Proceedings of the Twentieth International AAAI Conference on Web and Social Media (ICWSM 2026)

SoMeR: A Multi-View Social Media User Representation Learning Framework

Siyi Guo, Keith Burghardt, Valeria Pantè, Kristina Lerman

Information Sciences Institute, University of Southern California
Marina del Rey, California, USA
siyiguo@isi.edu

Abstract

Low-dimensional representations of social media users are crucial for modeling preferences, interests, and behavior, with applications ranging from behavior prediction to the detection of inauthentic accounts. However, existing approaches often focus narrowly on specific tasks or data modalities such as text, activity patterns, or platform metadata, limiting their ability to holistically capture the complexity of user behavior. We introduce SoMeR: Social Media user Representation learning, a multi-view framework that integrates temporal activity, textual content, profile attributes, and network interactions to learn comprehensive user embeddings. SoMeR encodes users' post streams as sequences of time-stamped text features, embeds them using transformers alongside profile data, and jointly trains on link prediction and contrastive objectives to learn representations that reflect both behavioral patterns and social similarity. We demonstrate the versatility and effectiveness of SoMeR in three applications: (1) identifying accounts driving information operations, (2) measuring online polarization following major events, and (3) predicting future participation in Reddit hate communities. By modeling user behavior across multiple modalities and tasks, SoMeR enables a deeper understanding of socio-political dynamics on social media and supports more informed interventions.

Code — https://github.com/fionasguo/SoMeR

Introduction

The ability to model user behavior on social media has important implications for social platforms, researchers, and society. As social media increasingly shapes public discourse, civic engagement, and responses to external events, accurately capturing user preferences, opinions, and behaviors has become essential. Low-dimensional user representations offer a powerful way to model such behavior, serving as foundational building blocks in applications including personalized content recommendation, bot detection (Nwala, Flammini, and Menczer 2023), identifying mental health risks (Sawhney et al. 2021), and hate speech detection (Qian et al. 2018; Del Tredici et al. 2019).

The main challenge stems from the inherently multimodal and dynamic nature of user behavior. Individuals express themselves through textual content, activity patterns, temporal rhythms, and social connections, which evolve over time and vary across platforms. While user representation learning (URepL) has made strides in commercial applications such as recommendation and targeted advertising (Bhargava et al. 2015; Wang, Wang, and Yeung 2015; Hou et al. 2022; Shin et al. 2023), these methods are often narrowly tailored to specific use cases or rely on commercially relevant features (Li and Zhao 2021; Purificato, Boratto, and De Luca 2024). Even within the social media domain, existing approaches typically focus on individual modalities, such as text (Hallac, Ay, and Aydin 2021), images (Pan and Ding 2019), activity logs (Nwala, Flammini, and Menczer 2023), or platform-specific metadata (AlMahmoud and AlKhalifa 2018; Dahiya, Kumar, and Yadav 2022), and often lack generalizability across platforms or tasks. Multi-view approaches exist, but they typically combine only text and network features, and many remain task- or platform-specific (Feng et al. 2021; Shen et al. 2023).

To address these challenges, we propose **SoMeR**, **So**cial **Me**dia user **R**epresentation learning framework that integrates four key behavioral signals into a unified, low-dimensional embedding: temporal activity, textual content, user profile information, and network interactions. SoMeR models user posts as sequences of triplets *(timestamp, textual feature, value)*, enriching sparse time series data with content-derived features. These triplets are embedded using transformer architectures (Vaswani et al. 2017), combined with profile embeddings, and trained jointly using two objectives: (1) network link prediction, to model social interactions, and (2) contrastive learning, to learn a user embedding space that reflects behavioral similarity. This enables SoMeR to be used in unsupervised settings where labeled data is scarce, while remaining adaptable via fine-tuning for task-specific downstream applications.

We demonstrate SoMeR's versatility and generalizability through three diverse case studies across two platforms:

1. Identifying Information Operation (IO) drivers, i.e., malicious accounts used to sow division or influence public opinion (Wen et al. 2020), using a newly constructed IO ground-truth dataset.
2. Measuring polarization following major events, by analyzing changes in user activity and language on X (formerly Twitter) after the U.S. Supreme Court ruling over-

turning federal abortion protections.

3. Predicting user participation in Reddit hate communities, a task linked to the early detection of online-to-offline hate (Müller and Schwarz 2021), even in the absence of recent posting history.

These applications highlight several key challenges in user representation learning from social media data: high heterogeneity of user behaviors, sparsity in temporal activity, variability in network structure, and limited availability of ground truth data. SoMeR is designed to tackle these by leveraging universal features across platforms and adapting to user behavioral diversity.

Our main contributions are:

- A **multi-view framework** that jointly models user opinions, temporal behaviors, profiles, and networks in a unified embedding, which is better designed to handle the sparsity, noise, and heterogeneity characteristic of real-world social media data.
- A **joint training** approach combining link prediction and contrastive learning, with a novel augmentation method for contrastive learning on irregular user histories using bootstrapped sampling and noise injection. This training paradigm enables robust unsupervised and fine-tuned representations.
- Demonstrated **generalizability and scalability** with successful deployment across diverse socio-political tasks, platforms and datasets comprising from 200K up to 17 million posts.

By bridging gaps between traditional user representation learning and real-world socio-political analysis, SoMeR offers a powerful tool for studying dynamic, heterogeneous user populations and supports more informed interventions in online spaces.

Related Work

User representation learning has gained widespread interest within recommender systems due to its ability to capture compact and meaningful embeddings of users' behaviors and preferences (Yuan et al. 2021). Over the years, researchers have developed many methods, mostly for commercial applications (Li and Zhao 2021; Purificato, Boratto, and De Luca 2024). These include matrix and tensor factorization (Bhargava et al. 2015), auto-encoders (Zhuang et al. 2017), transformer-based architectures (Cheng et al. 2021; Hou et al. 2022; Shin et al. 2023), contrastive learning (Oord, Li, and Vinyals 2018; Cheng et al. 2021), graph neural networks (Liu et al. 2023), and large language models (LLM) (Ren et al. 2024). Improving from task-specific methods (Guo et al. 2017), researchers have also explored universal user representation learning methods that can be generalized to different downstream tasks in recommendations (Yuan et al. 2021; Hou et al. 2022; Shin et al. 2023; Fazelnia et al. 2024).

Beyond recommendations, URepL also presents considerable opportunities in the social and political domains, such as enabling a deeper understanding of public sentiment and societal trends (Pan and Ding 2019). However, the approaches designed for recommendations are built on commercially relevant features or pre-trained using commercial data (Li and Zhao 2021; Purificato, Boratto, and De Luca 2024), and thus are not compatible in social analyses. In comparison, most current approaches in social domains rely on a narrow subset of user features. For example, Mueen et al. (2016) and Nwala, Flammini, and Menczer (2023) uses temporal activity features to detect online bot and IO drivers; Hallac, Ay, and Aydin (2021) experimented with different textual embedding methods; Perozzi, Al-Rfou, and Skiena (2014) and Wang et al. (2017) uses network features and node prediction to learn user interests or communities. Multi-view studies, meanwhile, are typically limited to text and network features, with many tailored to specific tasks or platforms (Ribeiro et al. 2018; Wang et al. 2019; Lai and Neville 2020; Feng et al. 2021; Shen et al. 2023). Our framework **SoMeR** addresses prior gaps by creating a more universal user representation approach, better fitted for socio-political domains and applicable across different platforms and tasks, by incorporating all of the textual, temporal, profile and network features into a multi-view embedding.

Methods

We propose a self-supervised framework to learn a latent user embedding space based on the architecture shown in Figure 1. From each user's history, a timeline of texts, we first extract certain textual features, such as sentence embeddings. Next, to better learn from users with sparse activities, we format textual features and timestamps into triplets of observations $(timestamp, feature, value)$. These triplets pass through a Triplet Encoder, a transformer-based contextual learning module, and a fusion attention layer, being encoded into a user history embedding. We concatenate it with user's profile embedding from a separate module, obtaining a complete user embedding. We train these encoding modules with two self-supervised objectives: *network link prediction* that learns patterns of interactions, including sharing, following or other connections, and *contrastive loss* that learns user similarity with respect to posting history. This method learns user similarity in a heterogeneous user population without the need for time-consuming human annotations. The method can be easily adapted for various downstream tasks, such as supervised classification and unsupervised similarity search.

User History Data Processing

In this study, we consider each user's history to be a collection of time-stamped texts. Each text can be an original post, a repost, a reply, etc. There can be other types of user activity, such as likes or views, that also provide valuable information that can be incorporated in future work. To learn from a user's history, we first extract textual features that are useful for a given downstream task. In our experiments, we find that the use of contextual sentence-BERT embeddings (Reimers and Gurevych 2019) leads to a powerful representation. However, BERT embeddings yield complex high-dimensional features that can slow performance. To re-

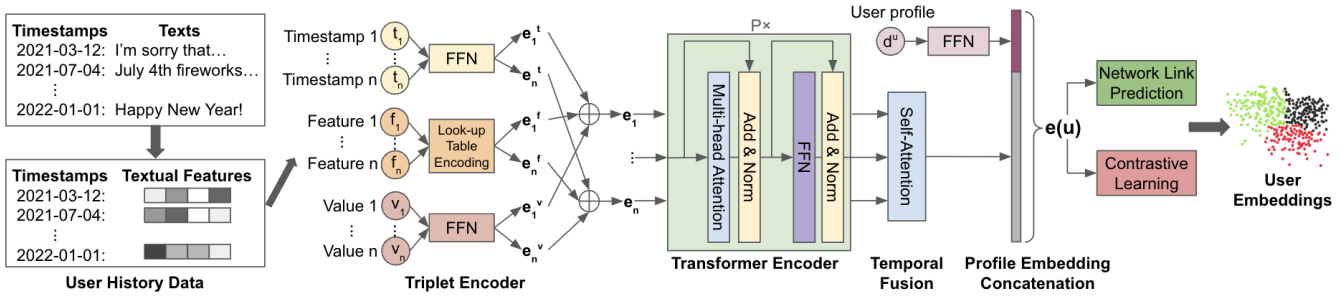

Figure 1: Model Architecture of **SoMeR**. We format a user's posting history into triplets of time, feature, and value, which undergo encoding via a Triplet Encoder, a transformer-based contextual learning module and a fusion attention layer, becoming a user history embedding that is then concatenated to the user profile embedding. Through training with two self-supervised objectives - network link prediction and contrastive loss - our method effectively captures user similarity in the latent space.

duce model size and complexity, we perform dimensionality reduction using principal component analysis (PCA) to reduce BERT embeddings to the first five components and treat them as textual features. Although we could vary the number of components, we chose five as a reasonable trade-off between a low-dimension and adequately complex representation. Previous works (Grootendorst 2022) have shown that meaningful contextual representation is still retained after dimension reduction operations on BERT embeddings.

