TIES: Temporal Interaction Embeddings For Enhancing Social Media Integrity At Facebook

Nima Noorshams
Core Data Science - Facebook
nshams@fb.com

Saurabh Verma
Core Data Science - Facebook
saurabh08@fb.com

Aude Hofleitner
Core Data Science - Facebook
aude@fb.com

ABSTRACT

Since its inception, Facebook has become an integral part of the online social community. People rely on Facebook to make connections with others and build communities. As a result, it is paramount to protect the integrity of such a rapidly growing network in a fast and scalable manner. In this paper, we present our efforts to protect various social media entities at Facebook from people who try to abuse our platform. We present a novel Temporal Interaction EmbeddingS (TIES) model that is designed to capture rogue social interactions and flag them for further suitable actions. TIES is a supervised, deep learning, production ready model at Facebook-scale networks. Prior works on integrity problems are mostly focused on capturing either only static or certain dynamic features of social entities. In contrast, TIES can capture both these variant behaviors in a unified model owing to the recent strides made in the domains of graph embedding and deep sequential pattern learning. To show the real-world impact of TIES, we present a few applications especially for preventing spread of misinformation, fake account detection, and reducing ads payment risks in order to enhance Facebook platform's integrity.

CCS CONCEPTS

• **Computing methodologies** → *Artificial intelligence*; *Knowledge representation and reasoning*;

KEYWORDS

temporal embeddings, sequence modeling, social media integrity

ACM Reference Format:
Nima Noorshams, Saurabh Verma, and Aude Hofleitner. 2020. TIES: Temporal Interaction Embeddings For Enhancing Social Media Integrity At Facebook. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3394486.3403364

1 INTRODUCTION

People use online social media such as Facebook to connect with family and friends, build communities, and share experiences every day. But the rapid growth of social media in recent years has introduced several challenges. First is the rise of fake and inauthentic accounts, which pose potential threats to the safety of online communities. Second is the rise of threatening and/or disparaging content such as hate-speech, misinformation, bullying, terrorist propaganda, etc. These can occur both through authentic as well as inauthentic accounts. In Q1-2019 Facebook has acted on 4 million and 2.6 million pieces of content for violating hate-speech and bullying policies, respectively [2, 3]. We broadly refer to these as *social media integrity* challenges.

There is a volume of research on fake account detection in online social media. Graph propagation techniques to detect spams over user-following graphs have been proposed in [24, 32]. Integro [8] focuses on predicting "victims" using user-level features and performs a random walk style algorithm on a modified graph. There have been other works that focus on utilizing the graph structure, by clustering and identifying cohorts of malicious actors that share common IP addresses and other common networking properties [29, 31, 33]. Researchers in [20] create graphs based on user activities in order to detect fake engagements. Similarly, hand-designed features based on activity has been used in [9].

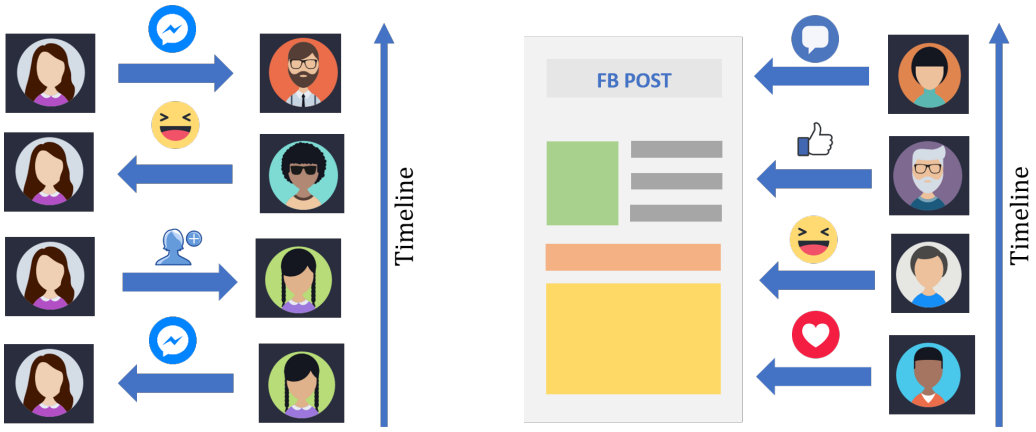
There is also a volume of research on content-based integrity problems [5, 6]. Natural language processing techniques have been widely used for hate-speech and cyberbullying detection [10, 14, 23, 26, 34]. Simple token and character-level n-grams are included in the feature set by [10, 26, 34]. Word topic distribution using Latent Dirichlet Allocation has been used by [14] to detect cyberbullying on Instagram. Alternatively, paragraph embedding for hate-speech detection was proposed in [23]. Dinakar et al. [17] presented a knowledge-based approach utilizing domain specific assertions in detecting hate-speech. In [14, 15] authors combine image and other media features with text, essentially incorporating context through multimodal information. User meta-data such as the violation history, number of profane words in prior comments, gender, etc. have also been shown predictive [22, 36]. More recently, deep-learning has been used to fight child pornography [25], hate-speech [13, 27], and misinformation [12, 16].

The majority of previous approaches tackling integrity challenges are static in nature. More specifically, they utilize engineered user-level, graph, or content features that do not alter in time. However, entities on social media (accounts, posts, stories, Groups, Pages, etc.) generate numerous interactions from other entities over time (see Figure 1). For instance,

- posts get likes, shares, comments, etc. by users, or
- accounts send or reject friend requests, send or block messages, etc. from other accounts.

