Causality based event sequence modelling

Nikita Paplavskii [1] **Polina Pilyugina** [1]

Abstract

This project is a part of the Models of Sequential Data course and is dedicated to the implementation of the Causal based Neural Hawkes model for event sequences. As a part of the project, we will identify the most suitable method for the creation of a causal graph for event sequences data. Further, we introduce the causal graph into the Neural Hawkes method to improve the model's performance. In the end, we will compare Causal based Neural Hawkes with original Neural Hawkes on real-world and synthetic data.

1. Introduction

The problem of event sequences modelling has drawn increased attention in recent years. Event sequences arise in many fields:

- *Banking*: events of customers' interactions with the banking system. Analysis of such sequences allows identifying fraudulent behaviour or creating individualised marketing products.
- *Medical events*: events of medical services usage by different patients. In this case, researchers are interested in exploring sequences of events of other patients to predict new patients' future medical occurrences.
- *Customer actions*: events of interaction of customers with some online services. To provide targeted marketing, researchers are interested in analysing sequential patterns in customers' activities.
- *Recorded activity*: events recorded using smart wearable electronics and additional tracking apps. Patterns in human behaviour can make individualised recommendations easier.
- *Other events*: such as social media behaviour, website activities etc.

With recent advances in Big Data gathering and storage, as well as increased computational capabilities, high-resolution event sequences data has become available for researchers. In order to model such type of data several approaches were introduced. In this work we will focus on the Neural Hawkes model from (Mei & Eisner, 2017a). We aim to introduce the concept of causality between event sequences into this model and evaluate whether this helps to improve the performance over the original model. However, detection of causality is an important problem and has been widely researched in the context of time series data. Even though there are many approaches for causality estimation in case of standard discrete-time time series data, in case of continuous-time event sequences the process of causality estimation is still being investigated. Therefore, we firstly aim to find and evaluate various approaches for causality estimation. And further we aim to change the Neural Hawkes model to incorporate causality.

This paper has the following structure: in section 2 we define the problem and motivation behind the research. In section 3 we identify the main existing works related to this topic. Further, in section 4 we identify the main methods we use in our implementation. The code structure and its main features are described in section 5. Finally, we will provide the results of our experiments in final section 6.

2. Problem statement

Given K types of possible events, the event sequence E is defined as:

$$E = ((k_1, t_1), (k_2, t_2), ..., (k_N, t_N)) \qquad (1)$$

where N is the total number of events in sequence, $k \in \{1, 2, ..., K\}$ is an event type that occurred and $t_1 < t_2 < ... < t_N$ is a continuous time of event occurrence. The problem of predicting the future of a sequence means identifying which events are likely to happen next and when they will happen. This problem is different from standard time series forecasting since the event sequence is in continuous time, and no fixed time-lagged observation is available. Moreover, events are discrete and sparse, which even more complexifies the problem. Therefore standard methods for time series

[*]Equal contribution [1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Nikita Paplavskii <Nikita.Paplavsky@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

analysis are not of particular use when dealing with event sequences and novel approaches were introduced.

A basic model for event streams is the non-homogeneous multivariate Poisson process. It assumes that an event of type k occurs at time t with probability $\lambda_k(t)dt$. The value $\lambda_k(t) \geq 0$ is an intensity function and can be also regarded as a rate of event occurring per unit time. A more flexible generalization which captures interconnections between event types is known as Hawkes Process and was firstly described in (Hawkes, 1971). It takes into account that historic events can raise intensities of not only the same type events, but also other event types. The original Hawkes model describes a set of event intensities as:

$$\lambda_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k} \text{exp}(-\delta_{k_h,k}(t - t_h)) \qquad (2)$$

