Quantification Over Time

Feiyu Li, Hassan H. Gharakheili, and Gustavo Batista (✉)

University of New South Wales, Sydney NSW 2052, Australia
{feiyu.li, h.habibi, gbatista}@unsw.edu.au

Abstract. Quantification is the supervised machine learning task that estimates the class distribution in a sample. Therefore, quantification applications typically involve predicting aggregated quantities, such as the prevalence of positive comments about a product, personality or company on a set of social media posts. However, quantification analysis is more informative when performed over time, such as when we are interested in tracking public opinion on social media and relating changes in opinion with relevant events. The vast majority of the literature considers quantification as a standalone task, assuming the output of quantifiers to be independent even when applied to temporal data. This paper proposes a new quantification task, Quantification over Time (QoT), that allies quantification with time series forecasting methods. We propose an approach based on the Kalman filter, which can help improve the performance of standalone quantifications and a general framework that includes both ours and SOTA methods. In an experimental comparison with several textual datasets and numeral datasets, we show that our method outperforms existing methods for QoT in the literature, such as a simple composition of the *classify and count* method with moving averages and *ReadMe2* as a standalone quantifier. We also show that our proposal can outperform several baselines, including recently proposed quantifiers used as standalone approaches. Codes are available at https://github.com/frieli11/quantification-over-time

Keywords: Machine learning · class distribution estimation · quantification · time series · Kalman filter.

1 Introduction

Quantification is the supervised Machine Learning task defined by Forman [9] as the induction of a system "*that takes an unlabeled test set as input and returns its best estimate of the number of cases in each class.*" This definition opposes quantification to classification that focuses on predicting individual instances' labels. Therefore, quantification finds application in areas where we are interested in understanding the behaviour of groups instead of predicting the class of individuals.

The simplest existing quantifier is *Classify and Count* (CC). It consists of the direct application of classifiers to quantification problems. This approach classifies each instance in the unlabeled set and counts the number of instances predicted in each class. CC is a biased quantifier, as previous research [8,9] showed a systematic error that increases linearly as the unlabeled set class distribution differs from the classifier's training distribution. CC's limitations have led to the proposal of several recent quantifiers that can accurately estimate the class distribution for a wide range of class prevalence.

Quantification papers have accessed their proposals as standalone methods, in which the current decision relies only upon the current data and ignores previous quantifications. The main reason is that the quantification literature has adopted the Artificial Prevalence Protocol (APP) in their experimental setups. The APP uses subsampling on a classification dataset to create many test sets with varying class distributions. However, such experimental protocol does not match how quantifiers are often used in practice. As Forman noticed in his seminal work [9], quantifiers are useful "*to monitor for changes or trends in the class distribution over time.*"

For example, a standalone quantification can estimate a presidential candidate's approval rate based on posts on social media from a certain period, such as the last 24 hours [26]. However, a more fundamental question is how the public support of the candidate varies over the election period, leading to a series of quantifications that are likely to show time dependency. Fig. 1 shows how Biden and Trump supporters' online activity relates to events such as the candidate's debates [1]. Other examples of applications requiring quantification over time are disease-vector mosquito [6] and pollinator surveillance [20] and disease outbreak forecasting [15].

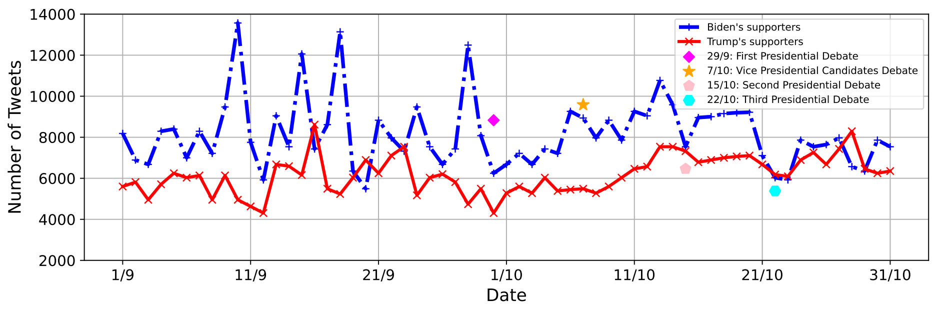

Fig. 1. Prevalence of polarized tweets on different presidential candidates from Sept. 1 to Oct. 31, 2020. Adapted from [1].

This paper introduces *Quantification over Time* (QoT), a quantification task that allies time series forecasting and quantification. In QoT, a series of quantifications give origin to a time series. The objective of QoT is to provide an accurate estimate of the class distribution for an unlabeled set S_t given the data in this set and previous estimates for unlabelled sets $S_1, \ldots, S_{t-1}$.

This paper also proposes KF-MA, which integrates the Kalman filter and moving average, aiming to adjust the result from a standalone quantification for QoT without increasing the requirement for extra labelled data. An algorithm framework is also introduced to conclude how selected time series forecasting methods conduct the adjustment in both KF-MA and state-of-the-art methods. We compare KF-MA with observed methods in literature on several textual and numeral datasets. We demonstrate in our experiments how KF-MA outperforms the state-of-the-art. We evaluate KF-MA in various conditions, including different classifiers and recently proposed quantifiers. The results show that our proposal provides evident improvements in the estimations of standalone quantifications.

This paper is organized into the following sections. Section 2 formalizes the quantification over time task and defines the notation used throughout this paper. Section 3 reviews the relevant literature, including the methods incorporated in our experimental comparison. Section 4 introduces our proposed approach. Section 5 describes the experiment settings and discusses the results. Section 6 provides our concluding remarks and suggests future developments of quantification over time.

2 Definitions, Notation and Background

This section formalizes the task of Quantification over Time (QoT). We start defining the quantification task to highlight the similarities and differences between these two tasks.

