Behavioral Sequence Modeling with Ensemble Learning

Maxime Kawawa-Beaudan
J.P. Morgan AI Research
383 Madison Avenue, New York NY USA
maxime.kawawa-beaudan@jpmorgan.com

Srijan Sood
J.P. Morgan AI Research
383 Madison Avenue, New York NY USA
srijan.sood@jpmorgan.com

Soham Palande
J.P. Morgan AI Research
383 Madison Avenue, New York NY USA
soham.palande@jpmorgan.com

Ganapathy Mani
J.P. Morgan AI Research
383 Madison Avenue, New York NY USA
ganapathy.mani@jpmorgan.com

Tucker Balch
Emory University
1300 Clifton Road NE, Atlanta GA USA
tucker.balch@emory.edu

Manuela Veloso
J.P. Morgan AI Research
383 Madison Avenue, New York NY USA
manuela.veloso@jpmorgan.com

Abstract

We investigate the use of sequence analysis for behavior modeling, emphasizing that sequential context often outweighs the value of aggregate features in understanding human behavior. We discuss framing common problems in fields like healthcare, finance, and e-commerce as sequence modeling tasks, and address challenges related to constructing coherent sequences from fragmented data and disentangling complex behavior patterns. We present a framework for sequence modeling using Ensembles of Hidden Markov Models, which are lightweight, interpretable, and efficient. Our ensemble-based scoring method enables robust comparison across sequences of different lengths and enhances performance in scenarios with imbalanced or scarce data. The framework scales in real-world scenarios, is compatible with downstream feature-based modeling, and is applicable in both supervised and unsupervised learning settings. We demonstrate the effectiveness of our method with results on a longitudinal human behavior dataset.

1 Introduction

Modeling human behavior is a complex task with applications spanning multiple domains such as user research, healthcare, payments, trading, and e-commerce. Applications range from classifying human activity (28), distinguishing humans from bots (16), detecting credit card fraud (25) etc. Behavior is often captured as sequences of actions or events over time, and understanding patterns within these sequences is crucial for tasks like classification, anomaly detection, and user modeling. A key challenge in utilizing the captured sequences is class imbalance, where underrepresented behavior profiles or a disproportionate number of anomalous examples hinder model generalization.

Many solutions leverage complex deep learning models (33), focusing on event-level classification with extensive feature engineering (6). However, such event-level or feature-aggregated methods frequently fall short in capturing the sequential dynamics essential for understanding and modeling behavior. These approaches are also susceptible to overfitting, with performance rapidly degrading in class-imbalanced scenarios.

NeurIPS 2024 Workshop on Behavioral Machine Learning.

Sequential context is crucial for effective behavior modeling. For example, in credit card fraud detection and anti-money laundering, a user's transaction history provides deeper insights into behavioral intent than isolated transactions or aggregated features. Yet, many datasets and approaches remain confined to the event level, overlooking the broader sequential context. In this study, we advocate for a sequence modeling approach to behavior modeling, particularly in scenarios where behaviors are represented as action sequences derived from unstructured data. Aggregating this data into coherent sequences that reflect an agent's decision-making process is a non-trivial challenge. Our method is tested in a real-world setting characterized by extreme class imbalance, with millions of diverse user behavior examples contrasted against only a few hundred instances of the anomalous class.

We propose a novel and lightweight ensemble-based framework for behavior modeling, and show efficacy on downstream (imbalanced) sequence classification tasks. While our approach is model-agnostic, we leverage Hidden Markov Models (HMMs) for their simplicity, interpratibility, and efficacy at capturing temporal dependencies and latent patterns. Our real-world deployment scales to millions of sequences, while being compatible with downstream machine learning methods. We demonstrate its effectiveness on a publicly available human behavior dataset (31).

Our contributions: (1) We introduce a behavior modeling framework based on sequences of events/actions, applicable to various domains; (2) We outline its application to supervised and unsupervised tasks; (3) We demonstrate its effectiveness on a longitudinal human behavior dataset.