where $\mu_k \geq 0$ is the base intensity of event type k, $\alpha_{j,k} \geq 0$ is the degree to which an event of type j initially excites an event of type k, and $\delta_{j,k} \geq 0$ is the decay rate of that excitation. The impact function of the process in this form is therefore $\alpha_{k_h,k}\text{exp}(-\delta_{k_h,k}(t - t_h))$ According to the Theorem 4.1 from (Eichler et al., 2017), learning whether event of type k_h Granger-causes event of type k is equivalent to detecting whether the impact function is all-zero. Given the functional formulation of the impact function, in general we can deduct that the binarized infectivity matrix $A = [sign(\alpha_{j,k})]$ is the adjacency matrix of the corresponding Granger Causality Graph. Therefore, the task of learning Granger Causality Graph is equivalent to learning the binarized infectivity matrix A. There are several existing approaches on how to estimate such matrix, which we will discuss in section 4.

There are different approaches to model this process, as was discussed in section 1, however this project is dedicated mainly to the Neural Hawkes model from (Mei & Eisner, 2017b). In this model authors introduce Continuous Time LSTM (CTLSTM) network for modelling of event sequences. For this they reformulate the original Hawkes process in the form of Neural Hawkes process. This approacn has many benefits, which allowed the authors to achieve the great performance on both real-world and synthetic data. However, their formulation of the process does not have an explicit formulation of the $\alpha_{j,k}$, which are present in the original formulation. Therefore, it by construction cannot take advantage of the known-in-advance matrix of such coefficients. However, if we know the estimated matrix, or at least the binarized infectivity matrix A, it should allow to more precisely model the process.

Given the above, our problem consists of two main parts. Firstly, given the unknown multivariate point process we want to estimate its infectivity matrix A, binarized version of which is the Granger Causality Graph of such a process. Secondly, we want to make use of such information in the Neural Hawkes model in order to improve its performance. Our hypothesis is that improved Causal Neural Hawkes model will outperform the original implementation in terms of prediction quality.

3. Related Work

Among the main approaches for analysis of event sequences is using Hawkes process modelling (Hawkes, 1971). The formal description of this process was described in section 2, equation 2. In recent years several approaches were introduced for Hawkes process modelling: parametric, such as (Xu et al., 2020), and neural-based, such as (Mei & Eisner, 2017a) and (Zuo et al., 2021). As we already mentioned, we focus our work on the Neural Hawkes CTLSTM model from (Mei & Eisner, 2017b). The authors suggest a method to improve Hawkes model's expressiveness (Hawkes, 1971). One limitation of the original approach is that positivity constraints on α and μ in the Hawkes process limit its expressivity. First, the positive interaction parameters $\alpha_{j,k}$ fail to capture inhibition effects, in which past events reduce the intensity of future events. Second, the positive base rates μ_k fail to capture the inherent inertia of some events, which are unlikely to happen until their cumulative excitation by past events crosses some threshold. To remove these limitations, authors suggested two self-modulating models.

At first, they relaxed positivity constraints over $\alpha_{j,k}$ and μ_k allowing them to range over $\mathbf{R}$, which allows inhibition ($\alpha_{j,k} < 0$) and inertia ($\mu_k < 0$). To solve possible negativity of resulting total activation they pass it through a non-linear transfer function $f : \mathbf{R} \to \mathbf{R}^+$ to obtain a positive intensity function as required:

$$\begin{aligned} \tilde{\lambda}_k(t) &= \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k} \, \text{exp}(-\delta_{k_h,k}(t - t_h)) \\ \lambda_k(t) &= f_k(\tilde{\lambda}_k(t)) \end{aligned} \qquad (3)$$

Secondly, to improve expressiveness of Hawkes model they introduced CTLSTM to deal with more complex dependencies and to override an assumption that previous events have independent, additive influence on $\tilde{\lambda}_k(t)$. They introduced modified Neural version of Neural Hawkes process. In this process each event type k still has a time-varying intensity $\lambda_k(t)$, which jumps discontinuously at each new event, and then drifts continuously toward a baseline intensity. However, in the new process these dynamics are modelled by a hidden state vector $h(t) \in (-1, 1)^D$. This hidden state vector depends on the vector $c(t) \in \mathbf{R}^D$ of memory cells in a CTLSTM model. CTLSTM model is inspired by the discrete-time LSTM (Graves, 2012). One of the main differences is that in CTLSTM, similar to the original Hawkes process, the intensity exponentially decays at some rate toward a baseline intensity. However, in terms of CTLSTM it is equivalent to the exponential decay of each memory cell c toward some steady-state $\bar{c}$ at a rate δ. Therefore the reformulation of the equations 3 in the CTLSTM problem works as follows.