2.1 Quantification

Quantification is a supervised machine learning task that predicts the class prevalence (prior probability or relative frequency) in an unlabeled dataset. Like other Machine Learning tasks, quantification requires a training set $D_{train} = \{(\mathbf{x}_i,\ y_i)\}_{i=1}^{N}$ which is a collection with N examples, where each example $\mathbf{x}_i \in \mathcal{X}$ is a vector with M attributes, and $y_i \in \mathcal{L} = \{l_j\}_{j=1}^{L}$ is the class label associated with $\mathbf{x}_i$. The goal of quantification is to learn a function h from D_{train} such that:

$$h : 2^{\mathcal{X}} \rightarrow \Delta^L \tag{1}$$

where $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$, *i.e.*, the set with all possible sets of samples with the representation $\mathcal{X}$. Δ^L is the L-probability simplex defined as:

$$\Delta^L = \{\{p_i\}_{i=1}^{L} | p_i \in [0,1], \sum_{i=1}^{L} p_i = 1\} \tag{2}$$

Given an unlabelled set $S \in 2^{\mathcal{X}}$, h returns an L-dimensional vector $\hat{\mathbf{p}} = [\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_L]^\top$, such that $\sum_{i=1}^{L} \hat{p}_i = 1$. The function h is the quantifier, and $\hat{\mathbf{p}}$ is the estimated class prevalence in set S.

2.2 Quantification over Time (QoT)

QoT requires a dataset with timestamps so instances can be grouped in periods of interest, such as hours, days or weeks. For some time t, there is an unlabelled sample S_t and a set of quantified class prevalences of previous samples, S_1 to S_{t-1}, represented as a vector $\hat{\boldsymbol{\pi}}_{t-1} = [\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots, \hat{\mathbf{p}}_{t-1}]$. Each $\hat{\mathbf{p}}_i$ is a vector with the estimated class prevalences for the set S_i. The goal of QoT is to find a function g such that:

$$g : (\hat{\boldsymbol{\pi}}_{t-1}, S_t) \rightarrow \Delta^L \tag{3}$$

where g returns the class distribution estimate for S_t considering previous estimates in $\hat{\boldsymbol{\pi}}_{t-1}$.

2.3 QoT, quantification and time series forecasting

QoT relates to quantification and time series forecasting in the following ways:

– If g only relies on data from S_t, then $g = h$, *i.e.*, the QoT quantifier will ignore the time dependency of the quantifications.
– If g only uses $\hat{\boldsymbol{\pi}}_{t-1}$, then g as a time series forecasting model with the additional constraint that the estimates must be a valid probability distribution.

Therefore, we can understand QoT as a research task at the intersection of time series forecasting and quantification. However, we must emphasize that existing methods from these two areas are inappropriate solutions for the QoT problems since:

1. Time series forecasting models assume a *prediction horizon*, which is the number of time steps ahead that the models should estimate. QoT problems have an *infinite* prediction horizon. For most QoT applications, we should never expect to receive labels for the unlabelled sets S_t. Suppose we use time series forecasting models for QoT problems by inputting forecasts into the model. In that case, we can expect that the error accumulation will make the model forecasts excessively inaccurate after a long period.
2. Quantification methods ignore the time dependency of the estimates. The incorporation of time information provides the model with an expected rate of change. This work hypothesizes that incorporating historical information with the current quantification provides more accurate estimates.

To conclude this section, we provide additional details about how most QoT applications work in practice regarding label availability. We use the social media sentiment analysis from Section 1 as an example.

For most QoT applications, we should *not* expect to see class labels after the system deployment. In the case of social media, we should not expect humans to label any posts manually during the system's operation. However, we often have a labelled dataset to train and tune the model's hyperparameters. We simulate this condition in our experiments by using the initial samples in the data stream as a training/validation set. This situation differs significantly from the typical operation of time series forecasting models. In regression applications, it is common for the target information to auto-reveal after the forecast horizon has passed. For instance, if we predict stock prices over 24 hours, we will know the actual value of the stocks after this period. This allows us to feed these actual prices to the model when we need to provide a forecast for the next horizon.

The lack of labels after model deployment gives the impression that the QoT task will accumulate errors, providing inaccurate predictions for longer forecast windows, as would happen if we fed time series forecasting methods with estimates instead of actual values. However, QoT inherits a pivotal assumption from quantification: $P_{te}(\mathbf{X}|Y) = P_{tr}(\mathbf{X}|Y)$, *i.e.*, the conditional distribution of the features given the class remains constant from training to test.

This assumption is reasonable for most applications. In the sentiment analysis example, $P(\mathbf{X}|Y)$ captures the relationship between words and sentiments and is mostly constant with small changes in the long term as the language evolves. Under this assumption, the quantifiers can provide relatively accurate prevalence predictions that will not allow the time series forecaster to drift away from reality. At the same time, the time series forecaster will provide temporal dependency to the quantifier, allowing it to improve its predictions.

3 Related Work

Due to the popularity of sentiment analysis for social media and the fact that social media posts are timestamped, plenty of research papers quantify sentiment over time. However, these papers fall into two categories: The first uses the classify and count quantifier without temporal information, and the second uses the same quantifier with a moving average to smooth the predicted prevalences.

Before reviewing the relevant literature, we should understand why the CC quantifier is a suboptimal choice despite its popularity in the sentiment analysis community [18]. Classify and count is a biased quantifier because the classifier assumes that the training and test samples come from the same underlying distribution. However, in quantification problems, the class distribution of the test samples can differ significantly from the test distribution.

Let us suppose that a binary-class classifier is trained with a balanced distribution. In this case, when the positive class prevalence increases, the CC quantifier tends to underestimate the prevalence of the positives. Essentially, the classifier expects the test samples to have the same class distribution as the training samples and thus pushes the class estimates towards a 50%/50% estimate. Similarly, when the prevalence of positives decreases, the same quantifier tends to overestimate its prevalence.

Forman [8,9] provides a formal analysis of the CC quantifier error and shows that it increases linearly as the test class distribution moves away from the training distribution. The classifier error, specifically the difference between the true positive and negative rate, defines the error slope.