2 Background & Related Work

HMMs Hidden Markov Models (HMMs) are statistical models for sequential data, which have a long history of use in natural language processing, finance, and bioinformatics (24; 5; 32; 22). HMMs have been used extensively for behavior modeling, including sensor surveillance (21), human-computer interfaces (11), and web user interactions (15), with recent applications in social media bot detection (19). While neural network-based approaches like CNNs, LSTMs, and Transformers have shown success in settings like sentiment analysis (13) and network intrusion detection (27), they face challenges such as high computational cost, overfitting, and reduced interpretability.

Resolution Event-level classification still dominates in areas like anti-money laundering and network security, where sequence-level labels are often missing (1; 8). This lends itself to aggregate feature based approaches, missing key historical context. While some work has tackled sequence modeling in network intrusion detection with favorable results (26), much remains to be done.

Data Imbalance and Anomaly Detection Many real-world problems, including intrusion detection (12), credit card fraud (25), and money laundering (2), involve detecting rare events and suffer from class imbalance. One-class anomaly detection focuses on robustly modeling the nominal class and identifying deviations (9), while more targeted approaches model both normal and anomalous sequences to detect specific behavioral anomalies (10).

3 Approach

Consider a sequence observation $\mathcal{O} = \{a_1, a_2, ..., a_T\}$, where each a_i is drawn from a discrete set of actions $\mathcal{A}$. Such sequences can represent various behaviors, such as user interactions in an app, trading actions in financial markets, or other human decision-making processes. Our goal is to model these behaviors, either discovering behavior clusters, or classifying behaviors when labels are available (e.g., online bot detection, credit card fraud detection, physical activity recognition (20; 33; 18).

3.1 Sequence Construction

One of the primary challenges lies not in modelling but in organizing coherent data streams from raw, fragmented data $\mathcal{D}$, which often contains interwoven behaviors from multiple agents/users. For instance, in trading, $\mathcal{D}$ may span billions of transactions across participants, assets, and exchanges, requiring grouping data streams by participant, and further by exchange or asset to capture specific behaviors. In network analysis, interactions between devices and servers can be grouped by source IP for individual user activity, or further by target IP to constitute specific behavior streams.

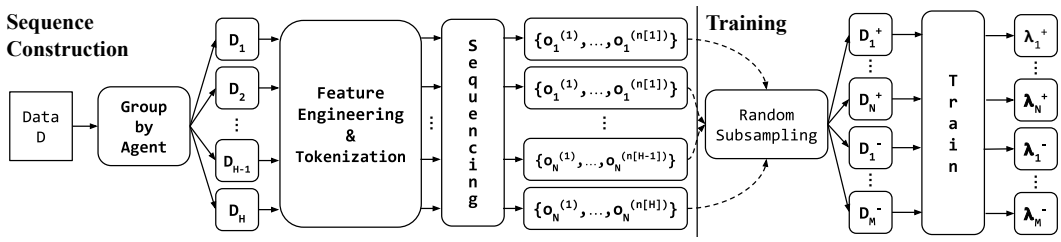

Figure 1: Flow diagram of our sequence construction approach, as detailed in Section 3.1. We disentangle the monolithic dataset $\mathcal{D}$ into data streams, then process these further into sets of observation sequences. Our subsequent HMM-*e* ensemble training approach is detailed in Section 3.2. While we adopt this approach using HMMs, the framework itself is model agnostic. The training data is broken into random subsets, and a diverse ensemble of learners is trained on these subsets.

As illustrated in Fig. 1, we begin by disentangling $\mathcal{D}$ into separate data streams $\mathcal{D}_1, ..., \mathcal{D}_H$, each corresponding to one of H agents. Feature engineering refines these data streams through dimensionality reduction, tokenization, or discretization, enhancing model generalization, particularly in the presence of imbalanced or sparse datasets.Continuous features can also be normalized and estimated directly, through techniques like Gaussian HMMs (23).