Given a time $t > 0$, the intensity of type k event $\lambda_k(t)$ is given by the following equations:

$$\lambda_k(t) = f_k(w_k^T h(t))$$
$$h(t) = o_i \odot (2\sigma(2c(t)) - 1) \text{ for } t_\in (t_i, t_{i+1}] \quad (4)$$

where $h(t)$ is hidden state and $c(t)$ is memory cell.

More information on this formulation can be found in the original paper. The authors used scaled SoftPlus function as non-linear transfer function f. Among benefits of this model is that it allows to fit datasets in which some pairs of event types do not influence one another, therefore having zero interaction coefficients. Although the Neural Hawkes provides promising results, these approach does not allow to add estimated information about internal causal relationships. Moreover, the authors identify that adding information on the causal relationship between event sequences will be an essential extension to the model. We hypothesise that one can increase the model performance by considering the existence or non-existence of causal relationships between series. If we know the infectivity matrix A in advance, it can help to train more reliable model.

Many recent works outline the importance of Granger Causality graphs for time series data analysis, such as (Arnold et al., 2007) and (Basu et al., 2015). Granger causality is a statistical concept of causality between stochastic processes based on prediction. It is a method to evaluate causality between processes and it can be described as follow:

- Let X and Y be stationary stochastic processes.
- Denote with $\mathcal{U}_i = (U_{i1}, ..., U_{i\infty})$ all the information in the universe until time i, and with $\mathcal{X}_i = (X_{i1}, ..., X_{i\infty})$ all information in X until time i.
- Denote with $\sigma^2(Y_i|\mathcal{U}_i)$ the variance of the residual of predicting Y_i using $\mathcal{U}$ at time i.
- Denote with $\sigma^2(Y_i|\mathcal{U}_i \setminus \mathcal{X}_i)$ the variance of the residual of predicting Y_i using all information in $\mathcal{U}_i$ at time i except $\mathcal{X}_i$.

If $\sigma^2(Y_i|\mathcal{U}_i) < \sigma^2(Y_i|\mathcal{U}_i \setminus \mathcal{X}_i)$ then we say that X Granger-causes Y, and write X⇒Y. If X⇒Y and Y⇒X by the described above definition, we say that feedback is occurring. Therefore, we can create a non-symmetric adjacency matrix of causal relationship between events. However, their approaches are mostly based on vector autoregressive models for discrete time-lagged variables. This approach is helpful for time series but is not directly applicable to continuous-time event sequences data. Therefore, other approaches arose for modelling Granger Causality on event sequences. Among them is the work of (Xu et al., 2016) and (Zhou et al., 2013) which propose an approach to learn Granger Causality for Hawkes process.

4. Methods

4.1. Causal-based Neural Hawkes

In order to include causality matrix into the Neural Hawkes we have slightly modified its implementation. The general idea was: given the binarized infectivity matrix $A \in 0, 1^K$ and estimated interaction coefficients matrix $W \in [0, 1]^K$, we multiply the weighted hidden state $w_k^T h(t)$ by $A \odot W$ to incorporate interrelationship between event types into the model. Firstly, we note that the matrix $w_k^T h(t)$ from equation 3 is of size (T, K) and it contains estimated unnormalised intensities of each event type. Further, matrices A and W are both of size (K, K) and the matrix A can be viewed as a mask matrix for W. Then if we multiply $w_k^T h(t)$ with $A \odot W$ from the right, we will obtain the same size matrix with intensities weighted via estimated interaction coefficients for each event type. We further pass it through SoftPlus transfer function f_k to obtain final intensities λ_k.

