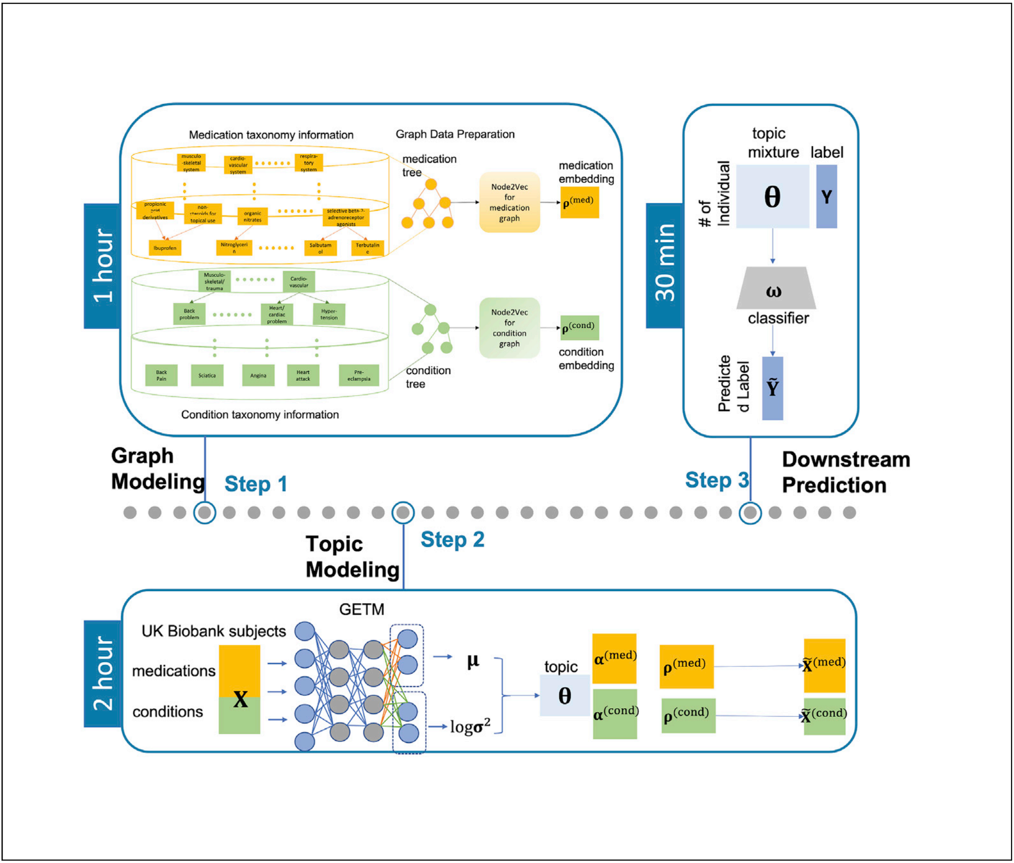
STAR Protocols

CellPress OPEN ACCESS

Protocol

Implementation of a graph-embedded topic model for analysis of population-level electronic health records



Yuening Wang, Audrey V. Grant, Yue Li

yueli@cs.mcgill.ca

Highlights

Preprocess the patient health record and graph taxonomy data

Apply graph learning scripts to obtain phenotype embedding

Infer distributions over medical codes to identify disease comorbidity

Infer phenotypic mixture membership to phenotype patients

To address the need for systematic investigation of the phenome enabled by ever-growing genotype and phenotype data, we describe our step-by-step software implementation of a graph-embedded topic model, including data preprocessing, graph learning, topic inference, and phenotype prediction. As a demonstration, we use simulated data that mimic the UK Biobank data as in our original study. We will demonstrate topic analysis to discover disease comorbidities and computational phenotyping via the inferred topic mixture for each subject.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Wang et al., STAR Protocols 4, 101966
March 17, 2023 © 2022
https://doi.org/10.1016/j.xpro.2022.101966

CellPress
OPEN ACCESS

Protocol

Implementation of a graph-embedded topic model for analysis of population-level electronic health records

Yuening Wang,[1] Audrey V. Grant,[2] and Yue Li[1,3,4,*]

