Time-series Forecasting using Markov Models

Filipa Rente*

***Abstract*—Time-series modeling and forecasting has become a popular topic over the last decades since it can be applied to a wide range of practical domains. In this work, we compare parametric stochastic Markov models, such as *hidden (semi-)Markov models* (*HMMs* and *HSMMs*), with artificial neural networks, in particular, *long-short term memory* (*LSTM*), for the task of time-series modeling and forecasting. We propose a context aware model, based on a parametric stochastic Markov model (the HSMM), that is conditioned on external features through a logistic regression approach. We evaluate the prediction accuracy of the different models using a social media dataset from Twitter, about *bitcoin*. The results indicate that the Markov models can achieve a higher prediction accuracy than neural networks in the less training data regime. Moreover, these models are easier to train and less complex to interpret when compared to neural networks. The model interpretability is very sought after nowadays, in the context of automatic decision making based on machine learning. The second contribution regards the model order selection problem in Markov models. We propose a novel approach to select the optimal number of states, as well as the state duration in HSMMs. One of the main contributions is a formal proof of equivalence between models that allows us to focus only on the selection of the number of states. We propose a unique order selection criterion based on that proof. Furthermore, the optimal number of states is found through a sequential pruning strategy using a so-called *mixture minimum description length* (*MMDL*) criterion. We demonstrate the effectiveness of the approach using synthetic experiments.**

***Index Terms*—Time-series forecasting, Markov models, Logistic Regression, Artificial neural networks, Model selection.**

I. Introduction

Many real phenomena produce data that can be represented as a *time-series*, i.e., a list of the time instants of occurring events. Earthquakes, road accidents, child births and cancer diagnosis are some examples of such data. Common to these examples is that the number and exact time instant of future events is not known. Usually complex mechanisms are behind these seemingly random times, for example cancer diagnosis can cause new cancer diagnosis in the form of metastases.

Time-series prediction models allow us to predict where in the future the next events are going to happen with a certain level of confidence. Examples of areas where time-series forecasting has been applied include medicine [1], finance [2], weather [3], engineering [4] and computer vision [5], to name a few.

The time-series prediction models should be able to identify trend and seasonality patterns in the data and properly reflect those patterns in the predictions. Building such models is not trivial because some assume certain characteristics about the data that may not be verified, e.g., abrupt or smooth signal dynamics, while others do not account for data inter-dependencies, e.g., cancer metastases.

In this work, we use parametric stochastic Markov models, such as *hidden (semi-)Markov models* (*HMMs* and *HSMMs*), detailed in the following section.

II. Background and related work

We are interested in modeling time-series data - a collection of n events that occur at random times -, in order to forecast future events of the time-series. In this work, we assume that the events are generated by a *non-homogeneous Poisson process*, which is fully specified by its *arrival rate* $\lambda(t)$, also called *intensity function*. The arrival rate defines the expected number of event arrivals in a bounded interval as a Poisson distribution. That is, the probability that the number of event arrivals in any time interval of length T equals $n_e \in \mathbb{N}^0$ is given by

$$\mathbb{P}[N\big([t,t+T)\big) = n_e] \quad = \frac{(\int_t^{t+T} \lambda(\tau)d\tau)^{n_e} e^{-\int_t^{t+T} \lambda(\tau)d\tau}}{n_e!}, \quad (1)$$

where $N(t)$ is the counting process of the number of events. By following this assumption, in order to model the time-series data, we need to estimate the underlying intensity function $\lambda(t)$ of the non-homogeneous Poisson process. A common approach is to assume a specific parametric form for $\lambda(t)$ and estimate it though maximum likelihood using (1). Yet, it is more interesting to explore *Cox processes* [6], where the unobserved intensity function $\lambda(t)$ is a realization of a non-negative stochastic process Λ, given by a random field. We proceed to describe *Markov modulated Poisson processes*, a type of Cox process extensively used to model time-series data.

A. Markov modulated Poisson process

A *Markov modulated Poisson process* (MMPP) [7] models the intensity function $\lambda(t)$ as a piece-wise constant function, with a finite set of intensity levels, through a *Markov chain* (MC) with k states. Each state $s_i \in \mathcal{S} = \{s_1, ..., s_k\}$ is associated with a homogeneous Poisson process of constant intensity $\lambda_{s_i} \equiv \lambda_i \geq 0$. The intensity level of the MMPP at time t is given by the parameter λ_i of the Poisson process that is active, according to the state (s_i) of the MC at time t. The MC is fully specified by the set of parameters $\mu = \{\boldsymbol{\pi}, \boldsymbol{A}\}$, where $\boldsymbol{\pi} \in \mathbb{R}^k$ is the initial state probability distribution and $\boldsymbol{A} \in \mathbb{R}^{k \times k}$ represents the state transition probability matrix. The MMPP is fully specified by the parameters of the MC and by the set of Poisson parameters $\boldsymbol{\lambda} = \{\lambda_1, ..., \lambda_k\}$, one per state. That is, $\mu_{\text{MMPP}} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\lambda}\}$. We have considered two types of Markov chains for the MMPP: the standard Markov chain and a *semi-Markov chain* (SMC). A MMPP with a standard Markov chain is a *hidden Markov model*, whereas a MMPP with a semi-Markov chain is a *hidden semi-Markov model*, both with constant intensity levels described by Poisson processes. In the next sections, we define both models and their estimation procedures.

B. Hidden Markov models with Poisson emissions

A hidden Markov model (HMM) is a statistical model used to represent sequential data with *hidden* states, using the accessible information generated by them, i.e., the observations.[1]

The HMM representation covers the dependency between observations and states in two stochastic processes, namely, a hidden (unobserved) process of states represented by a first-order Markov chain[2] and a visible process of observations.

HMMs are widely used in different areas, including speech recognition [8] and medical image processing [9]. Applied to time-series forecasting, HMMs can be used, for example, to predict stock market price trends [10].

Formally, a HMM with n observations and k states is represented by the tuple $(S, O, \boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{\lambda})$, where $S = (S_1, ..., S_n) \equiv S_{1:n}$ is the hidden state sequence and $O = (O_1, ..., O_n) \equiv O_{1:n}$ is the observation sequence. A HMM with Poisson emissions has the following set of parameters: $\mu = \{\boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{\lambda}\}$, which completely specify it. At each time $t \in [1, n]$, the state $S_t = s_i \in \mathcal{S}$, where the Markov chain is at, dictates the HMM emission probability

$$b_i(t) = \mathbb{P}[O_t = o_t | S_t = s_i] = \frac{e^{-\lambda_i} {\lambda_i}^{o_t}}{o_t!}, \quad (2)$$

* Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
filipa.rente@tecnico.ulisboa.pt

[1] Observations are also called emissions or symbols.

[2] We consider discrete first-order Markov chains. However, the HMM can also be defined by continuous time Markov chains.

from where the observation $O_t = o_t$ is obtained. The joint probability that the HMM will generate the observation sequence $O_{1:n}$ with underlying state sequence $S_{1:n}$ is given by

$$\begin{aligned}\mathbb{P}(S, O; \mu) &\underset{Bayes}{=} \mathbb{P}[O|S, \mu]\mathbb{P}(S; \mu) \\ &= \pi_{S_1} b_{S_1}(o_1) \prod_{t=2}^{n} \boldsymbol{A}_{S_{t-1} \to S_t} b_{S_t}(o_t),\end{aligned} \tag{3}$$

which is called the *joint likelihood* of the states with the observations. Since these models are parametric, the training phase consists in estimating the parameters μ. The parameters are usually estimated through maximum likelihood estimation, i.e., $\hat{\mu} = \arg\max \mathbb{P}(O; \mu)$, where the likelihood is obtained by marginalizing out all possible hidden state assignments in the joint likelihood of (3). However, two problems arise: **1)** there is a double-dependency between the states and the observations, and **2)** the complexity of computing the likelihood is $\mathcal{O}(k^n)$, which can easily become unfeasible when k and n are reasonably large. The latter is solved by using the *forward-backward* (FB) algorithm, which takes advantage of dynamic programming to reduce the complexity to $\mathcal{O}(k^2 n)$ [11]. The former requires using an iterative method to obtain the parameters such as the well-known *expectation-maximization* (EM) algorithm, which comprises two steps: the E-step and the M-step. In the E-step, the expectations of the parameters and states given the observations are computed using the FB algorithm. In the M-step, the parameters are updated according to those expectations. The E-step and the M-step for the HMM can be found in [12]. We will skip the derivations and present solely the final formulas for the parameter estimates:

$$\begin{aligned}\hat{A}_{ij} = \frac{\sum_{t=2}^{n} \zeta_t(i,j)}{\sum_{j=1}^{k}\sum_{t=2}^{n} \zeta_t(i,j)}, \qquad \hat{\pi}_i = \frac{\gamma_1(i)}{\sum_{i=1}^{k} \gamma_1(i)} \\ \hat{\lambda}_i = \frac{\sum_{t=1}^{n} \gamma_t(i) o_t}{\sum_{t=1}^{n} \gamma_t(i)},\end{aligned} \tag{4}$$

where $\gamma_t(i) = \mathbb{P}[S_t = s_i|O, \mu]$ is the posterior distribution of state s_i and $\zeta_t(i,j)$ is the expected transition probability from state s_i to s_j, both obtained in the E-step.

