Counterfactual Neural Temporal Point Process for Misinformation Impact Estimation on Social Media

Anonymous Author(s)
Affiliation
Address
email

A Appendix

A.1 Data and Code

Together with this appendix, we also submit the synthetic data and necessary code. However, the real-world data (from Twitter) is too huge for OpenReview platform. Thus, we can not submit it right now. We will opensource a version of the data with anonymous processing to protect privacies after this paper is accepted.

A.2 Proof of the theorems

Theorem 1. *For a user u, if the intensity function $\lambda(\boldsymbol{f}, t|Tr \cup X)$ is known, then we have:*

$$\mu(t, t+T, \lambda_1, Tr \cup X) = \int_{sup(\boldsymbol{f})} d\boldsymbol{f} \int_t^{t+T} \lambda(\boldsymbol{f}, t|Tr \cup X)dt \tag{1}$$

$$\phi(t, t+T, \lambda_1, Tr \cup X) = \int_{sup(\boldsymbol{f})} \boldsymbol{f} d\boldsymbol{f} \int_t^{t+T} \lambda(\boldsymbol{f}, t|Tr \cup X)dt \tag{2}$$

Proof. The first equation can be trivially proved by replacing the $\boldsymbol{F}$ in the definition of λ with the support set. The second one can be proved with the Campbell's Theorem [1]:

Lemma 1. *Campbell's Theorem (Campbell, 1909): For a point process S, and a measurable function $f : \mathbb{R}^d \to \mathbb{R}^{d'}$:*

$$\mathbb{E}[\sum_{x \in S} f(x)] = \int_{\mathbb{R}^d} f(x)\mathbb{E}(N(x, x+dx)) \tag{3}$$

Note that the above theorem is for all point process. For the specific scenario of temporal point process, we also need to include the time range into consideration:

$$\mathbb{E}[\sum_{x \in S[t_1:t_2]} f(x)] = \int_{\mathbb{R}^d} f(x)\mathbb{E}(N(x, x+dx, t_1, t_2)) \tag{4}$$

From the definition of ϕ, we know that the $f(x) = x$. Recall the definition of intensity function (Equation. 1 in the main content):

$$\mathbb{E}(N(\boldsymbol{F}, T_1, T_2)|S_h) = \int_{\boldsymbol{F}} d\boldsymbol{f} \int_{T_1}^{T_2} \lambda(\boldsymbol{f}, t|S_h)dt \tag{5}$$

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

We can convert this integral to a derivative formula:

$$\mathbb{E}(N(\boldsymbol{f}, \boldsymbol{f}+d\boldsymbol{f}, T_1, T_2)|S_h) = d\boldsymbol{f}\int_{T_1}^{T_2} \lambda(\boldsymbol{f}, t|S_h)dt \tag{6}$$

With the above formula and Campbell's theorem, we have:

$$\phi(t, t+T, \lambda_1, Tr \cup X) = \int_{\sup(\boldsymbol{f})} \boldsymbol{f}d\boldsymbol{f}\int_{t}^{t+T} \lambda(\boldsymbol{f}, t|Tr \cup X)dt \tag{7}$$

□

Theorem 2. *Given the following min-max game:*

$$min_H max_{\hat{p}} \mathbb{E}_{Tr,X\sim p(Tr,X)} \log \hat{p}(Tr|H(X)) \tag{8}$$

the min gamer's Nash balanced solution H^*, ensures for any X_1, X_2, the following equation holds:

$$p(Tr|H^*(X_1)) = p(Tr|H^*(X_2)) \tag{9}$$

where p denote the ground-truth conditional distribution of treatment given encoding.

Proof. For the objective of the max gamer, we have:

$$\begin{aligned}\mathbb{E}_{Tr,X\sim p(Tr,X)} \log \hat{p}(Tr|H(X)) =& \mathbb{E}_{Tr,H(X)\sim p(Tr,H(X))} \log \hat{p}(Tr|H(X)) \\ =& \mathbb{E}_{H(X)\sim p(H(X))}\mathbb{E}_{Tr\sim p(Tr|H(X))} \log \hat{p}(Tr|H(X)) \\ =& \mathbb{E}_{H(X)\sim p(H(X))} - KL(p(Tr|H(X)), \hat{p}(Tr|H(X)) \\ & - Q(p(Tr|H(X)))\end{aligned} \tag{10}$$

where $Q(p(Tr|H(X)))$ is the entropy of $p(Tr|H(X))$. Note that for a fixed H, $Q(p(Tr|H(X)))$ is a constant. Thus, the max objective is the same as minimizing the KL-divergence between $p(Tr|H(X))$ and $\hat{p}(Tr|H(X)$ for every X:

$$\max_{\hat{p}} \mathbb{E}_{Tr,X\sim p(Tr,X)} \log \hat{p}(Tr|H(X)) \iff \min_{\hat{p}} \mathbb{E}_{H(X)\sim p(H(X))} KL(p(Tr|H(X)), \hat{p}(Tr|H(X)) \tag{11}$$