[1]School of Computer Science, McGill University, Montreal, QC H3A 0G4, Canada

[2]Department of Anesthesia, McGill University, Montreal, QC H2A 0G4, Canada

[3]Technical contact

[4]Lead contact

*Correspondence: yueli@cs.mcgill.ca
https://doi.org/10.1016/j.xpro.2022.101966

SUMMARY

To address the need for systematic investigation of the phenome enabled by ever-growing genotype and phenotype data, we describe our step-by-step software implementation of a graph-embedded topic model, including data preprocessing, graph learning, topic inference, and phenotype prediction. As a demonstration, we use simulated data that mimic the UK Biobank data as in our original study. We will demonstrate topic analysis to discover disease comorbidities and computational phenotyping via the inferred topic mixture for each subject.
For complete details on the use and execution of this protocol, please refer to Wang et al. (2022).[1]

BEFORE YOU BEGIN

Protocol overview

This protocol will give a step-by-step guide to develop a machine learning model called GETM, which is built up ETM.[2] The method was developed to infer interpretable topics using large-scale health-related data with two types of medical features. The protocol includes steps: 1. processing raw medical data as bag-of-words input for topic inference; 2. Constructing taxonomical knowledge graph using taxonomical structure of medical features 3. Running scripts of knowledge graph modeling, topic modeling and medical outcome prediction. The model enables systematic investigation of entire structured phenomes by modeling multi-modal discrete patient medical records. Besides, in application to UK Biobank (UKB) self-reported clinical phenotype data, the model demonstrates overall superior performance in imputing missing conditions and medications.

Acquiring datasets

Timing: 1–2 h

1. Apply for access to UK Biobank at https://www.ukbiobank.ac.uk/enable-your-research/apply-for-access.

   Note: Request access to UK Biobank and its data of specific data fields might take days to weeks.

2. Request data from the data-fields: 20002 and 20003. The relevant information for each field could be found at https://biobank.ndph.ox.ac.uk/ukb/field.cgi?id=x by replacing x with the corresponding data field identifiers.

   Note: data-field 20002 records participants' self-reported non-cancer illness code and data-field 20003 records participants' medication/treatment code.

Optional: to replace the UK Biobank dataset with the datasets of the user's choice, please ensure that the datasets of interest include each individual's taken medications and historical diseases/conditions. Besides, information of disease-disease and medication-medication relationships is also needed to construct knowledge graph.

3. Download anatomical therapeutic chemical (ATC) dataset from https://www.who.int/tools/atc-ddd-toolkit/atc-classification.

   Optional: to replace ATC dataset with other datasets of the user's choice, please make sure that the datasets include the medication-medication relation network, from which the medications could map to the medications of the data in previous step.

4. Request data from the data-fields: 6159, 3799, 4067, 3404, 3571, 3741, 3414, 3773 and 2956. The relevant information for each field could be found at https://biobank.ndph.ox.ac.uk/ukb/field.cgi?id=x by replacing x with the corresponding data field identifiers.

   Note: the data-fields describes participants' pain phenotypes as following:

6159: pain type(s) experienced in last month.

3799: headache for 3+ months.

4067: facial pains for 3+ months.

3404: neck/shoulder pain for 3+ months.

3571: back pain for 3+ months.

3741: stomach/abdominal pain for 3+ months.

3414: hip pains for 3+ months.

3773: knee pains for 3+ months.

2956: general pain for 3+ months.

   Optional: to replace UK Biobank pain-related data, please ensure that the datasets include the medical outcome (such as pain) of individuals from step 2.

Software installation

Timing: 2–4 h

5. Clone the code repository https://github.com/li-lab-mcgill/getm as following:

```
>git clone https://github.com/li-lab-mcgill/getm.git
```

6. Install packages according to the requirements file as following:

```
>pip install -r requirements.txt
```

Note: Graphical Processing Unit (GPU) usage is strongly recommended though not required. With the help of GPU, we could save about 1.5–2 times of training time. For larger datasets, the training is expected to benefit more.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
UK Biobank dataset	https://biobank.ctsu.ox.ac.uk/	N/A
Source code	https://github.com/li-lab-mcgill/getm	N/A
Software and algorithms		
Python 3.8.0	https://python.org/downloads/	SCR:008394
matplotlib 3.1.3	https://matplotlib.org/	SCR:008624
networkx 2.5	https://networkx.org/	N/A
NumPy 1.9.0[3]	https://numpy.org/	SCR:008633
node2vec 0.3.0	https://snap.stanford.edu/node2vec/	N/A
pandas 0.25.3	https://pandas.pydata.org/docs/	SCR:018214
PyTorch 1.4.0	https://pytorch.org/	N/A
scikit-learn 0.22.1	https://scikit-learn.org/stable	SCR:002577
scipy 1.4.1	https://scipy.org/	SCR:008508

STEP-BY-STEP METHOD DETAILS

The following are detailed instructions on how to implement the method. We have created a simulated dataset ***Simulation_data*** and show examples of each step in a tutorial jupyter notebook called ***Tutorial***. Those can be found in the repository https://github.com/li-lab-mcgill/getm.