Once data is organized into streams $\mathcal{D}_h$, sequences of observations $\mathcal{O}_h^{(1)}, ..., \mathcal{O}_h^{(n[h])}$ are then extracted from each stream, with domain knowledge or sessions guiding the sequence span (start and end points). For example, web user behavior may span minutes to hours, whereas medical trial observations could extend over days or weeks. Breaks in continuous data streams often demarcate sequences, with shorter pauses treated as wait events and longer breaks as sequence endpoints. The number of sequences can vary significantly across agents, reflecting differing activity levels (e.g. power users vs intermittent monthly users).

3.2 Ensembles of Hidden Markov Models

Singleton HMMs First, considering binary sequence classification, we separate training data by class and train two indivudal HMMs: one positive class HMM λ^+, and one negative class HMM λ^-. Given an unseen sequence $\mathcal{O}$, the predicted class $c(\mathcal{O})$ is determined by comparing the likelihoods:

$$c(\mathcal{O}) = \mathbb{1}\{p(\mathcal{O} \mid \lambda^+) > p(\mathcal{O} \mid \lambda^-)\} \tag{1}$$

Variable Sequence Length HMMs excel at sequence analysis (5), but struggle when comparing sequences of varying length, as length influences likelihood computation exponentially. We address this through model-driven normalization, computing likelihoods for a given sequence across multiple models, and deriving a rank-based composite score (rather than comparing sequence likelhihoods).

HMM Ensembles HMMs, while lightweight and efficient, can struggle to capture the complexity of behaviors in training data when using a singular model (per class). Ensemble methods train multiple models on subsets of the data (7), enabling each learner to specialize on distinct patterns or behaviors, while collectively capturing the full data distribution. This results in a more robust approach, particularly in scenarios with data imbalance, where monolithic models skew towards modeling the majority class (or underfitting for class specific models) (17). We propose **HMM-*e***, an ensemble framework that computes composite scores from individual learners (14). While HMMs are effective for our case, this framework is model-agnostic and can incorporate other model classes such as neural networks, SVMs, or decision trees.

3.2.1 Formalization and Algorithmic Framework

First, we train N models $\{\lambda_1^+, ..., \lambda_N^+\}$ on the positive class and M models $\{\lambda_1^-, ..., \lambda_M^-\}$ on the negative class, taking care to ensure diversity among the models by training each on a randomly selected subset of samples from the training data. Each model sees $s\%$ of the training data in its relevant class. While we set $N = M$ for all settings, these parameters (M, N, s) can be established using typical hyperparameter optimization approaches. For any given sequence in the training data, the probability of not being selected for any model's random subset is $(1-s)^N$. The expected number

of unsampled sequences is the same, so it is important to select s and N to keep this proportion of the training data small.

For an unseen observation sequence $\mathcal{O}$, we compute its likelihood scores under all models: $\{p(\mathcal{O} \mid \lambda_1^+), ..., p(\mathcal{O} \mid \lambda_N^+)\}$ and $\{p(\mathcal{O} \mid \lambda_1^-), ..., p(\mathcal{O} \mid \lambda_M^-)\}$. We then compute a composite score:

$$s(\mathcal{O}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{1}\{p(\mathcal{O} \mid \lambda_i^+) > p(\mathcal{O} \mid \lambda_j^-)\} \tag{2}$$

The score $s(\mathcal{O})$ represents the pairwise comparisons where positive-class models assign a higher likelihood than negative-class models, taking values in $[0, N \times M]$. A low score indicates that the sequence is more likely under the negative-class models, and vice versa. As likelihoods across different sequence lengths are not directly compared, this composite score acts as an implicit normalization technique. N and M should be chosen such that the score range adequately distinguishes the classes.

For our HMM and HMM-*e* approaches, we use 3 states in each of our models, and converge on an ensemble size of 250 and a subset factor of 1%. We perform hyperparameter search for ensemble size, trying other values in [10, 50, 100, 500, 1000]. We settle on 250 for its good performance at a relatively low complexity.