Triplet Data Encoder

Social media users exhibit highly diverse posting behaviors. A small fraction of users generate most of the content, whereas the majority of users only have a few posts over a long period of time (Krishnan, Sharma, and Sundaram 2018). This leads to a sparse matrix of multivariate time series as the input data, making it difficult to accurately learn an embedding space from sparse signals and posing a challenge in time complexity. We take inspiration from (Tipirneni and Reddy 2022) and represent the user's posting history as a collection of triplets $(timestamp, feature, value)$. This only takes into account the observations available in the data, allowing us to account for times with little to no activity. Thus, a dataset of M users can be represented as U:

$$\mathbf{U} = \{(d^u, \mathbf{X}^u)\}_{u=1}^{M} \; where \; \mathbf{X} = \{(t_n, f_n, v_n)\}_{n=1}^{N} \quad (1)$$

Each user u is characterized by their profile feature vector $d^u \in \mathbb{R}^D$ and their posting history $\mathbf{X}^u$, which is a set of N triplets including the timestamp t_n, the feature category f_n and the value of this feature category v_n. There can be more than one feature-value pair at one time point, and therefore N may be greater than the total number of time points.

Tipirneni and Reddy (2022) shows the effectiveness of using a feed-forward network to embed continuous values. We therefore use two separate feed-forward networks to encode timestamp t_n and value v_n into time embedding $e_n^t \in \mathbb{R}^K$ and value embedding $e_n^v \in \mathbb{R}^K$ respectively, where K is the hidden dimension of these embeddings. These networks have one linear layer followed by a tanh activation function. For feature categories, we use a lookup table encoder to generate a feature embedding $e_n^f \in \mathbb{R}^K$. Lastly, we add up these three embeddings to be the triplet embedding $e_n^{triplet} \in \mathbb{R}^K$. Adding them up, rather than concatenating, has been proven its effectiveness (Tipirneni and Reddy 2022), and allows us to reduce dimensions of model weights to save training time and memory.

$$e_n^t = W_1^t \, tanh(W_2^t t_n + b^t) \quad (2)$$
$$e_n^v = W_1^v \, tanh(W_2^v v_n + b^v) \quad (3)$$
$$e_n^f = LookupEncoder(f_n) \quad (4)$$
$$e_n^{triplet} = e_n^t + e_n^v + e_n^f \quad (5)$$

Transformer Encoder

The transformer architecture has been shown to have a high performance in representation learning for time series and user behavior sequences (Zerveas et al. 2021; Shin et al. 2023). We therefore choose to use the transformer encoder to better extract a latent representation from the triplet embedding $e_n^{triplet} \in \mathbb{R}^K$. We use L transformer layers. Each layer has H attention heads with learnable key, query, and value weights $W^k, W^q, W^v \in \mathbb{R}^{K \times P}$ where P is the hidden dimension size for these weights. After all attention heads are added up and layers normalized, the embedding vector is projected back to K-dimensions through a feed-forward network. This network includes two layers with hidden dimension $2K$ and a ReLU activation in the middle. Lastly, layer normalization is applied again. From this transformer module, we obtain $e_n^{trans} \in \mathbb{R}^K$. In our case studies, data sizes are within 20 million posts, and therefore we choose to have a small transformer module with $L = 2$, $H = 4$ and $P = K/H$. We choose the hidden dimension $K = 64$ with a grid search in $[32, 64, 128]$.

Temporal Fusion

After all the triplets for a user pass through the triplet encoder and the transformer encoder, we use an attention layer to learn the correlations between the different triplets. This gives us the integrated embedding of posting history $e^{hist} \in$

$\mathbb{R}^K$ for each user.

$$a_n = W_1^{attn}\, tanh(W_2^{attn} e_n^{trans} + b^{attn}) \quad (6)$$

$$a_n = \frac{exp(a_n)}{\sum_{i=1}^{N} exp(a_i)} \quad (7)$$

$$e^{hist} = \sum_{n=1}^{N} a_n\, e_n^{trans} \quad (8)$$

where $W_1^{attn} \in \mathbb{R}^{2K}$, $W_2^{attn} \in \mathbb{R}^{2K\times K}$, $b^{attn} \in \mathbb{R}^{2K}$ are trainable weights and intercept.

Profile Embedding

Other than the posting history of a user, their profile features, e.g., location and number of followers and friends, can also play an important role. Therefore, we add a feed-forward network to learn a profile embedding $e^{prof} \in \mathbb{R}^K$ from user's profile vector $d^u \in \mathbb{R}^D$, and concatenate it to the user history embedding e^{hist} to obtain a complete user embedding $e^u \in \mathbb{R}^{2K}$.

$$e^{prof} = W_1^{prof}\, tanh(W_2^{prof} d^u + b^{prof}) \quad (9)$$

$$e^u = e^{hist} \oplus e^{prof} \quad (10)$$

where $W_1^{prof} \in \mathbb{R}^{2K}$, $W_2^{prof} \in \mathbb{R}^{2K\times D}$, $b^{prof} \in \mathbb{R}^{2K}$ are trainable weights and intercept, and $\oplus$ is a concatenation operation.

Network Link Prediction

Individuals connected in social networks tend to be similar, and their behaviors are often affected by other users in the network (McPherson, Smith-Lovin, and Cook 2001). Therefore, it is crucial to include network connection features when learning user representations (AlMahmoud and AlKhalifa 2018). We design a self-supervised network link prediction objective to train our model to learn interaction activities such as sharing, following, and commenting. We use a feed-forward module to perform link prediction with a binary cross-entropy loss. During training, we consider all pairs of distinct users in each batch. The feature for link prediction is the concatenated embedding of a user pair, and we obtain the links from a self-defined network as the binary labels (e.g. whether a user reposts another user). This is described as following:

$$\tilde{y}_{i,j}^{link} = \sigma(W_1^{link}\, ReLU(W_2^{link}(e_i^u \oplus e_j^u) + b^{link})) \quad (11)$$

$$\mathcal{L}_{\text{network}} = -\frac{1}{N_{\text{distinct pairs}}} \sum_{\substack{i,j\in batch \\ i\neq j}} \Big[y_{i,j}^{link} \cdot log(\tilde{y}_{i,j}^{link}) + (1 - y_{i,j}^{link}) \cdot log(1 - \tilde{y}_{i,j}^{link}) \Big] \quad (12)$$

where i, j are indices of two users in a training batch, $W_1^{link} \in \mathbb{R}^{2K}$, $W_2^{link} \in \mathbb{R}^{2K\times 4K}$ and $b^{link} \in \mathbb{R}^{2K}$ are trainable weights and intercept, $\sigma(\cdot)$ is the sigmoid function, $N_{\text{distinct pairs}}$ is the number of all pairs of distinct users in a batch, and y^{link} is the binary label of whether i and j are connected in the network.

Contrastive Learning

Contrastive learning aims to obtain a latent embedding space in which similar samples are closer and distinct samples are farther away from each other. Previous papers on user representation learning have shown contrastive learning's utility (Cheng et al. 2021; Shin et al. 2023). We adopt self-supervised contrastive learning to learn user similarity in their posting histories. The InfoNCE (Oord, Li, and Vinyals 2018) loss function uses categorical cross-entropy loss to optimize the negative log probability of classifying one positive or similar sample correctly among a set of negative or unrelated samples. In our case, it can be written as:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{\text{b.s.}} \sum_{i\in\text{batch}} \left[log \frac{exp(e_i^u e_i^{u^+}/\tau)}{\sum_{j\in batch, i\neq j} exp(e_i^u e_j^u/\tau)} \right] \quad (13)$$

For e_i^u a user history embedding, $e_i^{u^+}$ is its positive pair. $e_j^u\ \forall j \in batch$ where $i \neq j$ are the embeddings of all other users in that randomly retrieved batch, which we consider as negative samples to e_i^u. We further perform temperature scaling with parameter τ, which is tuned during training using grid search in $[0.5, 1, 3, 6]$. b.s. represents batch size.

To generate a positive sample paired to each user history embedding, we perform data augmentation on users' triplet data. Although researchers have proposed many time series data augmentation methods (Wen et al. 2020), many classical methods are not applicable to our scenario. For example, a user history cannot be sub-sequenced or shuffled in the time domain. We use another efficient and simple way: bootstrapping with replacement of existing triplets $\{(t_n, f_n, v_n)\}_{n=1}^N$. This ensures that the generated set of triplets is similar. We incorporate noise into the augmentation by

1. varying the number of random draws between $range(start = (1-\gamma)N,\ end = (1+\gamma)N,\ step = 1)$, where N is the total number of triplets for a user and γ is a hyperparameter to control the noise level,
2. scaling the value v_n by a factor randomly selected between $range(start = 1-\gamma,\ end = 1+\gamma,\ step = 0.5)$, and
3. imposing a lag time, randomly selected between 1 -3 days, on the timestamp t_n of sampled triplets.

During training, we tune γ by grid searching in $[0.1, 0.5, 1, 2, 5]$. The results of the three applications below shows the effectiveness of this augmentation method.