Data preprocessing for topic modeling (scripts/data_utils.py)

Timing: 1 h

This section describes 1). The data filtration by participants' ethnic group, so to reduce potential confounding by ethnic groups 2). The creation of medication and condition sets according to medication/condition's relatedness to pain 3). The preparation of the input file for the topic model.

1. Filter out UK Biobank data of which the individuals are not European descent.
   a. To reduce potential confounding caused by ethnic groups, we focus on only the European descents.
   b. Based on UK Biobank Field 21000, we create a dictionary indicating whether an individual is a European descent. The key of the dictionary is the individual id and the value is the Boolean type object. We can refer to the function ***create_dict()*** for implementation.

```
/* Create a dictionary indicating whether an individual is a European descent*/

>white_dict = create_dict(eth_data, ''ethnicity'', ''white'')
```

   c. Filter UKB phenotype data by individual ids according to the dictionary from previous step.

```
>med_data_eth = [white_dict[u] for u in med_data[user_id]]

>med_data["ethnicity''] = med_data_eth

>filtered_med_data = med_data[med_data["ethinicity'']==1]
```

2. Use active ingredients as medications.
   a. Extract active ingredients of each medication's name in UKB field 20002.
   b. Replace medication codes from UKB 20002 with their corresponding active unique ingredients.
   c. Each active ingredient is considered as a unique medication.
3. Remove duplicate conditions.
4. Create different sets of medications and conditions. Of each set, keep/remove certain medication or condition based on their relatedness with medical outcome of interest (such as chronic pain, etc.) using physician's curated lists or odds ratios:
   a. For each interested pain type, get a medication list and a condition list curated by physicians that are correlated this type of pain.
   b. Calculate the odds ratio of each condition and medication with respect to this specific pain type as the outcome. Create a medication list and condition list with medications and conditions with top odds ratio.
   c. Create new different filtered medication sets by removing from the entire medication set the medications of physician curated list or odds ratio or the union of both. Create new filtered condition sets similarly.
   d. Please refer to function **create_filtered_set()** for implementation.

```
> new_med_set = create_filtered_set(med_set, gp_related_med)
```

5. For N individuals, selected medication set of M medications and selected condition set of C conditions, create a Python NumPy array $X \in \mathbb{R}^{N \times (M+C)}$, of which we set 1 if we observe medication or condition for the individual, otherwise we set 0. We can refer to the function ***combine_data()*** and ***create_input_file()*** for implementation.

```
>combined_data = combine_data(filtered_cond_data, filtered_med_data)

>X, med_idx, cond_idx = create_input_file(combined_data, new_med_set, new_cond_set)
```

6. Save the array using NumPy's *save()* method from previous step to the file location specified by *data_path* while running GETM.

```
> np.save(data_path, X)
```

Data preprocessing for graph learning (scripts/data_utils.py)

Timing: 2 h

This section describes 1). The construction of medication and condition knowledge graphs 2). The creation of input files for the graph model.

7. Create a tree graph of all non-duplicate conditions based on coding taxonomy designed by UKB team (i.e., data-field 20002).

   Note: The graph contains more condition nodes than that were observed of the individuals. The unobserved nodes are mainly from higher levels of the hierarchical tree (e.g., disease categories). We do not delete any nodes from the graph. Extra nodes help to learn more knowledge for those observed condition during message passing of graph model (Node2Vec).

8. Create a text file consisted of two columns in the format of **<node1, node2>**, where node1 and node2 are two connected nodes from tree graph in step 13. The number of rows is the number of all edges in the graph. We can refer to function ***create_cond_graph()*** **for implementation.**

```
> cond_graph = crate_cond_graph(cond_tree, saved_cond_graph)
```

9. Save the text file.
10. Create a tree graph of medications (or medication groups) based on Anatomical Therapeutical Chemical (ATC) classification system.
11. Map active ingredients from UKB data-field 20003 to the fifth level codes of ATC by ingredient name.
12. Replace the related ATC codes with the corresponding active ingredients from UKB data. For ATC codes mapped to same active ingredient, merge all corresponding nodes to one single node in graph created in step 16.
13. Create a text file consisted of two columns in the format of **<node1, node2>**, where node1 and node2 are two connected nodes from tree graph in step 18.
14. Please refer to function ***create_med_graph()*** for implementation of steps 16–20.

```
> med_graph = create_med_graph(med_tree, saved_med_graph)
```

15. Save the text file.

Data preprocessing for pain prediction

Timing: 1 h

This section illustrates how to extract UKB pain-related phenotype labels for supervised learning.