One of the main disadvantages of HMMs is that it is not possible to directly change the duration[3] of the system in a particular state. In fact, the duration in a k-state HMM is implicitly modeled by a geometric distribution with fixed parameter $A_{ii}, \forall i \in [k]$, which corresponds to the diagonal entries of the transition probability matrix $\boldsymbol{A} \in \mathbb{R}^{k \times k}$ [12]. This can be a limitation for real data. Instead, it is advantageous to explicitly specify a state duration distribution as it provides greater expressiveness to the states.

Hidden semi-Markov models, detailed in the next section, are an extension of HMMs designed to achieve this goal.

C. Hidden semi-Markov models with Poisson emissions

In *Hidden semi-Markov models* (HSMMs), the state duration is modeled explicitly through non-parametric (e.g. multinomial distribution) or parametric distributions (e.g. geometric distribution). HSMMs have been vastly used for speech synthesis [13] to model the phoneme duration so that the required naturalness of speech is achieved.

In HSMMs, the state sequence follows a semi-Markov chain (SMC), where each state has a variable duration taking values from the set $\mathcal{D} = \{1, 2, ..., d_{max}\}$ of possible durations. The duration corresponds to the number of observations produced while in a state. A SMC with k states is a system defined by a hidden random sequence $S_{1:n}$ and fully characterized by the set of parameters $\mu = \{\boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{D}\}$, defined below. Regarding the state durations, HSMMs define a state duration probability matrix $\boldsymbol{D} \in \mathbb{R}^{k \times d_{max}}$, where element $D_{jd} = p_j(d)$ represents the probability of occupying state s_j with duration d. In this work, we characterize the durations in the SMC through geometric distributions. Hence, the PMF of the durations for state s_i is given by

$$p_i(d) = \text{Geom}(d|\theta_i) = (1-\theta_i)^{d-1}\theta_i, \qquad \forall d \in \mathcal{D}, \tag{5}$$

where $\theta_i \in]0, 1]$ is the parameter of the geometric distribution of state s_i. In this work, we use the simplest version of HSMM - the explicit duration HMM (EDHMM) -, which assumes that a state transition is not dependent on the duration of the previous state. That is, the probability of a transition from state s_i with duration d' to state s_j with duration d is given by [14]

$$A_{id',jd} = A_{i,jd} = \mathbb{P}[S_{t:t+d-1} = s_j | S_{\to t|} = s_i] = A_{ij} p_j(d), \tag{6}$$

where the notation $S_{\to t|} = s_i$ means that state s_i ends at time t.[4] Note that the diagonal entries of the transition matrix are zero ($A_{ii} = 0$) since no self-transitions are allowed due to the explicit duration modeling. The initial state probability distribution is $\boldsymbol{\pi} \in \mathbb{R}^k$, with elements $\pi_i = \mathbb{P}[S_{|1\to} = s_i], \quad \forall i \in [k]$, where the notation $S_{|1\to} = s_i$ means that state s_i starts at $t = 1$. Moreover, in EDHMMs, it is considered that the observations are conditionally independent. That is,

$$\mathbb{P}[o_{t:t+d-1}|S_{t:t+d-1} = s_i] \underset{ind.}{=} \prod_{\tau=t}^{\tau=t+d-1} b_i(\tau), \tag{7}$$

where $b_i(\tau)$ is given by (2). The parameter estimation problem in EDHMMs, similarly to HMMs, exploits the EM algorithm. We proceed to define the E-step and the M-step.

D. E-step

Again, the E-step corresponds to the FB algorithm. The HSMM forward variable is defined as the joint probability that state s_j ends at t and the observations up to time t, i.e., [5]

$$\alpha_t(j) = \mathbb{P}[O_{1:t}, S_{\to t|} = s_j; \mu]. \tag{8}$$

Equation (8) can be re-written (proof can be found in [12]), within the EDHMM formulation as

$$\begin{aligned}\alpha_t(j) &= \sum_{i=1}^{k} \sum_{d=1}^{\min(d_{max}, t)} \alpha_{t-d}(i) A_{ij} D_{jd} \prod_{\tau=t-d+1}^{t} b_j(\tau) \\ &= \sum_{d=1}^{\min(d_{max}, t)} \alpha^*_{t-d}(j) D_{jd} \prod_{\tau=t-d+1}^{t} b_j(\tau),\end{aligned} \tag{9}$$

where $\alpha^*_t(j) \in \mathbb{R}^k$ is an auxiliary variable that represents the joint probability of the observations up to time t and that state s_j starts at $t+1$. That is,

$$\alpha^*_t(j) = \mathbb{P}[O_{1:t}, S_{|t+1\to} = s_j] = \sum_{i \neq j} \alpha_t(i) A_{ij}, \tag{10}$$

i.e., first we compute the forward variables for each possible duration $d \in \mathcal{D}$ in each state s_j at time step t in (9) and then aggregate the statistics in by weighting them w.r.t. the transition probabilities in (10). The auxiliary variable $\alpha^*_t(j)$ is initialized as follows

$$\alpha^*_t(j) = \begin{cases} \pi_j, & t = 0 \\ 0, & t < 0 \end{cases} \qquad \forall j \in [k]. \tag{11}$$

The backward variable for HSMMs is defined as the conditional probability of future observations from time $t+1$ up to the final instant n given that state s_j ends at t, i.e.,

$$\beta_t(j) = \mathbb{P}[O_{t+1:n}|S_{\to t|} = s_j]. \tag{12}$$

Likewise, (12) can be re-written within the EDHMM formulation as

$$\beta_t(j) = \sum_{i=1}^{k} A_{ji} \beta^*_t(i), \tag{13}$$

where $\beta^*_t(i)$ is an auxiliary variable that represents the conditional probability of future observations from time $t+1$ up to the final instant n given that state s_i starts at $t+1$. That is,

$$\begin{aligned}\beta^*_t(i) &= \mathbb{P}[O_{t+1:n}|S_{|t+1\to} = s_i] \\ &= \sum_{d=1}^{\min(d_{max}, t)} \beta_{t+d}(i) D_{id} \prod_{\tau=t+1}^{t+d} b_i(\tau),\end{aligned} \tag{14}$$

[3]The duration is the number of observations made while in a state.

[4]In HSMMs, the notation $S_t = s_i$ is not enough due to the incorporation of explicit durations.

[5]For ease of notation, we lose the dependency on the model parameters μ.

The backward procedure (from $t = n, ..., 1$) relies on the initialization of the backward variable as

$$\beta_t(j) = \begin{cases} 1, & t = n \\ 0, & t > n \end{cases} \qquad \forall j \in [k]. \tag{15}$$

E. M-step

In order to reestimate the HSMM parameters, the forward and backward variables from eqs. (8) and (12) are used. The unnormalized formulas for the estimates of the initial state, transition, emission and duration distributions are [12]

$$\begin{aligned} \hat{\pi}_i &= \beta_0^*(i)\pi_i \\ \hat{A}_{ij} &= \sum_{t=1}^{n} \alpha_t(i) A_{ij} \beta_t^*(j) \\ \hat{\lambda}_i &= \sum_{t=1}^{n} o_t \Big(\sum_{\tau<t} \alpha_\tau^*(i)\beta_\tau^*(i) - \sum_{\tau<t} \alpha_\tau(i)\beta_\tau(i) \Big) \\ \hat{D}_{jd} &= \sum_{t=1}^{n} \alpha_t^*(j) D_{jd} \beta_{t+d}(j) \prod_{\tau=t+1}^{t+d} b_j(\tau). \end{aligned} \tag{16}$$

From inspection of (9) and (14), it is clear that numerical underflow will occur as $n \to \infty$ since we are working with increasingly large products of probabilities. In this work, we use the implementation from [15], which avoids numerical underflow by proposing a new, efficient and practical implementation of the forward–backward algorithm for an EDHMM. The new FB algorithm redefines the forward and backward variables of (8) and (12) in terms of posterior probabilities, conditioned on the available observations. For example, the forward variable becomes [15]

$$\alpha_{t|x}(i, d) = \mathbb{P}\left[S_t = s_i, \tau_t = d \mid o_{1:x}\right], \tag{17}$$

where $x = t-1, t$ or n, resulting in the predicted, filtered or smoothed forward probabilities, respectively, and τ_t denotes the remaining sojourn time in the current state S_t. The trick behind this re-definition is that by using conditional probabilities normalized at each step, we work with relative changes, which give higher resolution than absolute ones. Thus, numerical underflow is less probable to occur in practice.