Downstream Modeling using HMM-*e* Scores Given a corpus of sequences and corresponding scores $\{\mathcal{O}, s(\mathcal{O})\}$, we classify sequences using a threshold s_{thresh}: $c(\mathcal{O}_i) = \mathbb{1}\{s(\mathcal{O}_i) \geq s_{thresh}\}$. Alternatively, base learner likelihoods can serve as features for downstream classifiers. For each sequence $\mathcal{O}_i$, we define a feature vector

$$f_i = \left[p(\mathcal{O}_i|\lambda_1^+), \ldots, p(\mathcal{O}_i|\lambda_N^+), p(\mathcal{O}_i|\lambda_1^-), \ldots, p(\mathcal{O}_i|\lambda_M^-)\right] \tag{3}$$

To account for sequence length sensitivity, f_i is normalized by $||f_i||_2$. This technique utilizes HMMs as feature extractors, where each feature $p(\mathcal{O}_i|\lambda_j)$ represents the similarity between the sequence $\mathcal{O}_i$, and the random subset of training data underlying λ_j.

Clustering HMM-*e* Scores in Unsupervised Settings In label-free settings, behavior clustering can be achieved using unsupervised learning approaches. We train N models $\{\lambda_1, ..., \lambda_N\}$ on random $s\%$ data subsets and generate feature vectors of base learner likelihoods f_i (Section 3.2.1). Unsupervised clustering like K-Means can be applied to discover behavioral groups, dimensionality reduction (e.g., PCA) can be helpful when N is large.

Figure 2: UMAP embeddings of features f_i, as discussed in Section 3.2.1, from a 500-model ensemble. Colors correspond to clusters discovered via K-Means.

4 Experiments

Data

We evaluate our approach on the GLOBEM dataset (31), a longitudinal human behavior study featuring over 3,700 attributes from 497 participants across four years (2018-2021). The dataset includes survey, smartphone, and wearable data, with a focus on depression detection (the task we consider). Data includes mood assessments, step counts, location variability, and sleep metrics. We utilize the data standardization platform for reproducibility provided by the authors (30).

While GLOBEM includes thousands of features, we select just four features to employ, each evaluated daily: smartphone moving/static time ratio, total screen time (minutes), total sleep time (minutes), and total steps. We train Gaussian HMMs on these continuous features, depicted for three anonymous participants in 4. Our small feature set allows us to learn meaningful correlations and avoid converging

	SVM (4)	Random Forest (29)	CNN (Reorder)	HMM	HMM-e
AUC-ROC	49.9 ± 2.9	53.6 ± 2.1	56.3 ± 0.8	51.8 ± 1.7	53.6 ± 1.3
Balanced Acc.	49.8 ± 1.2	50.7 ± 0.5	54.7 ± 1.6	51.8 ± 1.7	52.5 ± 0.9

Table 1: Balanced Accuracy and AUC-ROC on GLOBEM. Our approach outperforms baseline machine learning methods and achieves similar results to the best-performing deep learning approach.

to degenerate or redundant models due to insufficient samples. Most deep-learning based models included in the benchmark leverage the 14-day history versions of the provided features, which sum over the prescribed time period. This helps reduce the frequency of missing values, but results in a lagging time series. We use the raw feature for the current day, rather than the 14-day history, which we find boosts performance by $\sim 3\%$ for our HMM-*e* approach. Sequences are constructed from a 28-day history of normalized features, aggregated by participant and preprocessed using the provided data platform. We filter out days where more than half of these features are missing, and within each study participant, fill remaining missing values using median imputation. After filtering, we have 5,393 training samples from 2018, 3,228 from 2019, 2,036 from 2020, and 1,192 from 2020, with class ratios (not-depressed to depressed) of 1.13, 1.23, 1.29, and 1.14 respectively.

