for all $c_j \in C$ **do**
      $x_{\text{previous}}[v][c_j] \leftarrow x_{\text{previous}}[v][c_j] + x_v[c_j]$
    end for
  end if
  return $W_v^{k[v]}$
end function

function ensemblePredict(X_v, W_v, c_j, t)
  $\hat{\mathbf{x}}_{\text{experts}} \leftarrow [0,0,0,0]$, $\hat{x}_{\text{ensemble}} \leftarrow 0$
  for $e = 1$ to 4 **do** ▷ Experts predict
    $\hat{\mathbf{x}}_{\text{experts}}[e] \leftarrow \text{Expert}_e.predict(X_v[c_j], W_v[c_j], t)$
  end for
  $\hat{x}_{\text{ensemble}} \leftarrow W_v[c_j] \cdot \hat{\mathbf{x}}_{\text{experts}}$ ▷ Ensemble predicts
  return $\hat{x}_{\text{ensemble}}$
end function

function main( )
  for all $(d_i, t_i) \in D$ **do**
    $V_{d_i} \leftarrow \text{features}(d_i)$
    $V \leftarrow \text{sketch}(V_{d_i}, V)$
    for all $c_j \in C$ **do** ▷ Predict for priors and likelihoods
      $\hat{N}_c[c_j] \leftarrow \text{ensemblePredict}(N_c, W_c, c_j, t_i)$
      for all $v_{d_i} \in V_{d_i}$ **do**
        $\hat{N}_{v_{d_i}}[c_j] \leftarrow \text{ensemblePredict}(N_{v_{d_i}}, W_{v_{d_i}}, c_j, t_i)$
      end for
    end for
    ▷ Predict for document by eq.12
    $\hat{c}_d = \arg\max_{c_j \in C} P_{temporal}(c_j) \prod_i P_{temporal}(v_i|c_j)^{\text{freq}_{v_i}^d}$
    ▷ Train for prios and likelihoods
    Observe n_c for both classes at t_i
    $W_c \leftarrow \text{ensembleTrain}(n_c, W_c, \hat{N}_c, t_i)$
    for all $v_{d_i} \in V_{d_i}$ **do**
      Observe $n_{v_{d_i}}$ for both classes at t_i
      $W_{v_{d_i}} \leftarrow \text{ensembleTrain}(n_{v_{d_i}}, W_{v_{d_i}}, \hat{N}_{v_{d_i}}, t_i)$
    end for
  end for
end function

$$\begin{aligned} \hat{c}_{d_i} &= \arg\max_{c \in C} P_{\text{temporal}}(c|d') \\ &= \arg\max_{c \in C} P_{\text{temporal}}(c) \prod_{v \in V_{d_i}} P_{\text{temporal}}(v|c) \end{aligned} \quad (12)$$

Finally, the true class of the d_i is observed. We aggregate the observed values. When a new aggregation period (lP) is reached the ensemble observes the real outcome $n_c(lP)$, $n_{v,c}(lP)$, it incurs loss and updates its weights as shown in EnsembleTrain().

5 EXPERIMENTAL EVALUATION

We evaluated our approach on a real dataset from Twitter, which is related to sentiment classification (Section 5.1). The role of the sketch in feature space selection is investigated in Section 5.2. The prediction accuracy is evaluated in Section 5.3, where we compare our approach to several baselines, listed below:

- `accumulativeMNB` [3], the original MNB for streams where the word-class and class counts are accumulated over the stream. This is an incremental approach, however it does not deal with drifts [30].
- `fadingMNB` [32], adapts `accumulativeMNB` for concept drifts by introducing a fading function that decays (the accumulated over the stream) class counts and conditionals counts based on the recency of their observation. The fading function depends on the decay factor λ which controls how fast the observations age and on the aggregation period (e.g., daily vs weekly).
- `aggressivefadingMNB` [32], is a variation of `fadingMNB` that stores the decayed counts and applies ageing over them, thus leading to faster adaptation. The `aggressivefadingMNB` depends on the same parameters as `fadingMNB`.

