Modeling Multi-User Behaviour in Social Networks

Tiberiu Chis, Peter G. Harrison

Department of Computing, Imperial College London
South Kensington Campus, London, UK
Emails: {tiberiu.chis07, p.harrison}@imperial.ac.uk

***Abstract*—Social networks have revolutionized the way millions of users interact on the internet over the last decade. Therefore, leading social media companies such as Facebook, YouTube, LinkedIn and Twitter (to name but a few) have grown dramatically to manage such enormous online user-populations, which has led to an explosion of multimedia content on such social networking websites. Thus, the individual and collective behaviour of the ever-increasing user-base has been studied in more depth by researchers and, consequently, the properties of online social interactions have been modeled. In an attempt to rigorously analyse such internet traffic, we propose a Multi-dimensional HMM (MultiHMM) to act as a Multi-User workload classifier. The MultiHMM is an adaptation of the original HMM, using clustering methods and multi-processing of the Baum-Welch algorithm. The model is trained by multiple user traces simultaneously, whilst at the same time, the number of iterations of the costly expectation-maximization (EM) algorithm is reduced. The ultimate goals of our proposed MultiHMM are to classify multiple online user streams with minimal processing needs, to represent burstiness and correlation amongst groups of users and to improve security measures in the social network. Experiments have been carried out using multiple traces from Twitter data, where original traces are analysed and compared with the MultiHMM-generated traces. The metrics involved in validating our model include means, standard deviations, skewness, autocorrelation and Mean Absolute Percentage Error (MAPE). Applications and extensions of our model are considered and further statistical results are presented in the final section.**

I. Introduction

In the last decade, leading social media companies such as Facebook, YouTube, LinkedIn and Twitter (to name but a few) have grown to manage an enormous user-base, which has led to an explosion of multimedia content on such social networking websites. Thus, the individual and collective behaviour of the ever-increasing user-base has been studied in more depth by researchers and, consequently, the properties of online social interactions have been modeled. For example, researchers have used Twitter data to examine links within tweets and determine characteristic patterns that help define misinformed events [23]. Such work [23] has evolved the analysis of multiple crisis events (i.e. Boston Marathon Bombings in 2013) and aims to prevent bad information or "rumors" spreading via Twitter and similar social media sites.

To accurately represent such user activity, one of the simplest models is the Poisson process, which is a continuous-time stochastic point-process, in which inter-arrival times (between events) are independent and exponentially distributed. This process can be discretized, via partitioning time-stamped data into "bins", and turned into a portable, discrete-time stochastic process or time-series. One such parsimonious model is the hidden Markov model (HMM), which has garnered success characterizing and classifying Internet traffic data. HMMs were used in [20] to approximate self-similar traffic from CAIDA [22] traces and verify long-range dependence using the Hurst coefficient. In [13], HMMs are used to analyse Traffic Burstiness and Internet Packet-level Sources and variations of HMMs have been applied to identifying trends in user behaviour in social networks [19]. More specifically, [19] introduces a new class of Coupled HMMs to describe temporal patterns of user activity which incorporate users' neighbours in a social network. Each HMM corresponds to one user and the coupling of models represents user interaction. These Coupled HMMs provide better explanatory and predictive power compared with existing models such as those based on Renewal Processes or uncoupled HMMs. However, with increased interaction, the coupling of HMMs became more complex and the training more computationally expensive. Therefore, an improvement suggested in [19] involves developing more efficient models for social analysis on groups of users, although research involving a model for individual email communication [11], [24] has not considered Multiple User sessions.

Not surprisingly, then, a common problem that arises from such work as [11], [19] is the lack of efficient training on multiple traces, derived from communicating users, to represent and classify Multi-User behaviour. Here we propose a model that overcomes this issue, using a hybrid HMM which is both parsimonious and can classify Multi-User temporal activity, without decreasing in accuracy and computational efficiency. Essentially, this is a Multi-User HMM (MultiHMM), which uses K-means clustering and a Multi-Input Baum-Welch algorithm to obtain the required parameters to form a Discrete Markov Multi-Arrival Process (DMMAP). For our results, we validate this DMMAP[1] by comparing quantitatively means, standard deviations, skewness and autocorrelation computed from the raw (i.e. unclustered) and MultiHMM-generated traces of user actions. Another efficiency measure for the MultiHMM (compared to a standard HMM) is the number of steps required for the model parameters to converge in the Baum-Welch algorithm component. We conclude by summarizing the advantages of the MultiHMM over existing models and explain potential MultiHMM real-world applications, for example quantifying activity burstiness amongst groups of users and classifying fake or suspicious users for security purposes. Based on our results and experience, some ideas for enhancements to the MultiHMM are proposed as future research and the Appendix summarizes background information on HMMs and additional statistical results for groups of users.

[1]Note: MultiHMM and DMMAP will be used interchangeably in this paper

II. Related Work

We first give a brief introduction to some existing models and identify their limitations. We focus on research involving Hypergraphs as Twitter clones [4] and some variations of HMMs for parallel training.