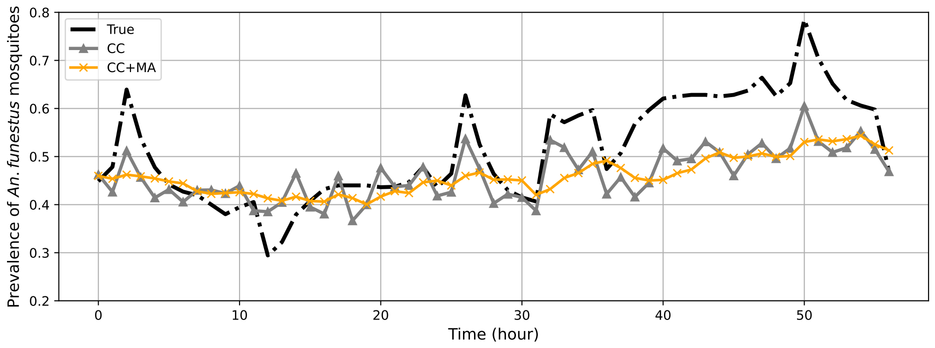

Fig. 2. A comparison of the performance of the classify and count (CC) quantifier with and without moving average (MA). The classifier was trained with a balanced training set with signals from female *Anopheles funestus* and *Culex quinquefasciatus* mosquitoes. CC underestimates high prevalences and overestimates low prevalences. CC+MA performs even worse as the moving average operates as a low-pass filter.

Fig. 2 illustrates this issue in a mosquito surveillance dataset. The data represents the number of *Anopheles funestus* mosquitoes captured by a mosquito trap that uses Machine Learning classifiers to recognize the mosquito species based on data collected from their wing movement. We trained a Machine Learning classifier with a balanced distribution of mosquitoes, which achieved a respectful 80% accuracy. When test class distribution increases, the CC quantifier underestimates the species prevalence. Such underestimation is proportional to the species prevalence and achieves its highest value for the peak at hour 50. The opposite occurs when class prevalence decreases, as we can observe at hour 12.

An even worse behavior occurs when we apply a moving average of window size of 4 hours to the classify and count output. As we will see later in this section, this is a popular approach in the sentiment analysis community. A moving average operates as a low-pass filter, providing a smoother sequence of predictions but worsening the under/over-estimation issue.

In the next two sections, we briefly summarize the relevant literature. Section 3.1 focuses on papers that approach the QoT problem using quantifiers only without incorporating time information. Section 3.2 summarizes the related work that combines the classify and count quantifier with a moving average.

3.1 QoT with quantification only

The literature contains various studies quantifying sentiments on social platforms over time and exploring the relations between public opinions online and real-world events. The vast majority of the relevant papers use the classify and count quantifier. We believe such a poor design decision can only be explained by a lack of understanding of the classification and count quantifier's limitations in the sentiment analysis community.

In what follows, we provide some examples of sentiment analysis papers that use the classify and count quantifier to solve QoT problems. Bollen et al. [2] investigate whether measurements of collective mood states derived from large-scale Twitter feeds correlate to the value of the Dow Jones Industrial Average (DJIA) over time. Borge-Holthoefer et al. [3] track the public opinion dynamics about political events in Egypt on Twitter and analyze the motivation of people switching political polarization. They believe that tracking the relevant change of proportions of the sentiment index instead of absolute values might diminish the quantification bias. Lamsal [15] analyzes tweets about COVID-19, aiming to understand the public opinion patterns related to the ongoing pandemic. Liu et al. [16] also study the aggregate sentiments on Twitter over time based on an algorithm related to CC to predict the presidential election. Notably, the authors claim their trained classifier with an evaluated accuracy of 57%, as it should be predictably biased when deploying classify and count.

Few papers use a quantifier other than classify and count in QoT applications. Hopkins and King [10] analyze the impact of political speech incidents on blog sentiments toward candidates. Aware of the bias of the classify and count quantifier, they propose *ReadMe* to quantify sentiments over time. Similarly, Ceron et al. [4] conducted sentiment analysis on tweets to estimate the distribution flow of citizens' political preferences. They also use ReadMe, applying this quantifier across time while disregarding eventual time dependency among forecasted prevalences.

3.2 QoT with classify and count and moving average

There is abundant literature in sentiment analysis that uses the classify and count quantifier with moving averages. The moving average can be seen as a basic time series analysis method that incorporates time dependency to smooth prevalence forecasts. Overall, the papers summarized in this section employ moving averages to improve the visualization of trends.

O'Connor et al. [19] argue social media sentiment can serve a role similar to that of traditional polling and surveys. The authors acknowledge the impact of misclassification on the estimates generated by the classify and count quantifier. However, they think the occurrence of false positives and negatives would balance each other out when aggregating the sentiment. Wen et al. [25] use a 3-day sliding window alongside the CC quantifier on sentiment polarity analysis to understand students' opinions towards the course and course tools. They also believe that false positives and negatives could potentially offset each other during the counting of classified instances.

Both O'Connor et al. [19] and Wen et al. [25] make the incorrect assumption that the classify and count is potentially an accurate quantifier as the errors nullifying each other. In fact, if a binary classifier makes the same number of false positive and negative errors, the quantification can be flawless despite the imperfection of the classifier [9]. However, as the test class distribution changes, one type of error may prevail. For instance, a classifier may make the equal number of false positive and negative misclassifications in a balanced training set. However, if the prevalence of positive instances increases to 90% in a test sample, then the number of false negative misclassifications will surpass the false positives, causing the classifier to underestimate the positives, as shown in Fig. 2.

In addition, Lai [14] correlates the sentiment trend on Twitter with a traditional presidential performance poll. The author applies a moving average to the survey data, which the paper regards as a gold standard for the sentiment trend. Rani and Kumar [21] focus on sentiment analysis within teaching research. They propose a system that analyzes student feedback sourced from online platforms and course surveys. A method similar to moving average, called a mean emotion vector, is used to smooth the quantified results.

