TigerLily: Finding drug interactions in silico with the *Graph*

Benedek Rozemberczki
United Kingdom

1 INTRODUCTION

Adverse drug-drug interactions cause every year thousands of fatalities in the United States [8] and lead to the hospitalization of many more; one can only imagine these numbers for the whole world. This extreme number of unnecessary death and burden on the health care system could be avoided if better drug-drug interaction indications would be available to drug discovery researchers, clinicians, and patients. Moreover, adverse drug interaction events affect the elderly population and those excluded from private healthcare systems and self-medicating disproportionately [4, 17]. In a time when the population of many countries is aging [5] and self-medication is becoming increasingly widespread [3] there is an urgent need for reliable drug-drug interaction indications that are available to all.

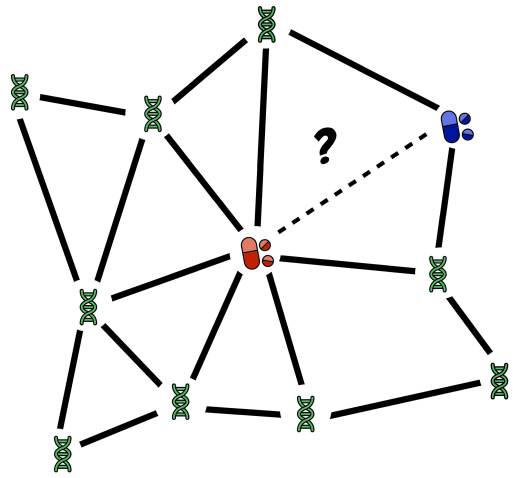


Figure 1: TigerLily solves the in silico drug interaction prediction problem [31] using a heterogeneous biological graph [24]. Given information about systematic interactions of drugs (red and blue pills) and gene targets (green double helix segments) our goal is to find unexpected potentially dangerous links between drugs.

However, collecting data about potential drug-drug interaction indications is an extremely challenging task due to multiple reasons: (a) the number of potential interactions increases quadratically with the number of drugs; (b) experimental validation is time-consuming and costly concerning equipment; (c) experimental results can be noisy and inconsistent across laboratories and patients; (d) the observed symptoms might be unrelated to the parallel administration of drugs. The reasons listed above make computational approaches to the drug-drug interaction predicting problem, particularly appealing in practical settings [31, 34].

The proposed framework TigerLily offers a computational solution to the drug-drug interaction prediction problem based on systems biology and graphs [21]. A cartoon-like graphical summary of the main idea behind TigerLily is depicted in Figure 1. Based on a biological network where edges are interactions, drugs and genes are nodes our goal is to predict interactions. We assume that based on the *biological network* based neighbourhood context of drugs one can predict the drug-drug interactions. TigerLily learns to embed the drug nodes in a feature space using personalized PageRank scores computed with TigerGraph. Using the drug embedding features drug pair features are defined which serve as input to supervised drug interaction classifiers which learn from known interactions and can predict novel adverse events.

1.1 Statement of Significance

Releasing TigerLily is significant for multiple reasons. We aim to briefly summarize these reasons and how the release of TigerLily is related to the main goals of the *Graph For All Million Dollar Challenge*.

1.1.1 Impactfulness. Drug-drug interactions affect everyone who has to take multiple medications in parallel, a tool that can indications of them can benefit drug discovery researchers, clinicians, patients, and drug safety regulators. A widely applicable and accessible tool that can indicate adverse interactions can reduce the number of fatalities and hospitalization rates.

1.1.2 Innovativeness. TigerLily uses a novel node embedding technique that is based on pruned approximate Personalized PageRank scores. This node embedding is innovative, because: (i) the technique was not described explicitly in the literature previously; (ii) exploits the existing TigerGraph ecosystem (iii) does not require the computation and network transfer of the whole dense personalized PageRank matrix.

1.1.3 Ambitiousness. The experimental evaluation of TigerLily uses a heterogeneous biological graph with two node types: drugs and genes. This graph has nearly a million edges, more than 1000 drugs, and 20,000 genes – which nearly covers the whole human genome. The personalized PageRank computation in TigerGraph Cloud exploits the sparsity and graph heterogeneity offered by TigerGraph to reduce the memory and computation requirements of TigerLily.

1.1.4 Applicability. Throughout the development of TigerLily we followed a pragmatic software engineering approach: the code base is covered by unit and integration tests, documented, continuous integration runs on the repository, and the library is pip installable and we provided tutorials for the users. The solution can be repurposed to solve other tasks such as the drug synergy prediction problem with little effort.

1.2 Summary of Contributions

Introducing TigerLily makes several significant contributions to the field of graph machine learning-based drug-drug interaction prediction. The main contributions can be summarized as:

- We release TigerLily an open-source TigerGraph-based system designed to predict drug-drug interactions in silico using heterogeneous biological graphs.
- We evaluate the performance of TigerLily using real-world biological and chemical data that we integrated from DrugBank [43, 44] and BioSNAP [24].
- We discuss those who would benefit from TigerLily, potential limitations to the approach, obstacles for the adoption, and potential future directions for development.