A. Hypergraphs

In social networks, studies have shown that Multi-User Queries result in multiple operations [4] and thus are expensive in terms of performance. To model these Multi-User interactions, a solution of Selective Replicated Partitioning was implemented through a Temporal Activity Hypergraph Model [4]. In this model, the vertices represented the users and the nets corresponded to the Multi-User Queries. The study in [4] used a Twitter clone (hosted on Amazon EC2) to obtain a realistic experimental test bed. The Hypergraph Model then performs simultaneous partitioning and replication to reduce the query span while respecting load balance and I/O load constraints under replication. Indeed, it was shown that the Hypergraph Model is the best choice (among other hash- or graph-based approaches) for predicting future query patterns and significantly improving latency and throughput. However, replicating whole sections of Twitter networks using the Hypergraph method proves costly (in terms of storage and computational complexity) for increased numbers of users and their interactions. Also, the proposed model has many parameters [4]. The HMM attempts to solve both of these issues, acting as a parsimonious (efficient training with few input parameters) model capable of Multi-User training.

B. Variations of HMMs

Recently, the literature has seen frequent research into parallelization of HMM training algorithms (i.e. Baum-Welch), mainly due to increasing applications of HMMs in User Modeling and Pattern Classification. One technique has involved GPUs to parallelize the Baum-Welch algorithm using CUDA implementations [17], [18]. Apart from synchronization of threads, other research has relied on variations of HMMs. For example, a Factorial HMM [6], where observed outputs are combined, requires separate hidden state sequences to train the observation chain. Each sequence of hidden states is independent from the next and there is no interaction between chains via state probabilities. Other variations include Coupled HMMs [19] (see Fig. 1), where the output (O_t) is separate and each hidden state (C_t) emits an observation with a given probability. The hidden state sequences interact with each other, resulting in many possible combinations for model setup. However, with increased interaction (i.e. coupling of HMMs), Baum-Welch training becomes much more computationally expensive; a potential problem noted earlier in the paper.

Layered HMMs have been used for multiple sensory channels [14] with multiple state sequences corresponding to one observation sequence, which iteratively updates based on its likelihood. This model is less likely to suffer from overfitting, but will be computationally more expensive on Baum-Welch training compared to the traditional HMM. In [7], two parallel and independent HMMs are combined, information about sign language being merged for each model using a token-passing algorithm. Despite the reduced computation

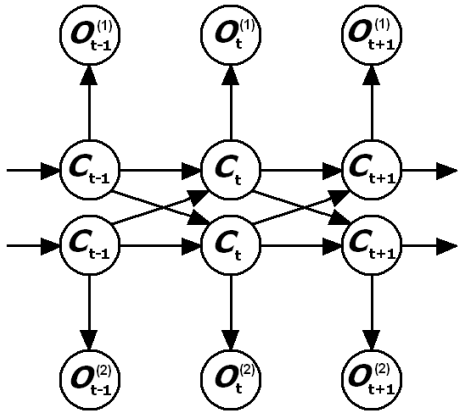

Fig. 1: Example of two Coupled HMMs [19]

requirement needed for the two models, scaling to higher numbers of models (i.e. to train on larger vocabularies) would not be as efficient. Here we propose a robust MultiHMM solution to optimize training of the HMM on multiple traces, whilst maintaining accuracy. The MultiHMM uses two layers of clustering and an adapted Baum-Welch algorithm, where multiple traces can be processed by a single HMM.

III. Background

The focus of this paper is to determine Multi-User characteristics and model the behaviour of groups of users on social networks. The technical aim is to implement a Multi-User unsupervised learning technique, which is essentially a hybrid HMM. The underlying properties of the HMM (including its stochastic, predictive and parsimonious nature) have made it appealing in classifying user profiles on social media. These defining characteristics of the HMM have led to a range of applications in a wide variety of fields: originally, HMMs were used in Speech Recognition [8], [16] and Genome Sequence Prediction [10], [12]; more recently, they have represented Storage Workloads [9], [21] and have modeled Hospital Patient Arrivals [21] as well as Social Network Interactions [19]. In all cases, these models have employed well-known statistical algorithms to solve three fundamental problems credited to HMMs. These problems are:

1) Find $P(O;\lambda)$, the probability of observing O given the model λ, using the **Forward-Backward algorithm** [1].
2) Maximize $P(O;\lambda)$ by adjusting the parameters of the model λ using the **Baum-Welch algorithm** [2].
3) Obtain the most likely hidden state sequence for the observation set O, using the **Viterbi algorithm** [3].

Using these algorithms to iteratively train on data sets, HMMs can faithfully represent workloads for discrete time processes. Therefore, HMMs can be used as portable benchmarks to explain and predict the complex behaviour of these processes, and more specifically for the social media phenomena such as Twitter and Facebook. As HMMs can efficiently represent workload dynamics, acting as parsimonious models that obtain trace characteristics, their popularity in the social media research sector is no surprise.

In the Appendix, we describe the aforementioned Forward-Backward algorithm (FBA) and Baum-Welch algorithm (BWA) in more detail, including their defining recursion equations. In fact, the BWA re-estimation formulas, defined by Eq.

(8), (9) and (10), only work on a single trace of observations and a useful upgrade for the BWA is to handle multiple stream analysis for characterizing users. Adapted versions of the FBA and BWA, for the purpose of modeling Multi-User processes using clustering techniques, are the novel contributions of this paper and are explained in the next section.