We implemented our approach in MOA, Version 2017.06 [5]. For the sketch, we used its publicly available implementation from the MOA-tweetreader package [6]. For ARIMA, we used the available Java library[1]. Our implementation is available at a public repository[2].

We employed prequential evaluation over a sliding window of 1,000 instances over each day of the dataset. As evaluation measures, we use accuracy and kappa [3]. Accuracy is the percentage of correctly predicted instances over all instances of the sliding window. However, accuracy cannot indicate if the model achieved a high value by overfitting on the current data distribution. To this end, the kappa statistic is also employed that normalizes the model performance by that of a chance classifier. We report on both average and standard deviation of those measures for each sliding window of each day.

5.1 Datasets

TSentiment [13] consists of 1.6M tweets collected over a period of three months (April 6 - June 25, 2009) annotated as positive or negative through distant supervision using emoticons as proxies for the class labels. We preprocessed the data set following the typical preprocessing steps for sentiment analysis [[16], [1]]. Firstly, we treated negations spliting them into two words for example we substitute "can't" with "can not". Then, we removed hashtags, mentions and URLs and converted all remaining words to lowercase. We replaced each positive emoticon with the word "EMO_POS" and each negative with the word "EMO_NEG". We substituted a word with repetitions of letters with the normal writing of the word for example "cooool" to "cool". We expanded acronyms using the dictionary of [1]. We removed punctuation, numbers and whitespaces. Also we used stemming, keeping the common base form of each word. Finally, we removed stop words using the list in [22].

The dataset is in overall balanced (50% positive, 50% negative tweets) but the empirical class distribution changes with time as shown in Figure 3 where the tweets are aggregated on a daily basis. As we can see, at the end of the monitoring period only the negative class is observed. This is an example of concept drift. No feature drifts are known for this data set.

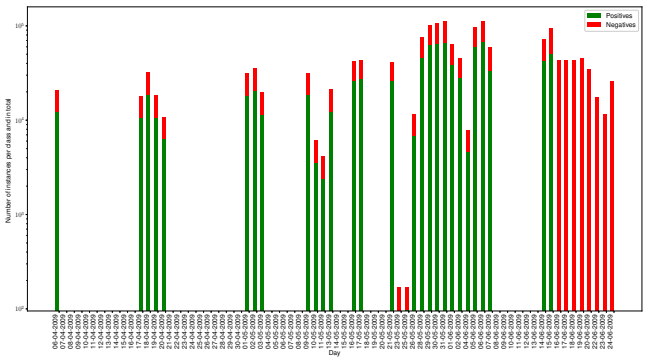


Figure 3: TSentiment: Class distribution over the stream (daily aggregation).

Email data [18] consists of 1,500 emails of 913 attributes. The task is to predict if an email is interesting or not for a given user. The authors know the trends of drift for some words, namely "medicine", "baseball" and "space".

Spam data [18] consists of 9,324 emails of 500 attributes. The task is to predict if an email is spam or ham. The authors use this data set as a complementary to **Email** data set to experiment with a data set where the features evolve smoother compared to the **Email** data set. For both of these data sets, we preprocessed the data set using the same techniques as before. As these two data sets are small we did not use the sketch component but we kept all found words. The goal using this data set is to verify that our proposed ensemble can capture such already known trends.

5.2 Sketch evaluation

The sketch (c.f., Section 4.2) allows us to adapt to feature drifts in a memory-efficient way, by controlling the number of maintained words (parameter n_f). The adaptation rate of the sketch and consequently of the feature space depends on the selected sketch type (`baselineSketch`, `fadingSketch`, `adwinSketch`). In this section we evaluate the effect of n_f (sketch size) and of the different sketch types (sketch adaptation type).

5.2.1 Sketch size. We experimented with three different sketch sizes n_f: 3,000, 5,000 and 10,000 distinct words. We fixed the type of sketch to `baselineSketch` that has the lowest ability to adapt to changes. We evaluate the impact of n_f on stream classification using the `accumulativeMNB` classifier.