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403364



(a) user-user interactions for fake account detection.

(b) post-user interactions for hate-speech detection.

Figure 1: Entities on social media interact with each other in numerous ways. Interactions generated by bad entities differ from normal entities. We can enhance the platform integrity by capturing/encoding these interactions.

These temporal sequences can potentially reveal a lot about entities to which they belong. The manner in which fake accounts behave is different from normal accounts. Hateful posts generate different type of engagements compared to regular posts. Not only the type but also the target of these engagements can be informative. For instance, an account with history of spreading hate or misinformation sharing or engaging positively with a post can be indicative of a piece of questionable content.

In this work, we present Temporal Interaction EmbeddingS (TIES), a supervised deep-learning model for encoding interactions between social media entities for integrity purposes. As its input, TIES takes a sequence of (source, target, action) in addition to miscellaneous source and target features. It then learns model parameters by minimizing a loss function over a labeled dataset. Finally, it outputs prediction scores as well as embedding vectors. There has also been other works on temporal interaction networks and embeddings. Recently and simultaneously to our work, JODIE, a novel embedding technique to learn joint user and item embeddings from sequential data, was proposed [18]. While these authors apply their algorithm to the problem of detecting banned accounts from Wikipedia and Reddit, which is similar to the problem of fake account detection, their work is different from ours. For example, they apply two recurrent neural networks to learn joint user-item embeddings based on temporal interaction sequences, while we just want to learn the embedding for the source entities. In order to leverage the target entities' current state, we also use pre-existing embeddings or application specific feature sets which JODIE does not. A notable limitation of JODIE and other existing approaches is scalability. On Facebook, we have billions of accounts and trillions of interactions per day. Therefore, scalability and reasonable computational costs are of utmost importance.

The remainder of the paper is organized as follows. We begin in Section 2 with the problem formulation and description of the model architecture. In Section 3, we discuss a couple of integrity case studies and results. Finally, we conclude the paper with discussions in Section 4.

2 PROTECTING FACEBOOK SOCIAL MEDIA INTEGRITY

We first start by providing a mathematical formulation for solving various social media integrity problems encountered on Facebook's platform and subsequently present the TIES model in detail.

2.1 Integrity Problem Formulation

At Facebook, we are interested in verifying the integrity of various social media entities such as accounts, posts, Pages, Groups, etc. As mentioned earlier, in this work we exploit the interaction information between such entities to determine their integrity. We refer to an entity under inspection as source (denoted by u) while other interacted entities are referred as targets (denoted by v). Suppose at time t, the source entity u interacts with target entity v_t by taking an action[1] a_t such as receiving a friend request, sending an event invite, or liking a post. We might also have source and target specific features f_t at each timestamp (e.g. text or image related features or time gaps between consecutive actions). As a result, we will have a sequence of temporal interactions represented by $\mathcal{I} = \{(u, v_1, a_1, f_1), (u, v_2, a_2, f_2), \ldots, (u, v_T, a_T, f_T)\}$. Based on the interaction sequence $\mathcal{I}$, TIES determines the integrity of u, for instance, how likely is the entity to be a fake account or a hateful post. We do so by training a supervised model using a labeled training set $\{(\mathcal{I}_k, l_k)\}_{k=1}^{N}$, where l_k is the ground truth label for the sequence $\mathcal{I}_k$. Thus, in our framework solving social media integrity is formulated as a sequential or temporal learning problem.

2.2 TIES Model

At the core of the TIES model, there are two types of embeddings: 1) graph based and 2) temporal based. These embeddings are constantly trained to capture the ongoing behavior of the entities. We first make use of a large-scale Facebook graph to capture the static behavior of the entities. Subsequently, these graph-based embeddings are used to initialize the temporal model that captures the

[1] Actions could originate from either source or target.

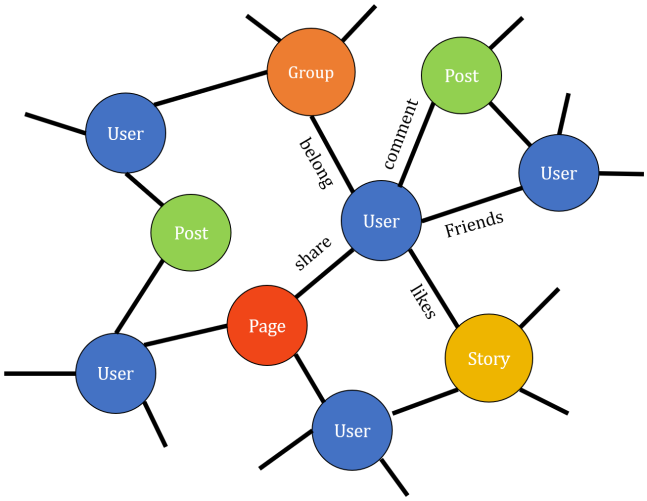


Figure 2: Facebook social media entity graph. To capture the static behavior of entities, Pytorch-BigGraph system is used to compute graph embeddings. These vectors are treated as pre-trained embeddings for the TIES model.

dynamic behavior of the entities. This is a distinguishing feature of the TIES that has not been explored before in prior integrity works. We now describe these parts in more detail.