16. Define the outcome (e.g., chronic musculoskeletal pain).
17. Create a binary vector $Y \in \mathbb{R}^{N \times 1}$ based on individuals' answers from the related data-field.

    Note: For details about all UKB pain-related phenotype labels we extracted, please refer to section 10.3.2 in Wang et al.[1] Briefly, we use the data from step 4 to collected 9 pain-related labels.

18. Save the vector Y as a NumPy array using *save()* method.

Graph modeling on taxonomical knowledge graphs

Timing: 1 h

This section describes how to run separate graph models (i.e., node2vec[4]) on medication and condition knowledge to get pre-trained medication and condition embeddings. The embeddings are vectors projected to latent space which capture the structural knowledge of medications or conditions. The embeddings are inputs for topic modeling described in later steps.

19. Modify the script subjob/run_node2vec.sh.
    a. graph_file = path to save *.txt* file containing graph information (i.e., text file from steps 15 or 21).

Note: if tuning other parameters is required, please refer to details about each parameter in *scripts/node2vec.py*.

```
> bash run_node2vec.sh
```

20. The output are medication embeddings of all medications on the medication knowledge graph and condition embeddings of all conditions on the condition knowledge graph.

    Note: The embedding of a medication or condition is a vector in dimension D on latent space. D is a hyperparameter which is 128 in our setting. The output is a .txt file of which each row is in the format <node_id, embedding>.

21. Find and extract the rows of medication ids and condition ids of the medication set and condition set selected in step 11 for topic modeling.
22. Save the medication embeddings and condition embeddings of the selected sets from the previous step. The embeddings are saved as NumPy matrix using the *save()* method.

Topic inference

Timing: 2 h

This section describes how to run the graph topic model by appropriately loading input files and setting up the hyperparameters.

23. Modify the script subjob/run_metm.sh.
    a. data_path = path to load input data from step 12.
    b. save_path = path to save outputs.
    c. vocab_size1 = the number of unique medications.
    d. vocab_size2 = the number of unique conditions.
    e. embedding1 = path to load the pretrained medication embedding obtained from step 28.
    f. embedding2 = path to load the pretrained condition embedding obtained from step 28.

    Note: if tuning other parameters is required, please refer to details about each parameter in *scripts/main_multi_etm_sep.py*.

```
> bash run_metem.sh
```

24. The outputs are medication-specific topic mixture $\beta^{(med)} \varepsilon \ \mathbb{R}^{K \times M}$ which is K topics' distribution over M medications, condition-specific topic mixture $\beta^{(cond)} \varepsilon \ \mathbb{R}^{K \times C}$ which is K topics' distribution over C conditions and individual topic mixture $\theta \ \varepsilon \ \mathbb{R}^{N \times K}$ which is N individuals' distribution over K topics.

Pain prediction

Timing: 30 min

25. Modify the script subjob/ run_prediction.sh.
    a. input_file = path to load θ.
    b. label_file = path to load Y.
    c. save_label = path to load the predicted labels.

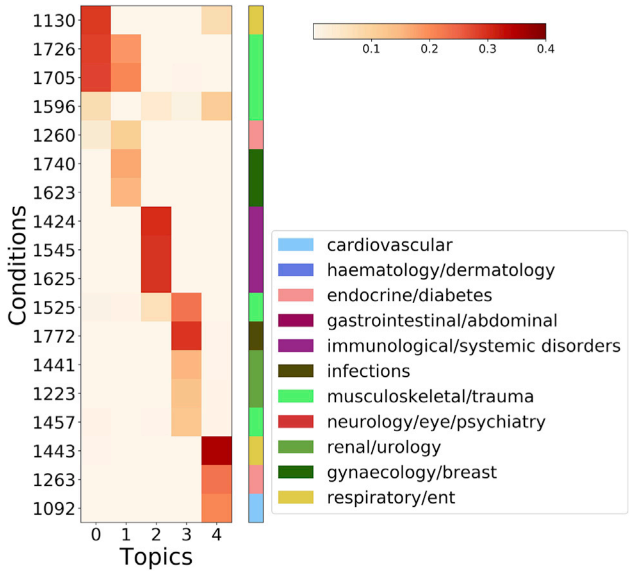


Figure 1. Condition topic quality visualization of the simulated data

> *Note:* if tuning other parameters is required, please refer to details about each parameter in *scripts/prediction.py*.

```
> bash run_prediction.sh
```