As shown in Figure 4, the performance has two phases, before and after the point of the class drift. Before class drift, the larger

[1] Java TimeSeries Library: https://github.com/signaflo/java-timeseries.
[2] GitHub Repository:https://github.com/damianosmel/temporalMNB

sketch size results to the best performance, after change the smallest sketch size performs better. This is reasonable, as during stable phases the larger the sketch the more frequent words are kept thus improving the performance of the learner. However, when a change occurs, a smaller sketch will faster forget the old words and will start maintaining more recent words. A trade-off is $n_f = 5,000$, which performs decently in terms of both accuracy and memory usage. So, for the rest of the experiments we fixed n_f to 5,000 distinct words.

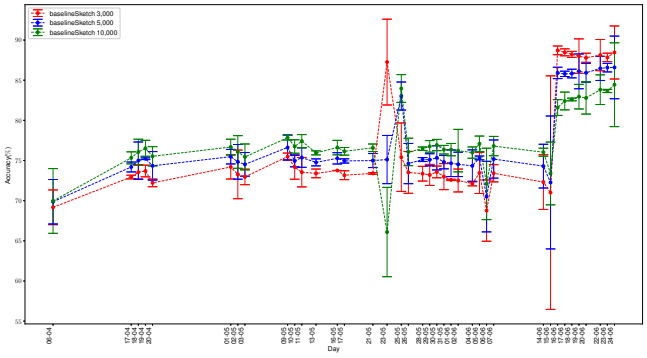


Figure 4: TSentiment: Effect of sketch size (daily aggregated stream, 1,000 instances evaluation window, `accumulativeMNB` classifier, `baselineSketch` adaptation type).

5.2.2 Sketch adaptation type. We evaluate the performance of the different sketch types, `baselineSketch`, `fadingSketch` and `adwinSketch`, using a fixed sketch size of $n_f = 5,000$ words. Again, we use `accumulativeMNB` as the learning model.

From the results, in Figure 5, we can see that the performance has again two phases separated by the point of class drift. Before the drift, the `fadingSketch` performs better closely followed by the `baselineSketch`. After drift, `adwinSketch` performs better, followed again by `baselineSketch`. The difference in the behavior of `fadingSketch` and `adwinSketch` before and after the drift point is due to their update rate. The `adwinSketch` adapts faster to changes as it replaces words which display significant drop in their usage. On the other hand, `fadingSketch` downgrades words based on their recency without counting for changes in their usage frequency. Since we are interested in a sketch that performs decently in both times of stability and times of change, we chose `baselineSketch` that was consistently the second best performing sketch for the experiments hereafter.

5.2.3 Sketch variability. Thus far, we have evaluated the effect of sketch size and sketch adaptation type on the performance of a stream classifier. In this section, we evaluate the sketch variability, i.e., how the words in the sketch are replaced over the stream. Recall that due to the memory constraint, only a fixed number of n_f words can be maintained over the stream.

To this end, we plot i) *the existence of change* as the percentage of instances introducing new words over the total number of instances of the day. Moreover, we are interested in the degree of

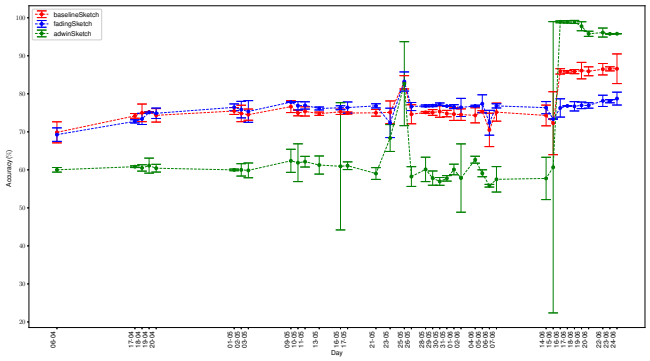


Figure 5: TSentiment: Effect of sketch adaptation type (daily aggregated stream, 1,000 instances evaluation window, `accumulativeMNB` classifier, $n_f = 5,000$ words).

such changes. To this end, we also plot ii) *the degree of change* as the percentage of new introduced words in the documents with new words normalized with the length of the document and averaged over all documents with new words. The results are shown in Figure 6 (blue for (i) and red for (ii)).