2.2.1 Graph Based Embeddings. One of the novel components of TIES model is making use of a large scale graph structure formed by various social media entities beside the dynamic interaction information. In social networks, entities may be connected to each other through friend relationships or belong to the same groups. Such prior knowledge can capture the 'static' nature of various entities. It should be noted that even though these static relations (such as friend relationships, group membership, etc.) are not truly static, they vary at a much lower pace compared to other more 'dynamic' interactions (such as post reactions, commenting, etc.). Past studies mainly focus either on static or dynamic behavior but do not account for both in the model. Moreover, the scale of the graph structure considered in this work is much greater than in previous works and thus presents unique challenges.

Let $G = (V, R, E)$ denote a large-scale multi-relations graph formed by social media entities (see Figure 2). Here, V denotes a set of nodes (i.e., entities), R is a set of relations and E denotes a set of edges. Each edge $e = (s, r, d)$ consists of a source s, a relation r, and a destination d, where $s, d \in V$ and $r \in R$. In order to learn graph embeddings for entities in the graph G, we utilize the PyTorch-BigGraph (PBG) [19] distributed system due to its scalability to billions of nodes and trillions of edges. This is essential for solving our real-world needs.

Suppose $\theta_s, \theta_r, \theta_d$ are trainable parameter vectors (embeddings) associated with source, relation, and destination. PBG assigns a score function $f(\theta_s, \theta_r, \theta_d)$ to each triplet, where higher values are expected for $(s, r, d) \in E$ as opposed to $(s, r, d) \notin E$. PBG optimizes a margin-based ranking loss (shown below) for each edge e in the training data. A set of judiciously constructed negative edges e' are obtained by corrupting e with either a sampled source or destination node

$$\ell = \sum_{e \in G} \sum_{e' \in S'_e} \max\{f(e) - f(e') + \lambda, 0\}$$

Here λ is the margin hyperparameter and $S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d') | d' \in V\}$. Finally, entity embeddings and relation parameters are learned by performing mini-batch stochastic gradient descent. More details about these embeddings can be found in [19]. The PBG-trained embeddings on a large-scale graph with billions of nodes and trillions of edges are fed as pre-trained source and target embeddings to the temporal model, to which we now turn.

2.2.2 Temporal Based Embeddings. Temporal based embeddings are designed to capture the information encoded in the sequence of interactions $\mathcal{I}$ as discussed in Section 2.1. Consider all the interactions of a source entity u in a given window of time period. Suppose at t time, source entity u interacted with target entity v_t by performing an action a_t. This whole interaction at time t is encoded into a single feature vector as follows (see Figure 3):

(1) **Action Features**: Action a, for instance commenting or liking, is represented by a fixed size vector that will be learned during the training process and initialized as random (also referred as trainable embedding). Depending upon the task, the same type of action can learn different embeddings and can have multiple meanings based on the context. As mentioned earlier, one can also encode the direction information in action by splitting it into two directional events as $a-$send or $a-$receive.
(2) **Source Entity Features**: Source entity u is represented by a pre-trained embedding. More specifically, we utilize graph-based embeddings obtained from the PBG system as described in Section 2.2.1. One can further finetune the pre-trained embeddings in the TIES model, but this is only possible if the number of unique source entities is not greater than a few million (due to computational cost).
(3) **Target Entity Features**: Similar to the source entity, the target entity v is also represented by a pre-trained embedding (if available) or by a trainable embedding (if problem dimension allows i.e., limited to few millions). Multiple types of pre-trained target embeddings can be utilized in the same TIES model.
(4) **Miscellaneous features**: We can also encode useful time-related information such as rate of interaction via $\Delta_t = t_{i+1} - t_i$ (may need to normalize the range appropriately). Rate of interaction is an important signal for detecting abusiveness. Other features like text or images can also be plugged into TIES in similar manner.

All these features are packaged into a single feature vector by performing an aggregation operation. We obtain a single embedding capturing the full interaction information in

$$\mathbf{x}_t = \mathbf{e}(u) \odot \mathbf{e}(v_t) \odot \cdots \odot \Delta_t,$$

where $\odot$ is a aggregation operator, $\mathbf{e}(\cdot)$ represents the entity embedding and $\mathbf{x}_t$ is the resultant embedding obatined at time t. In our case, we simply choose concatenation as the aggregation operation. Next, we pass the sequence of these temporal embeddings