Evaluation We use the "all-but-one" validation scheme (30), training on three years and testing on the fourth. Performance metrics include AUC-ROC and balanced accuracy (average of specificity and sensitivity), which we adopt as in (31) for its robustness to class imbalance (3). We compare our approach to the top-performing model from the GLOBEM study (31), *Reorder*, a CNN-based deep learning algorithm, along with a traditional SVM-based method (Canzian et al. (4)) from the benchmark. We also compare against a Random Forest-based approach included in the benchmark, based on Wahle et al. (29). We train the Random Forest approach on our selected four features rather than the original paper's six, using 450 trees, with the number of leaf nodes selected via K-Folds cross validation on a small training subset.

Results & Discussion Our lightweight approach using just four features outperforms machine learning approaches like (4) which uses up to 17 features, and performs comparably to or outperforms many other machine learning approaches included in the benchmark in (31). Our mean AUC-ROC and balanced accuracy using singleton HMMs beat out (4) by 1.9 and 2 percentage points, respectively. Using HMM-*e* , our AUC-ROC and balanced accuracy beat the same by 3.7 and 2.7 percentage points. In terms of balanced accuracy, HMM-*e* outperforms (29) 1.8 percentage points while achieving the same AUC-ROC. We also achieve similar performance as the complex deep-learning approach *Reorder*, falling 2.7 percentage points short in AUC-ROC and 2.2 in balanced accuracy. Notably, we achieve this performance with traditional machine learning techniques, far simpler models, and far fewer features.

For each of our $N \times M$ base learners with `num_states` states, on `num_features` features, we learn $\texttt{num_states} + (\texttt{num_states} \times \texttt{num_states}) + (\texttt{num_states} \times \texttt{num_features})$ parameters. In our case, with 4 features and 3 states, this results in 6,000 total parameters – versus *Reorder*'s 10,099 parameters. On our hardware, detailed in C, HMM-*e* trains on 2019-2021 data in 5 minutes versus 18 minutes for *Reorder*, 3 minutes for *Wahle*, and 29 seconds for *Canzian*.

In an unsupervised setting, training HMM-*e* without class labels and using subsampled data, we perform clustering with K-Means and dimensionality reduction with UMAP; results shown in Fig. 2.

5 Conclusion

We explore the connection between human behavior modeling and sequence analysis, providing a general framework for extracting coherent sequences from fragmented data. We present HMM-*e*, an ensemble learning approach that effectively models behavior sequences with minimal feature engineering. Our experiments demonstrate that HMM-*e* outperforms traditional machine learning baselines and delivers results comparable to complex deep-learning models, despite using fewer features. This highlights the efficiency and potential of our approach for scalable and interpretable sequence modeling in behavior-driven applications.

References

[1] Altman, E.; Blanuša, J.; von Niederhäusern, L.; Egressy, B.; Anghel, A.; and Atasu, K. 2024. Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. arXiv:2306.16424.

[2] Borrajo, D.; Veloso, M.; and Shah, S. 2020. Simulating and classifying behavior in adversarial environments based on action-state traces: An application to money laundering. In *Proceedings of the First ACM International Conference on AI in Finance*, 1–8.

[3] Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; and Buhmann, J. M. 2010. The Balanced Accuracy and Its Posterior Distribution. *2010 20th International Conference on Pattern Recognition*, 3121–3124.

[4] Canzian, L.; and Musolesi, M. 2015. Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, 1293–1304. New York, NY, USA: Association for Computing Machinery. ISBN 9781450335744.

[5] Choo, K. H.; Tong, J. C.; and Zhang, L. 2004. Recent Applications of Hidden Markov Models in Computational Biology. *Genomics, Proteomics & Bioinformatics*, 2(2): 84–96.

[6] Correa Bahnsen, A.; Aouada, D.; Stojanovic, A.; and Ottersten, B. 2016. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51: 134–142.

[7] Dietterich, T. G. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 1–15. Springer.

[8] Divekar, A.; Parekh, M.; Savla, V.; Mishra, R.; and Shirole, M. 2018. Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. IEEE.

[9] Gerych, W.; Agu, E.; and Rundensteiner, E. 2019. Classifying Depression in Imbalanced Datasets Using an Autoencoder- Based Anomaly Detection Approach. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, 124–127.