We can observe that approximately half of the tweets introduce new words (blue axis) over all the course of the stream. However, the percentage of change for each such tweet is roughly 20%, i.e one new word is found in every five words of a document (red axis). We observe a large variability in the sketch in the beginning of the stream (from 6/4 to 15/4), which is probably due to the instable initialization of the sketch on 6/4. The variability scores are lower after the class drift point (Figure 3). A large drop is also observed on 23/5 where we observe the smallest number of documents introducing new words. However, the actual number of such documents is small as we can see from Figure 3.

Consequently, we can state that the `baselineSketch` produces a feature space that does not drastically change. Thus the resulted space can be employed by the learning component.

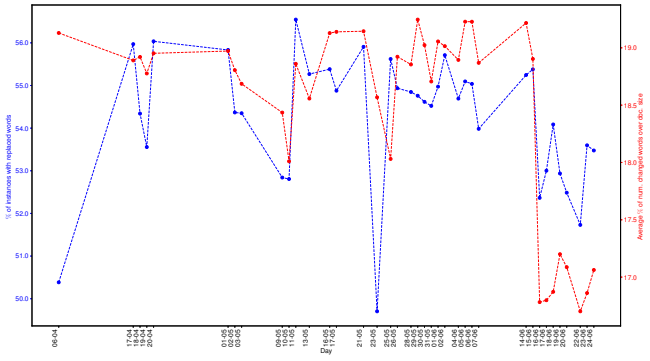


Figure 6: TSentiment: Variability of sketch, left y axis (blue): % of tweets introducing new words per day, right y axis (red): average % new words per document over all documents introducing new words (daily aggregated stream, `baselineSketch` with $n_f = 5,000$ words).

5.3 Performance evaluation

5.3.1 Parameter setting. Before comparing our approach to the baselines, we tuned their parameters. That is, for the `fadingMNB` and the `aggressivefadingMNB` we tested for $\lambda = \{0.1, 1, 2\}$ and aggregation of *second* and *hour* based on the results of [32]. The best values for these two baselines is *hourly* aggregation and $\lambda = 0.1$. This means halving the conditional count after 10 hours of observing the word.

For the ensemble, we set the aggregation period P based on two facts. First, we intend to use value of P such that the ensemble can follow potential drifts of the class or the word feature. Secondly, the series for the class has very short interval as it comes by each instance but the respective series for a word has a wider interval. Thus, we tuned the parameters to capture distinct trends using aggregation period of $P = 1$ *second* for the class prior and $P = 1$ *minute* for the conditional counts of each word. For the first model, *Poisson*, we do not need to tune parameters as it is parameter-free. For the *seasonal* model, we set the sliding window history to $\gamma_s = 1$ week, i.e $H_{\text{Seasonal}} = 2$ week, capturing seasonal events during two weeks period. As the first two models capture long term dependencies we will select parameters for the last two to capture short term dependencies. Thus, for the *ARIMA* model, we use $(p = 1, d = 1, q = 1)$ to capture non-stationary time dependencies, over the last 50 periods, $H_{\text{ARIMA}} = 50P$. Following the same intuition, with the *EWMA* model that captures short time-dependencies we used a sliding window of size $\gamma_e = 22$ periods, $H_{\text{EWMA}} = 22\ P$. We set the values for fading factors of *Seasonal* and *EWMA* equal to $\alpha_s = 0.9$ and $\alpha_e = 0.1$ respectively, using the corresponding equations from [15]. Experimentally, running the first three days of the stream and observing the maximum value for the class prior and the conditional count of a word we set $B = 30$, consequently $\tilde{c}_L = 1800$.