The remainder of this project report is structured as follows. In Section 2 we overview the relevant literature about drug-drug interactions, proximity preserving node embeddings, and biological knowledge graphs. Formal definitions of graph mining and the underlying mathematical model behind TigerLily are discussed in Section 3. We focus on the practical design of the framework in Section 4 with code snippets with a real-world running example. The system is evaluated in Section 5 with real-world data. The limitations of Tigerlily are discussed in Section 6 and the report concludes in Section 7 with potential future directions for research and development. The library is available under https://github.com/benedekrozemberczki/tigerlily.

2 RELATED WORK

Our high-level overview of the related work primarily aims to position TigerLily in the existing literature about drug-drug interaction prediction, heterogeneous biological graphs, and node embedding algorithms.

2.1 Drug-Drug Interaction Prediction

The drug-drug interaction prediction problem is part of the wider drug pair scoring task [31, 34]. In this one has to assign labels to a pair of drugs that describes the behaviour of the drugs in a biological context. This context can be drug-drug interaction [36], polypharmacology side effects [47] or the synergy of the drugs when administered together [33]. According to [31] machine learning-based solutions to this challenge can be categorized into three main groups: (a) molecular features based models [36] (b) network biology-based embedding models [47] (c) hierarchical models which use a mixture of molecular features and systems biology [33]. Our work is closest to the network biology-based solutions as it exploits a high-level biological entity graph to solve the problem by creating upstream drug embeddings and a downstream classifier.

2.2 Heterogeneous Biological Graphs

A heterogeneous biological graph consists of biological entities (node types) and heterogeneous interactions between these entities (edge types). Publicly available graphs [7] are differentiated by each other based on the types of nodes and edges present in the graph, various node type hierarchies [7], the size of the biological graph [9] and the use case that was driving the creation of the graph [11]. These heterogeneous biological graphs can be used by drug discovery researchers in the pharmaceutical industry [7] as an input for off- and on target drug repurposing [10], gene target identification [12], and compound interaction prediction [33] systems.

2.3 Node Embeddings

Node embeddings are unsupervised machine learning models which map the nodes of a graph into an Euclidean space [29] where various notions of graph-based proximity between nodes (e.g. neighbourhood overlap [13], personalized PageRank [26], adjacency [35]) are preserved. By doing this for each node a feature vector is assigned which can be used to solve various downstream machine learning task such as node classification [13], link prediction [13, 35] or node clustering [35]. The drug-drug interaction problem that is our interest can be formulated as a link prediction task on a heterogeneous biological graph, because of this we are going to take a customized node embedding-based approach.

3 PRELIMINARIES

In this section, we focus on the mathematical model that powers the predictions made by TigerLily and the biological graph dataset that we created to test the performance of the machine learning system. [1]

3.1 The Embedding Model

TigerLily relies on a custom node embedding model which exploits the advantageous functionalities of TigerGraph. Our goal is to give a concise and prompt description of this upstream machine learning model and how the features of this model are used by the downstream drug interaction predictor.

3.1.1 Graph Theory Basics. Drugs and genes are noted by the sets $\mathcal{D}$ and $\mathcal{P}$. The union of these two sets $\mathcal{V} = \mathcal{D} \cup \mathcal{P}$ defines the node set (biological entities), $\mathcal{E}$ is the set of edges between the entities and $G = (\mathcal{V}, \mathcal{E})$ is the heterogeneous biological graph. We postulate that the edge set does not contain any drug-drug edges hence $(d, d') \notin \mathcal{E}, \forall d, d' \in \mathcal{D}$. We use $n = |\mathcal{V}|$ and $m = |\mathcal{D}|$ to denote the cardinality of the biological entity and drug sets. Finally, $\tilde{\mathbf{A}}$ is the $n \times n$ normalized adjacency matrix of the graph G.

3.1.2 Upstream Drug Embeddings. A drug indicator matrix $\mathbf{S} \in \mathbb{R}^{n \times m}_{(0,1)}$ is a binary matrix where each row corresponds to a biological entity and columns correspond to drugs. Non zero entries of this matrix correspond to indicators of the drugs in the graph, meaning that $\forall v \in \mathcal{V}, d \in \mathcal{D}$ it only holds that $\mathbf{S}_{v,d} = 1$ if $v = d$. The approximate personalized PageRank scores [15, 22, 26] of drugs are defined by Equation (1).

$$\mathbf{X} = \sum_{r=0}^{t-1} \alpha \cdot (1-\alpha)^r \widetilde{\mathbf{A}}^r \mathbf{S} + (1-\alpha)^t \widetilde{\mathbf{A}}^t \mathbf{S} \tag{1}$$

Here r is a running index, t is the number of approximation iterations, $0 \leq \alpha \leq 1$ is the return probability, and $\mathbf{X} \in \mathbb{R}^{n \times m}_{0+}$ is the matrix of approximate personalized PageRank scores for the drug

[1] The embedding model described in Subsection 3.1 is involved technically, it gives reasoning to the reader why the proposed solution interfaces well with the architecture of TigerGraph. The reader can jump ahead and read the rest of the report if such details seem less relevant to the scope of the challenge.

nodes in the graph. An entry in this matrix is large when a source biological entity (row) is close to a drug (column) based on the approximate personalized PageRank score.