In this work, we focus on parametric Markov models (in particular, HMMs and HSMMs) with Poisson emissions and geometric duration distributions, for their simple estimation procedures, and we compare them with *Artificial neural networks* (ANNs). ANNs are yet another type of parametric model that can be used for time-series modeling. In particular, we use the *Long short term memory* (LSTM) [16], a type of ANN efficiently used to model sequential data, that can both capture long and short term dependencies within the time-series data. After training, these models can be used for time-series forecasting. The accuracy in time-series forecasting depends fully on the trained model, thus, it is important to learn the best parameters so that the patterns of the time-series data are well captured and less uncertainty is reflected in future predictions. In parametric Markov models, the hyperparameters - number of states k and maximum duration d_{max} - are unknown for practical applications and, therefore, must be estimated before training. The prediction accuracy of the model is very dependent on the hyperparameter selection problem, since they tune the granularity/smoothness of the predictions. We proceed to detail the hyperparameter strategy used in this work, in particular, for HSMMs.

III. Hyperparameter selection in a HSMM

The order estimation problem in HSMMs consists of estimating the optimal number of states (k^*) and maximum allowed state duration ($d_{max}{}^*$). Hence, the order estimation procedure is a multivariate optimization problem where we optimize both k and d_{max} w.r.t. the same criterion. We use an equivalence of models (proof omitted due to space constraints), which states that, for example, a EDHMM model with two states and a mixture of two geometric state duration distributions per state is equivalent to another model with 4 (augmented) states but only one geometric state duration distribution per state. Consequently, each state has one and only one parameter to specify its duration.[6] This strategy allow us to simplify the order estimation problem to a univariate optimization problem, where the focus is only on the selection of k^*. Using this equivalent model, d_{max} can be empirically estimated through the parameters of the geometric distributions. In fact, there is a correspondence between the parameter θ_i of the geometric distribution (of state s_i in the equivalent model with augmented states) and the durations d: a lower parameter is associated with higher durations. Therefore, d_{max} can be estimated through the following steps: **1)** find the lowest parameter between all (augmented) states ($\theta_i^* = \min(\{\theta_i\}_{i=1}^k)$; **2)** find the duration for which the geometric cumulative probability (easily derived from the PMF of (5)), with parameter found in **1)**, is smaller than $\epsilon \approx 0.001$. That is,

$$\begin{aligned} 1 - \mathbb{P}_i(D \leq d_{max}) &= \epsilon \\ \iff d_{max} &= \lfloor log_{(1-\theta_i^*)}(\epsilon) \rfloor, \end{aligned} \tag{18}$$

where D is the random variable of durations. After determining the hyper-parameter d_{max}, the state duration probability matrix $\boldsymbol{D} \in \mathbb{R}^{k \times d_{max}}$ (used in the EM algorithm) can be filled with values from the geometric PMF in (5) evaluated at the corresponding durations $d = 1, ..., d_{max}$.

In this work, we obtain k^* through heuristic order estimation strategies such as the well-known *Akaike information criterion* (AIC) [17], given by

$$\text{AIC}(k) = \log \mathbb{P}[O; \hat{\mu}_k] - \frac{N_k}{2}, \tag{19}$$

and the *Bayesian information criterion* (BIC) [18], given by

$$\text{BIC}(k) = \log \mathbb{P}[O; \hat{\mu}_k] - \frac{N_k}{2} \log(n). \tag{20}$$

In (19) and (20), the first term in the criterion is the model log-likelihood, whereas the second term penalizes the model complexity through the total number of free parameters (N_k), weighted by the total number of observations (n) in the case of the BIC criterion. Furthermore, we adapt the *mixture minimum description length* (MMDL) [19] criterion to HSMMs. The MMDL criterion was first proposed for mixture models, where the model complexity penalty term does not need to consider the whole data ($log(n)$), given that each component of the mixture is not estimated from all observations but only from those that were, in fact, generated by that component. Hence, the term $log(n)$ is replaced by a quantity that measures how much data was generated by a given component. For mixture models, this quantity is easily obtained as $log(nc_j)$, where c_j is the probability of the j^{th} mixture component. In [20], the authors adapted the MMDL criterion to HMMs. We follow the same reasoning to adapt it to HSMMs. The quantity that specifies the state probabilities[7] is given by the stationary probability distribution $\boldsymbol{p}_\infty = [p_\infty(1), ..., p_\infty(k)]$, which represents the "average" occupation of each (steady) state after the Markov chain has reached its equilibrium. The MMDL criterion is therefore given by

$$\text{MMDL}(k) = \mathbb{P}[O; \hat{\mu}_k] - \frac{N_k}{2} \log(n) - \frac{N_1}{2} \sum_{i=1}^{k} \log(np_\infty(i)), \tag{21}$$

where N_1 is the number of free parameters of a HSMM with just one state ($k = 1$). The second quantity is weighted by $\sum_{i=1}^{k} \log(np_\infty(i))/2$, where $np_\infty(i)$ is designated the "effective sampling size" of state s_i, which measures the percentage of the total data n presumably generated by steady state s_i. Since the EDHMM considers that the duration depends only on the corresponding state, $\boldsymbol{D}$ is actually estimated from the observations that were presumably generated by that state, and not from all the observed data. Therefore, the number of free parameters of $\boldsymbol{D}$ will appear in the second quantity of (21), together with the number of free parameters of $\boldsymbol{\lambda}$. The number of free parameters for $\boldsymbol{A}$ and $\boldsymbol{\pi}$ are $N_k^{\boldsymbol{A}} = k(k-2)$ and $N_k^{\boldsymbol{\pi}} = k-1$. In what the emissions and durations are concerned, since we use parametric distributions, then there is only one free parameter in a HSMM with just one state, i.e., $N_1^{\boldsymbol{\lambda}} = N_1^{\boldsymbol{D}} = 1$. Replacing the free parameters in (21) and dropping the terms that do not depend on k, yields

$$\text{MMDL}_{\text{HSMM}}(k) = \text{LL} - \frac{k^2 - k}{2} \log(n) - \sum_{i=1}^{k} log(np_\infty(i)). \tag{22}$$

[6]The parameter of the geometric distribution.

[7]The states in HSMMs are analogous to the components of the mixture in mixture models and the state probabilities correspond to the component probabilities (c_j) in mixture models.

The computation of the stationary distribution $\boldsymbol{p}_\infty$ in (22) for HSMMs is detailed in the next section.

A. HSMM stationary probability distribution

The stationary probability distribution for semi-Markov chains is not the same as for regular Markov chains. However, it can be found empirically. In [21], the authors proposed the following estimator

$$p_\infty(i) = \frac{\hat{v_i}(n)\hat{m}_i(n)}{\sum_{i=1}^{k} \hat{v_i}(n)\hat{m}_i(n)}, \quad \hat{v_i}(n) = \frac{N_i(n)}{N(n)}, \quad \forall i \in [k], \tag{23}$$

where $\hat{v_i}(n)$ is an estimator for the stationary probability distribution of the embedded[8] Markov chain. It is obtained, as shown in (23), as the ratio of the number of visits of the embedded Markov chain to state s_i up to time $t = n$ ($N_i(n)$) and number of jumps of the embedded Markov chain in the time interval $(0, n]$ ($N(n)$), both available from the estimated state sequence after running the *Viterbi* algorithm.
In (23), $\hat{m}_i(n)$ is an estimator of the mean sojourn time in state s_i, considering all observations up to time n. In [21], they define it as

$$\hat{m}_i(n) = \frac{\sum_{l=1}^{N_i(n)} d_{il}}{N_i(n)}, \qquad \forall i \in [k], \tag{24}$$

where d_{il} is the duration of the l^{th} visit of the embedded Markov chain to state s_i.
In practice, $\hat{m}_i(n)$ is obtained by averaging the EM estimate for the state duration probability matrix ($\hat{\boldsymbol{D}}$), weighted by the corresponding durations. For the sake of interpretation, we will replace (24) in (23), which yields the final estimator for the stationary probability distribution of a semi-Markov chain

$$p_\infty(i) = \frac{\sum_{l=1}^{N_i(n)} d_{il}}{\sum_{i=1}^{k}\sum_{l=1}^{N_i(n)} d_{il}} \approx \frac{\sum_{l=1}^{N_i(n)} d_{il}}{n}. \tag{25}$$

This represents the ratio between the total time spent in state s_i and the length of the observation sequence. Thus, as desired, the stationary probability distribution for semi-Markov chains ($\boldsymbol{p}_\infty$), given by (23), is a measure of the "average" state occupation. Moreover, the approximation in (25) is valid for a small maximum state duration d_{max}, where it is reasonable to assume that the truncation of the semi-Markov chain at time $t = n$ corresponds to the exact time of a state transition.