In terms of network implementation, the matrix $w_k^T h(t)$ is obtain via passing the hidden state $h(t)$ through the linear layer of size (T, K), which is further passed through SoftPlus function. We will further refer to matrix $A \odot W$ as *weighted_A*. For experiments we have created three versions of the model:

1. *CausalNeuralHawkesMasked*: we initialize matrix W as all-ones with no gradient
2. *CausalNeuralHawkesMaskedWeighted*: we initialize matrix W using the estimated interaction coefficients with no gradient
3. *CausalNeuralHawkesTrainableWeighted*: we initialize matrix W using the estimated interaction coefficients with gradient

The first model thus uses only the fact that estimated causality between events may or may not be non-zero and alters intensities with the coefficient equal to 1 accordingly. However, such an approach seems to be too strict and limiting. Moreover, using most causality estimation methods we are able to estimate not only matrix A, but also its non-binarized form W, and it seems wise to incorporate this information additionally in the model. But this model still does not allow for variability and harshly depends on the quality of W estimation. Therefore we finally came up with the third model, in which the model can learn weights of the matrix W while still incorporating the estimated absence of causal relationship.

4.2. Evaluation process

In order to evaluate model performance we rely on log-likelihood loss, which we optimize during training.

$$l = \sum_{h:t_h<T} \log \lambda_{k_h}(t_h) - \int_{t=0}^{T} \lambda(t)dt \tag{5}$$

It is the sum of the log-intensities of the events that happened, minus an integral of the total intensities over the observation interval. The problem of the integral estimation is solved by the authors of original work (Mei & Eisner, 2017b) using Monte Carlo trick and we also rely on this approach. Additionally, we estimate type-validation accuracy at each epoch on the validation dataset. For this we use the model to estimate the type of event, given the actual time it happened. We calculate accuracy as follows:

$$Accuracy = \frac{\#Correct\ guesses}{\#All\ guesses} \tag{6}$$

In order to evaluate prediction performance of the model, we calculate root mean square error of predictions of time durations. Additionally, we plot the predicted and true values of events and their timings. In order to obtain predictions, we rely both on the original paper an implementation in NeuralHawkesPytorch. Given that we have already estimated intensities, the probability density function of the next event happening at time t is given by:

$$p(t) = \lambda(t) \exp\left(-\int_{t_{i-1}}^{t} \lambda(s)ds\right) \tag{7}$$

Therefore estimated time can be calculated as an expectation from this probability function:

$$\hat{t} = \int_{t_i-1}^{\infty} tp(t)dt \tag{8}$$

However, direct calculation of this integral is problematic and therefore we rely on the trick from NeuralHawkesPytorch implementation. This allows us to estimate time durations of future events and thus their types as well, via applying the pretrained model to this data.

4.3. Granger causality estimation

In order to obtain matrix A we use several methods for estimation of Hawkes process parameters. Following the results from section 2, A is the binarized version of interaction coefficients $\alpha_{k_h,k}$. Therefore, we firstly focus on estimating matrix W which contains interaction coefficients $\alpha_{k_h,k}$. Secondly, we obtain matrix $A = [sign(\alpha_{j,k})]$. For estimation of W we use approaches from (Zhou et al., 2013) and (Xu et al., 2016). The first approach will be referred to as ADM4 and this model relies on exponential parametrization of the kernels and a mix of Lasso and nuclear regularization for Hawkes parameters estimation. The formal definition of the Hawkes process with exponential parametrization is then as follows:

$$\lambda_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k}\beta \exp(-\beta(t-t_h)1_{t>0} \tag{9}$$

