2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems

Modeling Multi-User Behaviour in Social Networks

Tiberiu Chis, Peter G. Harrison

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

***Abstract*—Social networks, and the behaviour of groups of online users, are popular topics in modeling and classifiyng Internet traffic data. There is a need to analyze online network performance metrics through suitable workload benchmarks. We address this issue with a Multi-dimensional Hidden Markov Model (MultiHMM) to act as a Multi-User workload classifier. The MultiHMM is an adaptation of the original HMM, using clustering methods and multiple trace-training for the Baum-Welch algorithm. The goals of the MultiHMM are to classify multiple online user streams with minimal processing needs, represent burstiness and correlation among groups of users and to improve security measures in the social network. Experiments are 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 and autocorrelation, and we discuss applications and extensions of our model.**

I. Introduction

In the last decade, leading social media companies, such as Twitter and Facebook, have grown to manage enormous online user-populations, which has led to an explosion of stored multimedia content. Thus, the individual and collective behaviour of the ever-increasing user-base is a popular research topic and, consequently, properties of online social interactions are being modeled. For example, researchers have used Twitter data to examine links within tweets and determine characteristic patterns that help define misinformed events [4]. Such work [4] 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 are used to analyse Traffic Burstiness from Internet Packet-level Sources [13] and variations of HMMs have identified 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 user's neighbours in the 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. Another example is HMMs used for individual email communication [11], where classifying characteristics of different groups was impossible for one model.

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. We propose a model that overcomes this issue, using a hybrid HMM to classify Multi-User temporal acitivity, 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). We validate this DMMAP[1] by comparing quantitatively means, standard deviations, skewness and autocorrelation from the raw (i.e. unclustered), HMM 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 converge in the Baum-Welch algorithm component. We conclude with 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, ideas for enhancements to the MultiHMM are proposed as future research and the Appendix summarizes background information on HMMs.

II. Related Work

We briefly introduce some existing models and identify their limitations. The focus is on related research involving Hypergraphs [5] and variations of HMMs for parallel training.

A. Hypergraphs

In social networks, studies show that Multi-User Queries result in multiple operations [5] and are expensive in terms of performance. To model these Multi-User interactions, one solution of Selective Replicated Partitioning was implemented through a Temporal Activity Hypergraph Model [5], where vertices represented users and nets corresponded to Multi-User Queries. Further, Twitter clone (hosted on Amazon EC2) provided a realistic experimental test bed [5]. The Hypergraph Model performs simultaneous partitioning and

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

1526-7539/14 $31.00 © 2014 IEEE
DOI 10.1109/MASCOTS.2014.29

replication to reduce 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 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 number of users and their interactions. Also, the proposed model has many parameters [5]. The MultiHMM 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 to 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 [7], 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], where the output is separate and each hidden state emits an observation given a 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 more computationally expensive; a potential problem introduced 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 [8], 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 requirement needed for the two models, scaling to higher numbers of models (i.e. to train on larger vocabularies) would not be as efficient. We propose a robust MultiHMM solution to optimize training 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 [9], [16] and Genome Sequence Prediction [12]; more recently, they have represented Storage Workloads [10], [20] and have modeled Hospital Patient Arrivals [20] as well as Social Network Interactions [19]. In all cases, these models have employed the well-known statistical algorithms to solve three fundamental problems credited to HMMs. These problems are: First, find $P(O;\lambda)$, the probability of observing O given the model λ, using the Forward-Backward algorithm [1]; Secondly, maximize $P(O;\lambda)$ by adjusting the parameters of model λ using the Baum-Welch algorithm [2]; Thirdly, 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. (7), only work on a single trace of observations and a useful upgrade for the BWA is to handle multiple stream analysis for characterizing users. Therefore, we adapt the FBA and BWA to model Multi-User processes using clustering techniques, forming the MultiHMM.

IV. MultiHMM

The novel contribution of this paper, namely a hybrid HMM for Multi-User training, comprises a 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 are trace means, standard deviation, skewness, autocorrelation and Mean Absolute Percentage Error (MAPE). We present our clustering methodology and specialized algorithm in detail.

A. Clustering Methodology

Clustering is used in pre-processing of input traces for training the BWA. One method involves K-means clustering to group the data points from H traces (i.e. into H-tuples) into K distinct clusters [20]. Each trace has data points belonging to one of C categories, which produces many possible combinations for the H-tuple. When H is large, this leads to unacceptably high values for K (i.e. H^C). Therefore, we propose our own clustering method, using K-means, which reduces H traces to K: we choose a value for C by inspection, and set K=C. If K<H, reduce H to K by grouping together data points from the same cluster; if H≤K, grouping points is omitted. This extra clustering reduces the computational burden further, before the BWA trains on the doubly-clustered traces. We assign weights to all traces and run 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 (ω_k) 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. This model now contains Multi-User information, and traces generated synthetically can be compared to invidual 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.