After computing $\boldsymbol{p}_\infty$, the MMDL criterion of (22) is fully defined. However, in order to obtain the optimal number of states k^*, a greedy approach of testing multiple values for k and selecting the one with the highest criterion value is not desirable. On the one hand, the greedy approach is highly dependent on the initialization, which requires training the model, with the same k, multiple times to obtain a reliable result for the log-likelihood, the parameter estimates and, consequently, the chosen criterion. On the other hand, it is a slow and inefficient approach. In the next section, we describe a strategy that counters these constraints.

B. Sequential pruning strategy

In [20], the authors propose a sequential pruning strategy for the selection of the number of states in HMMs, which can also be used for HSMMs. The sequential pruning strategy entails the following steps:

1) Set the minimum and maximum allowed number of states: k_{min} and k_{max}, respectively;
2) Initialize the EM algorithm parameters randomly, using $k = k_{max}$ as the initial number of states;
3) While $k \geq k_{min}$: Run the EM algorithm until convergence and obtain the estimated parameters; Compute and store the value of the model selection criterion given by (22); Find the least probable state (i.e., the smallest element of $\boldsymbol{p}_\infty$); Prune it by removing the corresponding rows/columns from the estimated parameters; Normalize the estimated parameters that after pruning need normalization[9]; Set these as the initial parameters of the EM algorithm and set $k = k - 1$. Repeat, i.e., run EM again with one less state.
4) The final model (with parameters $\mu^*_{k^*}$) is the one which corresponds to the maximum stored value for the model selection criterion, with the attached optimal number of states k_*.

Using this strategy combined with some criterion (we consider AIC, BIC and MMDL criteria), the optimal number of states is readily obtained. Subsequently, the trained model (with the optimal number of states found) is used to forecast future values of the time-series, as detailed in the next section.

IV. Time-series forecasting in Markov models

To predict future values of a time-series modeled by Markov models, we consider one-step ahead prediction. One-step-ahead prediction corresponds to predicting one observation at a time, using all past observations up to that instant. One step-ahead prediction in HMMs is well-known and can be found in [22]. We proceed to detail the equations for one-step ahead prediction for HSMMs (with Poisson emissions) only, considering the efficient forward-backward algorithm implementation from [15].

A. One-step ahead prediction in HSMMs

In [15], the probability for the one step prediction of observation o_t is given by

$$\mathbb{P}(o_t|o_{1:t-1}) = \sum_{i,d} \alpha_{t|t-1}(i,d) b_i(t) = \sum_{i=1}^{k} \gamma_{t|t-1}(i) b_i(t), \tag{26}$$

where $\alpha_{t|t-1}(i,d)$ is the forward variable from (17) and $b_i(t)$ is the Poisson PMF from (2). In the RHS of (26), we marginalized out the durations to obtain the posterior of state s_i at time t, i.e., $\gamma_{t|t-1}(i) = \mathbb{P}[S_t = s_i|o_{1:t-1}]$. In order to predict the observation at time $t+1$, we need to obtain $\gamma_{t+1|t}$ from $\gamma_{t|t-1}$. The forward variable (defined in (17)) is computed using the following recursion [15]

$$\begin{aligned}\alpha_{t|t-1}(i,d) = &\left(\sum_j \alpha_{t-1|t-2}(j,1) b^*_j(t-1)\ \hat{A}_{ji}\right)\hat{D}_{id} \\ &+ b^*_i(t-1)\,\alpha_{t-1|t-2}(i,d+1),\end{aligned} \tag{27}$$

where $b^*_i(t)$ is the "filtered" emission probability,[10] given by [15]

$$b^*_i(t) = \frac{\hat{b}_i(t)}{\mathbb{P}(o_t|o_{1:t-1})} = \frac{\hat{b}_i(t)}{\sum_{j,d}\alpha_{t|t-1}(j,d)\hat{b}_j(t)}. \tag{28}$$

Note that all parameter estimates in (27) and (28) ($\hat{A}_{ij}, \hat{D}_{id}, \hat{b}_i(t)$) are obtained in the M-step of the EM algorithm after convergence. The model is trained using the observations from $t \in [1, T]$. Then, we make predictions about the observations using the test set from $t \in [T+1, n]$, according to Algorithm 1.

a) Estimation of parametric state duration distributions: In (16), we considered multinomial distributions for the durations. However, in practice, we used geometric state duration distributions, with PMF given by $p_i(d)$ in (5) for state s_i with parameter $\theta_i \in]0, 1]$. The M-step update of θ_i is [23]

$$\hat{\theta}_i = \frac{\sum_{t=2}^{n}\gamma_{t|n}(i)}{\sum_{t=2}^{n}\sum_{d=1}^{d_{\max}} d\mathcal{G}_{t|n}(i,d)}, \tag{31}$$

where $\mathcal{G}_{t|n}(i,d)$ is given by

$$\begin{aligned}\mathcal{G}_{t|n}(i,d) &= \mathbb{P}[S_t = s_i, \tau_{t-1} = 1, \tau_t = d|o_{1:n}] \\ &= \sum_j \alpha_{t-1|t-2}(j,1) b^*_j(t-1)\hat{A}_{ji}\hat{D}_{id}\beta_t(i,d),\end{aligned} \tag{32}$$

where $\beta_t(i,d)$ is the backward variable given by $\dfrac{\mathbb{P}[S_t = s_i, \tau_t = d|o_{1:n}]}{\mathbb{P}[S_t = s_i, \tau_t = d|o_{1:t-1}]}$ and computed according to

[8]The embedded Markov chain of a semi-Markov chain with state sequence $\{4,4,1,1,1,2,3,3,3\}$ is $\{4,1,2,3\}$ with the corresponding jump times $\{2,5,6\}$.

[9]Set distributions to uniform if normalization is not possible (this problem can arise when considering a large amount of states).

[10]Which measures how well the model fits the observations.

Algorithm 1 One-step-ahead prediction algorithm for HSMMs

1: Initialize $t \leftarrow T+1$
2: Collect the forward variable $\alpha_{t-1|t-2}(i,d)$ from the end of training ($\alpha_{T|T-1}(i,d)$)
3: **while** $t < n$ **do**
4: **Conditioning**
5: Conditioned on the previous (true) observation o_{t-1}, compute: $b_i^*(t-1) \leftarrow$ (28)
6: **Filtering**
7: Compute the next forward variable using the estimated parameters from the EM algorithm ($\hat{\mu} = \{\hat{\boldsymbol{A}}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{D}}\}$): $\alpha_{t|t-1}(i,d) \leftarrow$ (27).
8: Compute the state marginal probability ($\gamma_{t|t-1}(i)$) using the forward variable from the previous step
9: **Prediction**
10: Predict the current observation, $\hat{o}_t$. If we consider a MAP prediction, then

$$\hat{o}_t \sim \mathcal{P}\left(\hat{\lambda}_{\arg\max_i(\gamma_{t|t-1}(i))}\right). \tag{29}$$

Another way to predict is to consider not only the most probable state but all states, weighted by the corresponding posteriors found in step (4). That is,

$$\hat{o}_t = \sum_i \gamma_{t|t-1}(i)\hat{c}(\hat{\lambda}_i), \tag{30}$$

where $\hat{c}(\lambda_i) \sim \mathcal{P}(\hat{\lambda}_i)$ is a sample from the Poisson distribution of state s_i with parameter $\hat{\lambda}_i$.
11: $t \leftarrow t+1$
12: **end while**

the FB formulas in [15]. In this section, we have defined how to execute one-step-ahead prediction in HMMs and HSMMs for sequential data. It is often the case, in real sequential datasets, that the observations depend not only on previous observations but also on external factors. In the next section, we discuss how to add external information into a HSMM.