IV. MULTIHMM

The novel contribution of this paper, namely a hybrid HMM for Multi-User training, comprises a simple K-means clustering algorithm for user traces and then a weighted BWA (MultiBWA), which trains on multiple discrete traces simultaneously and maintains accuracy with respect to moments of trace comparisons. The metrics for validating our MultiHMM technique include the mean, standard deviation, skewness, and autocorrelation of traces, as well as the Mean Absolute Percentage Error (MAPE) of model-generated time series. We begin by explaining our clustering methodology and present our specialized algorithm in the following sections.

A. Clustering Methodology

Clustering is used in pre-processing of input traces for training the BWA. One method involves K-means clustering to group each observation point from all traces [21], which results in a tuple of size H (where H is the number of input traces). Each combination of data points in this tuple will produce a distinct cluster, and in total there exist many possible clusters. When H is large, this leads to unacceptably high computational complexity for the BWA. For example, with only three traces (H=3), there exists up to 3^m possible combinations of 3-tuples, where m is the number of distinct trace categories (i.e. "No tweets", "Few", "Medium", "Frequent", etc.). This highlights the need for a more computationally efficient method of clustering traces.

Our proposed method of clustering involves reducing all H traces to K, where K is the number of clusters used in the K-means algorithm. If K $<$ H, then the number of traces is reduced to K through grouping together data points from the same cluster (of tuples); if H $\leq$ K, then this process of partitioning data points into cluster groups is omitted. This extra clustering reduces the computational burden further, before the BWA trains on the doubly-clustered traces. The next step is to assign weights to each trace, after which we train using the MultiBWA.

B. MultiHMM Algorithm

The full pseudo-code for the MultiBWA is provided in our specialized algorithm (Algorithm 1). The algorithm initializes its weights with equal probabilities, but a possible extension would be to prioritize the weights, according to the respective user streams. These priorities can define variations of the MultiHMM, depending on the groups of users used to train it. However, a potential pitfall of using certain types of user traces for model training can result in a more specific, but less applicable model. With any unsupervised learning technique, the more data provided (with more diverse inputs), the more accurate classification obtained. We apply this principle when training our model on multiple traces, simultaneously.

Algorithm 1 Training using MultiHMM

Require: K Clusters $\wedge$ H Traces $\wedge$ $size$ = Trace.length
  for $i = 1 : H$ Raw Traces **do**
    while Cluster Points not Fixed **do**
      K-means Clustering on Trace$_i$
    end while
  end for
  while $K < H$ **do**
    for $j = 1 : K$ **do**
      $Group_j$ = $\{\}$
      for $t = 1 : size$ **do**
        for $i = 1 : H$ **do**
          if Data Point Trace$_i(t) \in Cluster_j$ **then**
            if $Group_j \neq \{\}$ **then**
              $H \leftarrow H - 1$
            end if
            Add Trace$_i(t)$ to $\{Group_j\}$
          end if
        end for
      end for
    end for
  end while
  for $1 : K$ Observation Traces **do**
    while MultiBWA Parameters not Converged **do**

$$\hat{\alpha} = \sum_{i=1}^{N} \omega_i \alpha_i;\ \hat{\beta} = \sum_{i=1}^{N} \omega_i \beta_i;\ \hat{\xi} = \sum_{i=1}^{N} \omega_i \xi_i$$

$$\gamma_t(i) = \sum_{j=1}^{N} \hat{\xi}_t(i,j);\ \pi'_i = \gamma_1(i)$$

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \hat{\xi}_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \hat{\xi}_t(i,j)};\ b_j(k)' = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

    end while
  end for

This model now contains Multi-User information, and traces generated synthetically can be compared to invididual user profiles for validation. In the next section, we explain the simulation of the MultiHMM for various groups of Twitter users along with a collection of corresponding results.

V. RESULTS

We simulate a two-state MultiHMM for different groups of Twitter users: The first simulation analyzes only three Twitter users; The second simulation analyzes three groups of Twitter users, where each group has 3134, 17594, and 42729 users, respectively. Timestamped "tweet" information was captured from each user and we refer to this time series of tweets as a user's "Twitter trace". Each Twitter trace was partitioned into one hour intervals (aka binned trace) by counting the number of tweets present in each interval or "bin". This binned trace was then filtered through a K-means clustering algorithm, where we set K=5 and thus obtained five clusters to which we assigned integer observation values for our discrete time series (aka observation trace). Each data point in the observation trace is an integer between one and five (inclusive). The Twitter traces all have length 3000 (i.e. user observed for 3000 hours) and are input (in various groups) to the MultiHMM, where iterative training, using the MultiBWA, results in model-parameter convergence (i.e. A, B, π converge). The MultiHMM, using its

fixed parameters, can proceed to generate synthetic traces on all types of user groups involved in the training. Therefore, we obtain synthetic Twitter traces, which are simulated 1000 times using random generation sampling. In fact, the MultiHMM uses its own distribution of user Twitter data, defined by mean and standard deviation, and 95% confidence intervals are determined on both datasets. We compare our MultiHMM-generated results with mean and standard deviation for raw and (standard) HMM-generated traces; the HMM-generated trace is a result of a traditional HMM trained on an observation trace of length 3000.