It is known that the KL-divergence is minimized if and only if $\hat{p}(Tr|H(X)$ perfectly fit $p(Tr|H(X))$ at each point. Thus, the balance solution of the max gamer is to learn $\hat{p}(Tr|H(X)) = p(Tr|H(X))$. Meanwhile, if there are two X_1 and X_2 such that $p(Tr|H(X_1)) \neq p(Tr|H(X_2))$, then the min gamer can always update the H by swapping $H(X_1)$ and $H(X_2)$ while maintaining other points the same. This is doable when H is a universal function approximator. As a result, the KL-divergence will increase, which violates the definition of Nash balance. Therefore, at the balance point for any X_1 and X_2, $p(Tr|H(X_1)) = p(Tr|H(X_2))$ □

A.3 Synthetic Dataset

In this section, we introduce the how to generate the synthetic data in details. We present the basic ideas in Figure 1. To generate the data, we design a system to simulate the social media activities and events in real world. The basic elements in this system are **users** and **news**.

- **User**: Each user i is represented with a **hidden** vector u_i, which represent the hidden status of a social media user, such as interests, ideas and political trends in reality. If a user engage with a piece of news, then this user's hidden vector will change accordingly. We randomly assign each user with an randomly initialized vector to simulate the fact that users may hold different ideas and interests before starting to use social media.
- **News**: Each piece of news n is characterized by two randomly generated **hidden** feature vectors: a topic vector $v_{topic}(n)$ and an inherent influence vector $v_{in}(n)$. These two vectors decide which users will be attracted by the news and how they will be influenced by the news.

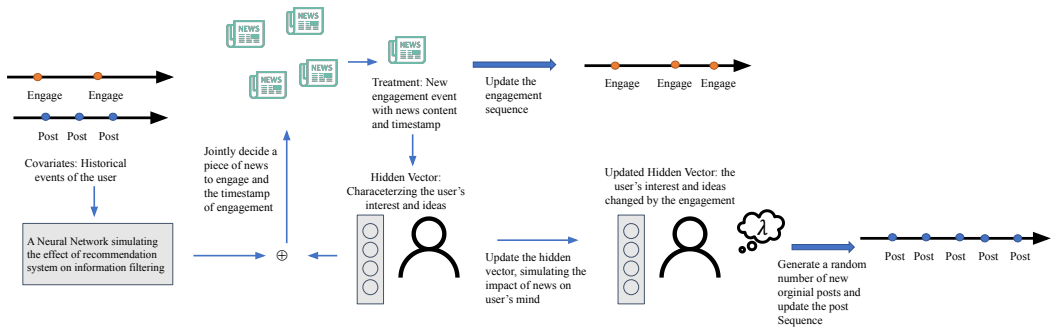


Figure 1: The general idea of generating the synthetic dataset

In this system, a user has two kinds of activities: (1) engaging with a piece of news and (2) posting a post with original contents. These activities are characterized by two temporal point process respectively. For the process of engaging with a piece of news, the intensity function of a user's (i) engagement with a specific piece of news n is defined as:

$$\lambda_e(n, t|u_i, S_h^{(e)}, S_h^{(g)}) = v_{topic}(n) \cdot u_i + \sum_{(\boldsymbol{f}_j, t_j) \in S_h^{(e)} \cup S_h^{(g)}} \exp(t_j - t) NN_{rec}(\boldsymbol{f}_j, v_{topic}(n)) \quad (12)$$

where $S_h^{(e)}$ and $S_h^{(g)}$ denotes the historical sequences of a user's engagement and posting activities respectively. In this equation, the first term models the consistency of news topic and user interest and the $NN_{rec}(\cdot, \cdot)$ in the second term is a random parameterized neural network simulating the recommendation system that recommend contents based on past history. $\exp(t_j - t)$ make sure that the influence of a history event decrease as time goes by, simulating the property that recommendation system usually give higher attention to recent events.