B. Context aware HSMM

We propose a context aware hidden semi-Markov model, named *HSMM-LR*, with external information introduced as a state conditionality. In order to condition the state transitions on time-varying feature vectors with p features, $\boldsymbol{C} \equiv \{\boldsymbol{c}_t\}_{t=1}^n \in \mathbb{R}^{n\times p}$, the stationary transition matrix has to be redefined as non-stationary, i.e., $\boldsymbol{A} \equiv \{\boldsymbol{A}^{(t)}\}_{t=1}^n \in \mathbb{R}^{n\times k\times k}$, with elements

$$A_{ij}^{(t)} = \mathbb{P}[S_t = s_j | S_{t-1} = s_i, \boldsymbol{c}_{t-1}]. \tag{33}$$

The conditional probability of (33) can be fit, through a logistic regression (LR) model, as a function of the features, given by

$$A_{ij}^{(t)} = \mathbb{P}[S_t = s_j | S_{t-1} = s_i, \boldsymbol{c}'_{t-1}] = \frac{e^{\boldsymbol{\beta}_{ij}{}^\intercal \cdot \boldsymbol{c}'_{t-1}}}{\sum_{g=1}^k e^{\boldsymbol{\beta}_{ig}^\intercal \cdot \boldsymbol{c}'_{t-1}}}, \tag{34}$$

where $\boldsymbol{\beta}_{ij} = [\beta_{ij_0} \quad \beta_{ij_1} \quad \dots \quad \beta_{ij_p}]^\intercal \in \mathbb{R}^{p+1}$ are the bias and weights of the logistic regression model with p features and $\boldsymbol{c}'_{t-1} = [1 \quad \boldsymbol{c}_{t-1}]^\intercal \in \mathbb{R}^{p+1}$. Here, the (multiple) classes are associated with state transitions, and the same goes for the logistic regression weights $\boldsymbol{\beta}_{ij}$. Each class is linked to an indicator variable $y_{t_{ij}} \in \{0,1\}$, which indicates whether or not there was a state transition $s_i \to s_j$ at time t. The class probability is given by $A_{ij}^{(t)}$ in (34) and it represents the probability of a state transition $s_i \to s_j$, supported by the feature vector $\boldsymbol{c}'_{t-1}$. The parameters of the logistic regression are estimated using a maximum likelihood approach. The log-likelihood (LL) of this model is given by[11]

$$\text{LL} = \sum_{t=2}^n \sum_{i=1}^k \sum_{j=1}^k y_{t_{ij}} \left(\boldsymbol{\beta}_{ij}^\intercal \cdot \boldsymbol{c}'_{t-1} - \log\left(\sum_{g=1}^k e^{\boldsymbol{\beta}_{ig}^\intercal \cdot \boldsymbol{c}'_{t-1}} \right) \right). \tag{35}$$

After defining the log-likelihood, the maximum likelihood estimation of the logistic regression weights $\boldsymbol{\beta}_{ij}$ requires computing the partial derivative of the log-likelihood w.r.t the parameters and equating it to zero. That is, fixing one feature $z \in \{1, ..., p\}$, we obtain

$$\frac{\partial \text{LL}}{\partial \beta_{ijz}} = \sum_{t=2}^n y_{t_{ij}} \left(1 - A_{ij}^{(t)}\right) c'_{t-1,z} = 0, \tag{36}$$

where $c'_{t-1,z}$ is the z^{th} feature of the feature vector.
Here, we considered that the state sequence is observed. However, in real scenarios, the state sequence is hidden and we only observe the feature vector $\boldsymbol{C}$ and the observation sequence $o_{1:n}$. Therefore, the hard label $y_{t_{ij}}$ is replaced by a soft label, which corresponds to the transition posterior learnt in the E-step of the EM algorithm. [12] It is well known that the logistic regression problem has no closed form solution given that the sigmoid function is non linear. Thus, in (36), it is not possible to set the derivative equal to zero analytically. To solve this problem, we use a numerical method named *Limited-memory Broyden–Fletcher–Goldfarb–Shanno* (L-BFGS) [24]. Furthermore, to avoid the problem of complete separation [25], we include a regularization term, that penalizes large logistic regression weights, both in the log-likelihood and in the gradient function.
Since we are considering a EDHMM model for the HSMM-LR model, in which self-transitions are not allowed ($A_{ii}^{(t)} = 0$), the logistic regression weights $\boldsymbol{\beta}_{ii}$ need not to be estimated.
The rest of the weights are estimated as the output of k L-BFGS blocks, one for each state $s_i \in \mathcal{S}$. That is,

$$\boldsymbol{\beta}_i = \text{L-BFGS}\left(\boldsymbol{c}'_{t_i-1}, \zeta_{t_i}(i,\cdot)\right) \in \mathbb{R}^{(p+1)\times k}. \tag{37}$$

The inputs of the i^{th} L-BFGS block in (37) are:

- $\boldsymbol{C}'_{t_i-1} \in \mathbb{R}^{r\times(p+1)}$: a subset of the feature vector $\boldsymbol{C}' \in \mathbb{R}^{n\times(p+1)}$ which selects only the features at r time-steps t_i. Those timesteps correspond, according to the predicted model, to the instants where a transition from state s_i (to any other state) occurred. These instants are obtained from the predicted state sequence, output of the *Viterbi* algorithm, which we run simultaneously with the E-step of the EM algorithm.

- $\zeta_{t_i}(i,\cdot) \in \mathbb{R}^{r\times k}$: the transition posteriors from state s_i to all states, also selected only at time-steps t_i. In fact, instead of the state transition posteriors ζ, we use the normalized transition probabilities from the M-step $\left(\hat{A}_{i(\cdot)}^{(t_i)}\right)$, without the diagonals due to the explicit duration formulation. This is an approximation for numerical reasons, since the transition posteriors are not normalized.

The output of the i^{th} L-BFGS block in (37) is the weight vector $\boldsymbol{\beta}_i \in \mathbb{R}^{(p+1)\times k} = [\boldsymbol{\beta}_{i1}, ..., \boldsymbol{\beta}_{ik}]$, where $\boldsymbol{\beta}_{ij} \in \mathbb{R}^{p+1}$, and k is the number of states. The parameter estimation problem for the HSMM-LR model is an iterative method based on the EM algorithm. In the E-step, we estimate the state, transition and duration posteriors using the FB algorithm. In the M-step, the HSMM parameters are (re-)estimated ($\hat{\mu} = \{\hat{\boldsymbol{A}}^{(\boldsymbol{t})}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{D}}\}$) based on those posteriors. The L-BFGS is included in the M-step, where the logistic regression weights are (re-)estimated using the transition matrix $\hat{\boldsymbol{A}}^{(\boldsymbol{t})}$ and the feature vector, according to (34). The resulting matrix is used as the input of the EM algorithm in the next iteration. This process repeats until the EM converges or some stop criteria is met. The one-step-ahead prediction problem in HSMM-LR is addressed in the same way as for HSMMs. The only adaptation needed is the replacement of the stationary transition matrix in (27) by the time-dependent version from (34).

V. Results

A. Comparison of model order selection criteria

In the present experiment, we compare the AIC, BIC and MMDL model order selection criteria, in their standard and pruning versions. The standard version is similar to the pruning version, but instead of initializing the EM parameters with the result of the last iteration with the less probable state pruned, we simply initialize them at random with one less state. The synthetic data consists of one sequence of 1000 observations ($n = 1000$), sampled from

[11] The terms that do not depend on the LR weights are removed.

[12] Note that the soft labels only affect, in a simple manner, the likelihood function, but not the way it depends on the parameters.

a HSMM with $k = 5$ states. To implement the model equivalence described in section III, we modeled the duration distribution of each state through geometric distributions with parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_k]^\intercal$, where $\theta_i \in \mathbb{R}^k$ is the geometric parameter of the duration distribution of state s_i. The rest of the parameters $\mu = \{\boldsymbol{A}, \boldsymbol{\pi}, \boldsymbol{\lambda}\}$ are obtained at random (except $\boldsymbol{\lambda}$), according to (38).

$$\boldsymbol{\pi} = \begin{bmatrix} 0.0062 \\ 0.0926 \\ 0.1914 \\ 0.2593 \\ 0.4506 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} 2 \\ 15 \\ 36 \\ 49 \\ 105 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} 0.7572 \\ 0.8123 \\ 0.7970 \\ 0.8077 \\ 0.8293 \end{bmatrix},$$
$$\boldsymbol{A} = \begin{bmatrix} 0 & 0.5714 & 0.1364 & 0.2727 & 0.0195 \\ 0.2718 & 0 & 0.2953 & 0.3255 & 0.1074 \\ 0.3043 & 0.0435 & 0 & 0.4174 & 0.2348 \\ 0.3511 & 0.2801 & 0.2660 & 0 & 0.1028 \\ 0.0198 & 0.2673 & 0.2178 & 0.4950 & 0 \end{bmatrix}. \tag{38}$$

The maximum allowed state duration (d_{max}) can be computed from (18), using $\epsilon = 0.001$, yielding a value of $d_{max} = 4$.