A. Mean, Standard Deviation and Skewness

We calculated statistics on discrete traces of User Tweets (original and synthetic) in two formats: First, each trace corresponds to one user; results are presented in Tables I, II and III; Secondly, each trace represents the activity of large groups of users, which is summarized in Table IV (3134 Users), Table V (17594 Users) and Table VI (42729 Users). Note, the same MultiHMM is used to generate each trace (one-to-many relationship), whereas, for the standard model, a new HMM produces only one trace. This is an obvious advantage of the MultiHMM over the standard HMM.

TABLE I: User 1 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	1.0	1.54	1.54
HMM	1.0 ± 0.007	1.53 ± 0.004	1.56 ± 0.011
MultiHMM	0.99 ± 0.002	1.54 ± 0.002	1.56 ± 0.004

TABLE II: User 2 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	1.0	1.56	1.59
HMM	1.0 ± 0.007	1.56 ± 0.004	1.61 ± 0.01
MultiHMM	0.92 ± 0.002	1.58 ± 0.002	1.66 ± 0.004

TABLE III: User 3 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	1.0	1.6	1.62
HMM	1.0 ± 0.007	1.6 ± 0.004	1.64 ± 0.011
MultiHMM	0.97 ± 0.002	1.61 ± 0.002	1.65 ± 0.004

TABLE IV: Group 1 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	3.13	1.9	0.84
HMM	3.14 ± 0.008	1.89 ± 0.02	0.85 ± 0.025
MultiHMM	3.35 ± 0.006	1.97 ± 0.002	0.59 ± 0.007

TABLE V: Group 2 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	17.59	7.5	0.59
HMM	17.56 ± 0.045	7.47 ± 0.014	0.61 ± 0.01
MultiHMM	17.64 ± 0.023	7.51 ± 0.007	0.58 ± 0.007

TABLE VI: Group 3 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	42.73	20.84	0.87
HMM	42.6 ± 0.163	20.68 ± 0.073	0.9 ± 0.019
MultiHMM	45.42 ± 0.066	21.83 ± 0.019	0.57 ± 0.007

B. Correlation

Correlation between users and groups of users is imperative for finding the social "intruder" and therefore provides a useful initial measurement for security in social networks. We define Pearson's Correlation Coefficient [5], as applied to a sample, as:

$$c = \frac{\sum_{t=1}^{N}(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{N}(x_t - \bar{x})^2}\sqrt{\sum_{t=1}^{N}(y_t - \bar{y})^2}} \quad (1)$$

where $\bar{x}$ and $\bar{y}$ are the means of observations $x_1, x_2, \ldots, x_N$ and $y_1, y_2, \ldots, y_N$, respectively.

Average Correlation Coefficients were generated (after 1000 runs) by pairing HMM and MultiHMM-generated user traces with the original raw traces. Table VII presents statistics for individual users.

TABLE VII: Average Correlation Coefficients for Twitter Users on HMM and MultiHMM-generated traces after 1000 simulations

Trace	User 1	User 2	User 3
HMM	0.4555	0.9356	0.9785
MultiHMM	0.5865	0.9931	0.9992

Analyzing the pairwise correlation between users can be beneficial in terms of finding trends in their "online relationships." This could be interpreted as how often a pair of users tweet each other, whether they are online at similar times, etc. Information on pairwise user correlation is summarized in Tables VIII and IX.

TABLE VIII: Pairwise Correlation Coefficients for Three Twitter Users using MultiHMM

User	1	2	3	4
1	1.0	0.233	-0.306	0.416
2	-	1.0	0.315	0.456
3	-	-	1.0	0.232
4	-	-	-	1.0

TABLE IX: Pairwise Correlation Coefficients for Five Twitter Users using MultiHMM

User	1	2	3	4	5
1	1.0	0.197	0.174	-0.011	0.274
2	-	1.0	0.206	0.124	0.148
3	-	-	1.0	0.208	0.065
4	-	-	-	1.0	-0.013
5	-	-	-	-	1.0

C. Autocorrelation

Autocorrelation is a computational method for comparing two time series, where the second series is a lagged version of the first time series over a number of time periods. Hence, another name for autocorrelation is *lagged correlation*. The autocorrelation between a time series and its lagged version is therefore a number between -1 and $+1$. If the result is -1 then we have a perfect negative correlation, and if it is $+1$ then there exists a perfect positive correlation – i.e. the time series are identical. As autocorrelation is just normalised *autocovariance*, the two terms are, unfortunately, sometimes used interchangeably in industry.

The autocorrelation function (ACF) for observations $y_1, y_2, \ldots, y_N$ (with mean $\bar{y}$) is defined as:

$$p_k = \frac{\sum_{t=1}^{N-k}(y_t - \bar{y})(y_{t+k} - \bar{y})}{\sum_{t=1}^{N}(y_t - \bar{y})^2} \tag{2}$$

One of the main purposes of the ACF is to find trends or cycles in the underlying time series. We apply the ACF (as defined above) to the raw, unclustered traces and then to corresponding MultiHMM-generated traces.

The group of graphs (Figs. 2, 3, and 4) have ACFs for a group of 3134 users with a lag of up to 500. There is little autocorrelation observed in the raw trace and this is matched by the HMM and MultiHMM, with ACF fluctuating between values of 0.1 and -0.05.