5.3.2 Performance. In Figure 7, the accuracy of the different approaches over the TSentiment dataset is depicted. We can observe that our approach performs always better than the baselines except for two days 23/05 and 25/05. After manual observing the distribution of timestamps in these two days we found that on both days we had only 169 instances having a time span of 2 minutes, on 23/05 there are instances from 18:04:32 to 18:06:34 and on 25/05 are from 10:42:48 to 10:44:22. Given our parameter selection, the ensemble samples every second for the class prior and every hour for the likelihood of each word, all changed words found in these days cannot be used for learning and consequently for their prediction the Laplace correction is used. From Figure 6, we can observe that the percentage of documents with changed words is close to 50% and 56% and the average number of changed words in these documents is close to 2 out of 10. Thus, in addition having words with short life span, the half of the documents introduce new words which again span a short period of time not allowing the ensemble to be used for prediction. On the contrary, a method that the update of the model for a word is per instance, and not per aggregation period, should not show this drop in accuracy. This is true for the other two competitive baselines, the `accumulativeMNB` and `fadingMNB`.

We also plot the *kappa* measure in Figure 8 that normalizes the accuracy of the classifier with that of a chance predictor. Again, our approach outperforms the rest of the approaches over the stream,

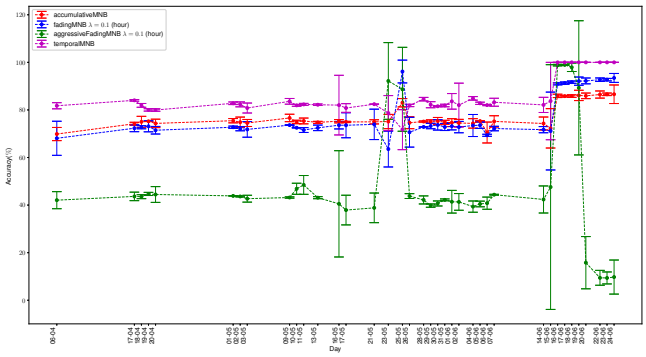


Figure 7: TSentiment: Accuracy over the stream for the different methods.(daily aggregated stream, 1,000 instances evaluation window, `baselineSketch`, $n_f = 5,000$ words)

with the only exception of one day (25/05). This is caused because the time span of instances is shorter than the sampling rate of the ensemble as previously explained. To check the statistical

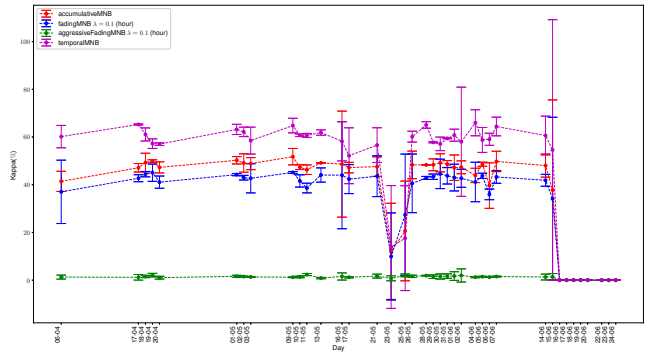


Figure 8: TSentiment: Kappa over the stream for the different methods.(daily aggregated stream, 1,000 instances evaluation window, `baselineSketch`, $n_f = 5,000$ words)

significance of our findings, we performed the McNemar's test[3]. We found that `temporalMNB` makes different predictions compared to the best baseline, `accumulativeMNB`, significantly with $p < .001$.

5.4 Efficiency

The running times for the different methods are presented in Table 1. Our approach is 713 times slower for total learning per instance comparing to the `accumulativeMNB` baseline; this is due to the employment of four different predictors for the ensemble. However, our approach can still perform at modest speed with an average of $2.53 \cdot 10^{-2}$ seconds per instance for the total learning process. Finally, the needed main memory to run this algorithm for the **TSentiment** is maximum 20 *GB*.

5.5 Ensemble Validity

5.5.1 Parameter setting. Compared to the previous data set we note that this data set does not have time stamps so we assume

[3] McNemar's test: https://en.wikipedia.org/wiki/McNemar's_test.