[10] Gong, Z.; and Chen, H. 2016. Model-Based Oversampling for Imbalanced Sequence Classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, 1009–1018. New York, NY, USA: Association for Computing Machinery. ISBN 9781450340731.

[11] Hevizi, G.; Biczo, M.; Poczos, B.; Szabo, Z.; Takics, B.; and Lorincz, A. 2004. Hidden Markov model finds behavioral patterns of users working with a headmouse driven writing tool. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, volume 1, 669–674.

[12] Jyothsna, V.; Prasad, R.; and Prasad, K. M. 2011. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7): 26–35.

[13] Kansara, D.; and Sawant, V. 2020. Comparison of Traditional Machine Learning and Deep Learning Approaches for Sentiment Analysis. In Vasudevan, H.; Michalas, A.; Shekokar, N.; and Narvekar, M., eds., *Advanced Computing Technologies and Applications*, Algorithms for Intelligent Systems, 347–359. Singapore: Springer.

[14] Kawawa-Beaudan, M.; Sood, S.; Palande, S.; Mani, G.; Balch, T.; and Veloso, M. 2024. Ensemble Methods for Sequence Classification with Hidden Markov Models. arXiv:2409.07619.

[15] Kawazu, H.; Toriumi, F.; Takano, M.; Wada, K.; and Fukuda, I. 2016. Analytical method of web user behavior using Hidden Markov Model. In *2016 IEEE International Conference on Big Data (Big Data)*, 2518–2524.

[16] Kudugunta, S.; and Ferrara, E. 2018. Deep neural networks for bot detection. *Information Sciences*, 467: 312–322.

[17] Kuncheva, L. I. 2014. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

[18] Li, Y.; Yin, R.; Park, H.; Kim, Y.; and Panda, P. 2022. Wearable-based Human Activity Recognition with Spatio-Temporal Spiking Neural Networks. In *NeurIPS 2022 Workshop on Learning from Time Series for Health*.

[19] Mannikar, R.; and Di Troia, F. 2024. Enhancing Botnet Detection in Network Security Using Profile Hidden Markov Models. *Applied Sciences*, 14(10).

[20] Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; and Tesconi, M. 2019. RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter. arXiv:1902.04506.

[21] Mitterbauer, J.; Bruckner, D.; and Velik, R. 2009. Behavior Recognition and Prediction With Hidden Markov Models for Surveillance Systems. *IFAC Proceedings Volumes*, 42(3): 206–211. 8th IFAC Conference on Fieldbuses and Networks in Industrial and Embedded Systems.

[22] Nguyen, N.; and Nguyen, D. 2021. Global Stock Selection with Hidden Markov Model. *Risks*, 9(1).

[23] Piyathilaka, L.; and Kodagoda, S. 2013. Gaussian mixture based HMM for human daily activity recognition using 3D skeleton features. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, 567–572.

[24] Rabiner, L.; and Juang, B. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1): 4–16.

[25] Robinson, W. N.; and Aria, A. 2018. Sequential fraud detection for prepaid cards using hidden Markov model divergence. *Expert Systems with Applications*, 91: 235–251.

[26] Su, T.; Sun, H.; Zhu, J.; Wang, S.; and Li, Y. 2020. BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset. *IEEE Access*, 8: 29575–29585.

[27] Ullah, F.; Ullah, S.; Srivastava, G.; and Lin, J. C.-W. 2024. IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic. *Digital Communications and Networks*, 10(1): 190–204.

[28] Vrigkas, M.; Nikou, C.; and Kakadiaris, I. A. 2015. A Review of Human Activity Recognition Methods. *Frontiers in Robotics and AI*, 2.

[29] Wahle, F.; Kowatsch, T.; Fleisch, E.; Rufer, M.; and Weidt, S. 2016. Mobile Sensing and Support for People With Depression: A Pilot Trial in the Wild. *JMIR Mhealth Uhealth*, 4(3): e111.