where β is the decay parameter, while all other parameters share the same meaning as in equation 3. Another approach we use relies on parametrization of the kernels as a sum of Gaussian basis functions and a mix of Lasso and group-Lasso regularization. We will refer to this approach as SumGaussians, and the parametrization techniques used in this approach allow to represent the Hawkes process as follows:

$$\begin{gathered} \lambda_k(t) = \mu_k + \sum_{h:t_h<t} \phi_{k_h,k}(t-t_h) \\ \text{where } \phi_{k_h,k} \sum_{m=1}^{M} \alpha_{k_h,k} f(t-t_m), \\ \text{and } f(t) = (2\pi\sigma^2)^-1 \exp(t^2/(2\sigma^2)) \end{gathered} \tag{10}$$

with σ being a standard deviation of Gaussian basis function. Both of these approaches allow to estimate matrix W, but differ in methodology and parametrization. The main problem of both of this approaches is their poor scalability with increased number of event types.

4.4. Synthetic datasets creation

In order to create synthetic datasets we use simulation techniques available in *tick* library. We rely on the exponential parametrization of kernels for simulation. We initialize matrix W for the process such that it can have zeros on diagonal and elsewhere, but the non-zero values we set to 0.2. Except for the case of synthetic dataset with 2 event types, in which we initialize W equal to:

$$W = \begin{bmatrix} 0.2 & 0 \\ 0.1 & 0.2 \end{bmatrix} \tag{11}$$

Overall we have created 5 synthetic datasets, which we will describe further in section 6.

5. Implementation

We base our work on some existing available implementations in terms of the code. Firstly, existing implementation of Neural Hawkes with PyTorch provides us with a base implementation of CTLSTM model in NeuralHawkesPytorch repository. It relies on the CTLSTM Cell code from the original Neural Hawkes repository in https://github.com/xiao03/nh. From this repository we have used the original model implementation, training procedure and prediction procedure. However, we made significant changes to the original code to make it more flexible and structured. Firstly, we have created a structured python package called `causal_nh`, in which we introduced modules for model training and evaluation, synthetic dataset creation and granger causality estimation. In the model section we have added three versions of the Causal Neural Hawkes as described in section 4.1. For this we have adopted the code from repo with necessary modifications. We have also added several modifications to make all code compatible with CUDA in order to speed up the training and testing. Training and predicting examples are contained in notebooks.

We have found several existing implementations of Granger Causality estimation for event sequences, described in section 4.3. One of the implementations of (Xu et al., 2016) approach can be found in `tick` library method HawkesSumGaussians. Another method based on the work (Zhou et al., 2013) is also available in the `tick` library: ADM4.

In order to create synthetic datasets we use the *tick* library. We provide the code for synthetic dataset creation in *utils* module and an example of its usage in *Obtaining_synthetic_data* notebook.

The code of our implementation is provided on GitHub repository. More details on the repository structure can be found in B.

6. Results

6.1. Baseline

As for the baseline, we will use Neural Hawkes implementation described in section 5. In our opinion, this would be the best choice as we base our new models implementations on this code.

6.2. Experimental setup

For the synthetic datasets, we rely on *tick* [1] library. Description of these datasets is contained in table 1 where K is a number of event types, N — a number of sequences and T is a maximal sequence length.

We train all models with following parameters: number of epochs equal to 100 , batch size equal to 100, and learning rate of 0.01.

[1] `https://x-datainitiative.github.io/tick/index.html`

dataset_name	K	T	N
data_synth_5_events	5	54	300
data_synth_2_events	2	31	100
data_synth_3_events	3	55	300
data_synth_10_events	10	31	300