Let us define $\widetilde{\mathbf{X}} \in \mathbb{R}_{0+}^{m \times n}$ the pruned approximate personalized PageRank matrix between drugs and biological entities by Equation (2). The pruning operation takes a matrix as an input and returns a sparse matrix wherein each row the top k largest values are kept, everything else is zeroed out. An entry in this matrix is large when a source drug (row) is close to a biological entity (column) based on the pruned approximate personalized PageRank score. It must be emphasized that this matrix only takes negligible $O(mk)$ space compared to the $O(n^2)$ space required by personalized PageRank for all biological entities.

$$\widetilde{\mathbf{X}} = \mathbf{PRUNE}(\mathbf{X}^\top, k) \tag{2}$$

This sparse matrix can be easily computed by using the personalized PageRank query of TigerGraph on the biological graph. Finally, we can learn the drug embeddings from this matrix by solving the non-negative matrix factorization problem [19, 38] defined by Equation (3).

$$\min \left\| \widetilde{\mathbf{X}} - \mathbf{HW} \right\|_F \text{ subject to } \mathbf{H} \in \mathbb{R}_{0+}^{m \times d}, \mathbf{W} \in \mathbb{R}_{0+}^{d \times n} \tag{3}$$

In Equation (3) $d << m$ is the number of embedding dimensions, the non negative matrix $\mathbf{H}$ is the drug embedding and $\mathbf{W}$ is the biological entity embedding. Each row of $\mathbf{H}$ is a drug and columns can be interpreted as hidden features of the embedding; we are going to use the features of this matrix as drug features to define the drug pair features and to train the interaction predictor.

3.1.3 Downstream Drug Pair Classifier. Given the drug set $\mathcal{D}$ and drug embedding matrix $\mathbf{H}$ the feature vector describing the drug pair $d, d' \in \mathcal{D}$ is defined as $\mathbf{H}_{(d,d')} = g(\mathbf{H}_d, \mathbf{H}_{d'})$ where the function $g(\cdot)$ is a so called operator function [13]; an example operator function can be the concatenation of the two drug vectors or the Hadamard product of the vectors. A downstream classifier takes such drug pair feature vector for a pair of drugs $d, d' \in \mathcal{D}$ as an input and outputs the probability that there is an interaction between them.

3.2 The Integrated Drug Interaction Dataset

Our experiments use a manually integrated biological network from BioSNAP [24] and a DrugBank DDI [36] based drug pair dataset.

3.2.1 The Biological Graph. We took all of the available non-cell-specific gene-gene and drug-gene interaction networks [39, 47] from BioSNAP and integrated them into a single heterogeneous biological graph. Genes have been mapped to the Entrez identifier system [23] and drugs have been mapped to DrugBank identifiers [44] in the data cleaning process. The result is an undirected heterogeneous graph with two node types; 1,106 drug nodes and 20,754 gene nodes, 38,393 drug-gene target interactions, and 778,290 gene-gene regulatory interactions.

3.2.2 The Drug-Drug Interactions. The target drug-drug interactions were taken from the DrugBank DDI dataset [36]; we filtered for those drug pairs where both drugs are present in the biological network we created. After this filtration we have 187,850 labeled drug pairs; 106,362 of these pairs have an adverse interaction (positive label) and 81,488 of these have no known interaction (negative label). This dataset was made available via the GitHub repository of TigerLily and the library also has a built-in data loader class to access this integrated dataset.

4 THE TIGERLILY FRAMEWORK DESIGN

Our discussion about the Tigerlily design focuses on two things: (a) a real-world drug-drug interaction prediction use case that showcases the API step-by-step and interfacing with other machine learning libraries; (b) the software engineering principles that ensure that the TigerGraph based solution is maintainable and robust.