We sampled 50 independent observation and state sequences from the setup of (38). Subsequently, we initialized, at random, the parameters μ used as starting point of the EM algorithm. The initialization was the same for all three criteria. Then, we ran the standard and pruning versions of BIC, AIC and MMDL and we obtained the optimal number of states (k^*) and optimal model ($\mu^*_{k^*}$) chosen by each criterion. Since the true number of states is $k = 5$, we considered $k_{max} = 10$ and $k_{min} = 2$ for pruning. The maximum number of iterations for the EM algorithm was set to 100.

In Fig. 1, we show a histogram of the number of times each state was chosen as the optimal total number of states, for each criterion. We observe that the pruning version of the BIC and MMDL criteria are the most accurate ones, since they hit most often the right total number of states ($k = 5$). We also note that the only criterion to estimate 7 and 8 states is the AIC (in the pruning and standard versions), which confirms that the penalization on the model complexity is lower in this criterion when compared to the others. In

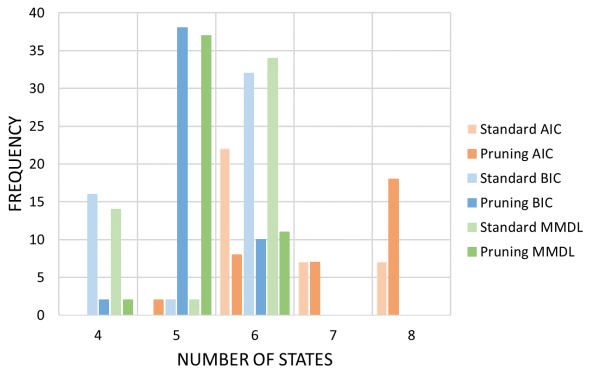


Fig. 1. Histogram of the chosen optimal number of states per criterion, with both the standard and pruned versions. The BIC is shown in blue, the AIC in orange and the MMDL in green.

Table I, we present the mean and standard deviation (for the 50 samples) of the chosen optimal number of states, *hit rate* and number of EM iterations at convergence, for each criterion. The hit rate measures the percentage of correct state assignments between the estimated and true state sequences. That is,

$$\text{Hit rate} = 1 - \frac{||S - \hat{S}||_0}{n \times N}, \tag{39}$$

where $||\cdot||_0$ is the ℓ_0-norm, S is the true state sequence and $\hat{S}$ is the predicted state sequence. From inspection of the table, we conclude that the pruning version of the BIC and MMDL are the heuristics that perform best, with a average number of states closer to the real one, a high hit rate and with the lowest number of iterations due to the pruning strategy. We also note that the number of iterations for the AIC criterion, from the standard to its pruning

TABLE I
ESTIMATION RESULTS FOR ALL CRITERIA: CHOSEN OPTIMAL NUMBER OF STATES, HIT RATE AND NUMBER OF EM ITERATIONS AT CONVERGENCE.

Criteria	k^*	Hit rate	Nº EM iterations
Std. BIC	5.32 ± 0.94	0.81 ± 0.12	85.32 ± 22.15
Std. AIC	7.38 ± 1.48	0.83 ± 0.08	99.48 ± 3.68
Std. MMDL	5.40 ± 0.90	0.82 ± 0.12	87.50 ± 20.73
Pr. BIC	$\mathbf{5.16 \pm 0.47}$	$\mathbf{0.90 \pm 0.06}$	$\mathbf{65.50 \pm 37.17}$
Pr. AIC	7.78 ± 1.27	0.77 ± 0.06	88.14 ± 28.42
Pr. MMDL	$\mathbf{5.18 \pm 0.48}$	$\mathbf{0.90 \pm 0.06}$	$\mathbf{67.36 \pm 36.50}$

versions, do not differ as much as for the BIC and MMDL criteria because the complexity is increased by having more states. However, for the other criteria, the pruning saves more than 20 iterations while achieving a better hit rate. Finally, in (40), we present the mean and standard deviation of the parameter estimates of the geometric state duration distributions ($\boldsymbol{\theta}$) and the Poisson rates ($\boldsymbol{\lambda}$), for the MMDL criterion with the pruning strategy. These results are averaged only for the runs where the chosen number of states corresponded to the ground truth, i.e., $k^* = 5$.

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} 0.7549 \\ 0.8140 \\ 0.7978 \\ 0.8331 \\ 0.8311 \end{bmatrix} \pm \begin{bmatrix} 0.0280 \\ 0.0238 \\ 0.0607 \\ 0.0617 \\ 0.0326 \end{bmatrix}, \quad \hat{\boldsymbol{\lambda}} = \begin{bmatrix} 2.3339 \\ 15.1223 \\ 36.4917 \\ 49.1569 \\ 105.1510 \end{bmatrix} \pm \begin{bmatrix} 0.0957 \\ 0.2842 \\ 1.0271 \\ 1.0271 \\ 1.1813 \end{bmatrix}. \tag{40}$$

We observe that the parameter estimates in (40) are pretty accurate when compared to the real ones from (38).

B. Time-series forecasting on social media

In this section, we present the results of time-series forecasting models using the previously described models (HMM, HSMM, HSMM-LR and LSTM) applied to a social media dataset.

1) Twitter dataset: We used an open-source Kaggle dataset,[13] which consists of 16 million tweets about bitcoin, from 01/01/2016 to 23/11/2019. We selected only the last 5 months, and, after preprocessing and cleaning, we used only the english sentences, which consist of 73.42% of the whole dataset. Moreover, the dataset was discretized so that discrete hidden (semi-)Markov models can be used. Since discretization by itself is not desirable because it is problem-dependent, we have considered a wide range of different discretization values: $\triangle t = \{1, 3, 4, 6, 12, 24, 48\}$ hours, which means that the observations ($\{o_t\}_{t=1}^n$) are the number of tweets in each interval of 1 hour, 3 hours, and so on. As bitcoin is a relatively recent topic, the focus on this area is still on exponential growth. It is essential to add external information so that the predictions are more realistic. For example, in a given week, the number of tweets about bitcoin may drastically increase because there was a drop in the price of the virtual currency, or vice versa. Since this information may be present in the text of the tweets, we focused on extracting textual features for time-series prediction of bitcoin data. We extracted both TF-IDF [26] features from the text corpus, and sentence embeddings from the *pretrained BERT*[14] [27]. We only present the results for BERT features since they are more representative than TF-IDF. We average all embeddings inside a discretized window. Furthermore, to account for the effect of previous windows, we introduce another variable of discretization, $\triangle w = \{0, 1, 3, 5, 7\}$. A value of $\triangle w = 0$ means that the feature vector for time t ($\boldsymbol{c}_t \in \mathbb{R}^p$)[15], contains only the average embedding from the current time window, whereas a value of $\triangle w = 7$ averages the embeddings of the previous 7 windows. To average windows, we use a weighted summation which gradually gives less weight to the average embedding of the farther back in time windows, according to an exponential factor. This process is illustrated in Fig. 2. Both TF-IDF and BERT features are normalized according to the z-score normalization. We

[13]That can be found in https://www.kaggle.com/alaix14/bitcoin-tweets-20160101-to-20190329.

[14]The BERT sentence embeddings were obtained from https://github.com/UKPLab/sentence-transformers and correspond to a vector of dimension 768 (per sentence).

[15]The number of features of BERT embeddings is $p = 768$.

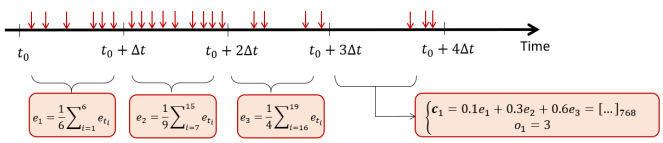


Fig. 2. Process of obtaining the embedding of a window by weighted summation of the average embeddings of the previous windows. The red arrows represent tweet arrivals in time.

trained the following models: HMM, HSMM, HSMM-LR and LSTM. For the Markov models, we performed the sequential pruning strategy separately for HMMs and HSMMs with the MMDL criterion,[16] using $k_{max} = 50$ and $k_{min} = 5$. The optimal chosen hyperparameters were $k^*_{\text{HMM}} = 36$, $k^*_{\text{HMM}} = 14$ and $d^*_{max} = 20$. Using the trained models, we performed one-step-ahead prediction in the test set. The results are shown in the next section.