Method	Training	Testing	Total
accumulativeMNB	$4.85 \cdot 10^{-5}$ (1)	$1.67 \cdot 10^{-5}$ (1)	$6.53 \cdot 10^{-5}$ (1)
fadingMNB	$7.14 \cdot 10^{-6}$ (0.1)	$2.31 \cdot 10^{-5}$ (1.3)	$3.03 \cdot 10^{-5}$ (0.5)
aggressivefadingMNB	$9.66 \cdot 10^{-6}$ (0.2)	$2.04 \cdot 10^{-5}$ (1.2)	$3.01 \cdot 10^{-5}$ (0.5)
temporalMNB	$8.34 \cdot 10^{-5}$ (1.72)	$2.52 \cdot 10^{-2}$ (1, 508.9)	$2.53 \cdot 10^{-2}$ (387.4)

Table 1: Execution times per instance in seconds and relative times compared to the accumulativeMNB in parenthesis.

periods over number of instances. Then, we tuned the parameters of the baselines. For the fadingMNB and aggressivefadingMNB, we found $\lambda = 2.0$ and $\lambda = 0.1$ over a period of 50 instances for the **Email** and **Spam** data sets respectively. We also tuned the parameters for temporalMNB . We found the same parameters for both data sets. For both the class prior and the conditional counts of words we use $P = 2$ instances. The *Poisson* model is parameter-free. For the *seasonal* model, we set the sliding window history to $\gamma_s = 2$ instance groups, i.e $H_{\text{Seasonal}} = 2$ groups, where an instance group is $25P$ instances. For the *ARIMA* model, we used $(p = 1, d = 1, q = 1)$ to capture non-stationary time dependencies, over the last 150 periods, $H_{\text{ARIMA}} = 150\ P$. For the *EWMA* model, we used a sliding window of size $\gamma_e = 22$ periods, $H_{\text{EWMA}} = 22\ P$. We set the values for fading factors of *Seasonal* and *EWMA* equal to $\alpha_s = 0.9$ and $\alpha_e = 0.1$ respectively, using the corresponding equations from [15]. Experimentally, running the first instances of the stream and we observed the maximum value for the class prior and the conditional count of a word we set $B = 2$ and $B = 6$, consequently $\tilde{c}_L = 8$ and $\tilde{c}_L = 72$ for prior and the conditional count respectively.

5.5.2 Performance. We plot the accuracy and kappa measure every 50 instances in order to be compared to the evaluation of the original research work of [18]. The accuracy plots are shown in Figure 9. We can observe that the ensemble achieves competitive results for both data sets. For the **Email** data set the temporalMNB cannot achieve the best results as it uses experts of sliding windows and this data set contains abrupt class changes. However, after an abrupt class change, for example at the batch of 350 instances, the ensemble recovers to its normal accuracy after two batches of instances. For the **Spam** data set which has a more gradual class change the temporalMNB achieved better performance competitive to fadingMNB.

5.5.3 Validity. As for the **Email** data set the authors know the trends of three words, "medicine", "space" and "baseball", we aim to validated the first stated theorem 4.1. To this end, we plotted the observed value and the predicted values of the best single expert (ARIMA) and of the ensemble for "medicine" and "space" as shown in the Figure 11. From this figure, we observe that truly the ensemble follows the best single expert, and the observed value, in an optimal way much lower than the given upper $c \ln M = 99.8$.

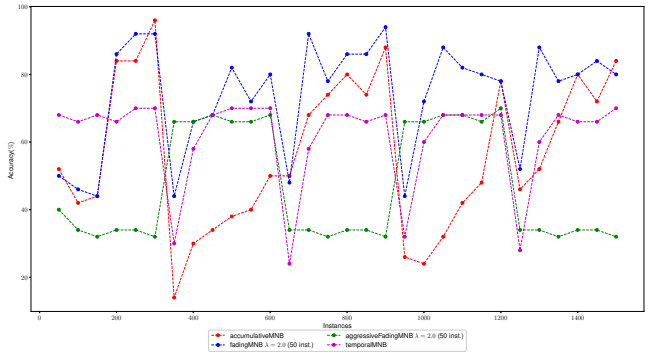


Figure 9: Email: Accuracy of batches of 50 instances over the stream.