4.1 A Real World Use Case

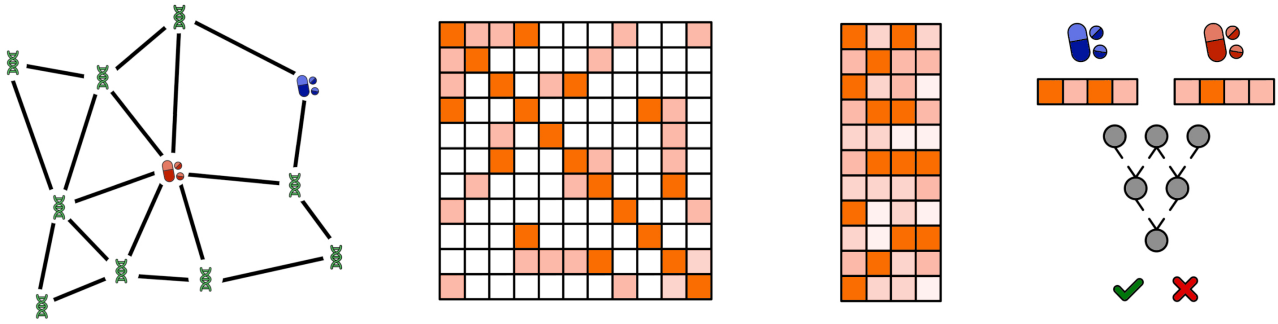
In this section, we solve a real-world drug-drug interaction problem that uses data from DrugBank and BioSNAP. The details of these datasets and the performance TigerLily on this problem are discussed in Section 5 in great detail. A high-level step-by-step overview of the TigerLily based solution to this problem is described in Figure 2 and our demonstration will follow this workflow.

4.1.1 Heterogeneous Graph Definition and Upload. Our goal is to create a heterogeneous biological graph and store it as a drug-gene interaction network with TigerGraph. The way we achieve this is described by the code snippet in Listings 1.

```
1  from tigerlily.dataset import ExampleDataset
2  from tigerlily.embedding import EmbeddingMachine
3  from tigerlily.operator import hadamard_operator
4  from tigerlily.pagerank import PersonalizedPageRankMachine
5
6  dataset = ExampleDataset()
7
8  edges = dataset.read_edges()
9  target = dataset.read_target()
10
11 machine = PersonalizedPageRankMachine(host="host",
12                                        graphname="graph",
13                                        username="user",
14                                        secret="secret",
15                                        password="password")
16
17 machine.connect()
18 machine.install_query()
19
20 machine.upload_graph(new_graph=True, edges=edges)
```

Listings 1: Loading the example drug-drug interaction dataset, creating a TigerLily `PersonalizedPageRankMachine` instance and populating the Graph with a heterogeneous biological graph.

We start by importing classes and functions from TigerLily that we will use later (lines 1-4). We create a `ExampleDataset` instance, load the edges and the target drug pairs with the respective class methods. Both of these parts of the dataset are returned as `pandas dataframe` objects by the class methods (lines 6-9). We must note, that the `edges` dataframe must have columns named `node_1`, `node_2`, `drug_1` and `drug_2` columns which respectively



(a) Heterogeneous Graph Definition and Upload

(b) Personalized PageRank Computation and Pruning

(c) Node Embedding

(d) Inference

Figure 2: TigerLily provides a workflow for biological graph-based drug interaction prediction that consists of multiple main steps. (a) Using TigerGraph a heterogeneous systems biology graph is defined with drug and gene nodes. This graph does not contain the drug-drug interactions. (b) The Personalized PageRank vector for each drug node is computed using TigerGraph and this vector is pruned to contain the top-k most similar nodes based on proximity. (c) Based on the Personalized PageRank matrix we learn drug embeddings which serve as *Systems Biology* based features of drugs. (d) Using the drug features we define drug pair features and train a classifier to predict the interactions of pair combinations.

contain the node identifiers and the node types. The edge dataframe must not contain edges where both of the nodes have a drug node type as our goal is to predict the existence of drug-drug interactions. In a similar fashion the `target` dataframe must have the columns `drug_1`, `drug_2` and `label` which contain the drug identifiers and the indicator for the existence or non-existence of an interaction.

We create a `PersonalizedPageRankMachine` instance (lines 11-14) that is parametrized by the hostname and graph names with the appropriate credentials. This step requires that there is a running and existing TigerGraph Cloud instance with a graph that has the appropriate Graph schema - in our case, it means drug and gene nodes with interaction edges. The instance is connected, the personalized PageRank query is installed and the edges of the graph are uploaded (lines 16-19). By setting the `new_graph` flag to be true we make sure that the graph is empty before the upload starts. After the graph is populated it is time to compute some approximate personalized PageRank scores.

4.1.2 Personalized PageRank Computation and Pruning. Our focus is on the drug nodes and we want to query those to get an understanding of which are those biological entities that are in close proximity to the drugs. We do this with the piece of code described in Listings 2. First, we query the `PersonalizedPageRankMachine` to get a list of the drug nodes (line 1). Based on this list using the `get_personalized_pagerank` class method we query those nodes for each drug that have a large approximate personalized PageRank score. For each drug we return the top-k highest scoring entry (line 3); this ensures that the returned dataset is small even when the number of biological entities is large compared to the number of drugs. The returned `pagerank_scores` data frame has the `node_1`, `node_2` and `score` columns; it describes a sparse matrix where rows are drugs, columns are biological entities (including drugs) and the values are pruned personalized PageRank scores. Let us move on to the learning drug embeddings from this matrix and creating drug pair features from the learned embeddings.

```
1 drugs = machine.connection.getVertices("drug")
2
3 pagerank_scores = machine.get_personalized_pagerank(drugs)
```

Listings 2: Querying the previously created TigerLily `PersonalizedPageRankMachine` instance for the list of drug nodes in the Graph and computing the personalized PageRank of nodes in the proximity of drugs.