Model Training

Finally, the contrastive objective function and the network link prediction objective are jointly trained. The overall loss is

$$\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \lambda \mathcal{L}_{Network} \quad (14)$$

where λ is a hyperparameter to balance between two losses. We perform grid search on the IO driver dataset for λ in $[0.1, 1, 5, 10]$ and find $\lambda = 1$ gives the lowest validation loss. We use Adam optimizer, a learning rate of 5e-5, a cosine decay learning scheduler, an early stopping mechanism monitored by overall loss function, a maximum epoch of 60, and

a batch size of 128. See Appendix Table 4 for a summary of all hyperparameters and their selection procedures. We find that changes to these hyperparameters do not strongly affect results.

Model Evaluation on Synthetic Data

To verify that our self-supervised framework indeed learns *both* temporal activity and textual features from the triplet data, we test it on four synthetic datasets. We use synthesized numerical features to mimic textual features in real data. The four datasets simulate (1) a simple scenario with three clusters that are easier to distinguish, (2) a hard scenario with 10 clusters and more noise, (3) a scenario where clusters vary in temporal activity pattern but have the same features and values, and (4) a scenario where clusters vary only in features and values, but having all the same temporal patterns. On all datasets, our model can successfully detect the clusters, indicating it is able to learn heterogeneity in both temporal activities and numerical feature values (see Appendix).

In the next sections, we use three applications - IO driver detection, political polarization analysis, and hate group joiner prediction to illustrate the effectiveness and versatility of our framework.

Detecting Information Operation (IO) Drivers

Information operations use strategically organized efforts to manipulate public opinions at scale, contaminating the online information ecosystem with disinformation. Social media platforms provide fertile grounds for these operations (Pacheco et al. 2021), necessitating their identification. IO drivers use different tactics from each other and from authentic accounts thus leading to distinct behaviors flagged by previous researchers, such as (1) co-sharing the same posts or URLs, (2) using an identical sequence of hashtags, (3) synchronized behaviors, and (4) sharing very similar posts (Pacheco et al. 2021; Nwala, Flammini, and Menczer 2023; Luceri et al. 2023). Our framework captures temporal, textual, profile, and network features, identifying platform-agnostic synchronized behaviors. We apply supervised fine-tuning on this framework and test it across three campaigns on X to demonstrate its effectiveness.

Data

We evaluate our method on an X dataset that prior IO driver detection methods are benchmarked on (Nwala, Flammini, and Menczer 2023; Luceri et al. 2023). X suspended and released these drivers of multiple information operations in 21 countries because they were associated with malicious IOs and violated the platform terms. We select four campaigns—one based in China involving a large number of accounts, one small operation in Egypt and UAE, and two operations in Venezuela, which are combined into one dataset for testing the multi-campaign scenario.

We collect a set of control users by first collecting the top five keywords and the top five hashtags for each IO driver within our X datasets. We then extract 10 random posts that were posted within the timeframe that the IO driver was active (between their first and last post). For each post, we find the post author and query all their posts made within the timeframe the IO driver is active. Table 1 shows the information about these campaigns. We randomly split data into 70% training, 15% validation and 15% testing.

Campaign	Time Range	# IO Drivers	# Control Users	# Posts
China	2019 - 2021	2016	11366	17M
Egypt-UAE	2016 - 2019	240	2164	4.5M
Venezuela	2017 - 2021	275	4183	10M

Table 1: Meta-data of Information Operation Datasets

Pre-training and Supervised Classification

We use a two-step approach to classify IO drivers. First, we pre-train our model in the self-supervised manner with both IO driver and control users using all of their posts and metadata, but without any IO label. For textual pre-processing, we remove user mentions, URLs, emojis, and all non-ASCII characters, but retain the hashtags. This is a multi-lingual datasets that include more than 50 languages. Therefore we compute the sentence-BERT embedding for each post using the `stsb-xlm-r-multilingual` [1], and compute the first five components from PCA. In the triplet *(timestamp, feature, value)*, value corresponds to the values of five PCA components, and feature is a categorical variable referring to which component this value corresponds to. Due to computational resource limitations, we use this technique to reduce the number of triplets and computation complexity. We aggregate data for each user by summing up their PCA embeddings of their posts in 3-day intervals and taking the middle day in the interval as the corresponding timestamp. We have conducted experiments with 1-day, 3-day and 5-day window sizes, and choose to use 3-day as it keeps the data relatively fine-grained and also aggregate data well for low-activity accounts. This gives us the temporal and textual features. For profile features, we use the number of followings and followers, as accounts such as news media outlets with lots of followers can behave very differently from other types of users. For the network link prediction component of our model, we use the repost network because co-repost has been shown as a potential indicator of an IO driver (Pacheco et al. 2021). With all these features, we train the model to learn a user embeddings space such that users with similar behaviors are closer.

In the next step, we perform supervised fine-tuning on the learned model parameters with an additional two-layer feed-forward network for binary classification. This network has linear layers with a hidden dimension of 128, a ReLU activation, a dropout layer with a rate of 0.3, a batch normalization, and a second linear layer that project embeddings onto $\mathbb{R}$ for binary prediction. The sigmoid function is then applied and a binary cross-entropy function is used as the loss. We use the same hyperparameters described in Methods.

[1] https://huggingface.co/sentence-transformers/stsb-xlm-r-multilingual

Baseline Models

We compare our method with common IO driver detection methods (Pacheco et al. 2021). In view of the observation that IO drivers have abnormally similar behaviors, these methods identify users with unusual similarity based on (1) co-repost and (2) co-URL sharing behaviors if cosine similarity is above or at the 99.5 percentile, (3) hashtag sequence (using a minimum sequence of 5 identical hashtags in the same order within a post) and (4) text similarity in averaged BERT embeddings over all posts from a user, with a cosine similarity threshold of 0.7 (parameters based on prior works (Pacheco et al. 2021; Luceri et al. 2023).)

In addition, we compare with a more advanced URepL baseline User2Vec (Amir et al. 2016). This approach trains on the BERT embeddings of all posts from a user and aims to predict which sentences are written by the corresponding authors. Compared to using averaged BERT embeddings, this method can better capture the relationships between users and their posts. This approach has been used to detect sarcasm, hate speech, and mental health conditions from social media posts (Amir et al. 2016, 2017; Irani, Wrat, and Amir 2021).

Last, we compare with a strong multi-view URepL baseline SATAR (Feng et al. 2021). This approach combines textual, profile and network features, and the authors uses it for social bot detection. In contrast to our work, SATAR does not utilize temporal features, and uses follower count prediction as the self-supervised objective, which is suitable for bot detection but less generalizable for other use cases. In addition, SATAR uses the Long Short-Term Memory model (LSTM) (Graves 2012) with word2vec embeddings (Church 2017) whereas SoMeR uses more advanced transformer module to learn contextual information from user posting histories. The details of evaluating User2Vec and SATAR are included in Appendix.

Model Performance and Ablation

Performance Table 2 shows the performance of different methods. Our models are evaluated on 10 random training-validation-test data splits. First, we compare our method with the baselines. For China and Venezuela campaigns, we outperform or match all the baselines. For the gap between the text similarity algorithm's F1-score and SoMeR for the Egypt-UAE campaign, our analysis shows that PCA dimensional reduction on BERT embeddings in our method can lower the performance, compared to the text similarity baseline using the full BERT embeddings. In addition, Egypt-UAE is the smallest campaign, and pre-training on a small dataset can also result in less ideal performance. See analyses in Appendix.

Moreover, our full model outperforms the URepL baselines User2Vec and SATAR in all scenarios. Interestingly, we observe that SATAR's pre-training step is ineffective, with the fine-tuning step contributing most significantly to its overall performance. SATAR's self-supervised pre-training objective, which predicts user follower counts, easily leads to overfitting. For example, for the Chinese campaign, F1-score for predicting follower counts reaches 0.88 but the F1 for predicting IO drivers is only 0.13 during pre-training. After fine-tuning, the F1 for predicting IO drivers reaches 0.98 as in Table 2. This reliance on labeled data highlights SATAR's limited generalizability to scenarios with sparse labels. In conclusion, the consistently high F1 scores achieved by SoMeR demonstrate the effectiveness of our method.

Scalability To evaluate the scalability of SoMeR, we conducted experiments by subsetting the training data for each IO detection dataset to 75%, 50%, and 25% (see Appendix). The results show that SoMeR maintains strong performance even when subsetting up to 50% of the training data, demonstrating its robustness.

	China	Egypt-UAE	Venezuela
BASELINES			
Co-Repost	0.00	0.15	0.26
Co-URL	0.19	0.27	0.30
Hashtag-sequence	0.08	0.20	0.05
Text Similarity	0.13	**0.93**	**0.82**
User2Vec	0.93 ± 0.01	0.63 ± 0.01	0.76 ± 0.02
SATAR	0.98 ± 0.01	0.72 ± 0.03	0.61 ± 0.04
OURS			
Temporal	0.96 ± 0.01	0.41 ± 0.13	0.57 ± 0.06
Textual	0.97 ± 0.01	0.69 ± 0.06	0.67 ± 0.04
Temporal+Textual	0.98 ± 0.01	0.77 ± 0.05	0.77 ± 0.04
SoMeR	$\mathbf{0.99 \pm 0.01}$	0.85 ± 0.04	$\mathbf{0.82 \pm 0.04}$

Table 2: F1-Scores on Detecting IO Drivers.

Ablations Next, we perform an ablation study on our model to dissect the impact of different features and modules we use. The temporal model only uses timestamps and three-day post counts as the feature but does not use any textual or network features. The textual model only uses the *average* textual embedding over three-day intervals, which does not reflect temporal activity. The Temporal+Textual model uses timestamps and the *summed* textual embeddings over three-day intervals, as described in Detecting IO Drivers. This reflects both temporal activity and textual features. The full model SoMeR incorporates network feature on top of the temporal and textual features. All of these models include profile features.