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
  for $j = 1 : K$ **do**
    $Group_j$ = {}
    for $t = 1 : size$, $i = 1 : H$ **do**
      if Point$_i(t) \in Cluster_j$ **then**
        $Group_j(t) \leftarrow$ Point$_i(t)$
      end if
    end for
  end for
  while MultiBWA Parameters not Converged **do**
    for $k = 1 : K$ Observation Traces **do**
      $\hat{\alpha} = \sum_{i=1}^{N} \omega_k \alpha_i$; $\hat{\beta} = \sum_{i=1}^{N} \omega_k \beta_i$; $\hat{\xi} = \sum_{i=1}^{N} \omega_k \xi_i$
    end for
    $\gamma_t(i) = \sum_{j=1}^{N} \hat{\xi}_t(i,j)$; $\pi'_i = \gamma_1(i)$
    $a'_{ij} = \frac{\sum_{t=1}^{T-1} \hat{\xi}_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \hat{\xi}_t(i,j)}$; $b_j(k)' = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$
  end while

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, where 95% confidence intervals are determined on both datasets. We compare our MultiHMM-generated results with mean, standard deviation and skewness 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 Traces: Raw, HMM and MultiHMM

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

TABLE II: User 2 Traces: Raw, HMM and MultiHMM

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 III: User 3 Traces: Raw, HMM and MultiHMM

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

TABLE IV: Group 1 Traces: Raw, HMM and MultiHMM

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

TABLE V: Group 2 Traces: Raw, HMM and MultiHMM

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

TABLE VI: Group 3 Traces: Raw, HMM and MultiHMM

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

B. Correlation

Correlation between users and groups of users, imperative for finding the social "intruder," provides a useful initial measurement for security in social networks. We define Pearson's Correlation Coefficient [6], 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 are generated (after 1000 runs) by pairing HMM and MultiHMM-generated traces with raw traces. Table VII presents statistics for individual users.

TABLE VII: Average Correlation Coefficients for Twitter Users: HMM and MultiHMM-generated traces

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

Analyzing pairwise correlation between users is 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 Table VIII.

TABLE VIII: Pairwise Correlation Coefficients for Four 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

C. Autocorrelation

Autocorrelation is a computational method for comparing two time series, where the second series is a lagged version of the first over a number of time periods. 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 follows:

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

One of the main purposes of the ACF is to find trends or cycles in the autocorrelated time series. We apply the ACF (as defined above) to the raw, unclustered traces and then to the corresponding MultiHMM-generated traces. The group of graphs (Figs. 1, 2, and 3) have ACFs for a group of 3134 users with a lag of 500. There is little autocorrelation observed in the raw trace and matched by the HMM and MultiHMM, with ACF fluctuating between values of 0.15 and −0.06.

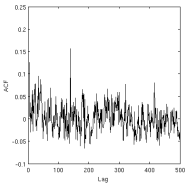

Fig. 1: Raw

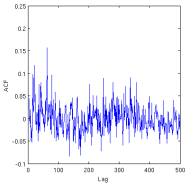

Fig. 2: HMM

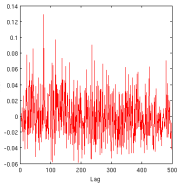

Fig. 3: MultiHMM

D. Burstiness

We measure trace burstiness for the HMM and MultiHMM in relation to individual users and groups of users. Figs. 4, 5, and 6 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.

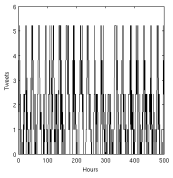

Fig. 4: Clustered

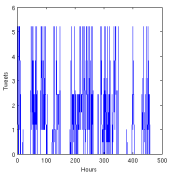

Fig. 5: HMM

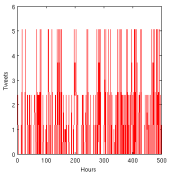

Fig. 6: MultiHMM

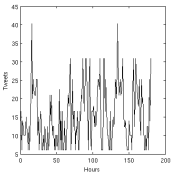

Fig. 7: Clustered

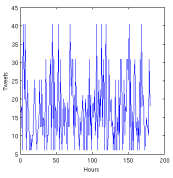

Fig. 8: HMM

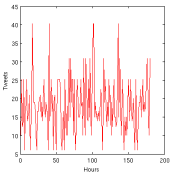

Fig. 9: MultiHMM

Similarly, we analyse burstiness for one large group of users (i.e. 17594 users) in Figs. 7, 8, and 9. The Clustered trace is best matched by the MultiHMM; both show two instances of very high activity (or extreme "bursts"), at the beginning of the trace and 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.