4.1.3 Drug Node Embedding. Using the previously computed personalized PageRank matrix we learn node embeddings for each drug with the Python script in Listings 3. We create an `EmbeddingMachine` instance and learn drug node embeddings (lines 1-5); the returned `embedding` is a a `pandas` dataframe where the firs column named `node_id` contains the drug identifiers and the remaining columns contain the embedding dimensions. Using the `target` and the drug pair feature computation `hadamard_operator` the `create_features` class method of the `EmbeddingMachine` instance allows the creation of drug pair features (lines 7-8). Using these drug pair features we are ready to solve the drug-drug interaction task!

```
1 embedding_machine = EmbeddingMachine(seed=42,
2                                      dimensions=32,
3                                      max_iter=100)
4
5 embedding = embedding_machine.fit(pagerank_scores)
6
7 features = embedding_machine.create_features(target,
8                                    hadamard_operator)
```

Listings 3: Creating a `EmbeddingMachine` instance, learning drug embeddings from the personalized PageRank scores and creating drug pair features with the `hadamard_operator` for the drug pairs.

4.1.4 Drug Pair Interaction Prediction and Inference. Our final step to learning a drug pair classifier and producing predictions for the drug pairs is summarized with code in Listings 4. This step is a pretty generic supervised machine learning workflow based on LightGBM [18] and scikit-learn [27].

We start by importing the gradient boosted classifier, the AUROC evaluation metric, and the function for creating train-test splits (lines 1-3). We split the drug pair features and the labels with the `train_test_split` function into training and testing parts (line 5). We create a `LightGBMClassifier` instance, learn the model from the training set drug pairs, score on the test set, compute the AUROC score and print the score by taking the first few digits (lines 7-17). This snippet demonstrated that TigerLily interfaces with existing machine learning libraries smoothly.

```
1  from lightgbm import LGBMClassifier
2  from sklearn.metrics import roc_auc_score
3  from sklearn.model_selection import train_test_split
4
5  X_train, X_test, y_train, y_test = train_test_split(features,
6                                                      target)
7
8  model = LGBMClassifier(learning_rate=0.01,
9                         n_estimators=100)
10
11 model.fit(X_train, y_train["label"])
12
13 y_hat = model.predict_proba(X_test)
14
15 auroc_score_value = roc_auc_score(y_test["label"],
16                                    y_hat[:,1])
17
18 print(f'AUROC score: {auroc_score_value :.4f}')
```

Listings 4: Splitting the target and the TigerLily generated drug pair features to train and evaluate a gradient boosting based drug pair scoring model.

4.2 Maintaining and Supporting TigerLily

As we have seen in the previous section TigerLily was engineered with an end-user-friendly API in mind and this design is supported by continuous integration, documentation, tutorials, package indexing, and unit- and integration tests.

4.2.1 Documentation and Example Notebook. The complete codebase of TigerLily is documented with docstrings and type annotations. Using these and restructured text files we automatically release new documentation that reflects the current state of the TigerLily Github repository. This documentation is available under https://tigerlily.readthedocs.io/en/latest/ and it covers the API reference and includes a tutorial for the new potential users. The same tutorial is available in the form of a Jupyter Notebook in the repository that explains line by line a typical Tigerlily-based graph analytics workflow.

4.2.2 Inclusion in the Python Package Index. The submission 0.1.0 release of Tigerlily is publicly available on the Python Package Index. This means that different versions of the library (including the submission release) can be accessed via the https://tigerlily.readthedocs.io/ webpage and that it can be installed via the command line using the `pip install tigerlily` command. This allows the end-users to install TigerLily in the Python environment that they use efficiently and also the same users can install different versions of the library based on their needs.

4.2.3 Test Suite and Code Coverage Reports. The continuous integration of Tigerlily with Github Actions allows the automated testing of the codebase. The Tigerlily namespaces are covered by unit and integration tests which ensure that software components behave as expected. Each automated test suite run generates a coverage report hosted on Codecov. These reports are publicly available under https://app.codecov.io/gh/benedekrozemberczki/tigerlily and allow inspecting the coverage rate of the TigerLily namespaces.

5 EXPERIMENTAL EVALUATION

The main goal of this section is to demonstrate that TigerLily can solve a real-world problem. Using the biological network and drug-drug interaction data discussed in Section 3 we will test the predictive performance of classifiers that use TigerLily-based drug embedding features.

5.1 Predictive Performance

The predictive efficacy of TigerLily is a primary driver of the potential impact that the solution can have in the real world. Because of this we investigate this and compare the performance under various drug pair feature computation operators with multiple binary classification metrics.