2) Forecasting on social media - Markov models vs. LSTM: In this experiment, we analyze how the amount of training data affects the prediction accuracy. We held-out the last 10% of the data for the validation and test sets (5% for each). The model is trained on different percentages of the training data, ranging from 90% to 5%. In this experiment, we compare the Markov models with and without features (HSMM-LR vs. HMM and HSMM) and the LSTM without features. We proceed to describe our LSTM architecture.

a) LSTM architecture: In Fig. 3, we show a sketch of the architecture of our LSTM. The LSTM inputs are sliding windows of the 1 hour discretized

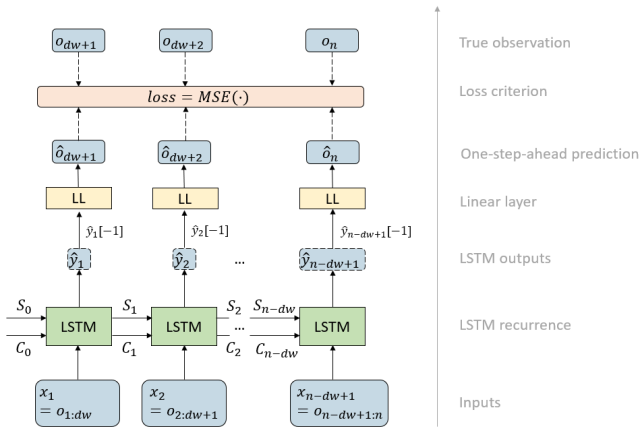


Fig. 3. Our LSTM architecture, where x represent the inputs, S and C the LSTM hidden and cell states, respectively, $\hat{y}$ the outputs of the LSTM, and o and $\hat{o}$ the true and predicted observations, repectively.

dataset ($\triangle t = 1$) without features, i.e., the first input x_1 corresponds to the true observations from $t = 1$ to $t = d_w = 24$, the next input is x_2 which corresponds to the previous input shifted one unit to the right, i.e., $x_2 = o_{2:25}$ and so on until all the dataset is fed into the LSTM. The sliding windows are needed for one-step-ahead time series forecasting in LSTMs. The use of $d_w = 24$ has only to do with our dataset. Given that each timestep represents a 1h interval, we have considered that sliding windows of dimension 24 were sufficient, since the LSTM manage to extract information even more far back by itself (through the hidden and cell states).

Furthermore, the observations, before being fed to the LSTM, are normalized using a MinMax scaler between $[-1, 1]$, fit to the training data. We now describe the two main blocks of Fig. 3: the LSTM and the Linear Layer (LL). The LSTM has an input size of 1, since we are not considering features, only twitter counts. We considered a one-layer LSTM with a hidden layer size of 128, i.e., 128 nodes per layer. Regarding the LL, it has an input dimension of 128 and an output size of 1.

[16]The number of states chosen for the HSMM-LR model is considered the same as the HSMM.

The LSTM act as an encoder, by upsampling the dimension of the inputs, i.e., in each timestep, we input $d_w = 24$ consecutive observations. The LSTM augments this dimension by increasing the size of the feature space from 1 to 128. Then, it outputs a matrix of dimension 24 by 128, from where we will use only the last vector of dimension 128. We feed that vector through the linear layer[17], which acts as a decoder, since it transforms the LSTM output into a one-step ahead prediction. Then, the linear layer weights are learnt along with the LSTM parameters so as to minimize the loss, which corresponds to the mean-squared-error between the LL output (the one-step-ahead prediction) and the true observation, as shown in Fig. 3.[18]

To sum up, the LSTM architecture predicts a single next observation for each sliding window input. During training, we evaluate the LSTM parameters on the validation set and, after training, on the test set. To evaluate, we use the last 24 true observations from the training set plus the initial hidden state, which act like the state posteriors for the one-step-ahead prediction in parametric Markov models, described in Section IV-A. This allows us to predict the first observation from the validation set. Then, we append the true observation value and we shift the sliding window by one to the right, thus predicting the second observation from the validation set, and so on. The same procedure is done for the test set.

We note that, in Fig. 3, the parameters of the Linear Layer and the LSTM are shared for all timesteps.

The initial hidden and cell states, S_0 and C_0, are initialized as random parameters that must be learnt along with the other LSTM parameters. This way, we guarantee that the dependence between sliding windows is included in the information passed by the hidden states.

In Fig. 4, we present a graphical visualization of the MAE prediction errors, along with error bars, for the different percentages of training data, for the test set and all models with $\triangle t = 1$.

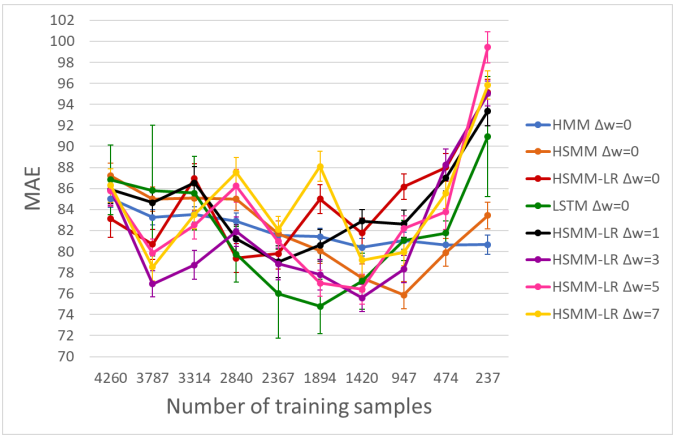


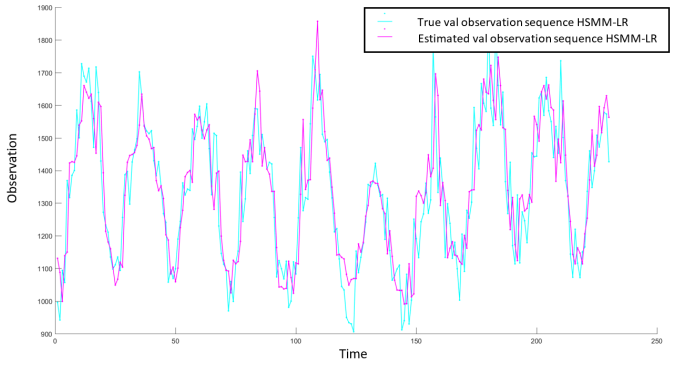
Fig. 4. Evolution of the MAE prediction errors with different percentages of training data for the test set. The models considered are: HMM, HSMM and LSTM, all for $\triangle t = 1$ and $\triangle w = 0$ and HSMM-LR for $\triangle t = 1$ and $\triangle w = \{0, 1, 3, 5, 7\}$.

The X-axis in Fig. 4 is the number of training samples. The Y-axis is the *mean-absolute error* (MAE) prediction errors.

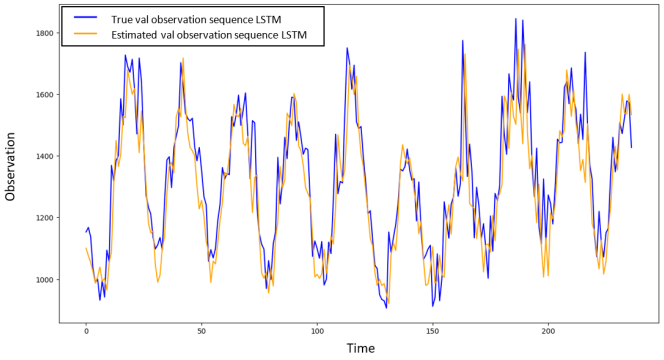
We can observe that the LSTM achieves the lowest MAE prediction errors in the interval from 1894 to 2840 training samples. We note that the LSTM is the second worst model, only after the HSMM, when considering 90% of the training data (4260 samples). It is also interesting to notice that the HSMM-LR is usually better than the HMM and HSMM, but with less training data, the logistic regression weights are not properly estimated from the features, which worsen the predictions. Regarding the HSMM-LR model, we note that the best window for the test set is $\triangle w = 3$, which is halfway in between of not averaging anything and averaging too much information, leading to noise. We can also state that the HSMM has, in general, worst predictions than the

[17]The vector corresponds to the features of the linear layer model.