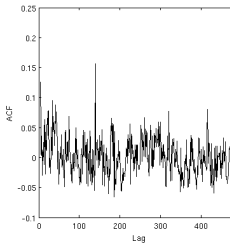

Fig. 2: Raw

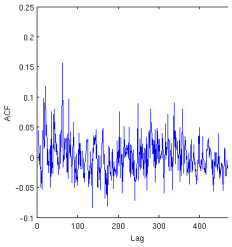

Fig. 3: HMM

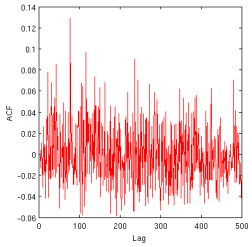

Fig. 4: MultiHMM

D. Convergence of BWA

The order of convergence of the BWA is $O(T^2N)$, where T is the trace length and N is the number of hidden states. We performed a simulation to analyse the computational efficiency of the BWA (trained on one trace) compared to that of the MultiBWA (multiple traces trained at once). Fig. 5 summarizes these findings, displaying the number of iterative BWA steps against the logarithm of the error (i.e. the maximum error between the state transition matrix entries at each step). Generally, the number of steps required to train the MultiHMM on H traces simultaneously (through the MultiBWA) has order $O(T^2N)$, compared to $O(T^2NH)$ for the standard HMM.

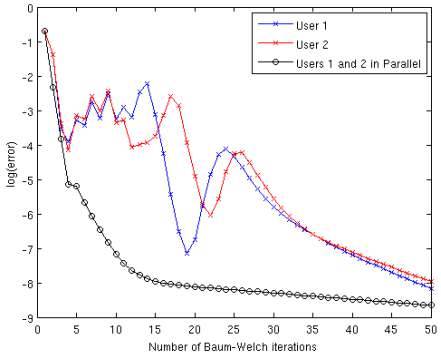

Fig. 5: log(error) vs. number of BWA and MultiBWA iterations for different user traces

Fig. 5 reveals the difference in rates of convergence of the standard BWA compared to its Multi-User, adaptive counterpart. The graph shows that both models effectively eliminate the error of the parameter updates (difference between a_{ij} and a'_{ij} entries). This proves, from a convergence point of view, that the MultiBWA is more efficient than the BWA for training on multiple traces. In the next section, we discuss the advantages of MultiHMM in terms of burstiness, which, we will see, is supported by our results.

E. Burstiness

We measure trace burstiness for the HMM and MultiHMM in relation to individual users and groups of users. Figs. 6, 7, and 8 present Twitter activity for one user using Clustered, HMM, and MultiHMM traces, respectively. It seems the HMM has large periods of no user activity (i.e. no tweets), unlike the Clustered and MultiHMM traces, which have similar and less frequent absences in tweets. Similarly, we analyse burstiness for one large group of users (i.e. 17594 users) in Figs. 9, 10, and 11. It is quite clear that the Clustered trace is best matched by the MultiHMM; both show two instances of very high activity (or extreme "bursts"), one at the beginning of the trace and one near the end. The HMM also shows signs of high spikes in user activity, but randomly throughout the trace, and thus fails to capture the extreme bursty behaviour exhibited by the Clustered and MultiHMM traces.

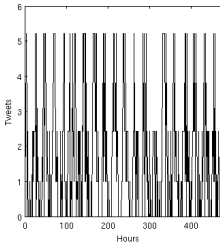

Fig. 6: Clustered

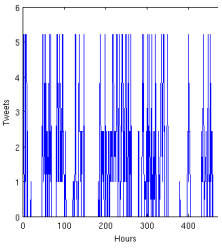

Fig. 7: HMM

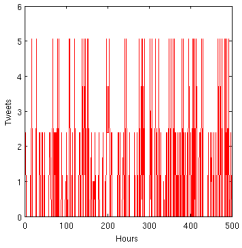

Fig. 8: MultiHMM

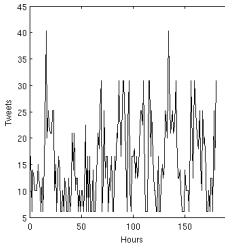

Fig. 9: Clustered

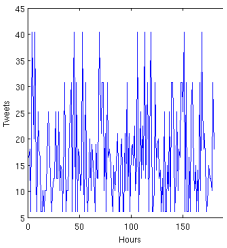

Fig. 10: HMM

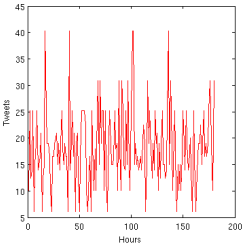

Fig. 11: MultiHMM

F. MAPE

Mean Absolute Percentage Error (MAPE) is a measure of accuracy for time series models that helps validate the synthetic MultiHMM-generation of Twitter traces against raw data. The value of MAPE is given by:

$$m = \frac{1}{N} \sum_{t=1}^{N} \frac{|x_t - y_t|}{x_t} \tag{3}$$

where x_t is the actual value and y_t is the forecast value.

Simulations of BWA and MultiBWA were executed 1000 times, and MAPE values were produced by comparing raw with HMM-generated traces and also raw with MultiHMM traces. Like before, we separate statistics for individual users (Table X) and groups of many users (Table XI).