5.1.1 Experimental Design. We compute approximate personalized PageRank scores with the `PersonalizedPageRankMachine` for the drugs and their closest 50 neighbors, from 25 PageRank approximation iterations with a return probability of 0.7. Drug embeddings are learned with the `EmbeddingMachine` in 32 dimensions by doing 100 iterations. From the drug embeddings, drug pair features are computed with the operators listed in Table 1. We use 80% of the pairs to train a gradient boosted tree machine (LightGBM implementation [18]) and compute AUROC, AUPR, F_1 scores on the remaining 20% of pairs. The average performance computed from 10 random seeded experimental runs is in Table 1 with standard errors around the mean performance.

Table 1: The mean predictive performance (standard deviations below) of a TigerLily based gradient boosted machine computed from 10 seeded train-test splits. Rows represent drug-drug feature computation routines described by [13].

Binary Operator	Definition of component $\mathbf{H}^i_{(d,d')}$	AUROC	AUPR	F_1
Absolute	$\lvert\mathbf{H}^i_d - \mathbf{H}^i_{d'}\rvert$	.953 ±.002	.961 ±.002	.874 ±.003
Squared	$(\mathbf{H}^i_d - \mathbf{H}^i_{d'})^2$	.951 ±.001	.963 ±.002	.870 ±.002
Difference	$\mathbf{H}^i_d - \mathbf{H}^i_{d'}$	.948 ±.003	.961 ±.002	.860 ±.004
Hadamard	$\mathbf{H}^i_d \cdot \mathbf{H}^i_{d'}$	.951 ±.003	.955 ±.002	.871 ±.003

5.1.2 Experimental Results. Most importantly the results in Table 1 are a strong signal that TigerLily-based embedding features can be used to predict the drug-drug interactions. We can also observe that there is no clearly superior operator and that there is a negligible difference in performance between the various feature computation operators across the performance metrics. A further error analysis could show which drugs participate in drug pairs that are being consistently misclassified by TigerLily.

5.2 Training Data Ratio

Getting ground truth about known drug-drug interactions is a costly process. Because of this, data-efficient solutions which require a limited amount of labeled drug pairs to solve the drug-drug interaction problem are extremely valuable. We are going to investigate the predictive performance of TigerLily under various training data ratio regimes with a range of downstream classifiers.

5.2.1 Experimental Design. We train LightGBM [18] based gradient boosting and scikit-learn [27] based random forest, logistic regression, and neural networks to predict the interactions using drug pair features computed with the Hadamard operator. The personalized PageRank calculation and embedding hyperparameter settings were taken from Subsection 5.1. We modulate the amount of training data and plot the test set average predictive performance computed from 10 random seeded experimental runs on the subplots of Figure 3 as a function of the training data amount.

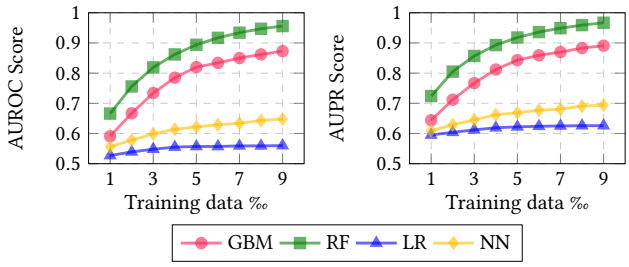


Figure 3: The mean drug interaction prediction performance of TigerLily based classifiers conditioned on the ratio of training data (in permille) calculated from 10 train-test splits.

5.2.2 Experimental Results. The line charts of Figure 3 clearly demonstrate that the non-parametric methods (gradient boosting and random forest) are extremely data-efficient compared to the parametric ones. Given less than 1% of data, these methods achieve similar results to the number in Table 1. This showcases that the two-stage machine learning system of TigerLily is able to solve the drug interaction problem with high predictive efficacy even when the amount of training data is extremely limited.

5.3 Embedding Dimension Sensitivity

TigerLily is a highly modular framework with the upstream drug embeddings and downstream drug pair classifier-based design. However, the number of dimensions used for the drug embeddings is a highly important hyperparameter of the upstream model; in a certain sense, it is a key hyperparameter to tune and optimize.

5.3.1 Experimental Design. Using the personalized PageRank, upstream embedding and downstream model settings from the previous section we train gradient boosted tree, logistic regression, and neural network classifiers while the number of embedding dimensions is modulated in $\{2^2, \ldots, 2^7\}$. We compute average AUROC and AUPR scores from 10 random seeded experimental runs and plotted the average predictive performance as a function of the embedding dimensions in Figure 4.