[30] Xu, X.; Liu, X.; Zhang, H.; Wang, W.; Nepal, S.; Sefidgar, Y.; Seo, W.; Kuehn, K. S.; Huckins, J. F.; Morris, M. E.; Nurius, P. S.; Riskin, E. A.; Patel, S.; Althoff, T.; Campbell, A.; Dey, A. K.; and Mankoff, J. 2023. GLOBEM: Cross-Dataset Generalization of Longitudinal Human Behavior Modeling. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(4).

[31] Xu, X.; Zhang, H.; Sefidgar, Y.; Ren, Y.; Liu, X.; Seo, W.; Brown, J.; Kuehn, K.; Merrill, M.; Nurius, P.; Patel, S.; Althoff, T.; Morris, M. E.; Riskin, E.; Mankoff, J.; and Dey, A. K. 2023. GLOBEM Dataset: Multi-Year Datasets for Longitudinal Human Behavior Modeling Generalization. arXiv:2211.02733.

[32] Yoon, B.-J. 2009. Hidden Markov models and their applications in biological sequence analysis. *Curr. Genomics*, 10(6): 402–415.

[33] Yu, C.; Xu, Y.; Cao, J.; Zhang, Y.; Jin, Y.; and Zhu, M. 2024. Credit Card Fraud Detection Using Advanced Transformer Model. arXiv:2406.03733.

A Supplementary Material: Extended Results

Method		2018	2019	2020	2021	Mean
Random Forest (*Wahle et al.*)	*AUC-ROC*	51.9	51.8	55.8	54.9	53.6 ± 2.1
	Balanced Acc.	51.2	50.0	51.1	50.7	50.8 ± 0.5
SVM (*Canzian et al.*)	*AUC-ROC*	49.1	48.4	48.0	54.2	49.9 ± 2.9
	Balanced Acc.	48.0	50.4	50.6	50.1	49.8 ± 1.2
CNN (*Reorder*)	*AUC-ROC*	56.7	56.4	55.2	57.1	56.3 ± 0.8
	Balanced Acc.	54.8	54.2	53.0	56.8	54.7 ± 1.6
HMM	*AUC-ROC*	52.4	50.9	54.0	50.0	51.8 ± 1.7
	Balanced Acc.	52.4	50.9	54.0	50.0	51.8 ± 1.7
HMM-e	*AUC-ROC*	51.9	54.7	54.4	53.3	53.6 ± 1.3
	Balanced Acc.	52.2	52.3	53.8	51.8	52.5 ± 0.9

Table 2: Balanced Accuracy and AUC-ROC on GLOBEM data. For each year indicated, we train on the other 3 years, and hold out that year for testing.

In addition to the aggregated results in Table 1, we present results for each of the four years in the GLOBEM dataset in Table 2. Each year indicated is the held-out test year, while the other three are used for training. We compare our approaches against the best-overall-performing approach in the GLOBEM study, *Reorder* (31). *Reorder* is a deep-learning based approach tailored for the GLOBEM problem, built on a CNN backbone.

We also present results from (4), an SVM-based traditional machine learning approach. We find that we are consistently able to outperform (4), and in terms of balanced accuracy, achieve comparable results to *Reorder* with far fewer features and much less complex models.

We illustrate the training and inference pipelines of the HMM-*e* approach detailed in Section 3.2.1, in Figure 3.

B Feature Selection

Figure 4 shows the four features we choose to model, across three anonymized participants from the 2018 study. While many baseline models included in the benchmark choose to model tens or even hundreds of features, we use a small feature set. This allows us to learn meaningful correlations and avoid converging to degenerate models due to insufficient samples. We include two smartphone features (screen time and location-based moving-to-static ratio) and two wearable features (sleep time and steps). All of these features are computed daily.