E. MAPE

Mean Absolute Percentage Error (MAPE) is 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:

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

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 IX) and groups of many users (Table X).

TABLE IX: 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 X: 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

F. Convergence of BWA

The order of convergence of the BWA is given by $O(T^2N)$, where T is the trace length and N is the number of hidden states. We perform 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. 10 plots 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) and is evidence that MultiHMM training on many traces simultaneously is more efficient than training the standard BWA for each user individually. 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.

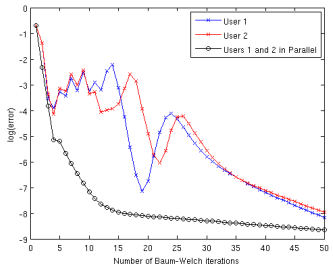

Fig. 10: log(error) vs number of BWA and MultiBWA iterations for different user traces

G. Advantages of MultiHMM

Based on results presented in this paper, 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. We list some of the advantages of the MultiHMM over other stochastic models:

1) Very few parameters for MultiHMM setup (i.e. A, B, π), compared to heavy parameterization in [11].
2) Periodic behaviour for MultiHMM not fixed to time period, which was the case in [11], where the inter-session rate of Poisson process was set to one week.
3) Other models (i.e. 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 characterizes group features of users and identifies "intruders," rather than prioritizing a high volume of users over model (as in [11]).
6) MultiHMM saves steps in training and convergence of its MultiBWA, whereas standard HMMs are ill-suited in situations where multiple processes interact.
7) The CoupledHMM of [19] is computationally expensive, relying on increased coupling between Markov chains to represent social influence amongst users. Our MultiHMM extends [19], acting as a basic "Social Influence" driven model for groups of users.

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 [10]. 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 straightforward 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. The MultiHMM has improved current Markovian temporal models, with some useful potential applications: *spam detection* by recognizing "fake" users; modeling *group behaviour* of users on trending topics; efficient *online resource allocation*, by exploiting the highs and lows of user burstiness; user classification for *security* (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 could predict "super busy times" in the 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] 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, In *Proc. iConference*, p. 654-662 (2014)

[5] 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)

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

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

[8] C. Vogler, D. Metaxas: Parallel Hidden Markov Models for American Sign Language Recognition, In *Proc. ICCV*, Kerkyra, Greece (1999)

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

[10] 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)

[11] R. D. Malmgren, J. M. Hofman, L. A. N. Amaral, D. Watts: Characterizing Individual Communication Patterns, In *Proc. KDD*, 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 (1997)

[13] J. Domanska, A. Domanski, T. Czachorski: A HMM Network Traffic Model, In *Proc. ICNFI*, 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 *Proc. 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, In *Proc. IEEE ICME*, p. 1-6 (2013)

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

APPENDIX

Here, important results are summarized to complement the research in this paper. To support the Background section, we define the Forward-Backward and Baum-Welch algorithms.

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 obtain the likelihood of O. The parameters of λ are: the state transition matrix (A), containing probabilities for moving from one state to another; the observation matrix (B), with probabilities for each state emitting an observation; the initial hidden state distribution (π). Based on the solution in [15], we present the "Forward" part of the algorithm (aka the α-pass), followed by the "Backward" part (β-pass). We define the forward variable $\alpha_t(i)$ as the probability of O up to time t and of state q_i at time t, given our model λ. So, $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = q_i; \lambda)$, where $i = 1, 2, \ldots, N$ (N is the number of states), $t = 1, 2, \ldots, T$ (T is the number of observations) and s_t is the state at time t. The solution of $\alpha_t(i)$ (for $i = 1, 2, \ldots, N$) is initially $\alpha_1(i) = \pi_i b_i(O_1)$ and defined recursively (for $t = 2, 3, \ldots, T$) as follows:

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

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 computed, the equations defining the Baum-Welch algorithm can be described.

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 maximizes $P(O; \lambda)$. As explained in Section 2.3.2 of [20], the parameters of the BWA are updated iteratively (for $i, j = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, T-1$) as follows:

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

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)};\ b_j(k)' = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)};\ \pi'_i = \gamma_1(i) \tag{7}$$

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 Eq. (7).