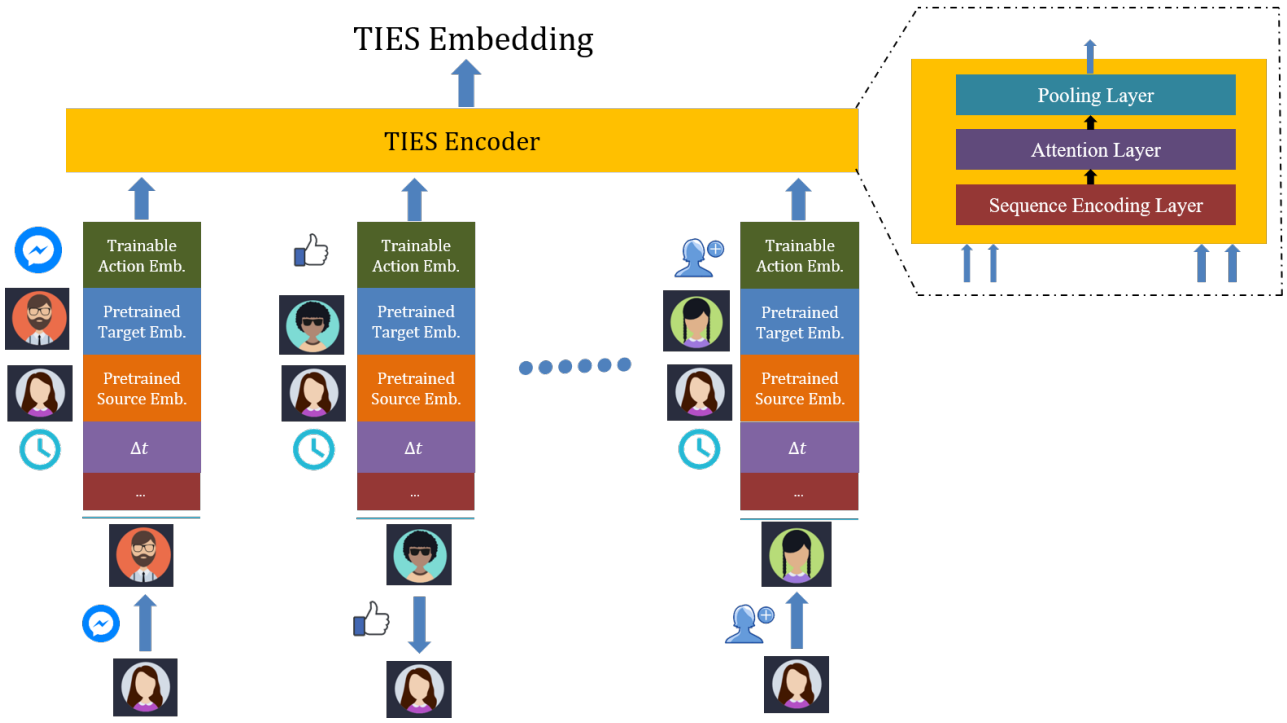


Figure 3: TIES Model Architecture: At each time step, the (source, target, action) triplet is converted into a feature vector that consists of trainable action embeddings, pre-trained source and target embeddings, and other miscellaneous features. The feature vectors are then fed into a deep sequential learning model to capture the dynamic behavior of entities.

i.e., $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T] \in \mathbb{R}^{T \times d}$ where T is sequence length and d is the input dimension, into a sequence encoder to yield TIES embedding $\mathbf{z} \in \mathbb{R}^h$ as follows

$$\mathbf{z} = TIESEncoder(\mathbf{X}).$$

TIES Encoder: It yields our final TIES embedding by capturing the temporal aspect present in the sequence of interaction embeddings. Our general purpose sequence encoder has the following components:

(1) **Sequence Encoding Layer**: This encoding layer transforms the input into a hidden state that is now aware of the interaction context in the sequence

$$\mathbf{H} = SeqEnocder(\mathbf{X}).$$

Here, $\mathbf{H} \in \mathbb{R}^{T \times h}$ is the hidden-state matrix with h as the dimension. We consider three types of sequence encoding layer in our TIES framework with varying training and inference costs and benefits:

(a) **Recurrent neural networks**: RNNs such as long short term memory networks (LSTM) are quite capable of capturing dependencies in a sequence [35]. But they are inherently slow to train.

(b) **Convolutional neural networks**: 1D sequence CNNs can also capture sequential information but are limited to local context and need to have higher depth for capturing global context, depending on the task in hand [7].

(c) **DeepSet**: When inputs are treated as sets and their order does not matter, we can use Deepsets as sequence encoders [21]. Here, we first pass each input in a sequence through an MLP (small neural network) and then perform a sum operation followed by another MLP layer, yielding a single embedding layer.

Besides the application in-hand and deployment challenges, the choice among RNN, CNN and DeepSet depends on the tradeoff between performance and inference time. Due to its recurrent nature, RNN is expensive while DeepSet has the lowest inference time in production.

(2) **Attention Layer**: Attention Layer [30] can be used to weigh the embeddings differently according to their contribution towards specific tasks. Attention values can also be used to visualize which part of the interaction sequence is being focused more than the others and that can provide more interpretable outcome. The output is given by

$$\mathbf{Z} = Attention(\mathbf{H}),$$

where $\mathbf{Z} \in \mathbb{R}^{T \times h}$ is the attention layer output.

(3) **Pooling Layer**: A final pooling layer such as mean, max, or sum operation is used to yield a single embedding for the whole interaction sequence. Here, we have

$$\mathbf{z} = Pooling(\mathbf{Z})$$

where $\mathbf{z} \in \mathbb{R}^h$ is the output of the pooling layer and serves as the final TIES embeddings.

2.2.3 Loss Function. Parameters of TIES are learned based on the task labels associated with each training sequence $\mathcal{I}_k$, $k = 1, 2, \ldots, N$. For instance, in case of abusive account detection we have a training set of sequences labeled as either abusive or benign and binary cross-entropy can be considered as the loss function. In general, TIES embedding $\mathbf{z}$ is fed to the feed-forward neural network for learning the parameters in end-to-end fashion as follows,

$$\ell = \sum_{i=1}^{N} \mathcal{L}(\mathbf{X}_i, f(\mathbf{z}_i))$$

where ℓ is the overall task loss, $\mathcal{L}$ is the loss function, $f(\cdot)$ is the feed-forward neural network, $\mathbf{X}_i \in \mathbb{R}^{T \times d}$ is the i^{th} input training sequence, $\mathbf{z}_i \in \mathbb{R}^h$ is the corresponding learned TIES embedding and N is the total number of training samples. Depending upon the task, different metrics are taken into consideration and class-imbalance issue is handled by weighting the classes properly. This completes the overall description of the TIES model architecture.

