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 classifying 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 analyzed 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 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). We validate this DMMAP 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

1Note: MultiHMM and DMMAP will be used interchangeably in this paper
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 $\lambda$, using the Forward-Backward algorithm [1]; Secondly, maximize $P(O; \lambda)$ by adjusting the parameters of model $\lambda$ 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 $H \leq K$, 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 ($\omega_k$) 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 individual 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 $\land H$ Traces $\land$ size = Trace.length

for $i = 1 : H$ Raw Traces do
    while Cluster Points not Fixed do
        K-means Clustering on Tracei
    end while
end for

for $j = 1 : K$ do
    Groupj = \{\}
    for $t = 1 : $size, $i = 1 : H$ do
        if Pointi(t) $\in$ Clusterj then
            Groupj(t) $\leftarrow$ Pointi(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
    $\gamma(t) = \sum_{j=1}^{N} \hat{\xi}(i,j); \pi_i^t = \gamma_1(i)$
    $a_{ij}^t = \frac{\sum_{k=1}^{T} \hat{\xi}(i,j) \gamma(k)}{\sum_{k=1}^{T} \hat{\xi}(i,j) \gamma(j)}; b_j^t = \frac{\sum_{t=1}^{T} \gamma(t) \gamma(j)}{\sum_{t=1}^{T} \gamma(j)}$
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: The first simulation analyzes only three Twitter 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.

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.
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:

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

(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

<table>
<thead>
<tr>
<th>Trace</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>0.4555</td>
<td>0.9356</td>
<td>0.9785</td>
</tr>
<tr>
<td>MultiHMM</td>
<td>0.5865</td>
<td>0.9931</td>
<td>0.9992</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>User</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.233</td>
<td>-0.306</td>
<td>0.416</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>1.0</td>
<td>0.315</td>
<td>0.456</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>1.0</td>
<td>0.232</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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:

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

(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.

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.

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:

\[
m = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{x_t - y_t}{x_t} \right|
\]

(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

<table>
<thead>
<tr>
<th>Trace</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>0.4179</td>
<td>0.4889</td>
<td>0.4736</td>
</tr>
<tr>
<td>MultiHMM</td>
<td>0.4410</td>
<td>0.3659</td>
<td>0.4237</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Trace</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>0.6471</td>
<td>0.7446</td>
<td>1.3442</td>
</tr>
<tr>
<td>MultiHMM</td>
<td>0.2570</td>
<td>0.2854</td>
<td>0.2530</td>
</tr>
</tbody>
</table>

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
plots the number of iterative BWA steps against the logarithm
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](image1.png)

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. $A, B, \pi$),
   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 “So-
   cial Influence” driven model for groups of users.

VI. CONCLUSION

In user classification, Individual Parameter Estimates fluc-
tuate 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
by providing 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. Addition-
ally, burstiness in a network of users has been replicated
by the MultiHMM for extended periods of time.

Such mathematical descriptions of workload must ulti-
mately 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
workload identifies “intruders,” rather than prioritizing a high
volume of users over model (as in [11]).

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


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 $\lambda$ 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 ($\pi$). Based on the solution in [15], we present the “Forward” part of the algorithm (aka the $\alpha$-pass), followed by the “Backward” part ($\beta$-pass). We define the forward variable $\alpha_t(i)$ as the probability of $O$ up to time $t$ and of state $q_t$ at time $t$, given our model $\lambda$. 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 $\alpha_t(i) = \pi_i b_i(O_1)$ and defined recursively (for $t = 2, 3, \ldots, T$) as follows:

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij} b_i(O_t) \quad (4)$$

where $\alpha_{t-1}(j) a_{ij}$ 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 $a_{ij}$); $b_i(O_t)$ 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 $\lambda$. 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:

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

where we note that the observation $O_{t+1}$ can be generated from any state $q_j$. With the $\alpha$ and $\beta$ 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 $A, B, \pi$, 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:

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

$$a'_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i, j)}{\sum_{t=1}^{T} \xi_t(i, j) a_{ij}}; b(j) = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (7)$$

We can now re-estimate our model parameters iteratively using $\lambda' = (A', B', \pi')$, where $A' = \{a'_{ij}\}$, $B' = \{b(j)\}$ and $\pi' = \{\pi'_i\}$, as defined in Eq. (7).