4 Methodology

This section presents our proposed method, Kalman Filter-Moving Average (KF-MA), for QoT problems. KF-MA is part of a more general framework that we name the *adjustment framework*. Such a framework also includes approaches based on the moving average, popularly employed in the sentiment analysis literature. We first introduce the framework in Section 4.1 and then introduce KF-MA in Section 4.2.

4.1 Adjustment Framework

Our framework adjusts class prevalence estimates for a sample S_t collected at a time point t by leveraging existing quantifiers and the temporal dependency of previous prevalence estimates.

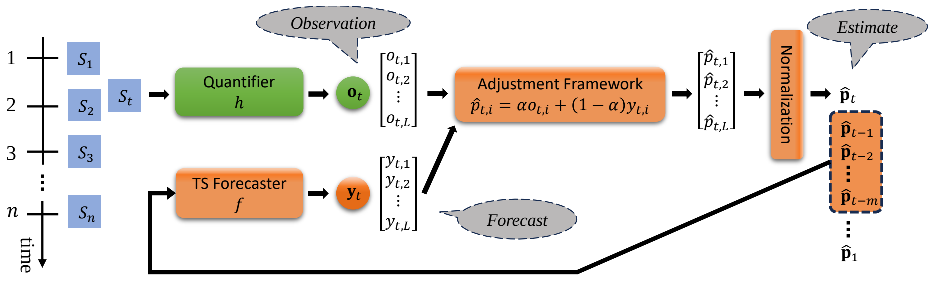

Fig. 3. A general view of the adjustment framework for QoT, which integrates predictions from a quantifier and a time series forecaster, having inputs from last m estimates.

We use the standard nomenclature of the Probabilistic Graphical Model literature and name the output of quantifiers as *observations*. At time point t, we have an observation $\mathbf{o}_t = [o_{t,1}, o_{t,2}, \cdots, o_{t,L}]^\top = h(S_t)$ from a quantifier h, being $o_{t,i}$ the observation corresponding to the estimate of class l_i's true prevalence

$p_{t,i}$. Besides, we have a *forecast* $\mathbf{y}_t = [y_{t,1}, y_{t,2}, \cdots, y_{t,L}]^\top$, where $y_{t,i} = f(\hat{\boldsymbol{\pi}}_{t-1})$ is the prevalence prediction for class l_i We integrate these two predictions into a single value

$$\hat{p}_{t,i} = \alpha\, o_{t,i} + (1-\alpha)\, f(\hat{\boldsymbol{\pi}}_{t-1}) \tag{4}$$

as the estimate of the true prevalence. For time series forecaster f, a model that does not require a large-scale training process is preferred. The reason is that deploying non-trivial time series analysis methods, such as recurrent neuron networks and autoregression models, requires high cost of data, while manually labelling enough data for training those methods is cumbersome in QoT tasks as time series in QoT is different from most that out of streaming data, each data point of time series in QoT is an aggregated mark over a sample instead of an instance.

$\hat{\boldsymbol{\pi}}_{t-1}$ is generallized as a representation of historical information. It can be set to storing either the previous integrated estimates $\hat{\boldsymbol{p}}$ or previous observation $\mathbf{o}$, which can be found in the literature and will be discussed in the following section. In addition, this historical information may encompasses only the last m observations, where m is a hyperparameter that limits how long the method can observe from the past.

The $\hat{p}_{t,i}$ are not necessarily normalized in the sense that $\sum_i^L \hat{p}_{t,i} \neq 1$. We thus normalize these predictions to turn them into a probability distribution. Fig. 3 illustrates this general approach that we named the *adjustment framework.*

In the remainder of this text, we simplify the notation by dropping the i index from symbols such as $p_{t,i}$, $o_{t,i}$, and $y_{t,i}$. The reason is that the methods in the adjustment framework often work with one class at a time. Therefore, for a multi-class problem, these methods make $L-1$ independent predictions integrated in the normalization step.

In the remainder of this section, we discuss how the moving average (MA) approach integrates into our framework. We will use it later as a baseline for comparison with our proposed methods. O'Connor et al. [19] was the first to apply MA to QoT problems. They started with daily sentiment ratios $[o_1, o_2, \ldots, o_n]$ of positive and negative tweets obtained with the Classify and Count method (CC). The authors adjusted each ratio using a m-size window as follows:

$$\begin{aligned} \hat{p}_t &= \frac{1}{m}(o_{t-m+1} + o_{t-m+2} + \ldots + o_t) \\ &= \frac{m-1}{m}\frac{(o_{t-m+1} + o_{t-m+2} + \ldots + o_{t-1})}{m-1} + \frac{1}{m}o_t \end{aligned} \tag{5}$$

Notice that, as they work with binary-class problems, they can track only the positive class prevalence and estimate the negative prevalence, as both need to sum to one.

According to Eq. (4), we have $y_t = f(\mathbf{o}_t^m) = \frac{(o_{t-m+1}+o_{t-m+2}+\ldots+o_{t-1})}{m-1}$ and $\alpha = \frac{1}{m}$. Note that, for MA, with a fixed window size m, the weights of prediction and observation are static, and it primarily presents a retrospective average without effectively capturing trends.

4.2 KF-MA

Our proposal, Kalman Filter-Moving Average (KF-MA), introduces a dynamic weighting approach for parameter α between observations and predictions. KF-MA uses moving average as a time series forecasting function f and Kalman filter [13] to the adjustment framework.

Kalman filter [13] is a recursive algorithm for estimating the state of a dynamic system. It assumes all uncertainties from the environment, observation and hidden states are Gaussian distributed.