We observe that the full model has better performance than any of the ablated models, implying the benefit of including the network link prediction objective. In addition, the Temporal+Textual model performs better than the temporal and textual models alone, indicating that each of the four features plays a role in detecting IO drivers. For the China campaign, all ablated and full models have high F1 scores. First, this implies the importance of the temporal features. The textual model uses *average* textual embeddings over three-day intervals. This lowers but does not completely eliminate temporal factors, and can result in behaviors similar to the temporal model. We further confirm the importance of combining different modules by t-SNE of the embeddings generated by different modules in Figure 6 in Appendix.

The performance of different SoMeR modules and different baseline approaches vary between different campaigns. For example, the baseline Text Similarity performs well for Egypt-UAE and Venezuela but poorly for China. On the other hand, our models that use temporal features show excellent performance in the China campaign. This shows that campaigns use different tactics and highlights that relying on a single behavior pattern is insufficient, whereas our framework's multi-view approach enables better detection.

Uncovering Shifts in Polarized Discussions

Next we use SoMeR to explore changes in polarization within U.S. social media discussions. The U.S. society has grown increasingly more polarized. Not only do liberals and conservatives hold sharply different opinions on a range of issues (Center 2017), but they also have more negative feelings towards members of the other party, compared to members of their own party (Iyengar and Krupenkin 2018). These differences show up not only in political speech, but also in the everyday behaviors (DellaPosta, Shi, and Macy 2015), on social media platforms, where liberals and conservatives segregate themselves in different echo chambers, they are reflected in the network structure (Conover et al. 2011). Prior research has found that events can shift public opinion, polarizing the population (Hebbelstrup Rye Rasmussen and Petersen 2023; Jiang et al. 2020; Rao et al. 2023). Although there are many alternative methods for measuring polarization on social networks, e.g., using network analysis (Conover et al. 2011), interaction behaviors and emotions (Del Vicario et al. 2016), and community embeddings (Waller and Anderson 2021), they are not well suited to measure shifts in polarization. Leveraging our multi-view user representation learning, we provide a new way to measure changes in polarization after significant events by tracking user embeddings, and apply it to measure polarization in the online discussions about abortion, a highly contentious issue in the American society. Our framework allows us to isolate specific topics and identify ones that grew more polarized.

Data

On June 24, 2022, the SCOTUS struck down federal protections for abortion rights. This event sparked many online discussions, in which users with different political ideologies expressed distinct views (Rao et al. 2023). We study a public dataset (Chang et al. 2023) containing posts with abortion-related keywords, such as "roevswade", "prochoice", and "prolife". The data spans the entire year of 2022. We select English posts in the U.S. from users with at least 20 posts. This gives us about 10M posts from 121K users. Using methodology described in Rao et al. (2023), we identify each user's political ideology (liberal/conservative), leaving us with 103K liberals and 18K conservatives. We use the learned user embeddings to track how these two ideological populations change in the user embedding space.

Measuring Event-Driven Polarization

We start with pre-training, which learns a user representation space for all liberal and conservative users using their posting histories in 2022. We perform the same text preprocessing as described in Detecting IO Drivers, except that we use `sentence-transformers/all-mpnet-base-v2`[2] for this English-only dataset. We also use following and follower counts as profile features and use a repost network to train link prediction objective.

Next, we fine-tune the model via few-shot learning with 358 political elites[3], such as U.S. politicians, with a known ideology. This aligns the embedding space to political ideology. We use a two-layer feed-forward network on top of SoMeR and fine-tune the model as described in Detecting IO Drivers. Figure 7 in Appendix shows t-SNE (Van der Maaten and Hinton 2008) representations of the learned embedding space for all users for the year 2022 both with and without fine-tuning. We see the separation of populations, with conservatives clustered in some regions, whereas liberals, who are 85% of all data, are distributed across the entire space. The analysis we discuss below describes results using the fine-tuned politically-aware embedding, but we find the same trends using the pre-trained embeddings.

In this politically-aware embedding space, to identify shifts in ideological polarization driven by the SCOTUS ruling, we measure how embeddings changed for users before versus after the ruling. We select the period of January 1st to May 2nd 2022 as the baseline period to avoid interference from a leak about this ruling on May 3rd, 2022. We then determine the period to observe the impact of ruling to be June 24th to November 11th 2022. We select this end date to reduce the confounding effect of the 2022 US midterm elections. Next, we select users who posted in both time periods, take their posting history in these two periods separately, and project these users onto the same politically aware embedding space we have learned. By visualizing the embeddings of the same users in the baseline period and after the SCOTUS ruling, we find a clear shift especially in the conservative population (Figure 2). In the baseline period, conservative users were more evenly distributed in the t-SNE embedding space, but moved closer together after the SCOTUS ruling. This indicates that conservative users became more similar in the content of their posts or in their behaviors.

To further quantify this effect, we find the k-nearest-neighbor (kNN) and check for each user the percentage of neighbors with the same ideology (in-group) and different ideology (out-group).We can infer how clustered each population is by the share of the nearest neighbors who are from their in-group. However, the share of out-group neighbors tells us how far away the two ideological populations are. Figure 3 shows the percent change in the mean of these four metrics across populations, from the baseline period to the after the ruling. Consistent with Figure 2, we see in the "All Data" row that the share of in-group neighbors increased for both conservatives and liberals and the share of out-group neighbors decreased. These indicate that *users with same ideology moved closer together, and users with different ideologies moved farther apart in the embedding space*, im-

[2]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3]https://github.com/sdmccabe/new-tweetscores

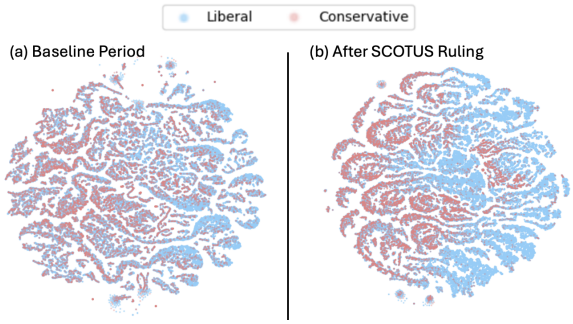

Figure 2: Users shifted in the embeddings space after the SCOTUS abortion ruling. Points in (a) are encoded with user post histories between January 1st to May 2nd, 2022. Points in (b) are encoded with the post histories from the same users between June 24th to November 8th, 2022. Points in (a) and (b) are both projected in the same embedding space.

plying that polarization increased. Conservatives grew especially clustered after the ruling. We perform the same analysis with k=50, 200, 500 nearest neighbors, all showing the same trends.

To dig deeper, we explored how this effect depends on the topic users discuss. Rao et al. (2023) identified topics discussed in each post, including religion, bodily autonomy, fetal rights, and women's health. We created one subset with posts related to liberal-centric topics, e.g., bodily autonomy and women's health, and another subset with posts related to conservative-centric topics, like religion and fetal rights. Then we perform the same analysis for each subset. Figure 3 shows a consistent overall trend that users with the same ideology move closer and users with different ideologies move farther away. Interestingly, we also observe that *each population coalesced when discussing partisan topics promoted by the opposite ideology* - in-group neighbors of conservative users increased more on Liberal-centric topics than on Conservative-centric topics, and similarly in-group neighbors of liberals increased more on Conservative-centric topics than on Liberal-centric topics. Users "united against a common enemy" in these online discourses.

Ablations

We also perform an ablation study on these data to assess how different types of evidence contribute to the shift of user embeddings we observe. Figure 4 compares three ablated models and the full model. Using the temporal model results in very little change in all four metrics, indicating that users did not change their activity patterns much after the ruling. Instead, we see much bigger changes when using the textual model. This implies that users of different ideologies diverged more in the topics and content they discussed after the ruling. The full SoMeR model shows smaller changes than the Temporal+Textual model. During pre-training, the full model learns the repost network aggregated over the entire year of 2022, including baseline period and after ruling period. With the repost network not changing between these two periods, user embeddings change less.

Figure 3: Users with same ideology moved closer after SCOTUS abortion ruling, and users with different ideologies moved away. The color represents the *percent change in the mean of these nearest neighbor metrics across populations* from baseline period to after ruling period. *** indicates that the means are significantly different in two time periods with p-value < 0.0001.

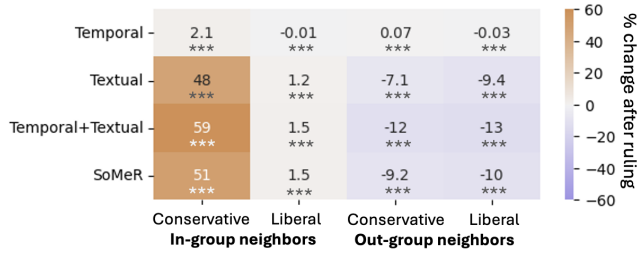

Figure 4: Comparison of changes observed in the embedding spaces learned by ablated models and the full model. Temporal features contributed little whereas textual features are the greater factor. The color represents the *percent change in the mean of these four metrics across populations* from baseline period to after ruling period. *** indicates that the means are significantly different in two time periods with p-value < 0.0001.

Predicting Participation in Hate Subreddits

Finally, we utilize SoMeR to predict participation in hate subreddits on Reddit. Online hate communities have numerous harmful effects on users (Schmitz et al. 2024) by radicalizing them (for the Study of Terrorism and to Terrorism START) and increasing hate crimes (Federal Bureau of Investigation 2021). Although there has been interest in what drives people to hate groups (Russo, Ribeiro, and West 2024), this analysis was often based on case studies of individual communities. Less understood is why users may be susceptible to extremism and hate. We aim to use SoMeR to predict whether a Reddit user who had never participated in any hate subreddits will become active weeks or even months in the future (on average, a user's last post fed into SoMeR was *22 days* before their first post in a hate subreddit). These results can help explain what drives people to join hate groups.

Data

We use a list of 168 hate subreddits collected by (Hickey et al. 2024), which consists of large banned subreddits with hate speech. We manually inspected the subreddits to verify

that they were in fact hate subreddits. We then extract all users who have ever posted in these subreddits until June, 2022 using data stored in Academic Torrent [4]. We take a random 200K subsample of all the hate users we collected. To construct a control user group, we collect the user names of 100K random users who ever made a post on Reddit until June, 2022.