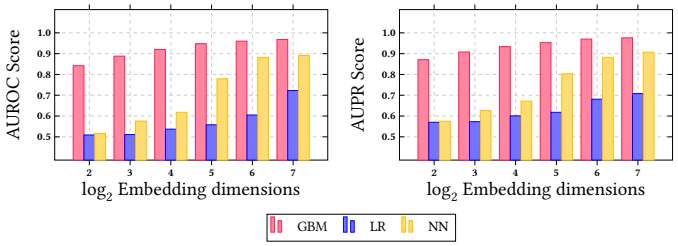


Figure 4: The mean drug interaction prediction performance of TigerLily feature based classifiers conditioned on the number of embedding features calculated from 10 train-test splits.

5.3.2 Experimental Results. Our results in Figure 4 demonstrate that increasing the embedding dimensions beyond 2^5 the default is beneficial for predicting novel drug-drug interactions. We can also observe that the non-parametric gradient boosted trees gain less with the increased number of embedding dimensions compared to the parametric models.

6 TARGET USERS AND LIMITATIONS

6.1 Potential Target Users

When TigerLily was designed we had specific user stories in mind about how it could be deployed. All of these stories showcase the potential of TigerLily to affect the life of millions of patients around the world.

6.1.1 Early Drug Discovery Researchers. Using TigerLily in the early discovery phase could allow drug discovery researchers to flag potentially adverse drug interactions early on. Indications could be followed up by experiments later in the clinical phase of the drug discovery process. This could reduce the attrition rate of drugs being developed as drugs with a lot of adverse interactions would not proceed to the late phases of the development process.

6.1.2 Clinical Practitioners. Using TigerLily could give clinical practitioners early warnings about potential interactions of drugs that are prescribed to patients. Based on these warnings the doctors can make informed decisions about the drug combinations. For example, the patients could be asked to look for specific polypharmacy side effects and monitored more closely when there is a risk of adverse drug interactions.

6.1.3 Pharmaceutical Industry Regulators. During the drug discovery process the potential drug-drug interactions are not a primary target for the researchers. Because of this potential drug safety concerns can be flagged during the drug approval. A TigerLily-like system that can create indications for potential interactions could

serve the pharmaceutical industry regulators who could get early warnings about potential safety issues and rare adverse events.

6.1.4 Self-medicating Patients. A large number of adverse drug interaction events happen when patients decide to use multiple drugs simultaneously on their own [1]. By extending TigerLily with a convenient and friendly user interface patients could make better self-medication decisions after consulting TigerLily. This could be extremely beneficial for those communities that are critically neglected and under-served by the healthcare system.

6.2 Limitations and Obstacles for Deployment

Our system TigerLily is a proof of concept for TigerGraph based in silico drug-drug interaction prediction. This means that the project and the presented approach have some limitations which we would like to highlight shortly.

6.2.1 Systems Biology Data of New Drugs is Lacking. The node embedding functionality of TigerLily implicitly assumes that the drugs of interest are connected to the gene-gene network so drugs can be contextualized by their location in the biological graph. This particular issue is a manifestation of the transductive node representation learning setting [14] of our embedding approach. However, this is not true for newly developed drugs, where the nature of the exact interactions with other genes is not understood beyond a primary gene target. This means that predicting interactions for compounds that are not known for long can be a challenging task and potentially the performance of TigerLily could be affected. This challenge could be overcome by using inductive graph neural networks [6, 14, 20, 22, 32, 41] which would require extrinsic drug and gene features.

6.2.2 Limited Number of Biological Entity Types. The graph which we integrated to demonstrate the predictive performance of our solution has two types of biological entities: drugs and gene targets. Existing heterogeneous biological graphs [11, 16, 42] used in the pharmaceutical industry have a larger variety of node types such as cell lines, protein variants, biological processes, and pathways. It is possible that the inclusion of more node and edge types would enrich the graph and result in better quality drug embeddings and interaction predictions.

6.2.3 Skewed Drug Interaction Data. Drug interaction and synergy databases are known to be skewed [31, 36, 47]. Practically this means that combinations that involve specific commonly used drugs are tested, but combinations of rarely used drugs are not present. Given that the primary goal of in silico drug interaction predictions is to give indications for combinations of rarely used pairs this can be problematic.

6.2.4 Homogeneous Proximity Preserving Node Embedding. Our approach creates homogeneous proximity preserving node embeddings based on the approximate Personalized PageRank scores; this is a considerable limitation given our graph. A number of current approaches are able to incorporate information about the heterogeneity of the graph with respect to node types and edges [2, 25, 40]. Moreover, certain embedding approaches are able to describe the structural roles of nodes [30] which could be important when it comes to the systems biology of compounds and genes.

6.2.5 Poor Interpretability of Predictions. TigerLily is a framework that consists of an upstream and downstream module; by the end of the upstream phase, the graph information is distilled into feature vectors. Downstream models trained on these features are not interpretable and because of this, we cannot answer questions about why the classifier flagged a drug pair to be a potential adverse one. However, given extrinsic drug and gene features and a graph neural network trained on these modern explanation techniques could allow the creation of such post-hoc explanations of the drug-drug interaction predictions [28, 37, 45, 46].