When using a Kalman filter to estimate a dynamic system with no known external influence, where the scales of measurement and states are identical, for state s_t, there is an observation $O_t \sim \mathcal{N}(\mu_{o_t}, R)$, where R is from the i.i.d.[1] random error of each measurement. A Kalman filter also models a forecast as $X'_t \sim \mathcal{N}(\mu_{x'_t}, \Sigma_{x'_t})$, such that:

$$\begin{cases} \mu_{x'_t} = F\,\mu_{x_{t-1}} \\ \Sigma_{x'_t} = F\,\Sigma_{x_{t-1}}\,F^\top + Q_t \end{cases} \tag{6}$$

where $X_{t-1} \sim \mathcal{N}(\mu_{x_{t-1}}, \Sigma_{x_{t-1}})$ is an estimate of s_{t-1}, and F is the matrix encapsulating the internal transition mechanism of the dynamic system. The transition covariance Q_t represents the uncertain influence of the external environment.

The filter integrates X'_t with O_t to have an estimate $X_t \sim \mathcal{N}(\mu_{x_t}, \Sigma_{x_t})$ for s_t. It is computed as follows:

$$\begin{cases} \mu_{x_t} = \mu_{x'_t} + K_t\,(\mu_{o_t} - \mu_{x'_t}) \\ \Sigma_{x_t} = (1 - K_t)\,\Sigma_{x'_t} \\ K_t = \frac{\Sigma_{x'_t}}{\Sigma_{x'_t} + R} \end{cases} \tag{7}$$

where K_t is named Kalman gain.

For a later state s_{t+1}, the filter recursively derives a prediction using estimate X_t of s_t. Only the estimate of the initial state should be specified manually since no prior estimates are made. To implement the Kalman filter, specifications for R, F, Q_t, and an initial estimate $X_0 \sim \mathcal{N}(\mu_{x_0}, \Sigma_{x_0})$ are required.

Since the forecast of the current state depends solely on the previous state, it is necessary to reformulate the state to integrate the Kalman filter into the adjustment framework. In KF-MA, for each class at time t, the forecast y_t of true prevalence p_t is modeled as:

$$y_t = f(\hat{\mathbf{p}}_t^m) = \mathbf{a}\,\hat{\mathbf{p}}_t^m \tag{8}$$

where the vector $\mathbf{a} = [a_1, a_2, \cdots, a_m]$, $\hat{\mathbf{p}}_t^m = [\hat{p}_{t-m}, \hat{p}_{t-m+1}, \cdots, \hat{p}_{t-1}]^\top$. For time t, define the state as a $m \times 1$ vector $\mathbf{p}_t^* = [p_{t-m+1}, \cdots, p_{t-1}, p_t]^\top$, hence we have the estimate $X_t \sim \mathcal{N}(\hat{\mathbf{p}}_t^*, P_t)$, in which $\hat{\mathbf{p}}_t^* = [\hat{p}_{t-m+1}, \cdots, \hat{p}_{t-1}, \hat{p}_t]^\top$, P_t is the $m \times m$ covariance matrix and $\hat{\mathbf{p}}_t^m = \hat{\mathbf{p}}_{t-1}^*$. Correspondingly, we have the forecast $X'_t \sim \mathcal{N}(\mathbf{y}_t^*, P'_t)$ and the observation $O_t \sim \mathcal{N}(\mathbf{o}_t^*, R)$. Therefore, we set the $m \times m$ transition matrix F as:

[1] Independent and identically distributed.

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1(m) & a_2(m) & a_3(m) & \cdots & a_m(m) \end{bmatrix} \tag{9}$$

s.t.

$$\mathbf{y}_t^* = F\,\hat{\mathbf{p}}_{t-1}^* \tag{10}$$

Now Eq. (8) is transformed into Eq. (10). KF-MA uses MA for time series forecasting f, hence $a_j(m) = \frac{1}{m}$ for each in F. R is modeled by the mean squared error r of the quantifier through validation multiplied by an $m \times m$ identity matrix I_m as $R = r \cdot I_m$. Since no former state can be used to update the initial state covariance, it is initiated by observations and R. Set the time index starting from 1, the initial state $\mathbf{p}_t^* = [p_1, \cdots, p_{m-1}, p_m]^\top$. The initial estimate is $\mathcal{N}(\mathbf{o}_m^*, R)$. Similar to other time series forecast algorithms, we cannot estimate the initial m class prevalence. Hence, the window size m is expected to be set small, as detailed in Section 5. Assuming a stable outside influence over time, we denote the transition covariance as Q, which is fine-tuned through the validation process. Recalling the Eq. 7 and Eq. (4), we can now obtain the estimate $X_t \sim \mathcal{N}(\hat{\mathbf{p}}_t^*, P_t)$ by

$$\begin{cases} \hat{\mathbf{p}}_t^* = K_t\,\mathbf{o}_t^* + (1 - K_t)\,\mathbf{y}_t^* \\ P_t = (1 - K_t)\,P_t' \\ K_t = \frac{P_t'}{P_t' + R} \end{cases} \tag{11}$$

in which $\alpha = K_t$. For each class at time t, value $\hat{p}_t = \hat{\mathbf{p}}_t^*\,[m]$ is the estimate of its true prevalence. In comparing KF-MA with the moving average, one notable advantage of KF-MA is its dynamic weights for observations and predictions. This advantage stems from the recursive updating of the Kalman gain as the states change over time.

5 Experimental Evaluation

This section outlines the experimental evaluation settings and discusses the results, focusing on how KF-MA improves quantification accuracy and whether it outperforms MA. Additionally, we compare KF-MA to the state-of-the-art QoT methods in the literature.

5.1 Experimental Setup

Implementing a QoT approach involves four main components: a classifier, a quantifier, a time series forecaster and an adjustment framework. Most existing quantifiers produce a class prevalence estimate using the scores generated by a classifier. The time series forecaster and adjustment framework adjust the output of the quantifier, improving its performance in the presence of historical data. To effectively evaluate our method, we created a diverse experiment involving multiple datasets, classifiers, and quantifiers. The objective is to introduce variations that enable a comprehensive comparison and simulate scenarios in which users might apply the method across various applications.