3 FACEBOOK INTEGRITY APPLICATIONS AND RESULTS

In this section, we briefly describe our implementation of TIES, introduce some of its integrity applications, and use cases. These applications cover a wide range of issues from content-based to account-based.

3.1 Production Implementation

Our framework is built on PyTorch, more specifically TorchScript to streamline productionization [4]. Training is generally performed on GPUs and when needed we use DistributedDataParallel provided by PyTorch for multi-GPU training. Inference on the other hand is performed in parallel on up to 200 machines (CPUs suffice during inference). To maintain relative consistency in embedding vectors overtime, we use warm-start—that is, initializing the model with a previously trained one. For our experiments, we use Adam optimizer with learning rate 0.0005 and clip gradients at 1. To mitigate overfitting we use dropout with probability 0.1. Finally, we weight positive samples such that the datasets are balanced.

3.2 Preventing Spread of Misinformation

False news can spread rapidly, either intentionally or unintentionally, by various actors. Therefore, proactive identification of misinformation posts is a particularly important issue. It is also very challenging as some of such posts contain material that are not demonstrably false but rather are designed to be misleading and/or reflecting only one side of a story.

Some of the existing techniques train classifiers on carefully crafted features [28]. There are also approaches that detects "rumors" based on users' reactions to microblogs overtime [12, 16]. To the best of our knowledge, users' histories (captured via graph-embeddings described in 2.2) and their interactions with posts have not been used to detect misinformation systematically. More recently, researchers at Facebook have devised a multimodal technique for identifying misinformation, inspired by the work of Kiela et al. [11]. In the multimodal model, separate encoders are learned

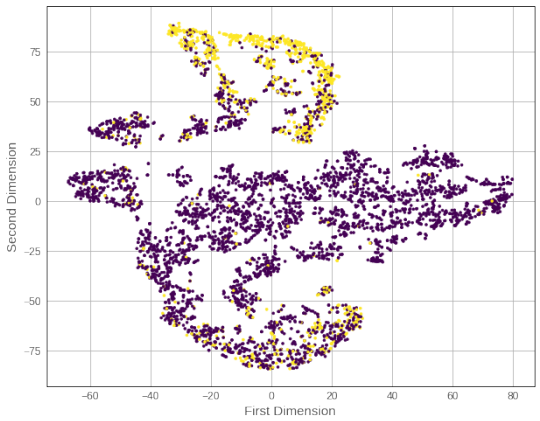


Figure 4: 2-dimensional TSNE projection of the TIES embeddings for misinformation. Clearly, the misinformation posts (yellow) have a different distribution than the regular posts (purple).

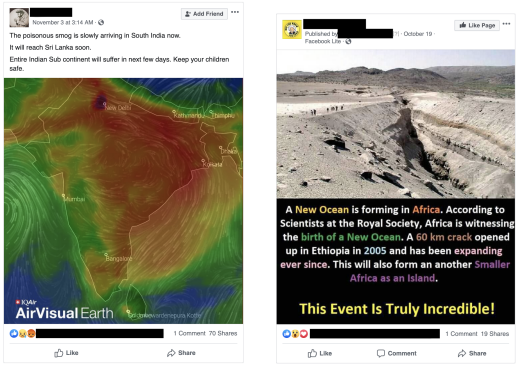
for various content modalities such as image, post-text, image-embedded-text, comments, etc. The encoded features are then collected into a set and pooled in an order-invariant way. Finally, the pooled vector is passed through a fully connected layer to generate predictions.

Model	PR-AUC Median Gap ± MAD
TIES-CNN	-0.1130 ± 0.0110
TIES-RNN	-0.0987 ± 0.0143
TIES-Deepset	-0.1152 ± 0.0124
Content+TIES-CNN	0.0386 ± 0.0087
Content+TIES-RNN	0.0474 ± 0.0102
Content+TIES-Deepset	0.0436 ± 0.0087

Table 1: Median Precision-Recall area under the curve difference with respect to the content-only model and median absolute deviation on the test dataset for misinformation detection. Combining TIES with content improves the performance significantly.

In the context of TIES, we have posts (sources) interacting with users (targets). Here, we consider a small set of interactions: like, love, sad, wow, anger, haha, comment, and share. Moreover, we use embeddings described in Section 2.2 for source and target entities. For this experiment, we split our training dataset, consisting of 130K posts (roughly 10% of which are labeled as misinformation), into *train-1*, *train-2*, and *test* sets[2]. It should be noted that this dataset is sampled differently for positive and negative cases and does not reflect the accurate distribution of the posts on our platform. We then use the set *train-1* to train a few TIES models (CNN, RNN, and Deepset). For all models, we set the interaction embeddings as well as hidden dimensions to 64. We consider sequences of length 512, where longer sequences are cropped from the beginning and shorter

[2]We use two disjoint training sets to prevent overfitting.