26. The output is predicted label array $\widehat{Y}$ which is a probability of being positive.

EXPECTED OUTCOMES

The expected results are displayed in Figures 1, 2, and 3 and described in detail below.

With $\beta^{(med)}$ and $\beta^{(cond)}$ from step 24, we could evaluate the medication (condition)-specific topic by calculating the topic coherence and topic diversity with different pre-defined topic numbers. Topic coherence evaluates the relatedness (i.e., whether the medications and conditions are from same category) of medications or conditions with top 5 probability for each topic. Topic diversity is the percentage of total unique medications/conditions of the top 5 medications/conditions of all topics. The exact codes to generate Figures 1 and 2 are shown in jupyter notebook *Tutorial*.

With $\widehat{Y}$ from step 26 we could evaluate performance of logistic regression (LR) by calculating area under the precision recall curve (AUPRC) and area under the receiver operating characteristic curve (AUROC). The exact codes to generate Figure 3 are shown in jupyter notebook *Tutorial.*

LIMITATIONS

To begin with, the datasets used in this protocol, i.e., UK Biobank and ATC, may be updated after they were accessed in this protocol. This may cause differences or irreproducibility of the results. In terms of topic evaluation, the lack of external categorical information of conditions results in poor topic coherence evaluation of condition-specific topic. Besides, the model is not able to integrate longitudinal healthcare information. In addition, graph modeling and topic inference are trained separately. Therefore, fine-tuning for both models is needed at the expense of computational resources.

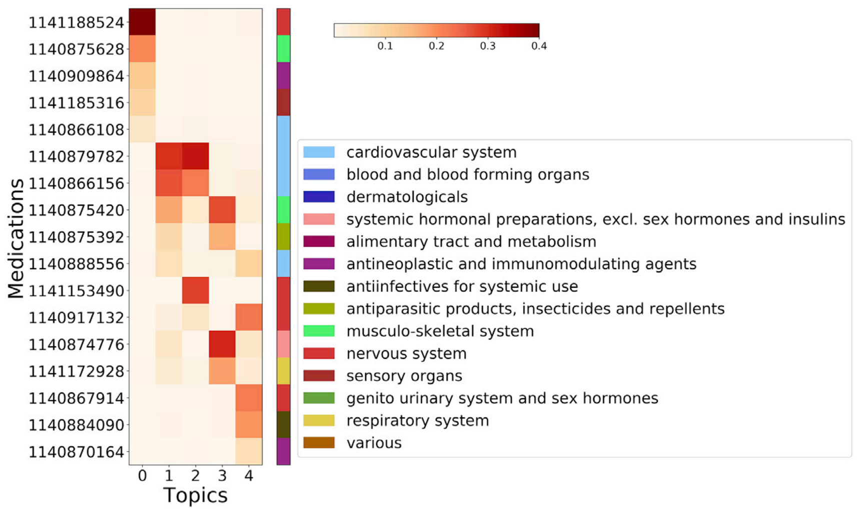


Figure 2. Medication topic quality visualization of the simulated data

TROUBLESHOOTING

Problem 1

The Python package libraries are not supported in the existing environment when installing required dependents (before you begin– software installation).

Potential solution

Create a virtual environment using *venv or conda*, and follow specific version of packages closely as instructed.

Problem 2

The medical knowledge graph contains edges with different weights, which cannot be ignored (step-by-step method details– graph modeling on taxonomical knowledge graphs).