We then collected all the posts made by each of these random and hate users and matched the random and hate users so that the random users and hate users were active roughly the same time to make them less trivially distinguishable (Luceri et al. 2023). To match random users with hate users, we kept track of which hate users had already been paired. For each random user, we filtered hate users who hadn't been matched, ensuring their first post was before becoming active in a hate subreddit. Then, we randomly selected a hate user active during the same time period as the random user. During training and testing, for each hate user, we only use their posts and comments before the date they became active on a hate subreddit, and we use the same time range for each matched control user. Within this dataset, we randomly select 4K matched pairs (8K users total) to have a manageable data size for fast experimentation. We keep the overall time range of the dataset from January 1, 2013 to December 31, 2022, because of the low amount of data before 2013. We then remove low-activity users who have posted and commented less than 10 times. This results in 3771 hateful users and 3444 control users, and 2.3 million posts and comments in total. We split data into 70%-15%-15% for training, validation, and testing, respectively.

Pre-training and Supervised Classification

We perform pre-training and supervised fine-tuning to predict whether a user would become active in a hate subreddit later. We first use the same data processing procedure as in IO driver detection, pre-processing text by removing URLs, emojis and all non-ASCII characters, using sentence-BERT `all-mpnet-base-v2` to compute the embeddings , and computing the first five components from PCA as the textual features. We aggregate data for each user by summing up their PCA embeddings of their posts in 3-day intervals and taking the middle day in the interval as the corresponding timestamp. This provides us with both temporal and textual features. For the network module, a reasonable option we have attempted is to use co-subreddit membership as the linkage (similar users participate in the same subreddits). However, users join subreddits at different times, resulting in a different co-activity network from time to time. These co-activity networks for user A may include user B who already joined the hate group, creating information leakage between training and testing. Moreover, because there is no fixed co-activity network, incorporating this feature into the training is very data-intensive and computationally infeasible. For this reason, we had to exclude the co-subreddit membership network feature, but encode users' subreddit membership information as the profile feature, by gathering the 50 most-posted subreddits of each user, and use a lookup table encoder to generate the profile embeddings.

With all the above features, we then pre-train the model to learn a user embeddings space in which users with similar behaviors are closer together. Next, we perform supervised fine-tuning using a two-layer feed-forward network for binary prediction of whether the user will become active in a hate subreddit later (hyperparameters and training procedure same as in Detecting IO Drivers).

Model Performance and Ablation

Performance There is no prior task-specific research for predicting user participation in hate communities, hence we first compare SoMeR with a simple baseline—using sentence-BERT embeddings averaged over user posts prior to becoming active in a hate subreddit to train the same two-layer feed-forward network for binary prediction. We also evaluate the more advanced URepL approach User2Vec and the multi-view approach SATAR as strong baselines (see User2Vec and SATAR evaluation details in Appendix). We evaluate all models with randomly bootstrapping training, validation and test data 10 times. Table 3 shows that SoMeR significantly outperforms the BERT baseline by 9%, SATAR by 20% and User2Vec by 5% on F1-scores, indicating the effectiveness of our method. User2Vec significantly outperforms averaged BERT embeddings by optimizing to predict the words and sentences from an author's previous posts. However, SoMeR still achieves superior performance compared to User2Vec. SATAR has surprisingly lower performance, possibly because its self-supervised objective is not fitted for detecting hate behavior.

	F1-Scores
BASELINE	
BERT	0.69 ± 0.01
User2Vec	0.74 ± 0.03
SATAR	0.59 ± 0.05
OURS	
Temporal	0.74 ± 0.01
Textual	0.76 ± 0.01
Temporal+Textual	0.77 ± 0.00
Temporal+Textual+Profile	$\mathbf{0.78 \pm 0.01}$

Table 3: F1 Scores on Predicting User Participation in Hate Subreddits.

Ablations Table 3 also shows our ablation study, where we see that the Temporal+Textual model outperforms the temporal model and textual model, and using temporal, textual, and profile data together further improves the F1 score. We confirm our results with t-SNE plots of embeddings generated by different ablated models in Appendix Figure 8.

Conclusion

In this work, we propose a universal multi-view user representation learning framework **SoMeR**. Our framework learns from a variety of user features including 1) temporal activities, 2) texts of their posts, 3) profile information, and

[4]https://files.pushshift.io/reddit/

4) network connections. We show that it is versatile and generalizable to different downstream tasks and across different social platforms, including detecting IO drivers, measuring online political polarization, and predicting future user participation in hate subreddits.

There are several limitations and promising avenues for future exploration. First, due to computational constraints, we applied PCA for dimensionality reduction on BERT embeddings when processing user history textual data, which lowers the model performance (see Appendix). We had explored alternative methods such as UMAP (McInnes, Healy, and Melville 2018), but UMAP on large data sets (often 10M or more) requires significant time and computation as well, which is a known challenge. If more compute resources are available, using more PC components or the full BERT embeddings would further improve the performance. Second, using network features has been shown to contribute to model learning. Future work may use network graph embedding (Grover and Leskovec 2016; Huang et al. 2021; Zhou et al. 2023) to better learn network-level representations. Third, while video, image, and audio modalities have been explored in representation learning, challenges remain in aligning heterogeneous modalities while preserving their characteristics. Pre-training these models often relies heavily on the availability of well-aligned multimodal datasets (Xu, Zhu, and Clifton 2023). Thus, current research on universal URepL that incorporates these modalities is limited. We acknowledge their potential and aim to incorporate them into future iterations of our framework. Fourth, in measuring polarization application, the machine-labeled user political ideologies, although extensively validated by (Rao et al. 2023), are not perfect, thus other ideology detection methods should be analyzed to test the robustness of these results. Last but not least, the utilization of temporal features also opens up exciting possibilities to track how users and communities evolve over time. Detecting change points in user representations (Guo et al. 2024) will be an interesting direction to expand this framework to a more powerful tool.

Ethical Statement

Our framework utilizes social media data collected from X and Reddit. Although raw data may include personally identifiable information or offensive content, we have taken proactive steps to mitigate privacy and ethical concerns. Specifically, we have documented the data and model usage processes in detail and implemented measures such as removing mentions (@) to minimize the use of personal information. All presented results are aggregated, ensuring no specific account information is disclosed. In addition, we acknowledge the potential risks associated with our framework, including its misuse for harmful purposes, such as predicting future user participation in hate communities. To address these concerns, we have carefully documented the ethical considerations of our framework and committed to tightly regulating training data access, as well as closely monitoring its applications. These measures are intended to mitigate risks to the greatest extent possible while enabling responsible scientific research and ensuring compliance with privacy norms and ethical standards. By emphasizing privacy, fairness, and transparency, we aim to contribute positively to the understanding of social media behaviors while minimizing risks of harm. We remain committed to refining our approach to align with evolving ethical standards and societal expectations.

Acknowledgments

This work was supported in part by DARPA under contract HR001121C0168.

References

AlMahmoud, H.; and AlKhalifa, S. 2018. TSim: a system for discovering similar users on Twitter. *Journal of Big Data*, 5(1): 39.

Amir, S.; Coppersmith, G.; Carvalho, P.; Silva, M. J.; and Wallace, B. C. 2017. Quantifying mental health from social media with neural user embeddings. In *Machine Learning for Healthcare Conference*, 306–321. PMLR.

Amir, S.; Wallace, B. C.; Lyu, H.; and Silva, P. C. M. J. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.

Bhargava, P.; Phan, T.; Zhou, J.; and Lee, J. 2015. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *Proceedings of the 24th international conference on world wide web*, 130–140.

Center, P. R. 2017. The partisan divide on political values grows even wider. *Trust, facts and democracy*.

Chang, R.-C.; Rao, A.; Zhong, Q.; Wojcieszak, M.; and Lerman, K. 2023. # RoeOverturned: Twitter Dataset on the Abortion Rights Controversy. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 17, 997–1005.

Cheng, M.; Yuan, F.; Liu, Q.; Xin, X.; and Chen, E. 2021. Learning Transferable User Representations with Sequential Behaviors via Contrastive Pre-training. In *2021 IEEE International Conference on Data Mining (ICDM)*, 51–60.

Church, K. W. 2017. Word2Vec. *Natural Language Engineering*, 23(1): 155–162.

Conover, M.; Ratkiewicz, J.; Francisco, M.; Gonçalves, B.; Menczer, F.; and Flammini, A. 2011. Political polarization on twitter. In *Proceedings of the international aaai conference on web and social media*, volume 5, 89–96.

Dahiya, S.; Kumar, G.; and Yadav, A. 2022. A Contextual Framework to Find Similarity Between Users on Twitter. In *Proceedings of Second Doctoral Symposium on Computational Intelligence: DoSCI 2021*, 793–805. Springer.

Del Tredici, M.; Marcheggiani, D.; Walde, S. S. i.; and Fernández, R. 2019. You shall know a user by the company it keeps: Dynamic representations for social media users in nlp. *arXiv preprint arXiv:1909.00412*.

Del Vicario, M.; Vivaldo, G.; Bessi, A.; Zollo, F.; Scala, A.; Caldarelli, G.; and Quattrociocchi, W. 2016. Echo chambers: Emotional contagion and group polarization on facebook. *Scientific reports*, 6(1): 37825.

DellaPosta, D.; Shi, Y.; and Macy, M. 2015. Why do liberals drink lattes? *American Journal of Sociology*, 120(5): 1473–1511.

Fazelnia, G.; Gupta, S.; Keum, C.; Koh, M.; Anderson, I.; and Lalmas, M. 2024. Generalized User Representations for Transfer Learning. *arXiv preprint arXiv:2403.00584*.

Federal Bureau of Investigation. 2021. Federal Bureau of Investigation Crime Data Explorer. https://cde.ucr.cjis.gov/LATEST/webapp/\#/pages/explorer/crime/crime-trend.