(a) **(b)**

Figure 5: Sample misinformation posts, as identified by independent human reviewers, with low content-only but high hybrid score.

sequences are padded accordingly. The CNN model consists of 2 convolution layers of width 5 and stride 1. The RNN model consists of a 1-layer bidirectional LSTM. And finally, the Deepset model consists of pre and post aggregation MLPs with one hidden layer of size 64. In addition to TIES, we train a multimodal model using post images and texts. Finally, we use the *train-2* dataset to train a hybrid model, a simple logistic-regression with two features, TIES-score and multimodal-score. In order to specify confidence intervals, we run the aforementioned experiment on several train/test data split. Table 1 illustrates the difference/delta performance for various models with respect to the content-only model on the *test* dataset.

At this point a few observations are worth highlighting: First, TIES-RNN seems to be the best performing TIES model. Second, Deepset appears to be the least performing TIES model, perhaps highlighting the importance of the ordered sequences (as opposed to sets) in identifying questionable posts. Third, the content-only model seems to outperform interaction-only models. Finally, combining the interactions signal with content (hybrid models) improves the performance significantly.

Figure 4 illustrates the TSNE projection of the post embeddings. It should be noted that the final hidden state of the TIES model is considered the source-entity and in this case post embeddings. Interestingly, the distribution of the misinformation posts (yellow dots), in the latent space, is clearly different from the regular posts (purple dots). Figure 5 illustrates a couple of positive posts that were not identified by the content-only model but registered high scores in the hybrid (content+TIES) model.

3.3 Detecting Fake Accounts and Engagements

It is understood that fake-accounts are a potential security and integrity risk to the online community. Therefore, identifying and removing them proactively is of utmost importance. Most existing approaches in detecting fake accounts revolve around graph propagation [24, 32] or clustering of malicious actors and/or activities [20, 29, 31, 33]. Fakebuster [9] trained a model on activity related features. To the best of our knowledge, sequence of accounts and associated activities have not been used in fake account detection.

One of the fake engagement baseline models at Facebook is a multilayer classifier trained on over one thousand carefully engineered features. These features cover a range of signals from metadata, activity statistics, among others. In this experiment, we train a few TIES models and combine them with the output of the baseline classifier in order to evaluate the effectiveness of the sequence data. Here, sources are accounts engaging with various targets. Targets on the other hand could be other accounts, posts, or pages. As engagements, we consider a set of 44 sentry-level actions that includes liking a post, following a page, messaging a user, friending, etc. As source and target features, we use graph-based embeddings described in Section 2.2 for accounts, as well as posts and pages creators, where applicable.

Our dataset consists of 2.5M accounts with 80/20 good/bad (fake) split. It should be emphasized that this dataset is sampled differently for positive and negative cases and does not reflect the accurate distribution of fake accounts on our platform. We randomly divide this dataset into *train-1*, *train-2*, and *test* sets consisting of 2M, 250K, and 250K accounts, respectively. We then use the set *train-1* to train a few TIES models similar to the previous experiment described in Section 3.2. The main difference is that here the CNN model has 3 convolution layers of width 5 and stride 1, and the RNN model consists of a 2-layer bidirectional LSTM. Subsequently, we use the *train-2* dataset to train a logistic-regression with two features, TIES-score and baseline-score. This experiment is repeated on several train/test splits in order to calculate confidence intervals. The performance gap between various models and the baseline on the *test* dataset are illustrated in Table 2.

Model	PR-AUC Median Gap ± MAD
TIES-CNN	-0.0566 ± 0.0022
TIES-RNN	-0.0521 ± 0.0011
TIES-Deepset	-0.0857 ± 0.0017
Baseline+TIES-CNN	0.0090 ± 0.0014
Baseline+TIES-RNN	0.0110 ± 0.0014
Baseline+TIES-Deepset	0.0055 ± 0.0012

Table 2: Median Precision-Recall area under the curve difference with respect to the baseline model and median absolute deviation on the test dataset for fake engagement detection. Combining TIES with the baseline improves the performance.

Our observations are largely consistent with the ones made in Section 3.2. Namely, TIES-RNN appears to be the best performing TIES model, baseline outperforms TIES, and as an additional feature, TIES can provide a boost to the baseline model. The fact that baseline outperforms TIES is not surprising, as it includes a lot more information through over 1000 carefully engineered features. Moreover, gains from TIES appear to be small but they are statistically significant and outside the confidence interval. *It should also be noted that, at the scale of Facebook, even a couple of percentage points improvement in recall for the same precision translates into significant*

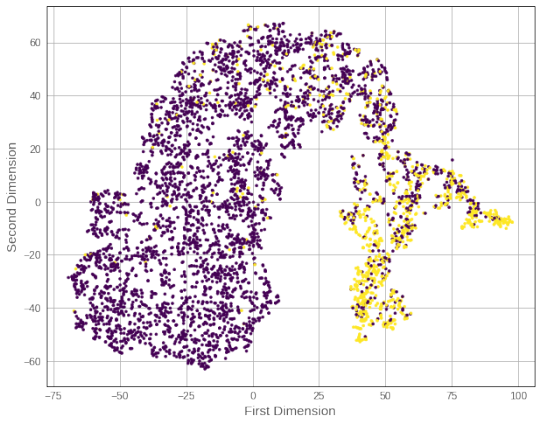


Figure 6: 2-dimensional TSNE projection of the TIES embeddings for fake engagements. Clearly, the fake accounts (yellow) have a different distribution than the regular accounts (purple).