[18]We use an Adam optimizer with a learning rate of 0.001 (default).



(a) HSMM-LR model for the dataset with $\triangle t = 1$, $\triangle w = 7$



(b) LSTM model with $\triangle t = 1$

Fig. 5. Prediction results for the validation set with a train/val/test percentage of $60 - 5 - 5$. The models are: (a) HSMM-LR for the dataset with $\triangle t = 1$, $\triangle w = 7$; (b) LSTM model with $\triangle t = 1$.

HMM, which may be due to the discretization unit. However, we note, by observing the Y-axis scale, that the variations in the MAE prediction errors between all models are not relevant. In Fig. 5, we show two examples of the temporal evolution of the true and predicted observation sequences for the validation set with a 60% of training. While Fig. 5a(a) refers to model HSMM-LR with $\triangle t = 1$ and $\triangle w = 7$, Fig. 5b(b) corresponds to the LSTM without features.

From Fig. 4, we observe that the HSMM-LR 1.7 has a higher MAE than the LSTM 1.0 for a 60% of training data (which corresponds to 2840 samples), which is confirmed by comparing the predictions from Fig. 5.

VI. Conclusions

A. Summary of contributions

This work adds two main contributions. The first regards the proposal of a context based HSMM, named HSMM-LR, which conditions state transitions on external features through a logistic regression.

The forecasting results indicate that, even though LSTMs are very popular nowadays, parametric models (such as HMM, HSMM and HSMM-LR) should not be underestimated as they can achieve close or even better prediction accuracy (with less training data) while remaining easy to train and more interpretable. Moreover, the parametric Markov models addressed in this work have way less parameters and hyperparameters to estimate and tune, and are still able to capture a lot of data patterns. Furthermore, these models are trained faster than any artificial neural network model.

However, in what the disadvantages are concerned, it is more difficult to extend or develop an end-to-end implementation of these models, when compared to neural networks. Moreover, parametric models make a lot of assumptions about the data and, in particular, the HSMM-LR model was proposed with a limitative assumption that the logistic regression weights only depend on state transitions.

Regarding the second contribution, we adapted the MMDL criterion, for order selection problems, to HSMMs. This adaptation was presented in a national pattern recognition conference - RECPAD2019.

Last but not least, the source code is publicly available at https://github.com/filiparente/Predtweet/.

B. Future Work

The main lines of future work to further improve our comparative study may explore:

- **Dataset limitations**

  Try out other real datasets and/or explore a smaller discretization unit ($\triangle t < 1$h), so that the observation signal is smoother and the features less noisy.

  Better feature engineering, i.e., collect more useful features by exploring entity recognition methods or combining textual with non-textual features.

- **Prediction limitations**

  One of the first limitations is that we have considered a simple one-step-ahead prediction, which is not very realistic. Following that mindset, a possible line of further work is to introduce:

  Multi-step-ahead prediction, since in real-life datasets we usually want to forecast multiple timesteps in the future; or

  Online learning, in which instead of updating the parameters offline, they are constantly being changed whenever a new observation is available. This option is particularly interesting for social media applications, given that the textual features have temporal attention drifts over time. Online learning mechanisms allow us to capture those drifts without requiring training the model from scratch. Moreover, some predictions may not be related with past predictions since social media topics are constantly changing.

- **Model limitations**

  Another caveat in our implementation is the use of MMPP models, which can only achieve good performances when the dataset is discretized or if the observation signal is non smooth. In that regard, it would be interesting to consider models such as the *Gaussian processes modulated Cox processes* (GPCox) [28] and *Markov modulated Gaussian Cox processes* (MMGCP) [29] models, which combine Gaussian processes to obtain the desired smoothness. However, those models require a transformation that guarantees that the observations are positive (e.g. Log-transformation), which makes the problem intractable, forcing the use of approximate inference methods such as *Markov chain Monte carlo* (MCMC) [30] or *Variational Bayesian inference* methods [31] [32] [33]. A more promising idea would be to find a non-Gaussian conjugate prior with interesting and smooth properties that has, at the same time, a closed-form expression for the likelihood function.

- **Context based models**

  In this regard, it would be interesting to explore other ways to add features to parametric models. One suggestion is to consider the use of a model (e.g. MMPP, MMGCP or GPCox) together with a Cox regression approach. The model captures the temporal characteristics of the time-series while the Cox regression allows us to introduce context into the model, by conditioning it in external features. We could consider a more general state of the Markov chain which includes the information about the features. For example, for the MMGCP model, each state of the Markov chain comprises a latent function with some intensity level modeled by a Gaussian process prior, and we propose that it could also include the feature characteristics. The overall intensity function of the arrival process is modeled through conditional jumps in the states of the Markov chain, by defining the Markov process variable to be conditioned on the features and applying Cox regression to decide to which state to move next given the features of the arrival and the previous state.

References

[1] A. F. Dugas, M. Jalalpour, Y. Gel, S. Levin, F. Torcaso, T. Igusa, and R. E. Rothman, "Influenza forecasting with google flu trends," *PloS one*, vol. 8, no. 2, p. e56176, 2013.

[2] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE, 2017, pp. 1643–1647.

[3] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by lstm," *Energy*, vol. 148, pp. 461–468, 2018.

[4] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.

[5] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[6] D. Lando, "On cox processes and credit risky securities," *Review of Derivatives research*, vol. 2, no. 2-3, pp. 99–120, 1998.

[7] W. Fischer and K. Meier-Hellstern, "The markov-modulated poisson process (mmpp) cookbook," *Performance evaluation*, vol. 18, no. 2, pp. 149–171, 1993.

[8] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 4277–4280.

[9] R. Meenakshi and P. Anandhakumar, "A hybrid brain tumor classification and detection mechanism using knn and hmm," *Current Medical Imaging Reviews*, vol. 11, no. 2, pp. 70–76, 2015.

[10] M. Zhang, X. Jiang, Z. Fang, Y. Zeng, and K. Xu, "High-order hidden markov model for trend prediction in financial time series," *Physica A: Statistical Mechanics and its Applications*, vol. 517, pp. 1–12, 2019.

[11] J. Read, "Hidden markov models and dynamic programming," *University of Oslo*, 2011.

[12] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[13] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *speech communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[14] S.-Z. Yu, "Hidden semi-markov models," *Artificial intelligence*, vol. 174, no. 2, pp. 215–243, 2010.

[15] S.-Z. Yu and H. Kobayashi, "Practical implementation of an efficient forward-backward algorithm for an explicit-duration hidden markov model," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1947–1951, 2006.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[18] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[19] M. A. Figueiredo, J. M. Leitão, and A. K. Jain, "On fitting mixture models," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 1999, pp. 54–69.

[20] M. Bicego, V. Murino, and M. A. Figueiredo, "A sequential pruning strategy for the selection of the number of states in hidden markov models," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1395–1407, 2003.

[21] V. Barbu, J. Bulla, and A. Maruotti, "Estimation of the stationary distribution of a semi-markov chain," *Journal of Reliability and Statistical Studies*, vol. 5, pp. 15–26, 2012.

[22] X. Wang, P. Whigham, D. Deng, and M. Purvis, "Time-line hidden markov experts for time series prediction," in *International Conference on Neural Networks and Signal Processing, 2003. Proceedings of the 2003*, vol. 1. IEEE, 2003, pp. 786–789.

[23] Z. Chen, "Estimating latent attentional states based on simultaneous binary and continuous behavioral measures," *Computational intelligence and neuroscience*, vol. 2015, p. 26, 2015.

[24] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[25] A. Albert and J. A. Anderson, "On the existence of maximum likelihood estimates in logistic regression models," *Biometrika*, vol. 71, no. 1, pp. 1–10, 1984.

[26] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[28] T. Gunter, C. Lloyd, M. A. Osborne, and S. J. Roberts, "Efficient bayesian nonparametric modelling of structured point processes," *arXiv preprint arXiv:1407.6949*, 2014.

[29] M. Kim, "Markov modulated gaussian cox processes for semi-stationary intensity modeling of events data," in *International Conference on Machine Learning*, 2018, pp. 2645–2653.

[30] R. P. Adams, I. Murray, and D. J. MacKay, "Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 9–16.

[31] C. Lloyd, T. Gunter, M. Osborne, and S. Roberts, "Variational inference for gaussian process modulated poisson processes," in *International Conference on Machine Learning*, 2015, pp. 1814–1822.

[32] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[33] A. Dezfouli and E. V. Bonilla, "Scalable inference for gaussian process models with black-box likelihoods," in *Advances in Neural Information Processing Systems*, 2015, pp. 1414–1422.