Similarly, the intensity function of a user's (i) process generating original posts is defined as:

$$\lambda_g(t|u_i, S_h^{(e)}, S_h^{(g)}) = u_i \cdot u_i + \sum_{(\boldsymbol{f}_j, t_j) \in S_h^{(e)} \cup S_h^{(g)}} \exp(t_j - t) NN_{post}(\boldsymbol{f}_j, u_i) \quad (13)$$

where $u_i \cdot u_i$ models the user's reliance to social media and the second term reflect the influence of history events to future event.

In real world, the news topic, influence on user mind and user's mind status are not observable. Instead we can only observe the tweet content text features, such as the representation extracted with a pre-trained language model. To simulate this fact, we use two neural networks NN_{news} and NN_{user} with random parameters to project the vectors to observable features:

$$\boldsymbol{f}_{news}(n) = NN_{news}(v_{topic}(n), v_{in}(n)), \quad \boldsymbol{f}_{user}(i, t) = NN_{user}(u_i, t) \quad (14)$$

where $\boldsymbol{f}_{news}(n)$ is the observable content feature of news n and $\boldsymbol{f}_{user}(i, t)$ is the observable content feature of the post generated by user i at time t. $\boldsymbol{f}_{news}(n)$ does not change over time because once a piece of news is reported, its content does not change significantly. Note that a normal user's activity is usually uncertain. To simulate such uncertainty, in each hidden layer of NN_{user}, we add a random noise. Since the two networks are with random parameters, the projection will be highly non-linear and thus brings enough complexity on the learning of causal impact.

The overall algorithm is shown in Algorithm 1:

A.4 Synthetic Data Experiments

We remove the users with only one event and then acquire around 13000 accounts. Then we split the training set, validation set and test set according to 8000/2000/3000 to train the proposed model. Specifically, we adopt the Adam optimizer with learning rate $1e-3$ and the max epoch is set to be

Algorithm 1 Synthetic Data Generation

Set the ending time T. {The simulation will end at time T}
Randomly generated user hidden representations and news features and a posting time of each news.{Each piece of news will only be engaged with after it is posted.}
for news n **do**
  $\boldsymbol{f}_{news}(n) = NN_{news}(v_{topic}(n), v_{in}(n))$
end for
for user i **do**
  $t_{end} = 0$
  $S_h^{(e)}, S_h^{(g)} = \emptyset, \emptyset$
  repeat
    Draw a posting event timestamp t_g based on intensity function $\lambda_g(t|u_i, S_h^{(e)}, S_h^{(g)})$
    $t_{cur} = t_g$
    $\boldsymbol{f}_{cur} = \boldsymbol{f}_{user}(i, t_g)$
    $flag =' generating'$
    for news n **do**
      if $t_p(n) \geq t_{cur}$ **then**
        Continue.
      end if
      Draw an event timestamp t_e based on intensity function $\lambda_e(n, t|u_i, S_h^{(e)}, S_h^{(g)})$
      if $t_e < t_{cur}$ **then**
        $t_{cur} = t_e$
        $\boldsymbol{f}_{cur} = \boldsymbol{f}_{news}(n)$
        $flag =' engagement'$
        $n_{cur} = n$
      end if
    end for
    $t_{end} = t_{cur}$
    if $flag$ is $'engagement'$ **then**
      $S_h^{(e)} = S_h^{(e)} \cup (\boldsymbol{f}_{cur}, t_{cur})$
      $u_i = u_i + NN_{scale}(v_{topic}(n_{cur}), u_i) v_{in}(n_{cur})$ {Update the user status. NN_{scale} is a neural network with random parameters to adjust news impact scale.}
    else
      $S_h^{(g)} = S_h^{(g)} \cup (\boldsymbol{f}_{cur}, t_{cur})$
    end if
  until $t_{end} \geq T$
end for

500 with batch size 128. In addition, we add the dropout with 0.1 rate. We train the models and baselines and evaluate their inference time on 1 NVIDIA GeForce RTX 2080 TI GPUs for the fair competition. The inference time with an error bar is presented in Table 1

Method	Inference Time
FullyNN (CNTPP w/o Adversarial Balance)	7.23±0.2 ms
CRN-VAE (Approximation)	**4.05±0.1ms**
CRN-VAE (Sampling)	29.34± 2.3ms
CNTPP(Ours)	7.12± 0.2ms