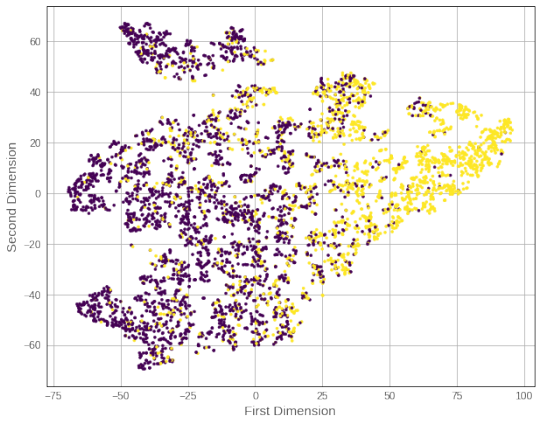


Figure 7: 2-dimensional TSNE projection of the TIES embeddings for ads payment risk. Clearly, accounts with unpaid fees (yellow) have a different distribution than the regular accounts (purple).

number of additional fake accounts being caught. Figure 6 illustrates the 2-d projection of the TIES embeddings and as expected the distribution of the fake accounts (yellow) is quite different from nor mal accounts (purple).

3.4 Reducing Ads Payment Risks

With more than two billion monthly active users, Facebook enables small- and medium-sized businesses to connect with a large audience. It allows them to reach out to their target audiences in an efficient way and grow their businesses as a result [1].

Some of the integrity issues facing this advertising platform include fraudulent requests and unpaid service fees or substantial reversals. Researchers at Facebook have devised various models to prevent such abuses. These models generally include thousands of carefully crafted features that cover a wide range of information sources, such as user metadata, activity history, etc. In order to test TIES' viability for identifying bad accounts that have failed to pay fees, we train a few models using interaction signals that generally include details about the account and associated payment trends. Here, sources are ads accounts, source features are graph-based embeddings, and targets are generally nulls. We follow the settings in the previous two experiments: we split our dataset into *train-1*, *train-2*, and *test* sets consisting of roughly 200K, 10K, and 10K accounts, respectively. The datasets are sampled such that we have roughly the same number of bad and good accounts. We then train TIES-CNN (2 layers of width 5), TIES-RNN (1 layer biLSTM), and TIES-Deepset (pre and post aggregation MLPs with 1 hidden-layer of size 64) as well as the baseline model on the set *train-1*. We then use *train-2* dataset to combine baseline and TIES scores via a simple logistic regression and finally test the outcome on the *test* dataset. In order to calculate the confidence intervals, we repeat this experiment on several random train/test splits. Precision-recall area under the curve gaps with respect to the baseline model are illustrated in Table 3. Figure 7, on the other hand, demonstrates the 2-d projection of embedding vectors and as expected the distribution of bad accounts is quite different from normal accounts.

Model	PR-AUC Median Gap ± MAD
TIES-CNN	-0.0705 ± 0.0051
TIES-RNN	-0.0672 ± 0.0045
TIES-Deepset	-0.1135 ± 0.0062
Baseline+TIES-CNN	0.0088 ± 0.0034
Baseline+TIES-RNN	0.0071 ± 0.0043
Baseline+TIES-Deepset	0.0060 ± 0.0043

Table 3: Median Precision-Recall area under the curve gap with respect to the baseline model as well as the median absolute deviation on the test dataset for ads payment risks. Combining TIES with the baseline improves the performance.

TIES models do a decent job in identifying bad accounts although they are not as predictive as the baseline model. Moreover, similar to the previous two experiments, RNN and CNN perform roughly the same and both outperform Deepset by a wide margin. Finally, combining TIES with the baseline provides about 1% gain in precision-recall area under the curve which is statistically significant. It is worth noting that even a small improvement in recall for the same precision would translate to a large monetary value in savings.

4 CONCLUSION

In this paper, we provided an overview of the temporal interaction embeddings (TIES) framework and demonstrated its effectiveness in fighting abuse at Facebook. Social media entities such as accounts, posts, Pages, and Groups interact with each other overtime. The type of interactions generated by bad entities such as fake accounts and hateful posts are different from normal entities.

TIES, a supervised deeplearning framework, embeds these interactions. The embedding vectors in turn can be used to identify bad entities (or improve existing classifiers). The TIES framework is quite general and can be used by various forms of account, post, Page, or Group integrity applications. Moving forward, we plan to continue exploring other applications of this methodology within Facebook. Moreover, we can add additional features such as model interpretability, hyperparameter optimization, unsupervised learning, etc. to the framework in order to create a more complete tool.

5 ACKNOWLEDGEMENTS

Authors would like to thank He Li, Bolun Wang, Yingyezhe Jin, Despoina Magka, Austin Reiter, Hamed Firooz, Shaili Jain for numerous helpful conversations, data preparation, and experiment setup.

REFERENCES

[1] [n. d.]. Facebook for Small Businesses. https://www.facebook.com/business/success/categories/small-business. Accessed: 2020-01-02.

[2] [n. d.]. Facebook Transparency Report - Bullying and Harassment. https://transparency.facebook.com/community-standards-enforcement#bullying-and-harassment. Accessed: 2019-10-28.

[3] [n. d.]. Facebook Transparency Report - Hate Speech. https://transparency.facebook.com/community-standards-enforcement#hate-speech. Accessed: 2019-10-28.