TABLE X: MAPE values for Twitter Users on HMM and MultiHMM-generated traces after 1000 simulations

Trace	User 1	User 2	User 3
HMM	0.4179	0.4889	0.4736
MultiHMM	0.4410	0.3659	0.4237

TABLE XI: MAPE values for Twitter Groups on HMM and MultiHMM-generated traces after 1000 simulations

Trace	Group 1	Group 2	Group 3
HMM	0.6471	0.7446	1.3442
MultiHMM	0.2570	0.2854	0.2530

G. Cumulative Distribution Function

For the raw data, HMM and MultiHMM, the Cumulative Distribution Function (CDF), based on the cumulative relative frequency histogram and underlying Gaussian distributions (obtained from the sample mean and variance), respectively, were plotted for the tweeting activity of 17594 users (Fig. 12) spanning 1000 hours. One can clearly infer, for example, the probability of the number of tweets being between 17 and 18 for raw, HMM and MultiHMM models. The MultiHMM distribution seems to match the raw CDF more closely in terms of shape than the HMM CDF.

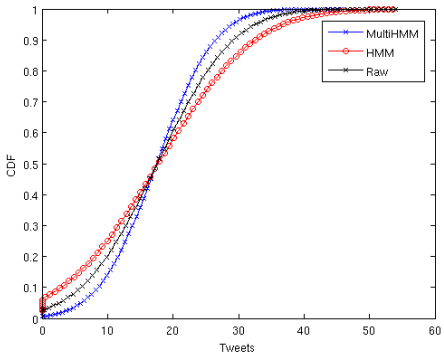

Fig. 12: CDFs for 17594 Twitter users for raw, HMM and MultiHMM fitting

H. Advantages of MultiHMM

Based on the above results, there is statistical evidence that the accuracy of the MultiHMM is superior to that of the standard HMM, in terms of synthetic generation of user traces. The number of steps to convergence of the MultiHMM on many traces is significantly less than training the standard BWA for each user individually. A list summarizing some of the advantages of the MultiHMM over other stochastic models is as follows:

1) Very few parameters are needed to set up the MultiHMM (i.e. A, B, π), compared to the heavy parameterization in [11].
2) Periodic behaviour for MultiHMM is not fixed to one length of time, which was the case in [11], where the inter-session rate of the Poisson process was set to one week.
3) Other models (e.g. Priority Queueing models) fail to account for cycles and sessions of high activity (i.e. burstiness), which the MultiHMM faithfully replicates for large groups of users.
4) MultiHMM provides inference into user behaviour through hidden states, and more so than trivial labelling of states as “Passive” or “Active” [11], [19].
5) MultiHMM aims to characterize groups of users with common features and identify “intruders”, rather than prioritising a high volume of users over model features (as in [11]).
6) The Cascading Poisson process [24] is too computationally intensive in comparison to the MultiHMM.

7) MultiHMM saves steps in training and convergence of its MultiBWA, whereas standard HMMs are ill-suited to situations where multiple processes interact.
8) The CoupledHMM of [19] is computationally expensive, relying on increased coupling between Markov chains to represent social influence amongst users. In fact, the MultiHMM acts as a basic hierarchical "Social Influence" driven model for groups of users, thus extending [19].

VI. Conclusion

In user classification, *Individual Parameter Estimates* fluctuate less over time than they do across individuals. Therefore, individual attributes are quite persistent, and can be good candidates for characterizing users. This assumption has been tested in our Multi-User classification model. It has already been established that HMMs, combined with the supporting clustering analysis and appropriate choice of bins, are able to provide a concise, parsimonious and portable synthetic workload [9]. However, the deficiency of such models is their heavy computing resource requirement, which essentially precludes them from any form of parallel or Multi-User analysis. The MultiHMM we have developed has a vastly reduced computing requirement making it ideal for modeling workload data in real-time, whilst at the same time providing excellent accuracy compared with both the resource-costly traditional HMM and the training traces themselves. Validation of the MultiHMM, in terms of means, standard deviation, and skewness, has proved a simple method for verifying average "bin" probabilities of storage workloads, in simple terms. ACFs have also validated the dynamics of the generated workload traces, focusing on the inter-bin correlation. Additionally, burstiness in a network of users has been replicated by the MultiHMM for extended periods of time.

Such mathematical descriptions of workload must ultimately be assessed quantitatively against independent data (i.e. traces not used in model construction) that they purport to represent, and more extensive tests are planned for our Multi-User model. Nonetheless, the specialized algorithm used in our MultiHMM has been successful after statistical comparisons between raw and model-generated traces. Our model has improved current Markovian temporal models, and we list some potential applications of the MultiHMM:

1) Spam detection by recognizing "fake" users.
2) Model behaviour of groups of users on trending topics.
3) Efficient online resource allocation, by exploiting the highs and lows of user burstiness.
4) Analysis of multiple sessions of users online, to understand the dynamic behaviour of Twitter groups.
5) User classification for security purposes (i.e. Normal, Aggressive, Intruder, etc.)

Extensions to our model include using hierarchical clustering to improve the cluster allocation in different user traces. This will give a better choice for the number of clusters and improve the accuracy of model-generated trace distributions and dynamics. Also, we plan to adapt the MultiHMM training algorithm to use varying weights for each trace to represent priorities in streams of users. Another possible extension is to look specifically at retweets for very popular tweets (i.e. millions of users retweeting celebrities). This might provide an initial prediction of "super busy times" in a network and thus help in resource allocation.