Feng, S.; Wan, H.; Wang, N.; Li, J.; and Luo, M. 2021. Satar: A self-supervised approach to twitter account representation learning and its application in bot detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3808–3817.

for the Study of Terrorism, N. C.; and to Terrorism (START), R. 2017. Profiles of Individual Radicalization in the United States. http://www.start.umd.edu/pirus.

Gebru, T.; Morgenstern, J.; Vecchione, B.; Vaughan, J. W.; Wallach, H.; III, H. D.; and Crawford, K. 2021. Datasheets for datasets. *Commun. ACM*, 64(12): 86–92.

Graves, A. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, 37–45.

Grootendorst, M. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.

Guo, S.; He, Z.; Rao, A.; Morstatter, F.; Brantingham, J.; and Lerman, K. 2024. The Pulse of Mood Online: Unveiling Emotional Reactions in a Dynamic Social Media Landscape. *arXiv preprint arXiv:2401.06275*.

Hallac, I. R.; Ay, B.; and Aydin, G. 2021. User representation learning for social networks: An empirical study. *Applied Sciences*, 11(12): 5489.

Hebbelstrup Rye Rasmussen, S.; and Petersen, M. B. 2023. The event-driven nature of online political hostility: How offline political events make online interactions more hostile. *PNAS nexus*, 2(11): pgad382.

Hickey, D.; Fessler, D. M.; Lerman, K.; and Burghardt, K. 2024. The Peripatetic Hater: Predicting Movement Among Hate Subreddits. *arXiv preprint arXiv:2405.17410*.

Hou, Y.; Mu, S.; Zhao, W. X.; Li, Y.; Ding, B.; and Wen, J.-R. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 585–593.

Huang, H.; Shi, R.; Zhou, W.; Wang, X.; Jin, H.; and Fu, X. 2021. Temporal Heterogeneous Information Network Embedding. In *IJCAI*, 1470–1476.

Irani, D.; Wrat, A.; and Amir, S. 2021. Early Detection of Online Hate Speech Spreaders with Learned User Representations. In *CLEF (Working Notes)*, 2004–2010.

Iyengar, S.; and Krupenkin, M. 2018. The strengthening of partisan affect. *Political Psychology*, 39: 201–218.

Jiang, J.; Chen, E.; Yan, S.; Lerman, K.; and Ferrara, E. 2020. Political polarization drives online conversations about COVID-19 in the United States. *Human Behavior and Emerging Technologies*, 2(3): 200–211.

Krishnan, A.; Sharma, A.; and Sundaram, H. 2018. Insights from the long-tail: Learning latent representations of online user behavior in the presence of skew and sparsity. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 297–306.

Lai, Y.-Y.; and Neville, J. 2020. MERL: Multi-view edge representation learning in social networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 675–684.

Li, S.; and Zhao, H. 2021. A survey on representation learning for user modeling. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4997–5003.

Liu, H.; Chen, Y.; Li, P.; Zhao, P.; and Wu, X. 2023. Enhancing review-based user representation on learned social graph for recommendation. *Knowledge-Based Systems*, 266: 110438.

Luceri, L.; Pantè, V.; Burghardt, K.; and Ferrara, E. 2023. Unmasking the web of deceit: Uncovering coordinated activity to expose information operations on twitter. *arXiv preprint arXiv:2310.09884*.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

McPherson, M.; Smith-Lovin, L.; and Cook, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1): 415–444.

Mueen, A.; Chavoshi, N.; Abu-El-Rub, N.; Hamooni, H.; and Minnich, A. 2016. AWarp: Fast warping distance for sparse time series. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 350–359. IEEE.

Müller, K.; and Schwarz, C. 2021. Fanning the flames of hate: Social media and hate crime. *Journal of the European Economic Association*, 19(4): 2131–2167.

Nwala, A. C.; Flammini, A.; and Menczer, F. 2023. A language framework for modeling social media account behavior. *EPJ Data Science*, 12(1): 33.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pacheco, D.; Hui, P.-M.; Torres-Lugo, C.; Truong, B. T.; Flammini, A.; and Menczer, F. 2021. Uncovering coordinated networks on social media: methods and case studies. In *Proceedings of the international AAAI conference on web and social media*, volume 15, 455–466. AAAI Press.

Pan, S.; and Ding, T. 2019. Social media-based user embedding: A literature review. *arXiv preprint arXiv:1907.00725*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Purificato, E.; Boratto, L.; and De Luca, E. W. 2024. User Modeling and User Profiling: A Comprehensive Survey. *arXiv preprint arXiv:2402.09660*.

Qian, J.; ElSherief, M.; Belding, E. M.; and Wang, W. Y. 2018. Leveraging intra-user and inter-user representation learning for automated hate speech detection. *arXiv preprint arXiv:1804.03124*.

Rao, A.; Chang, R.-C.; Zhong, Q.; Lerman, K.; and Wojcieszak, M. 2023. Tracking a Year of Polarized Twitter Discourse on Abortion. *arXiv preprint arXiv:2311.16831*.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ren, X.; Wei, W.; Xia, L.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, 3464–3475.

Ribeiro, M.; Calais, P.; Santos, Y.; Almeida, V.; and Meira Jr, W. 2018. Characterizing and detecting hateful users on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Russo, G.; Ribeiro, M. H.; and West, R. 2024. Stranger Danger! Cross-Community Interactions with Fringe Users Increase the Growth of Fringe Communities on Reddit. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 18, 1342–1353.

Sawhney, R.; Joshi, H.; Shah, R. R.; and Flek, L. 2021. Suicide Ideation Detection via Social and Temporal User Representations using Hyperbolic Learning. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2176–2190. Online: Association for Computational Linguistics.

Schmitz, M.; Muric, G.; Hickey, D.; and Burghardt, K. 2024. Do users adopt extremist beliefs from exposure to hate subreddits? *Social Network Analysis and Mining*, 14(1): 22.

Shen, Y.; Jiang, X.; Li, Z.; Wang, Y.; Xu, C.; Shen, H.; and Cheng, X. 2023. UniSKGRep: A unified representation learning framework of social network and knowledge graph. *Neural Networks*, 158: 142–153.

Shin, K.; Kwak, H.; Kim, S. Y.; Ramström, M. N.; Jeong, J.; Ha, J.-W.; and Kim, K.-M. 2023. Scaling law for recommendation models: Towards general-purpose user representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 4596–4604.

Tipirneni, S.; and Reddy, C. K. 2022. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6): 1–17.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Waller, I.; and Anderson, A. 2021. Quantifying social organization and political polarization in online platforms. *Nature*, 600(7888): 264–268.

Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1235–1244.

Wang, W.; Yin, H.; Du, X.; Hua, W.; Li, Y.; and Nguyen, Q. V. H. 2019. Online user representation learning across heterogeneous social networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 545–554.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Wen, Q.; Sun, L.; Yang, F.; Song, X.; Gao, J.; Wang, X.; and Xu, H. 2020. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*.

Xu, P.; Zhu, X.; and Clifton, D. A. 2023. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10): 12113–12132.

Yuan, F.; Zhang, G.; Karatzoglou, A.; Jose, J.; Kong, B.; and Li, Y. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 696–705.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 2021. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, 2114–2124. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383325.

Zhou, W.; Huang, H.; Shi, R.; Song, X.; Lin, X.; Wang, X.; and Jin, H. 2023. Temporal Heterogeneous Information Network Embedding via Semantic Evolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 13031–13042.

Zhuang, F.; Zhang, Z.; Qian, M.; Shi, C.; Xie, X.; and He, Q. 2017. Representation learning via dual-autoencoder for recommendation. *Neural Networks*, 90: 83–89.

Checklist

1. For most authors...

   (a) Would answering this research question advance science without violating social contracts, such as violating privacy norms, perpetuating unfair profiling, exacerbating the socio-economic divide, or implying disrespect to societies or cultures? See details in Ethical Statement.

(b) Do your main claims in the abstract and introduction accurately reflect the paper's contributions and scope? Yes

(c) Do you clarify how the proposed methodological approach is appropriate for the claims made? Yes

(d) Do you clarify what are possible artifacts in the data used, given population-specific distributions? Yes

(e) Did you describe the limitations of your work? Yes

(f) Did you discuss any potential negative societal impacts of your work? Yes

(g) Did you discuss any potential misuse of your work? Yes

(h) Did you describe steps taken to prevent or mitigate potential negative outcomes of the research, such as data and model documentation, data anonymization, responsible release, access control, and the reproducibility of findings? Yes

(i) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes

2. Additionally, if your study involves hypotheses testing...

(a) Did you clearly state the assumptions underlying all theoretical results? Yes

(b) Have you provided justifications for all theoretical results? N/A

(c) Did you discuss competing hypotheses or theories that might challenge or complement your theoretical results? N/A

(d) Have you considered alternative mechanisms or explanations that might account for the same outcomes observed in your study? Yes

(e) Did you address potential biases or limitations in your theoretical framework? Yes

(f) Have you related your theoretical results to the existing literature in social science? Yes

(g) Did you discuss the implications of your theoretical results for policy, practice, or further research in the social science domain? Yes

3. Additionally, if you are including theoretical proofs...

(a) Did you state the full set of assumptions of all theoretical results? N/A

(b) Did you include complete proofs of all theoretical results? N/A

4. Additionally, if you ran machine learning experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? Yes

(e) Do you justify how the proposed evaluation is sufficient and appropriate to the claims made? Yes

(f) Do you discuss what is "the cost" of misclassification and fault (in)tolerance? Yes

5. Additionally, if you are using existing assets (e.g., code, data, models) or curating/releasing new assets, **without compromising anonymity**...

(a) If your work uses existing assets, did you cite the creators? Yes

(b) Did you mention the license of the assets? Yes

(c) Did you include any new assets in the supplemental material or as a URL? Yes

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? Yes

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? Yes

(f) If you are curating or releasing new datasets, did you discuss how you intend to make your datasets FAIR? N/A

(g) If you are curating or releasing new datasets, did you create a Datasheet for the Dataset (see Gebru et al. (2021))? N/A

6. Additionally, if you used crowdsourcing or conducted research with human subjects, **without compromising anonymity**...