Datasets The requirements for datasets include true or hand-coded labels, timestamp features, and a task that involves counting a time-related topic or entity. We selected seven datasets meeting the criteria, three of which are textual data for sentiment classification:

NpSenti Sentiment analysis of COVID-19-related Tweets in Nepali [24].
AppleSenti User sentiments towards Apple company on Twitter[2].
GlobalSenti A collection of worldwide Tweets related to COVID-19[3].

The rationale for choosing these datasets is they represent applications similar to those discussed in Section 3, specifically in monitoring opinions of particular topics. In addition to textual datasets, we collected four other datasets for various applications:

Mosquito A mosquito surveillance dataset to count mosquitoes by species.
Bike An UCI dataset for bike sharing prediction[4].
Energy An UCI dataset for predicting energy consumption[5].
News An UCI dataset for predicting online news popularity[6].

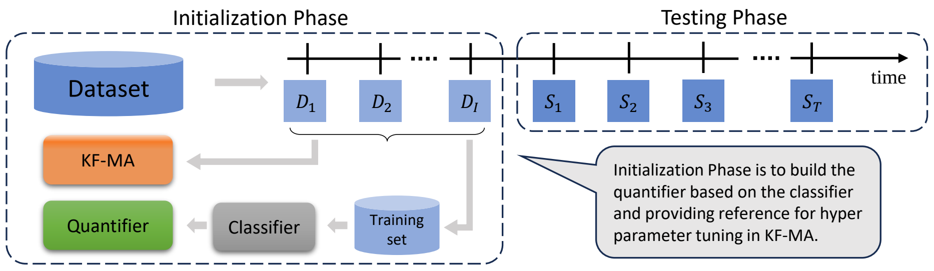
Initialization Phase
Testing Phase
Dataset
D_1
D_2
D_I
S_1
S_2
S_3
S_T
time
KF-MA
Quantifier
Classifier
Training set
Initialization Phase is to build the quantifier based on the classifier and providing reference for hyper parameter tuning in KF-MA.

Fig. 4. Deployment of quantification over time with KF-MA

[2] https://data.world/crowdflower/apple-twitter-sentiment
[3] https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification/data
[4] https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset
[5] https://archive.ics.uci.edu/dataset/374/appliances+energy+prediction
[6] https://archive.ics.uci.edu/dataset/332/online+news+popularity

The datasets are preprocessed to align with our evaluation criteria. For each dataset, instances are assigned to time intervals, such as hours or days, and then grouped into subsamples based on these intervals.

An initialization process is necessary for training classifiers and quantifiers when not using pre-trained ones or searching for KF-MA hyperparameters. This phase uses the first subsamples, as illustrated in Fig. 4. This data serves two purposes:

1. It is used to train classifiers and quantifiers when not using pre-trained models and evaluate their performance.
2. It estimates the random error matrix, R, for the Kalman Filter used in KF-MA. Such a matrix estimation is obtained from the quantifier's mean squared error obtained in the initialization data.

The number of samples in the initialization phase, I, must follow the requirements found in practical applications. Large values of I allow us to train better classifiers and quantifiers and estimate hyperparameters better. However, large training sets are unavailable for many applications due to the high cost of labelling data. In our experiments, we used around 15% of the data for initialization of the non-textual datasets. We used the first 15 time units for the textual datasets, such as hours or days. After setting I, T samples remain for testing. The final models are trained in the concatenation of all $\{D_i\}_{i=1}^{I}$ as a single training set. Table 1 summarises the parameters of our experimental setup for each dataset.

Table 1. Summary of the experimental setup. #Classes is the number of classes. *Tr. Size* is the number of instances in the training set after concatenation. I is the number of time units in the training set. *Avg. Size/t* is the average number of instances per sample in the testing sets. T is the number of time units in the training set.

Dataset	#Classes	Tr. Size	I	Avg. Size/t	T	Time Unit
NpSenti	3	233	15	104	320	day
AppleSenti	3	1927	15	104	18	day
GlobalSenti	3	4454	15	1397	29	day
Mosquito	2	958	24	334	312	hour
Bike	2	2634	55	47	273	hour
Energy	3	2880	20	144	103	day
News	2	6943	36	155	185	day

Classifiers and quantifiers We chose to use two pre-trained sentiment classifiers on the three textual datasets: *VADER* [11] a lexicon-based sentiment analyzer, and AutoNLP from HuggingFace[7], referred as *Solanki*. We used Logistic

[7] https://huggingface.co/amansolanki/autonlp-Tweet-Sentiment-Extraction-20114061

Regression and Random Forest classifiers for the other four datasets with default hyperparameters.

For the quantifiers, we selected *Adjusted Classify and Count* (ACC) [9], *Distribution y-Similarity* (DyS) [17], *Generalized Probabilistic ACC* (GPACC) [7], and *Energy Distance-y* (EDy) [5]. These are popular quantifiers often ranked among the best performing according to recent empirical studies [22].

ACC adjusts the output of CC using the *true positive* (*tpr*) and *false positive rates* (*fpr*) through a validation process on the training set. DyS models the scores provided by a classifier using histograms. The model searches for a parameter that minimizes the distance between a mixture of positive and negative scores from the training set and the unlabelled scores from the test sample. GPACC is a generalization of ACC for multiclass problems with probabilistic classifiers. It uses a soft variation of the confusion matrix obtained from the training set using cross-validation. EDy interprets the dimensional density of the data as the posterior distribution. This approach allows EDy to extract more detailed information than GPACC in feature space.

CC, GPACC, and EDy are naturally capable of quantification on multi-class data, while ACC and DyS are designed for binary data. Therefore, ACC and DyS are deployed using *One versus All* (OVA) for multi-class datasets.

Hyperparameters One hyperparameter is the window size m for the time series forecasting model f. In our experiments, we set $m = 4$, taking into account the following factors:

(*i*) Larger window sizes are wasteful because the recommended model f does not capture the seasonality patterns; (*ii*) Covariance decreases as time points move further apart; and, (*iii*) Minimizing the number of data points that can not be adjusted in the initial phase.