References

[1] L. E. Baum, T. Petrie: Stastical Inference for Probabilistic Functions of Finite Markov Chains, In *The Annals of Mathematical Statistics*, **37**, p. 1554-63 (1966)

[2] L. E. Baum, T. Petrie, G. Soules, N. Weiss: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, In *The Annals of Mathematical Statistics*, **41**, p. 164-171 (1970)

[3] A. J. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, In *IEEE Transactions on Information Theory*, **13**, p. 260-269 (1967)

[4] C. Aykanat, A. Turk, O. Selvitopi, H. Ferhatosmanoglu: Temporal Workload-Aware Replicated Partitioning for Social Networks, In *IEEE Transactions on Knowledge and Data Engineering*, p. 1-14 (2014)

[5] K. Pearson: Notes on regression and inheritance in the case of two parents, Proceedings of the Royal Society of London, **58**, p. 240-242 (1895)

[6] Z. Ghahramani, M. Jordan: Factorial hidden Markov models, In *Machine Learning*, **29**, p. 245-273 (2012)

[7] C. Vogler, D. Metaxas: Parallel Hidden Markov Models for American Sign Language Recognition, Proceedings of the International Conference on Computer Vision, Kerkyra, Greece (1999)

[8] J. Ashraf, N. Iqbal, N. S. Khattak, A. M. Zaidi: Speaker Independent Urdu Speech Recognition Using HMM (2010)

[9] P. G. Harrison, S. K. Harrison, N. M. Patel, S. Zertal: Storage Workload Modeling by Hidden Markov Models: Application to Flash Memory, In: *Performance Evaluation*, **69**, p. 17-40 (2012)

[10] A. Krough, M. Brown, S. Mian, K. Sjolander, D. Haussler: Hidden Markov Models in Computational Biology, In: *Journal of Molecular Biology*, p. 1501-1531, Computer and Information Sciences, University of California, Santa Cruz, USA (1994)

[11] R. D. Malmgren, J. M. Hofman, L. A. N. Amaral, D. Watts: Characterizing Individual Communication Patterns, Proceedings of the International Conference on Knowledge Discovery and Data Mining, Paris, France (2009)

[12] C. Burge, S. Karlin: Prediction of complete gene structures in human genomic DNA, In: *Journal of Molecular Biology*, p. 78-94, Computer and Information Sciences, Stanford University, California, USA (1997)

[13] J. Domanska, A. Domanski, T. Czachorski: A HMM Network Traffic Model, Proceedings of the Internation Conference on Networking and Future Internet, p. 17-20 (2012)

[14] N. Oliver, A. Garg, E. Horvitz: Layered Representations for Learning and Inferring Office Activity from Multiple Sensory Channels, In *Computer Vision and Image Understanding*, **96**, p. 163-180 (2004)

[15] L. R. Rabiner, B. H. Juang: An Introduction to Hidden Markov Models, In *IEEE ASSP Magazine*, **3**, p. 4-16 (1986)

[16] L. R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, In *IEEE*, **77**, p. 257-286 (1989)

[17] C. Liu: cuHMM: a CUDA Implementation of Hidden Markov Model Training and Classification. Online: http://liuchuan.org/pub/cuHMM.pdf (2006)

[18] S. Hymel: Massively Parallel Hidden Markov Models for Wireless Applications, MSc Thesis, Virginia Polytechnic Institute and State University (2011)

[19] V. Raghavan, G. Steeg, A. Galstyan, A. Tartakovsky: Coupled Hidden Markov Models For User Activity In Social Networks, Proceedings of the IEEE International Conference on Multimedia and Expo Workshops, p. 1-6 (2013)

[20] J. Domanska, A. Domanski, T. Czachorski: Internet Traffic Source based on Hidden Markov Model, Proceedings of the International Conference on Smart Spaces and Next Generation Wired/Wireless Networking, Lecture Notes in Computer Science, Springer, **6869**, p. 395-404 (2011)

[21] T. Chis: Hidden Markov Models: Applications to Flash Memory Data and Hospital Arrival Times, Department of Computing, Imperial College London (2011)

[22] The Cooperative Association for Internet Data Analysis: http://www.caida.org.

[23] K. Starbird, J. Maddock, M. Orand, P. Achterman, R. M. Mason: Rumors, False Flags, and Digital Vigilantes: Misinformation on Twitter after the 2013 Boston Marathon Bombing, Proceedings of iConference, iSchools, p. 654-662 (2014)

[24] R. D. Malmgren, D. B. Stouffer, A. E. Motter, L. A. N. Amaral: A Poissonian explanation for heavy tails in e-mail communication, Proceedings of the National Academy of Sciences of the USA, **105**, p. 18135-18158 (2008)

APPENDIX

Here, important results are summarized to compliment the research in this paper. First, to support the Background section, we define the Forward-Backward and Baum-Welch algorithms. Secondly, statistical results are presented for a group of five Twitter users.