Table 1. Description of synthetic datasets used

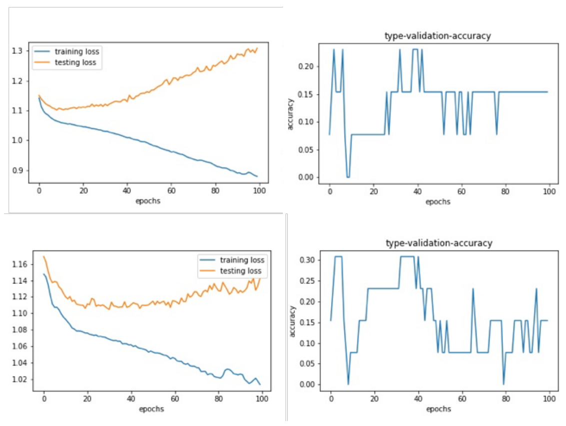


Figure 1. Training metrics of original NH (top) and CNHTW (bottom) on the 10 event synthetic dataset

6.3. Results

We firstly ran experiments on these synthetic datasets. With this we aimed to evaluate the performance of 3 models we described in section 4.1 and compare them. The resulting statistics can be found in table 2. In this table L — log likelihood at the end of training, rmse is root mean squared error of time durations estimation on test data, and accuracy is accuracy of predicted event types on test data. In case of synthetic datasets, since we have created them with predefined matrices A and W, we supply models exactly with these matrices. In general, we can see that original Neural Hawkes seem to perform better in terms of training loss, however our training figure 1 shows that these results are because NH harshly overfits on train, although the training parameters were the same for all models. The same picture holds for all synthetic datasets. While our model also slightly overfits, the results are not as bad as in case of NH. Moreover, validation accuracy of type prediction is slightly higher for our model.

Interestingly, we can see that the model without possibility to learn and thus adjust W performs better in terms of accuracy for datasets with small number of event types. However in general the CNHTW performs better than other models. Also we tried to explore the quality of predictions on test data. An example of such comparison can be found on 2. We can see that both models have actually reasonable predictions and are definitely capturing the intensity of events.

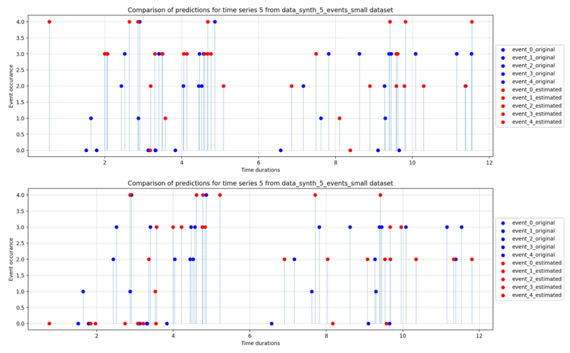


Figure 2. Predictions of original NH (top) and CNHTW (bottom) on the sample time series from 5 event synthetic dataset

In our repository we also provide a link to the folder with the detailed results of all runs and saved models.

7. Conclusion

In general, we can see that on synthetic datasets with known in advance matrix of interaction coefficients, the best performance among the causal based models is obtained using the *CausalNeuralHawkesTrainableWeighted* model. It provides the best scores in terms of log likelihood and RMSE of predicted time durations among causal models. Although Neural Hawkes outperforms our models in terms of loss, it significantly overfits and thus the resulting accuracies are not the best when evaluationg on test. Moreover, our model provides reasonable predictions of both event types and timings.

In general, we believe that we were successful in implementing the Causal Based Neural Hawkes model and our results suggest that this model indeed works. Moreover, our implementation is fully available for future usages and allows CUDA training, which is extremely useful in this case, as training of Neural Hawkes is extremely computationally intense. We have tested several variations of the model in order to evaluate their performance, and we believe our findings will be extremely useful for further investigations.

7.1. Limitations

One limitation that we were struggling with is the computational intensity of Neural Hawkes. Even with CUDA one epoch takes up to 60-70 seconds on mdeium sized datasets. Similar limitation concerns the module for granger causality estimation, as it scales exceptionally badly with increased number of events.

References

Arnold, A., Liu, Y., and Abe, N. Temporal causal modeling with graphical granger methods. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 66–75, 2007. doi: 10.1145/1281192.1281203.