Table 1: Inference Time with Error Bar

A.5 Real World Dataset

To collect the Twitter datset related to COVID-19 vaccines, we use the tracked keywords including vaccine, Pfizer, BioNTech, Moderna, Janssen, AstraZeneca, Sinopharm to filtering all tweets via the streaming Twitter API which returns a 1% sample of all tweets. We collect the tweets just before Pfizer-BioNTech and Moderna were approved by the FDA for Emergency Use Authorization (EUA)

Methods	ASD↓	$\text{ASD}_{in}\downarrow$	$\text{ASD}_{mis}\downarrow$
Event Feature	0.123	0.123	0.122
FullyNN	0.073	0.069	0.072
CNTPP (ours)	**0.045**	**0.042**	**0.044**

Table 2: Comparison of (normalized) Average Sum of Distances with different methods on the real-world dataset

from Dec 9, 2020 - April 24, 2021. After that, we only use the tweets with labels of 'MisInformation' and 'Information', including a total of 169,008 tweets from 24,192 users.

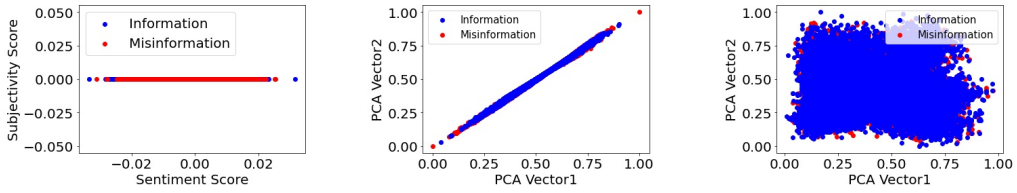
A.6 Real World Data Experiments

For the impact on sentiments and subjectivity experiments, we use first use TextBlob sentiment analyse tool [3] to generate the continuous sentiment and subjectivity score of the content user posts, where the former reflects the emotional charge of a statement and the latter reflects the degree of objectivity. After that, we use the tweets' timestamp to calculate the time interval, and use the sentiment and subjectivity score including whether the news is positive (1) or not (0) as the covariates of our causal structure. We split the training set, validation set and test set according to the ratio of 80%/10%/10% to train the proposed model. Specifically, we adopt the Adam optimizer with learning rate $1e-5$ and the epoch is set to be 500 with batch size 128. In addition, we add the dropout with 0.1 rate.

For the impact on text representation, we use the pre-trained BERT model [2] to extract the representation vectors of each tweet's content with 768 dimension. After that, we use PCA tools to reduce into 2 dimensions and use the new vectors as the covariates in a new model. The training setting is same with the impact on sentiments and subjectivity experiments.

Note that all the experiments are finished on the NVIDIA GeForce RTX 2080 TI GPUs for the fair competition.

Besides, we also evaluate FullyNN (a baseline) on the real-world dataset. Its results are visualized in Figure 2(a) and 2(b). As we can see, it can not acquire recognizable differences between information and misinformation. We further visualize the raw event features (BERT embedding with dimension reduction of PCA, not finetuned on fake news detection) in Figure 2(c). As we can see, it is also hard for a pre-trained language model to distinguish information and misinformation in an unsupervised manner.



(a) Impact on sentiment scores, estimated by FullyNN.

(b) Impact on text representation, estimated by FullyNN.

(c) Visualization of raw event feature.

Figure 2: Visualization of the results from baselines on real world

We also calculate Normalized Averaged Sum of Distances (NASD) for information cluster and misinformation cluster. More specifically, we first normalize the scale range of each embedding dimension (ITE vector or raw input) to be same for different methods to ensure that the scales embedding spaces of different methods are comparable. Then, we calculate their Averaged Sum of Distances (ASD), which is defined as $ASD = \sum_{i=0}^{k} \sum_{p \in C_i} (p - m_i)/n$, where C_i is the set of ITE results (or event feature) and $i = 1$ means the information results, $i = 0$ means the misinforamtion results, p stands for an instance in C_i, m_i is the center of all instance representations in C_i, n is the

total number of instance. The lower these metric are, the better that information and misinformation are distinguished. We also separately report the ASD on two clusters $ASD_{in} = \sum_{p \in C_0}(p - m_0)/n_0$ and $ASD_{mis} = \sum_{p \in C_0}(p - m_1)/n_1$ The comparison of our model against FullyNN and event features on this metric is shown in Table 2.

References

[1] N. Campbell. The study of discontinuous phenomena. *Proc Cambr. Phil. Soc*, vol. 15:pp. 117–136, 1909.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Steven Loria et al. textblob documentation. *Release 0.15*, 2:269, 2018.