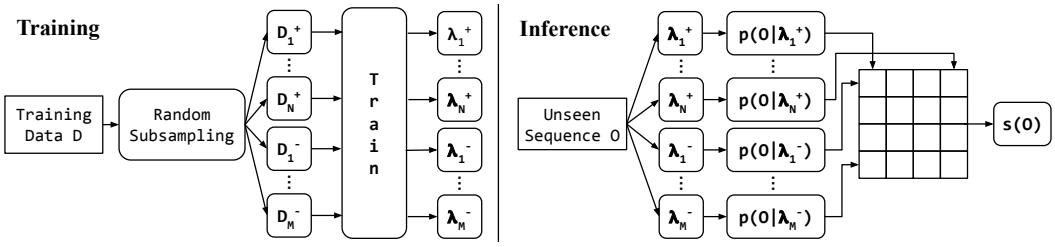

Figure 3: Flow diagram of our HMM-*e* ensemble training and inference approach, as detailed in Section 3.2. While we adopt this approach using HMMs, the framework itself is model agnostic. The training data is broken into random subsets, and a diverse ensemble of learners is trained on these subsets. At inference time, pairwise matchups of likelihoods given by the models are compared, giving the composite score s.

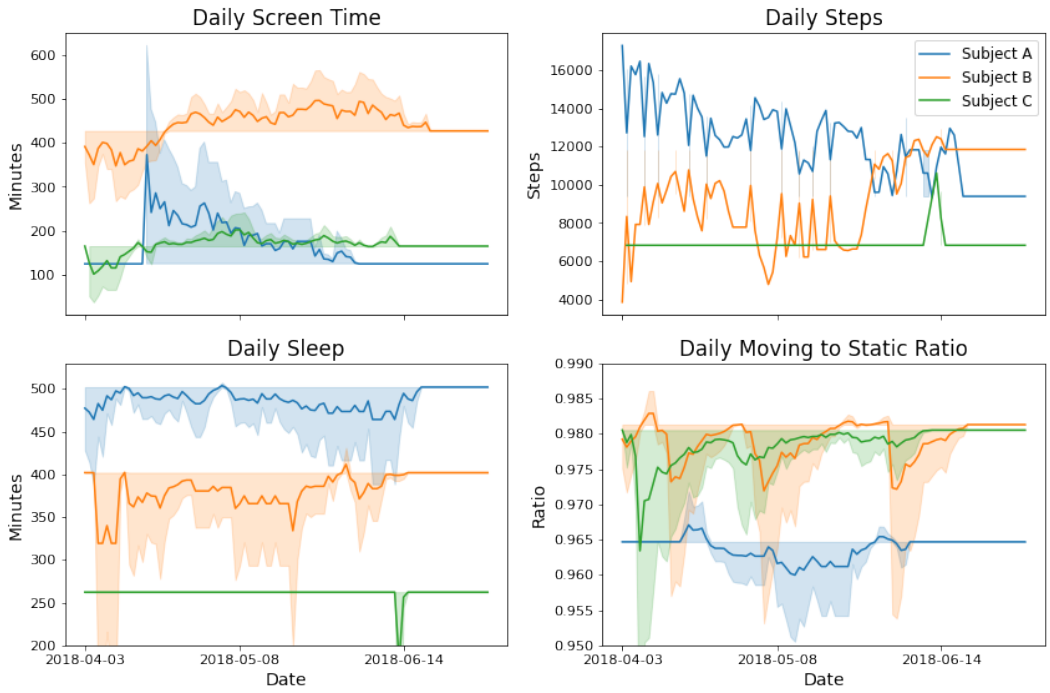

Figure 4: Samples of the daily features we model, across 3 anonymized 2018 participants. For visualization purposes, features are lightly smoothed using an exponential weighted moving average with a half-life of 4 days. For modeling, features are normalized.

C Hardware and Software Stack

Our experiments are performed on an AWS *r5.24xlarge* EC2 instance featuring 96 virtual CPUs and 768 GB of memory. Due to the lightweight nature of the models trained, we do not have to leverage GPU acceleration. The environment is configured with Ubuntu 20.04 LTS as the operating system, and we use Python version 3.8.10. Aside from standard machine-learning libraries like Pandas, NumPy, Pytorch, Scikit-Learn, and Tensorflow, we also use HMMLearn to train our HMMs, and Ray to parallelize training and data processing.