Basu, S., Shojaie, A., and Michailidis, G. Network granger causality with inherent grouping structure. *Journal of Machine Learning Research*, 16(13):417–453, 2015. URL `http://jmlr.org/papers/v16/basu15a.html`.

Eichler, M., Dahlhaus, R., and Dueck, J. Graphical Modeling for Multivariate Hawkes Processes with Nonparametric Link Functions. *Journal of Time Series Analysis*, 38(2):225–242, 2017. ISSN 14679892. doi: 10.1111/jtsa.12213.

Graves, A. Supervised sequence labelling with recurrent neural networks. 2012.

Hawkes, A. G. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971. ISSN 00063444. URL `http://www.jstor.org/stable/2334319`.

Mei, H. and Eisner, J. The neural hawkes process: A neurally self-modulating multivariate point process, 2017a.

Mei, H. and Eisner, J. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, Long Beach, December 2017b. URL `https://arxiv.org/abs/1612.09328`.

Xu, G., Wang, M., Bian, J., Huang, H., Burch, T. R., Andrade, S. C., Zhang, J., and Guan, Y. Semi-parametric learning of structured temporal point processes. *Journal of Machine Learning Research*, 21(192):1–39, 2020. URL `http://jmlr.org/papers/v21/18-735.html`.

Xu, H., Farajtabar, M., and Zha, H. Learning granger causality for hawkes processes, 2016.

Zhou, K., Zha, H., and Song, L. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. *Journal of Machine Learning Research*, 31:641–649, 2013. ISSN 15337928.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer hawkes process, 2021.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Nikita Paplavskii (50% of work)

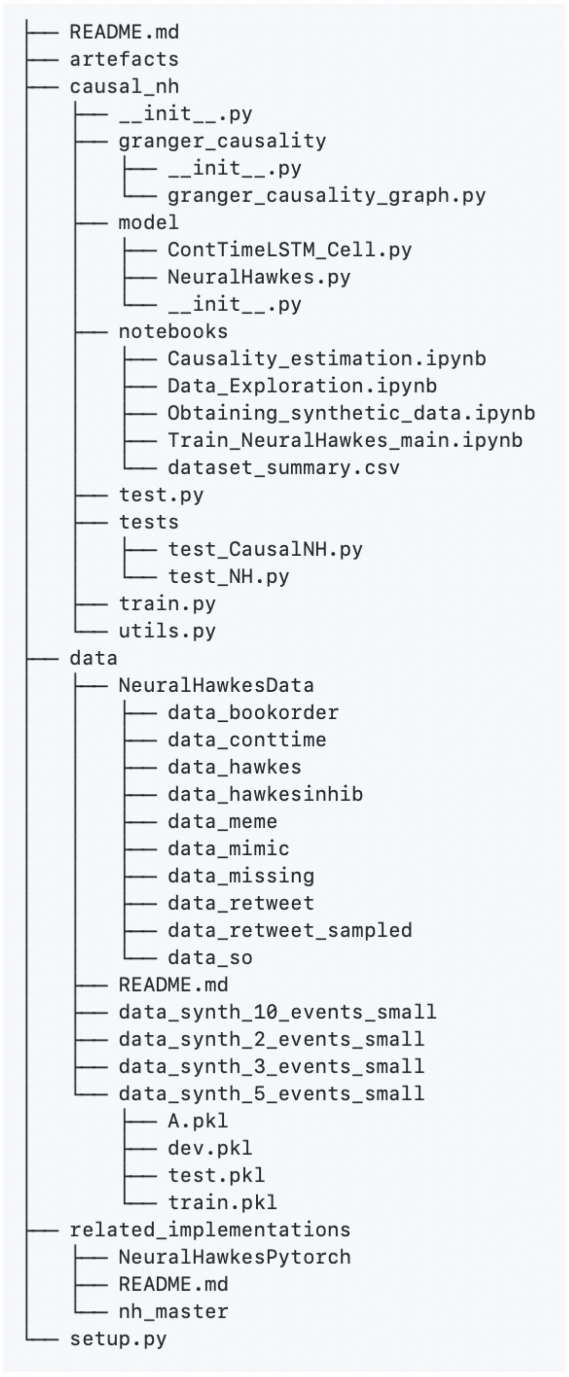
- Review of the literature on Granger Causality for event sequences and possible implementations;
- Integration of Granger Causality into Neural Hawkes Model;
- Refactoring of the models to make CUDA compatible;
- Experiments with Causal based Neural Hawkes;