(a) Did you include the full text of instructions given to participants and screenshots? N/A

(b) Did you describe any potential participant risks, with mentions of Institutional Review Board (IRB) approvals? N/A

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? N/A

(d) Did you discuss how data is stored, shared, and deidentified? N/A

Appendix

Verifying Model Validity with Synthetic Data

In this section, we use ablation to test the effectiveness of SoMeR embeddings.

First, we generate two datasets with different number of clusters. These clusters, each containing 1000 samples, mimic user populations with different behaviors. Each sample has a time series with length of 400 in the time domain, and five independently modeled feature categories. We draw each sample and its values of each feature independently from Poisson distributions $X \sim Poisson(\lambda)$ to mimic the daily post counts, and control λ for noise level. In addition, we inject peaks into the timeline of these samples, to mimic significant events happening. A peak is independently drawn from a Poisson distribution $X_{peak} \sim Poisson(\lambda_{peak})$ and then *imposed* on top of the baseline.

The first dataset simulates an easy scenario and contains three clusters which are more distinguishable from each other. We make it simple by aligning the events (injected peaks) at the same time points for all the samples in each cluster.

Data 1 Simple scenario with 3 clusters:

- *Cluster 0*: all samples have three peaks at specific time points, each peak lasts for 3 time steps and are drawn from $Poisson(\lambda_{peak} = 10)$; samples have higher activity levels - feature values vary by changing their $\lambda \in range(start = 4, stop = 6, step = 0.5)$.
- *Cluster 1*: all samples have three peaks exactly same as in Cluster 0; samples have lower activity levels - feature values vary by changing their $\lambda \in range(start = 0.5, stop = 2, step = 0.5)$.
- *Cluster 2*: all samples have three peaks different from Cluster 0; samples have lower activity levels - feature values vary by changing their $\lambda \in range(start = 0.5, stop = 2, step = 0.5)$.

We run our model with just the textual (represented by these activity feature values) and the temporal modules (represented by the indices of the data points in each sample). The hyperparameters and the setups are the same as described in Methods.

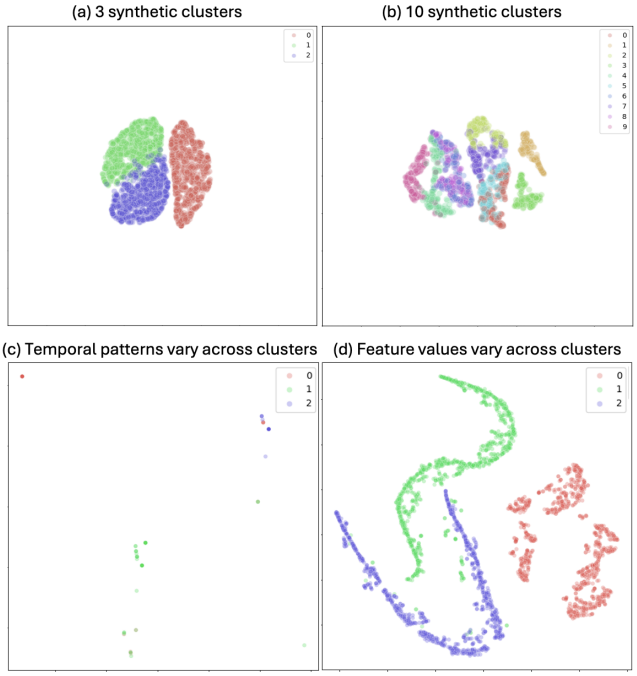

Figure 5: t-SNE of embedding spaces learned from synthetic datasets that (a) simulates a simple scenario with three clusters, (b) simulates a hard scenario with 10 clusters and more noise, (c) simulates the scenario that temporal patterns vary across clusters, and (d) simulates the scenario that feature values across clusters.

From Figure 5(a) we see that our approach nicely distinguish these three clusters. Interestingly, Cluster 1 and 2, having the same population activity level are closer, whereas Cluster 0 with higher activity levels are farther. This implies that activity level, rather than temporal patterns of peaks, might have a heavier impact on the embedding space learned.

The second dataset simulates a harder scenario where the data contains 10 different clusters. Some clusters are more similar with each other, and some clusters have higher noise level. Thus these clusters are harder to distinguish from each other.

Data 2 Harder scenario with 10 clusters:

- We vary the number of peaks across different clusters: $\#peaks \in [2, 3, 4, 5]$. For clusters with the same number of peaks, the positions of the peaks are the same.
- We vary the intensity of peaks across different clusters. The peak intensity is changed with λ_{peak} when drawing from the Poisson distribution. $\lambda_{peak} \in [1, 2, 5, 10]$;
- We vary the activity levels across different clusters. Five clusters have lower activity level ($\lambda \in range(start = 0.5, stop = 2, step = 0.5)$), and the other five clusters have higher activity level ($\lambda \in range(start = 4, stop = 6, step = 0.5)$).

Figure 5(b) shows the learned embeddings from 10 different clusters. We can see that most clusters are well separated from the others, whereas some clusters that are more similar are closer to each other or overlapping. In general, the samples from each cluster are closely placed in the embedding space, meaning that our model has learned the similarity of samples very well.

Furthermore, to show that it can learn from both temporal activities (timestamps in the triplets) and textual features (features and values in the triplets). We design the following two datasets. One contains three clusters, of which member samples have highly similar temporal patterns but very distinct feature values. In addition, the temporal patterns among the three clusters are different. This is to test if the model can learn heterogeneity in terms of temporal patterns. The second dataset contains three clusters of which member samples have highly similar feature values but distinct temporal patterns. This is to test if model can learn from heterogeneous features and values.

Data 3 Temporal patterns vary across clusters:

- *Cluster 0*: all samples have three peaks at specific time points and each peak lasts for 3 time steps;
- *Cluster 1*: all samples have three peaks different from Cluster 0 and each peak lasts for 3 time steps;
- *Cluster 2*: all samples have six peaks at specific time points and each peak lasts for 3 time steps.

Within each cluster, samples have low activity level - feature values vary by changing their $\lambda \in range(start = 0.5, stop = 2, step = 0.5)$.

Data 4 Feature values vary across clusters:

- *Cluster 1*: all samples have low activity ($\lambda = 0.5$)
- *Cluster 2*: all samples have medium activity ($\lambda = 2$)

- *Cluster 3*: samples all have high activity ($\lambda = 5$)

For all three clusters in Data 4, each sample has three peaks randomly placed in the time domain ($\lambda_{peak} = 10$ when drawing for the peak); and each peak lasts for 3 time steps.

Figure 5 shows that on both datasets we can see the segregation of clusters in the t-SNE of embedding space. Surprisingly, on data 3 which varies in temporal patterns across clusters, clusters are very condensed with many samples overlapping on top of each other. For example, almost all the samples in the red cluster in Figure 5(c) are concentrated on the top left corner. This is probably due to the highly synthetic data we created where all samples have the exact temporal patterns in a cluster. It also indicates that the model can capture the temporal feature well, producing highly similar embeddings for samples with similar temporal patterns. In Figure 5(d), clusters vary in feature values, specifically the amount of activities. We also observe that data are segregated in alignment with the ground truth cluster labels, indicating the model's ability to learn heterogeneity in features and values. In summary, this part shows that our model can learn from both temporal and feature values in the data.

Hyperparameters

Hyperparameters used in this work and their selection procedure are presented in Table 4. We used 2 RTX A6000 GPUs for training.

Evaluating User2Vec Baseline

For both IO driver detection and Reddit hate user detection, we use the code repository https://github.com/samiroid/U2V to compute the User2Vec embeddings from the exact same data as in other experiments. We than train a two-layer feed-forward neural network, with same parameters and procedures as in SoMeR fine-tuning step, and obtain the F1-scores.

Evaluating SATAR Baseline

Detecting IO Drivers We compare our framework with a strong baseline SATAR (Feng et al. 2021) on detecting IO drivers. SATAR is a more advanced multi-view user representation learning approach that combines textual, profile and network features. The authors uses it for social bot detection. To adapt SATAR for IO driver detection, we keep the model architecture and most of the setup the same as the original paper [5]. For SATAR's profile property module, it originally uses 15 true-or-false property items, e.g. "profile uses background image", 5 numerical property items, e.g. "favorites count", and "location". However, the authors did not specify the 15 binary attributes. In addition, some Twitter user attributes like location are inaccessible in our data. We therefore use the textual `user profile description` as the profile features, encoding the first 10 words into word2vec embeddings. The description often include a short summary of the user and many times include locations too, and we believe it is a good substitute for the profile property features. Similar to the original paper, we first perform self-supervised pre-training for follower count prediction, and fine-tune SATAR with IO driver labeled data. We pre-train for 60 epochs to make sure of model convergence. The following-follower network module in SATAR iteratively uses and updates the neighbor embeddings for each user in the network, resulting in more epochs to converge. We then fine-tune for 5 epochs. The batch size is 4. The data used and the train-validation-test splits are the same as those for training SoMeR. Other model structure parameters are kept the same as the original SATAR paper.

Predicting Participation in Hate Subreddits SATAR is originally designed for X platform with many X-specific features. For predicting participation in hate subreddit, we had to adapt SATAR to Reddit by (1) disregard the followinng-follower network feature, (2) using top 50 subreddit membership as the profile feature, and (3) train SATAR with the self-supervised objective to predict total number of "ups" a user receives instead of the follower counts. We then supervisedly fine-tune SATAR with hate user labels. These modifications align with our setup for SoMeR, offering a fair comparison. We pre-train for 5 epochs. We do not include the following-follower network module for SATAR in this application and therefore the model quickly overfits to the pre-training "ups" count signals. We then fine-tune for 5 epochs. The batch size is 4. Other model structure parameters are kept the same as the original SATAR paper.