Potential solution

The data format for graph modeling from step 14 and step 20 could be modified to *<node1, node2, weight>*, which adds one column of edge weight. Therefore, the model will consider different importance of the edges while do message passing.[1]

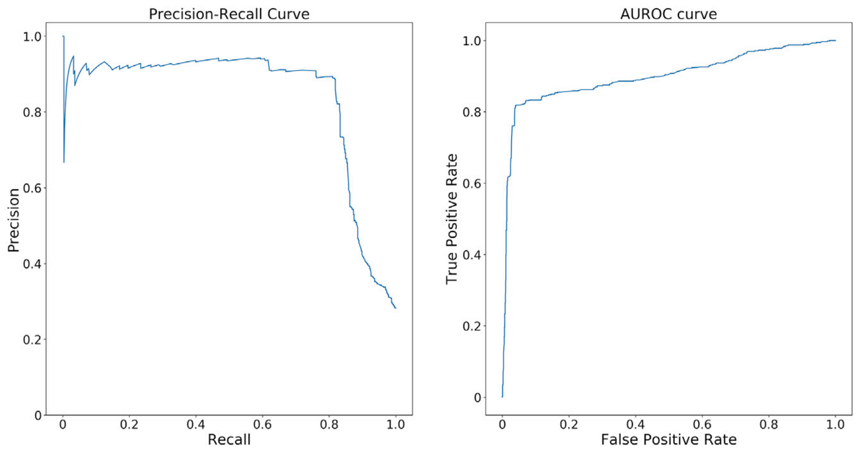


Figure 3. Prediction performance of the simulated data

Problem 3

Pre-trained embeddings are not informative enough and thus does not help improve much on inferring meaningful topics (step-by-step method details– topic inference).

Potential solution

Create knowledge graph with more enriched information: 1. using nodes with more details. For example, it could help to use medication of specific dosage instead of just active ingredients; 2. Introducing more entities to the graph. For instance, adding international classification of disease (ICD) code (https://www.who.int/standards/classifications/classification-of-diseases) as node to condition knowledge graph could probably further increase the knowledge learned by embeddings.

Problem 4

The datasets are too large for topic modeling in step 29 (step-by-step method details–topic inference).

Potential solution

Decrease batch size or embedding size could help fit in device memory limit.

Problem 5

When predicting different medical outcomes as in step 31, the labels have more imbalanced label distribution, causing poor performance (step-by-step method details—pain prediction).

Potential solution

There are several technologies that could mitigate the issue of imbalanced data. For instance, the user could use focal loss[5] which applies a modulating term to the cross entropy in order to improve on hard misclassified examples. Another way is to increase the regularization via weight decay.[6]

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Yue Li (yueli@cs.mcgill.ca).

Materials availability

This study did not generate any reagents.

Data and code availability

- The UK Biobank data access has been approved by McGill IRB under the project title "A replication study of pain interactions with comorbidities". The approval number is A03-M20-21B.
- All code associated with this paper can be freely accessed and downloaded via https://github.com/li-lab-mcgill/getm and are also provided in an open access desposition at Zenodo: https://zenodo.org/badge/latestdoi/318269616.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

Y.L. is supported by the Natural Sciences and Engineering Research Council (NSERC) Discovery Grant (RGPIN-2019-0621), Fonds de recherche Nature et technologies (FRQNT) New Career (NC-268592), and Canada First Research Excellence Fund Healthy Brains for Healthy Life (HBHL) initiative New Investigator start-up award (G249591). A.V.G. and Y.W. were partly funded by the Canadian Excellence Research Chairs (CERC09). The current study was conducted under UK Biobank application 20802.

AUTHOR CONTRIBUTIONS

These authors jointly supervised this work: A.V.G. and Y.L. Y.L. and A.V.G. conceived the study. Y.L. and Y.W. developed the methodology. Y.W. created the computational software and ran the analyses. Y.W., Y.L., and A.V.G. analyzed the results and wrote the initial manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Wang, Y., Benavides, R., Diatchenko, L., Grant, A.V., and Li, Y. (2022). A graph-embedded topic model enables characterization of diverse pain phenotypes among UK Biobank individuals. iScience *25*, 104390.
2. Dieng, A.B., Ruiz, F.J.R., and Blei, D.M. (2020). Topic modeling in embedding spaces. Trans. Assoc. Comput. Linguist. *8*, 439–453.
3. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (2020). Array programming with NumPy. Nature *585*, 357–362.
4. Grover A., Leskovec J. Node2vec: scalable feature learning for networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD); Springer. 2016. p. 855–864.
5. Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). In Focal loss for dense object detection (IEEE International Conference on Computer Vision (ICCV)), pp. 2999–3007.
6. Xie, Z., Issei, S., and Sugiyama, M. (2020). Understanding and scheduling weight decay. Preprint at arXiv. https://doi.org/10.48550/arXiv.2011.11152.