Polina Pilyugina (50% of work)

- Review of literature on Neural Hawkes model and Hawkes processes;
- Integration of Granger Causality into Neural Hawkes Model;
- Synthetic datasets creation for experiments;
- Implementation of prediction code and plotting functions

B. Repo structure

Figure 3 contains the resulting structure of the repository. Our implementation can be installed from the setup.py file as a package. The main folder is causal_nh, which contains two main modules: granger causality and model. Granger causality module contains a function to estimate matrices A and W from the given dataset. An example of its usage on our synthetic data can be found in the notebooks. The model module contains modified versions of Neural Hawkes models. The code for training and testing the model is contained in the root of causal_nh, while the example of its usage can be found in notebooks. Additionally, module with utils contains useful functions for plotting and synthetic dataset generation. Data folder contains all possible datasets on which the model can be run, including the original Neural Hawkes datasets and our synthetic ones. In the folder with related implementations we store the original implementations of Neural Hawkes, on which we base our code.



```
├── README.md
├── artefacts
├── causal_nh
│   ├── __init__.py
│   ├── granger_causality
│   │   ├── __init__.py
│   │   └── granger_causality_graph.py
│   ├── model
│   │   ├── ContTimeLSTM_Cell.py
│   │   ├── NeuralHawkes.py
│   │   └── __init__.py
│   ├── notebooks
│   │   ├── Causality_estimation.ipynb
│   │   ├── Data_Exploration.ipynb
│   │   ├── Obtaining_synthetic_data.ipynb
│   │   ├── Train_NeuralHawkes_main.ipynb
│   │   └── dataset_summary.csv
│   ├── test.py
│   ├── tests
│   │   ├── test_CausalNH.py
│   │   └── test_NH.py
│   ├── train.py
│   └── utils.py
├── data
│   ├── NeuralHawkesData
│   │   ├── data_bookorder
│   │   ├── data_conttime
│   │   ├── data_hawkes
│   │   ├── data_hawkesinhib
│   │   ├── data_meme
│   │   ├── data_mimic
│   │   ├── data_missing
│   │   ├── data_retweet
│   │   ├── data_retweet_sampled
│   │   └── data_so
│   ├── README.md
│   ├── data_synth_10_events_small
│   ├── data_synth_2_events_small
│   ├── data_synth_3_events_small
│   └── data_synth_5_events_small
│       ├── A.pkl
│       ├── dev.pkl
│       ├── test.pkl
│       └── train.pkl
├── related_implementations
│   ├── NeuralHawkesPytorch
│   ├── README.md
│   └── nh_master
└── setup.py
```

Figure 3. Structure of the repository

	NH			CNHM			CNHW			CNHTW		
Dataset	L	rmse	acc	L	rmse	acc	L	rmse	acc	L	rmse	acc
data_synth_10_events	0.88	0.131	0.1579	1.05	0.201	0.1316	1.06	0.324	0.1447	1.01	0.36	0.1316
data_synth_2_events	1.02	0.728	0.4762	0.93	0.876	0.4762	0.98	1.029	0.7143	0.94	0.47	0.619
data_synth_3_events	0.86	0.357	0.3056	0.87	0.345	0.2778	0.84	0.367	0.4722	0.88	0.35	0.25
data_synth_5_events	0.87	0.42	0.1818	0.89	0.417	0.3333	0.89	0.411	0.303	0.89	0.38	0.3333

Table 2. Results of synthetic experiments