A. Forward-Backward Algorithm

The Forward-Backward Algorithm (FBA) solves the following problem: Given the observation sequence $O = (O_1, O_2, \ldots, O_T)$ and the model $\lambda = (A, B, \pi)$, calculate $P(O; \lambda)$ (i.e. the probability of O given the model), and thus determine the likelihood of O. The parameters of λ are defined as follows: A is the state transition matrix and contains probabilities for moving from one state to another; B is the observation matrix and gives emission probabilities for each state emitting each observation; π is the initial hidden state distribution. Based on the solution in [15], we present the "Forward" part of the algorithm, which is the α-pass, followed by the "Backward" part or the β-pass. We define the forward variable $\alpha_t(i)$ as the probability of the observation sequence up to time t and of state q_i at time t, given our model λ. In other words, $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = q_i; \lambda)$, where $i = 1, 2, \ldots, N$ (where N is the number of states), $t = 1, 2, \ldots, T$, (where T is the number of observations) and s_t is the state at time t. The solution for $\alpha_t(i)$ (for $i = 1, 2, \ldots, N$) is initially given by $\alpha_1(i) = \pi_i b_i(O_1)$ and defined recursively (for $t = 2, 3, \ldots, T$) as follows:

$$\alpha_t(i) = \left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji} \right] b_i(O_t) \tag{4}$$

where $\alpha_{t-1}(j)a_{ji}$ is the probability of the joint event that $O_1, O_2, \ldots O_{t-1}$ are observed (given by $\alpha_{t-1}(j)$) and there is a transition from state q_j at time $t-1$ to state q_i at time t (given by a_{ji}); $b_i(O_t)$ is the probability that O_t is observed from state q_i.

Similarly, we can define the backward variable $\beta_t(i)$ as the probability of the observation sequence from time $t+1$ to the end, given state q_i at time t and the model λ. Then, $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots O_T; s_t = q_i, \lambda)$. The solution for $\beta_t(i)$ (for $i = 1, 2, \ldots, N$) is initially given by $\beta_T(i) = 1$ and defined recursively (for $t = T-1, T-2, \ldots, 1$) as follows:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{5}$$

where we note that the observation O_{t+1} can be generated from any state q_j.

With the α and β values now computed, the equations defining the Baum-Welch algorithm are described in the next section.

B. Baum-Welch Algorithm

Given the model $\lambda = (A, B, \pi)$, the Baum-Welch algorithm (BWA) trains a HMM on a fixed set of observations $O = (O_1, O_2, \ldots, O_T)$. By adjusting its parameters A, B, π, the BWA aims to maximise $P(O \mid \lambda)$. As explained in Section 2.3.2 of [21], the parameters of the BWA are updated iteratively by the following formulas (for $i, j = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, T-1$) as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O; \lambda)} \tag{6}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{7}$$

$$A = a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)} \tag{8}$$

$$B = b_j(k)' = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{9}$$

$$\pi'_i = \gamma_1(i) \tag{10}$$

We can now re-estimate our model-parameters iteratively using $\lambda' = (A', B', \pi')$, where $A' = \{a'_{ij}\}$, $B' = \{b_j(k)'\}$ and $\pi' = \{\pi'_i\}$, as defined in Eqs. (8), (9) and (10), respectively.

C. Extensive Results

This section summarizes some more extensive results on Twitter users, where synthetic workload traces were generated. These results support the work presented earlier in the paper, justifying the applicability of the MultiHMM to a wide range of Multi-User traces. Five Twitter users (three followers of each other, and two non-followers) had their tweets recorded into timestamped traces. These were processed using our MultiHMM Algorithm 1 and once converged, the model generated five synthetic traces, producing sets of statistics. We present these statistics in Tables XII, XIII, XIV, XV, XVI. From these results, one can deduce the non-followers as User 2 (Table XIII) and User 3 (Table XIV).

TABLE XII: User 1 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	0.47	0.63	0.75
HMM	0.48 $\pm$ 0.003	0.64 $\pm$ 0.002	0.74 $\pm$ 0.001
MultiHMM	0.51 $\pm$ 0.001	0.64 $\pm$ 0.0003	0.63 $\pm$ 0.003

TABLE XIII: User 2 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	1.38	1.39	0.85
HMM	1.37 $\pm$ 0.003	1.4 $\pm$ 0.002	0.86 $\pm$ 0.002
MultiHMM	1.3 $\pm$ 0.002	1.35 $\pm$ 0.001	0.98 $\pm$ 0.003

TABLE XIV: User 3 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	2.74	2.72	0.65
HMM	2.75 $\pm$ 0.002	2.71 $\pm$ 0.002	0.64 $\pm$ 0.002
MultiHMM	2.84 $\pm$ 0.003	2.4 $\pm$ 0.002	0.61 $\pm$ 0.002

TABLE XV: User 4 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	0.68	0.82	1.47
HMM	0.67 $\pm$ 0.003	0.81 $\pm$ 0.002	1.48 $\pm$ 0.002
MultiHMM	0.67 $\pm$ 0.001	0.78 $\pm$ 0.001	1.56 $\pm$ 0.001

TABLE XVI: User 5 Statistics on the raw, HMM and MultiHMM-generated traces

Trace	Mean	Std Dev	Skewness
Raw	0.49	1.1	2.18
HMM	0.5 $\pm$ 0.007	1.09 $\pm$ 0.004	2.17 $\pm$ 0.01
MultiHMM	0.52 $\pm$ 0.001	0.92 $\pm$ 0.001	2.01 $\pm$ 0.004