In addition to m, the transition matrix Q in the Kalman filter is tuned during the initialization phase in Fig. 4. Matrix Q is assumed to be a scalar matrix, similar to the observation covariance R. In our experiments, we perform a ternary search on Q within the interval $\left[10^{-4}, 10^{-1}\right]$.

Evaluation Our evaluation uses the *Absolute Error* (AE) due its interpretability [23]:

$$\mathrm{AE}(\mathbf{p} - \hat{\mathbf{p}}) = \frac{1}{L} \sum_{i=1}^{L} |\mathbf{p}(i) - \hat{\mathbf{p}}(i)| \tag{12}$$

where $\mathbf{p}$ is the true and $\hat{\mathbf{p}}$ the predicted class prevalence. We compute each sample's AE and take the mean across all samples, denoted as *Mean Absolute Error* (MAE).

With seven datasets, four quantification methods, and two classifiers for each dataset, we create 56 experimental conditions. Each condition undergoes ten iterations with different seeds, and the results are averaged to ensure the stability and reliability of the findings.

Table 2. MAE results on textual data for different combinations of datasets, classifiers and quantifiers.

Quantifier	DyS		ACC		GPACC		EDy	
Classifier	VADER	Solanki	VADER	Solanki	VADER	Solanki	VADER	Solanki
Dataset	GlobalSenti							
QFY	0.0182	0.0590	0.0095	0.0640	0.0310	0.0730	0.0236	0.0620
MA	0.0300	**0.0465**	0.0280	0.0489	0.0359	0.0637	0.0325	0.0530
KF-MA	**0.0156**	0.0498	**0.0087**	**0.0483**	**0.0280**	**0.0609**	**0.0208**	**0.0502**
Best Method	KF-MA	MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	NpSenti							
QFY	0.1858	0.1712	0.1823	0.1828	0.2075	0.2529	0.1921	0.2575
MA	**0.1350**	**0.1617**	0.1505	**0.1546**	0.1672	**0.2434**	0.1549	**0.2486**
KF-MA	0.1354	0.1644	**0.1473**	0.1609	**0.1640**	0.2442	**0.1527**	0.2499
Best Method	MA	MA	KF-MA	MA	KF-MA	MA	KF-MA	MA
Dataset	AppleSenti							
QFY	0.1706	0.1289	0.1968	0.1187	0.1518	0.1134	0.1538	0.1146
MA	0.1215	0.0999	0.1688	0.0999	0.1334	0.1018	0.1340	0.1032
KF-MA	**0.1128**	**0.0929**	**0.1670**	**0.0935**	**0.1254**	**0.0975**	**0.1269**	**0.0986**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA

5.2 Comparison with QoT approaches in literature

We compare KF-MA to other approaches previously used in related work. From our knowledge, only three methods have been used for QoT: standalone CC, CC with moving average, and standalone ReadMe [10]. ReadMe is a quantification method specifically designed for textual data. It does not require a trained classifier as it operates directly on text features. Jerzak et al. [12] recently proposed an improved version of ReadMe, referred to as *ReadMe2*, which has shown competitive performance in sentiment quantification tasks. We assess ReadMe2 combined with our proposed method KF-MA, comparing it with CC, CC with moving average, and standalone Readme2 on the three textual datasets. Other datasets are not included in this experiment as they do not have the textual data expected by ReadMe2. *Solanki* classifier was used in the CC approach. This experiment evaluates if KF-MA improves upon the state-of-the-art approaches.

5.3 Results

We evaluate three QoT methods across 56 experimental conditions: standalone quantification (QFY), quantification with moving average (MA), and quantification with KF-MA (KF-MA). According to the results presented in Table 2 for textual data and Table 3 for numeral data, our proposed method KF-MA achieved the best performance in most conditions.

Table 3. MAE results on non-textual datasets for different combinations of datasets, classifiers and quantifiers. *LR* represents Logistic Regression classifier and *RF* represents Random Forest classifier.

Quantifier	DyS		ACC		GPACC		EDy	
Classifier	LR	RF	LR	RF	LR	RF	LR	RF
Dataset	Bike							
QFY	0.2113	0.2622	0.1780	0.2443	0.1982	0.2648	0.1827	0.2604
MA	**0.1916**	0.2054	0.1671	0.1886	**0.1858**	0.2164	0.1726	0.2111
KF-MA	0.1918	**0.1971**	**0.1667**	**0.1840**	0.1885	**0.2112**	**0.1725**	**0.2049**
Best Method	MA	KF-MA	KF-MA	KF-MA	MA	KF-MA	KF-MA	KF-MA
Dataset	Energy							
QFY	0.2813	0.2575	0.3294	0.3179	0.3442	0.4378	0.3281	0.4305
MA	0.2285	0.2259	0.3136	0.2531	0.2867	0.3906	0.2769	0.3788
KF-MA	**0.2089**	**0.1858**	**0.3041**	**0.1862**	**0.2611**	**0.3533**	**0.2514**	**0.3297**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	News							
QFY	0.2143	0.2282	0.3253	0.2248	0.2243	0.1981	0.2175	0.2039
MA	0.1694	0.1951	0.2208	0.2177	0.1803	0.1732	0.1735	0.1779
KF-MA	**0.1518**	**0.1772**	**0.0959**	**0.2162**	**0.1597**	**0.1638**	**0.1534**	**0.1680**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA
Dataset	Mosquito							
QFY	0.0399	0.0191	0.0468	0.0223	0.0403	0.0182	0.0404	0.0187
MA	0.0445	0.0411	0.0459	0.0420	0.0440	0.0404	0.0440	0.0406
KF-MA	**0.0327**	**0.0190**	**0.0348**	**0.0219**	**0.0315**	**0.0178**	**0.0316**	**0.0184**
Best Method	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA	KF-MA

MA won 8 out of 56, while QFY did not win any. Standalone quantification methods exhibit varying performances across the seven datasets. However, the MAE results demonstrate that KF-MA consistently improves the quantification accuracy of the standalone quantifiers.