[4] [n. d.]. TorchScript documentation. https://pytorch.org/docs/stable/jit.html. Accessed: 2020-01-03.

[5] Michael Wiegand Anna Schmidt. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. 1–10.

[6] Kalina Bontcheva Maria Liakata Rob Procter Arkaitz Zubiaga, Ahmet Aker. 2018. Detection and Resolution of Rumours in Social Media: A Survey. *ACM Computing Surveys (CSUR)* 51, 32 (2018).

[7] Umme Zahoora Aqsa Saeed Qureshi Asifullah Khan, Anabia Sohail. 2019. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. *ArXiv* abs/1901.06032 (2019).

[8] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, and Konstantin Beznosov. 2015. Integro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs.. In *NDSS*, Vol. 15. 8–11.

[9] Yeh-Cheng Chen and Shyhtsun Felix Wu. 2018. FakeBuster: A Robust Fake Account Detection by Activity Analysis. In *Proceedings of 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. IEEE, 108–110.

[10] Achint Thomas Yashar Mehdad Yashar Mehdad Chikashi Nobata, Joel Tetreault. 2016. Abusive Language Detection in Online User Content. In *Proceedings of 25th International World Wide Web Conference*.

[11] Armand Joulin Tomas Mikolov Douwe Kiela, Edouard Grave. 2018. Efficient Large-Scale Multi-Modal Classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 5198–5204.

[12] Shu Wu Liang Wang Tieniu Tan Feng Yu, Qiang Liu. 2017. A Convolutional Approach for Misinformation Identification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 3901–3907.

[13] Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language Online*. 85–90.

[14] Anna Squicciarini Sarah Rajtmajer Christopher Griffin David Miller Cornelia Caragea Haoti Zhong, Hao Li. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. 3952–3958.

[15] Rahat Ibn Rafiq Richard Han Qin Lv Shivakant Mishra Homa Hosseinmardi, Sabrina Arredondo Mattson. 2015. *Detection of Cyberbullying Incidents on the Instagram Social Network*. Technical Report. University of Colorado at Boulder. https://arxiv.org/abs/1503.03909

[16] Prasenjit Mitra Hamad Bin Khalifa Sejeong Kwon Bernard J. Jansen Kam-Fai Wong Meeyoung Cha Jing Ma, Wei Gao. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 3818–3824.

[17] Catherine Havasi Henry Lieberman Karthik Dinakar, Birago Jones and Rosalind Picard. 2012. Common Sense Reasoning for Detection, Prevention, and Mitigation of Cyberbullying. *ACM Transactions on Interactive Intelligent Systems* 2, 3 (2012).

[18] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1269–1278.

[19] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. *arXiv preprint arXiv:1903.12287* (2019).

[20] Yixuan Li, Oscar Martinez, Xing Chen, Yi Li, and John E Hopcroft. 2016. In a world that counts: Clustering and detecting fake social engagement at scale. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 111–120.

[21] Siamak Ravanbakhsh Barnabás Póczos Ruslan Salakhutdinov Alexander J. Smola Manzil Zaheer, Satwik Kottur. 2017. Deep Sets. In *Advances in Neural Information Processing (NIPS)*.

[22] Roeland Ordelman Maral Dadvar, Dolf Trieschnigg and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Proceedings of European Conference on Information Retrieval*. 693–696.

[23] Robin Morris Mihajlo Grbovic Vladan Radosavljevic Narayan Bhamidipati Nemanja Djuric, Jing Zhou. 2015. Hate Speech Detection with Comment Embeddings. In *Proceedings of 24th International World Wide Web Conference*. 29–30.

[24] Shirin Nilizadeh, François Labrèche, Alireza Sedighian, Ali Zand, José Fernandez, Christopher Kruegel, Gianluca Stringhini, and Giovanni Vigna. 2017. Poised: Spotting twitter spam off the beaten paths. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1159–1174.

[25] Mauricio Perez Anderson Rocha Paulo Vitorino, Sandra Avila. 2018. Leveraging deep neural networks to fight child pornography in the age of social media. *Journal of Visual Communication and Image Representation* 50 (2018), 303–313.

[26] Matthew L Williams Pete Burnap. 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science* 5, 11 (2016).

[27] Manish Gupta Vasudeva Varma Pinkesh Badjatiya, Shashank Gupta. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 759–760.

[28] Mona T Diab Sardar Hamidian. 2016. Rumor Identification and Belief Investigation on Twitter. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 3–8.

[29] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. {EVILCOHORT}: Detecting Communities of Malicious Accounts on Online Services. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 563–578.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[31] Cao Xiao, David Mandell Freeman, and Theodore Hwa. 2015. Detecting clusters of fake accounts in online social networks. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. ACM, 91–101.

[32] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. 2012. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 71–80.

[33] Fang Yu Qifa Ke Yuan Yu Yan Chen Yao Zhao, Yinglian Xie and Eliot Gillum. 2009. BotGraph: Large Scale Spamming Botnet Detection. In *6th USENIX Symposium on Networked Systems Design and Implementation*.

[34] Sencun Zhu Heng Xu Ying Chen, Yilu Zhou. 2012. Detecting Offensive Language in Social Media to Protect Adolescent Online Safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. IEEE.

[35] Charles Elkan Zachary Lipton, John Berkowitz. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *ArXiv* abs/1506.00019 (2015).

[36] Dirk Hovy Zeerak Waseem. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. 88–93.