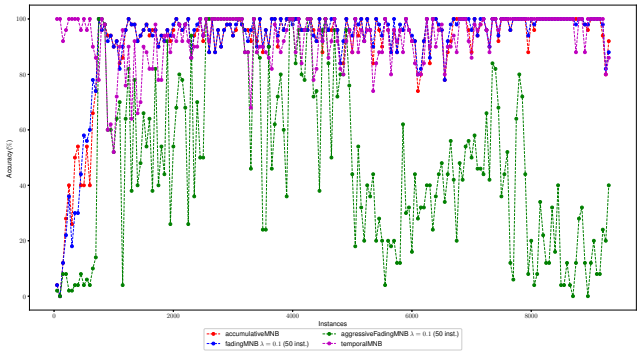


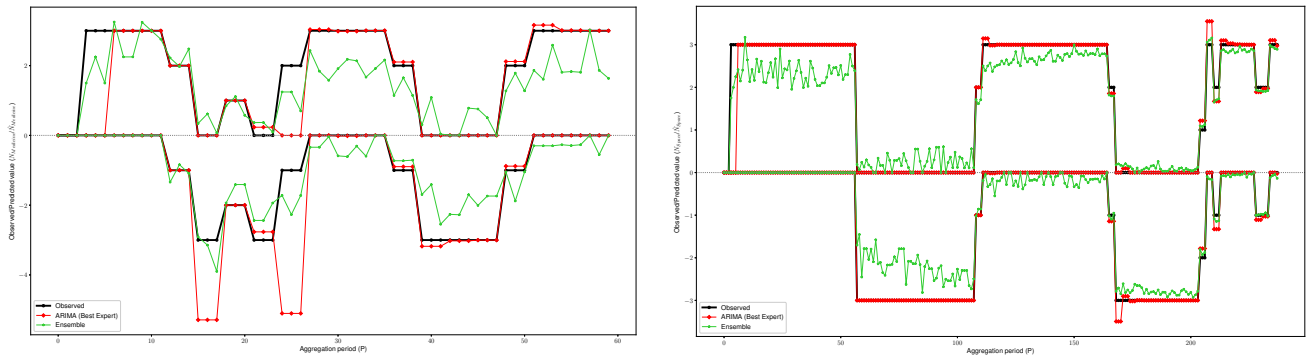
Figure 10: Spam: Accuracy of batches of 50 instances over the stream.

6 CONCLUSION

We have proposed a method to tackle concept and feature drift using two components. The first component is a sketch to maintain an updated feature space over the stream. The second is an ensemble to average out potential different trends of a feature. For the average out mechanism, we can guarantee that we can find the best single or probabilistic combination of trends representing the drift of each feature. Our experiments on textual streams demonstrated that our approach achieves competitive results compared to baselines.

7 ACKNOWLEDGEMENTS

The work of the first author was supported by the German Research Foundation (DFG) within the project OSCAR (Opinion Stream Classification with Ensembles and Active leaRners) for which the last two authors are the project's principal investigators. The last author and this work is partially supported by the European Commission within the ERC Advanced Grant ALEXANDRIA under grant No.



(a) "Medicine". **(b) "Space".**

Figure 11: Email: Observed and predicted conditional counts for the words "medicine" and "space" by the best single expert and the ensemble. The positive y-axis shows the counts for the positive class and the negative for the negative class.

339233. The first author would like to thank Dr. Luis Moreira Matias for his help understanding his research work [26], Jacob Rachiele the main developer of TimeSeries library in Java and Dr. Wouter M. Koolen ensuring the values for $\tilde{c}$ for the WWA.

REFERENCES

[1] Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. 2016. A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter.. In *KDWeb*.