Although not as effective as KF-MA, the moving average can improve the quantification accuracy over the standalone quantifiers in certain cases. However, for datasets such as *GlobalSenti* and *Mosquito*, in which standalone quantifiers underperform, applying a moving average tends to worsen the estimation due to its nature as a low-pass filter. In contrast, KF-MA provides adaptive filtering capabilities, which are particularly beneficial when observations are unbiased.

Table 4 shows that KF-MA improves the performance of the state-of-the-art approach ReadMe2. The quantification results of ReadMe2 with KF-MA on the three textual datasets show a consistent improvement over all methods previously applied in the literature.

Table 4. MAE results of different QoT methods on three textual datasets.

QoT Method	CC	CC+MA	ReadMe2	ReadMe2+KF-MA
NpSenti	0.2436	0.2465	0.1422	**0.1420**
GlobalSenti	0.2439	0.2264	0.0766	**0.0594**
AppleSenti	0.2000	0.1631	0.1182	**0.1148**
Mean	0.2292	0.2120	0.1123	**0.1054**

6 Conclusion

In this paper, we introduced the task of quantification over time (QoT), which is common in various fields, and proposed a method called the Kalman Filter-Moving Average (KF-MA) approach. Additionally, we propose a framework that accommodates both KF-MA and MA approaches. MA is a popular approach in the literature for adjusting the output of standalone quantifiers in QoT.

We evaluated our method through experiments under various combinations of datasets, quantifiers and classifiers, comparing them with state-of-the-art approaches. The results demonstrate that using time dependency enhances the performance of quantifiers in QoT problems. Our work provides practitioners with an accurate tool and offers fundamental guidelines and ideas to researchers interested in developing novel algorithms targeted at quantification over time.

In future work, we intend to develop and integrate more sophisticated approaches for time series forecasting that can learn from small quantities of data. One potential approach is the use of Gaussian Processes. We also intend to use multidimensional time series forecasters, using the dependencies among the class's prevalence to improve our results.

References

1. Belcastro, L., Branda, F., Cantini, R., Marozzo, F., Talia, D., Trunfio, P.: Analyzing voter behavior on social media during the 2020 us presidential election campaign. Social Network Analysis and Mining **12**(1), 83 (2022)
2. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. Journal of computational science **2**(1), 1–8 (2011)
3. Borge-Holthoefer, J., Magdy, W., Darwish, K., Weber, I.: Content and network dynamics behind Egyptian political polarization on Twitter. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. pp. 700–711 (2015)
4. Ceron, A., Curini, L., Iacus, S.M., Porro, G.: Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. New media & society **16**(2), 340–358 (2014)
5. del Coz, J.J.: Unioviedo (team2) at Lequa 2022: comparison of traditional quantifiers and a new method based on energy distance. In: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT (2022)

6. De Nadai, B., Maletzke, A., Corbi, J., Batista, G., Reiskind, M.: The impact of body size on Aedes [stegomyia] aegypti wingbeat frequency: implications for mosquito identification. Medical and Veterinary Entomology **35**(4), 617–624 (2021)
7. Firat, A.: Unified framework for quantification. arXiv preprint arXiv:1606.00868 (2016)
8. Forman, G.: Counting positives accurately despite inaccurate classification. In: European conference on machine learning. pp. 564–575. Springer (2005)
9. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**, 164–206 (2008)
10. Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. American Journal of Political Science **54**(1), 229–247 (2010)
11. Hutto, C., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Proceedings of the international AAAI conference on web and social media. vol. 8, pp. 216–225 (2014)
12. Jerzak, C.T., King, G., Strezhnev, A.: An improved method of automated nonparametric content analysis for social science. Political Analysis **31**(1), 42–58 (2023)
13. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
14. Lai, P.: Extracting strong sentiment trends from Twitter. https://nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf (2010)
15. Lamsal, R.: Design and analysis of a large-scale COVID-19 tweets dataset. applied intelligence **51**, 2790–2804 (2021)
16. Liu, R., Yao, X., Guo, C., Wei, X.: Can we forecast presidential election using Twitter data? an integrative modelling approach. Annals of GIS **27**(1), 43–56 (2021)
17. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: DyS: A framework for mixture models in quantification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4552–4560 (2019)
18. Moreo, A., Sebastiani, F.: Tweet sentiment quantification: An experimental re-evaluation. Plos one **17**(9), e0263449 (2022)
19. O'Connor, B., Balasubramanyan, R., Routledge, B., Smith, N.: From tweets to polls: Linking text sentiment to public opinion time series. In: Proceedings of the international AAAI conference on web and social media. vol. 4, pp. 122–129 (2010)
20. Parmezan, A.R., Souza, V.M., Seth, A., Žliobaitė, I., Batista, G.E.: Hierarchical classification of pollinating flying insects under changing environments. Ecological Informatics **70**, 101751 (2022)
21. Rani, S., Kumar, P.: A sentiment analysis system to improve teaching and learning. Computer **50**(5), 36–43 (2017)
22. Schumacher, T., Strohmaier, M., Lemmerich, F.: A comparative evaluation of quantification methods. arXiv preprint arXiv:2103.03223 (2021)
23. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Information Retrieval Journal **23**(3), 255–288 (2020)
24. Sitaula, C., Basnet, A., Mainali, A., Shahi, T.B., et al.: Deep learning-based methods for sentiment analysis on Nepali COVID-19-related tweets. Computational Intelligence and Neuroscience **2021** (2021)
25. Wen, M., Yang, D., Rose, C.: Sentiment analysis in MOOC discussion forums: What does it tell us? In: Educational data mining 2014. Citeseer (2014)
26. Yaqub, U., Chun, S.A., Atluri, V., Vaidya, J.: Analysis of political discourse on Twitter in the context of the 2016 us presidential elections. Government Information Quarterly **34**(4), 613–626 (2017)