Embeddings for IO drivers

In the application of detecting IO drivers, to further understand why SoMeR performs less satisfactory on the Egypt-UAE data, we investigate in how embeddings look like using different modules and plot the t-SNE in Figure 6. The sentence-BERT embeddings averaged across the entire time (Averaged BERT in the figure) corresponds to the features used in the Text Similarity baseline model. We notice that the IO drivers already separate well from the control users in this embedding space. This indicates that text similarity indeed is a very important feature in this campaign, consistent with the very high F1 score of this baseline model. The campaign might have promoted to use similar language as their tactic. On the other hand, the embeddings from our approach only using textual features are not separated so well. This again could be due to our dimension reduction step on the sentence-BERT embeddings. Second, Egypt-UAE is the smallest campaign here and our models are pretrained on only 2.5K users. We hypothesize that the models trained on small datasets and under-fitted can also result in less ideal performance.

Nevertheless, we observe that the full SoMeR model can generate more separable embeddings compared to the other ablated models. This is especially obvious when comparing embeddings from Temporal+Textual and the full SoMeR model (Temporal+Textual+Network). This shows the significance of the network module, which aligns well with the significant jump of F1-scores between these two models. From these embedding plots, we can see how combining different modules improves model's representation learning

[5] Code for SATAR is publically available at https://github.com/BunsenFeng/SATAR/tree/main

	Hyperparameter	Value	Grid Search / Notes
P	hidden dimension for transformer key query and value	16	P = K/H from prior work (Tipirneni and Reddy 2022)
K	hidden dimension for other model weights	64	[32, 64, 128]
L	number of transformer layers	2	small transformer sufficient for our data
H	number of attention heads	2	small transformer sufficient for our data
τ	temperature scaling parameter	0.5 for IO driver detection 3 for measuring political polarization 3 for predicting hateful Reddit users	[0.5, 1, 3, 6]
γ	noise parameter in data augmentation for contrastive learning	2	[0.1, 0.5, 1, 2, 5]
λ	loss balancing parameter	1	[0.1, 1, 5, 10]
lr	learning rate	5e-5	
e	maximum epoch	60	early stopping based on val. loss
b	batch size	32 when link prediction module used 128 for other cases	[32, 64, 128] doesn't impact performance
dr	drop-out rate in supervised finetuning	0.3	[0, 0.3, 0.5]
#PC	number of PCA components	5	[5,10,20]

Table 4: Hyperparameters used in this work.

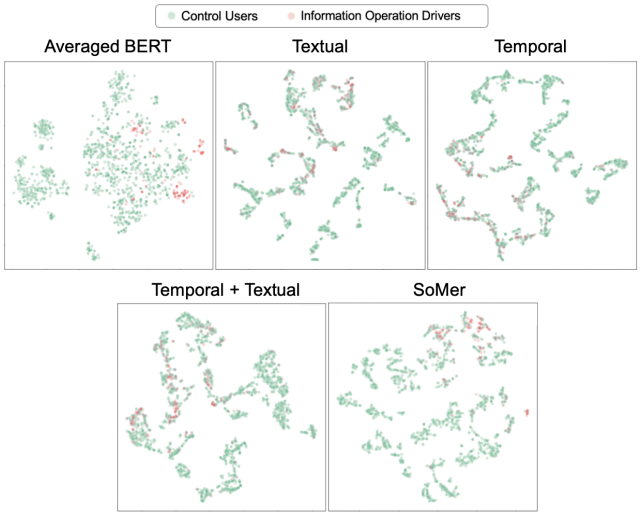

Figure 6: t-SNE of user embeddings in the Egypt-UAE data using different models.

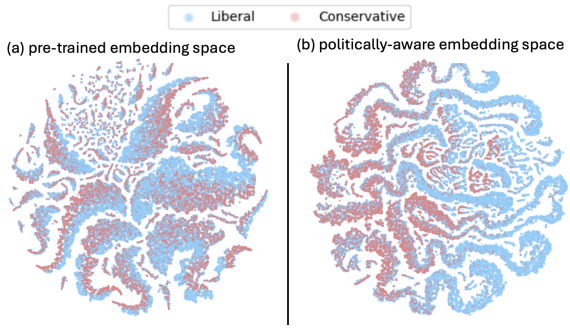

Figure 7: (a) t-SNE of pre-trained user embedding space learned with both ideology populations over the entire year of 2022. (b) t-SNE of the space fine-tuned with data from elite users with known ideologies, making the embedding space more politically-aware.

ability.

Effectiveness of fine-tuning models on partisanship

In the applications of uncovering polarization, in Figure 7 we show t-SNE representations of models trained on the SCOTUS decision dataset both with and without fine-tuning. We find that fine-tuning more effectively separates the political ideology dimension along the x-axis.

Embeddings for hateful Reddit users

For predicting hate users in the Reddit data, Figure 8 shows the t-SNE plots of embeddings learned by different models. We observe that the baseline, sentence-BERT embeddings averaged across time, do not distinguish between control and hateful users, resulting in a lower F1 score on user classification task. It is also interesting to see how Temporal+Textual+Profile model better separates the control and hateful users into two clusters, compared to other ablated models. This demonstrates the benefits of our multi-view approach that combines different features.

Effect of PCA on BERT Embeddings

Due to computational constraints, we applied PCA for dimensionality reduction on BERT embeddings when processing user history textual data. In IO driver detection, we notice the performance gap between the Text Similarity baseline and our SoMeR approach for the Egypt-UAE campaign. We hypothesize that it is due to our method utilizing only the

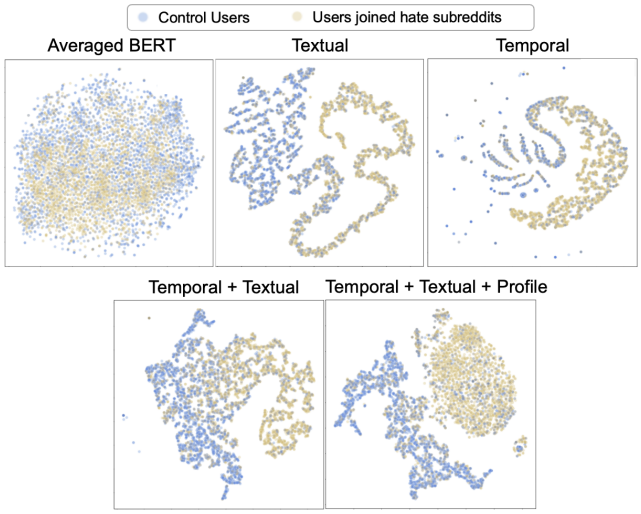

Figure 8: t-SNE plots of embeddings generated by different models.

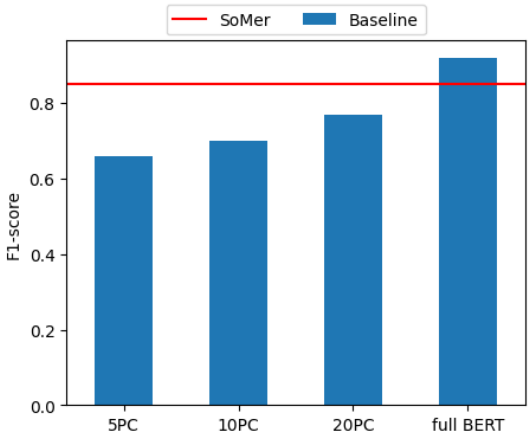

Figure 9: Classification performance using different numbers of PC components or the full BERT embeddings on Egypt-UAE IO driver detection data.

first five PCA components on BERT embeddings, versus the Text Similarity baseline using the full BERT embeddings.

Here we analyze the effect of applying PCA dimension reduction on classification performance. We use users' average BERT embeddings across time (same as Text Similarity baseline), and also obtain the first 5, 10, and 20 PCs from PCA on the average BERT embeddings. We then build a two-layer feed-forward network for binary classification using these embeddings. We train with the Egypt-UAE IO driver data in the same way as in other experiments, and compare the performances with different embedding settings. Figure 9 shows that performing PCA on top of BERT embeddings does significantly decrease the classification F1-scores, which is consistent with our hypothesis. Using fewer numbers of PCs also gives lower performance. If only using 5 PCs, the Text Similarity baseline would only have an F1-score of 0.66, which is significantly lower than our SoMeR performance 0.85, which also uses 5 PCs.

Interestingly, using 5, 10 or 20 PCs in the SoMeR approach only improves the overall performance a little (F1 ranges from 0.8 to 0.85). This may be because SoMeR not only uses textual features but also other multi-view features such as temporal activities and network. If more compute resources are available, using more PC components or the full BERT embeddings might further improve SoMeR performance.

Scalability Experiments

In three applications, we have already demonstrated the scalability of SoMeR, which is able to handle datasets with various sizes from 200K up to 17M texts. We further analyze the robustness of SoMeR by down-sampling theses datasets to 75%, 50% and 25% subsets, and compare the F1 scores with using the full datasets. Table 5 shows that down-sampling has little impact on bigger datasets. For example, by only using 25% subset of the China data, we still get a high F1 score of 0.94. For smaller data like Egypt-UAE, down-sampling has a bigger effect. However, the decrease of the performance shows an elbow-like pattern. When subsetting upto 50%, SoMeR can still hold relatively good performance with little F1 decrease. This shows the robustness of SoMeR.

Processing different sizes of subsets does not significantly change the run-time of SoMeR, this is due to data aggregation during triplet data embedding step. Even with larger subsets or full data, SoMeR is able to aggregate and transform the data into triplet time series for each user, which greatly reduces the computation burden in the following steps.

	China (17M)		**Egypt-UAE (4.5M)**		**Venezuela (10M)**	
Data Subset	F1-score	Run-time (min)	F1-score	Run-time (min)	F1-score	Run-time (min)
full data	0.99	95.6	0.88	25.7	0.84	44.8
75%	0.95	91.9	0.77	24.7	0.81	42.5
50%	0.94	87.2	0.73	23.9	0.77	41.1
25%	0.94	86.3	0.58	23.6	0.69	39.0

Table 5: SoMeR performance and run-time with different subset size of IO driver detection data.