[2] Jean Paul Barddal, Heitor Murilo Gomes, Fabrício Enembreck, and Bernhard Pfahringer. 2017. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software* 127 (2017), 278–294.

[3] Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in Twitter streaming data. In *International conference on discovery science*. Springer, 1–15.

[4] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 443–448.

[5] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. 2010. MOA: Massive online analysis. *Journal of Machine Learning Research* 11, May (2010), 1601–1604.

[6] Albert Bifet, Geoffrey Holmes, and Bernhard Pfahringer. 2011. MOA-tweetreader: real-time analysis in Twitter streaming data. In *International Conference on Discovery Science*. Springer, 46–60.

[7] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. 2009. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 139–148.

[8] George EP Box and Gwilym M Jenkins. 1976. *Time series analysis: forecasting and control, revised ed.* Holden-Day.

[9] J. Cryer and K. Chan. [n. d.]. Time Series Analysis with Applications in R.

[10] Joao Gama. 2010. *Knowledge discovery from data streams*. CRC Press.

[11] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46, 4 (2014), 44.

[12] Anastasia Giachanou and Fabio Crestani. 2016. Tracking sentiment by time series analysis. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 1037–1040.

[13] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1, 12 (2009).

[14] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. 2012. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence* 1, 1 (2012), 89–101.

[15] Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20, 1 (2004), 5–10.

[16] Zhao Jianqiang and Gui Xiaolin. 2017. Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis. *IEEE Access* 5 (2017), 2870–2879.

[17] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2006. Dynamic feature space and incremental feature selection for the classification of textual data streams. *Knowledge Discovery from Data Streams* (2006), 107–116.

[18] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2010. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems* 22, 3 (2010), 371–391.

[19] Jyrki Kivinen and Manfred K Warmuth. 1999. Averaging expert predictions. In *European Conference on Computational Learning Theory*. Springer, 153–167.

[20] Guy Lebanon and Yang Zhao. 2008. Local likelihood modeling of temporal text streams. In *Proceedings of the 25th international conference on Machine learning*. ACM, 552–559.

[21] Guanjun Lin, Nan Sun, Surya Nepal, Jun Zhang, Yang Xiang, and Houcine Hassan. 2017. Statistical Twitter Spam Detection Demystified: Performance, Stability and Scalability. *IEEE access* 5 (2017), 11142–11154.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[23] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. 2018. xStream: Outlier Detexion in Feature-Evolving Data Streams. (2018).

[24] Damianos P Melidis, Alvaro Veizaga Campero, Vasileios Iosifidis, Eirini Ntoutsi, and Myra Spiliopoulou. 2018. Enriching Lexicons with Ephemeral Words for Sentiment Analysis in Social Streams. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*. ACM, 38.

[25] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*. Springer, 398–412.

[26] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

[27] Hai-Long Nguyen, Yew-Kwong Woon, Wee-Keong Ng, and Li Wan. 2012. Heterogeneous ensemble for feature drifts in data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 1–12.

[28] Le T Nguyen, Pang Wu, William Chan, Wei Peng, and Ying Zhang. 2012. Predicting collective sentiment dynamics from time-series social media. In *Proceedings of the first international workshop on issues of sentiment discovery and opinion mining*. ACM, 6.

[29] Kyosuke Nishida, Takahide Hoshide, and Ko Fujimura. 2012. Improving tweet stream classification by detecting changes in word probability. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 971–980.

[30] Myra Spiliopoulou, Eirini Ntoutsi, and Max Zimmermann. 2016. Opinion Stream Mining. *Encyclopedia of Machine Learning and Data Mining* (2016), 1–10.

[31] Soroush Vosoughi, Helen Zhou, and Deb Roy. 2016. Enhanced twitter sentiment classification using contextual information. *arXiv preprint arXiv:1605.05195* (2016).

[32] Sebastian Wagner, Max Zimmermann, Eirini Ntoutsi, and Myra Spiliopoulou. 2015. Ageing-based multinomial naive bayes classifiers over opinionated data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 401–416.