6.2.6 Hesitance of Potential Users. Our proposed drug interaction prediction framework TigerLily serves as an intelligent system that can accelerate the work of clinicians, early discovery researchers, and drug safety experts. However, the black-box nature of this system might be something that cannot be overlooked by the end-users. Moreover, the system might have a potential bias that could affect specific groups of people. One such group could be a set of people who share genetic markers which make them more prone to experience a certain type of adverse events due to drug combinations. Issues like this could make the potential users of TigerLily use the system.

7 FUTURE DIRECTIONS AND CONCLUSIONS

7.1 New Research Directions

The open-source codebase, modularity, and extensible nature of TigerLily open up opportunities for future research and engineering solutions. We will highlight potential avenues for the further development of the framework which we see to be high impact and low effort.

7.1.1 User Interface Development. TigerLily is a Python library that requires that the users are familiar with the language and feel comfortable with reading the documentation. Allowing users to run simple queries on the TigerLily-based predicted scores could open up opportunities for a large number of people. Because of this, the development of a user interface where drug pairs associated with adverse events can be queried seems to be a promising and high-impact future research direction.

7.1.2 Large Biological Graphs. Our demonstration uses a graph that covers most of the human genome but ignores node types such as pathways or cell lines. We postulate that modeling the underlying biology systems biology better with data and encoding biology better can only be achieved via this. Hence, the integration of large publicly available biological graphs in TigerLily seems to be an extremely promising direction for future research.

7.1.3 Drug Synergy Predictions. The focus of our discussion about TigerLily is on adverse drug-drug interactions, but the system is sufficiently flexible to allow for solving other tasks. One interesting task defined on drug pairs is synergy prediction a central question of computational oncology research [33, 34].

7.2 Concluding Remarks and Summary

In this project report we discussed TigerLily a TigerGraph-powered open-source machine learning system for in silico drug interaction

prediction. We discussed existing artificial intelligence-based approaches to solving this interesting problem and how biological graphs can be used to tackle this challenge. We gave an overview of how we integrated data from public resources to solve this problem and formalized a mathematical model that can exploit biological graphs to solve the task; TigerLily is a conceptualization of this mathematical model. We demonstrated the main features of TigerLily with Python code examples and highlighted the software engineering principles that make the tool robust, reliable, and accessible. By doing extensive experiments we had shown that TigerLily can help to predict drug interactions in the real world. We reviewed potential users of the system, and its limitations and emphasized the most important future research directions.

REFERENCES

[1] Nathalie Asseray, Françoise Ballereau, Béatrice Trombert-Paviot, Jacques Bouget, Nadine Foucher, Bertrand Renaud, Lucien Roulet, Gerald Kierzek, Aurore Armand-Perroux, Gilles Potel, et al. 2013. Frequency and severity of adverse drug reactions due to self-medication: a cross-sectional multicentre survey in emergency departments. *Drug safety* 36, 12 (2013), 1159–1168.

[2] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 5188–5197.

[3] Darshana Bennadi. 2013. Self-medication: A current challenge. *Journal of basic and clinical pharmacy* 5, 1 (2013), 19.

[4] Aurélia Berreni, François Montastruc, Emmanuelle Bondon-Guitton, Vanessa Rousseau, Delphine Abadie, Geneviève Durrieu, Leila Chebane, Jean-Paul Giroud, Haleh Bagheri, and Jean-Louis Montastruc. 2015. Adverse drug reactions to self-medication: a study in a pharmacovigilance database. *Fundamental & clinical pharmacology* 29, 5 (2015), 517–520.

[5] Ingeborg K Björkman, Johan Fastbom, Ingrid K Schmidt, Cecilia B Bernsten, and Pharmaceutical Care of the Elderly in Europe Research (PEER) Group. 2002. Drug—drug interactions in the elderly. *Annals of Pharmacotherapy* 36, 11 (2002), 1675–1681.

[6] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate Pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2464–2473.

[7] Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William Hamilton. 2021. A review of biomedical datasets relating to drug discovery: A knowledge graph perspective. *arXiv preprint arXiv:2102.10062* (2021).

[8] Valeri Craigle. 2007. MedWatch: The FDA safety information and adverse event reporting program. *Journal of the Medical Library Association* 95, 2 (2007), 224.

[9] Gavin Edwards, Sebastian Nilsson, Benedek Rozemberczki, and Eliseo Papa. 2021. Explainable Biomedical Recommendations via Reinforcement Learning Reasoning on Knowledge Graphs. *arXiv preprint arXiv:2111.10625* (2021).

[10] Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. 2021. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics* 22, 6 (2021), bbab159.

[11] David Geleta, Andriy Nikolov, Gavin Edwards, Anna Gogleva, Richard Jackson, Erik Jansson, Andrej Lamov, Sebastian Nilsson, Marina Pettersson, Vladimir Poroshin, et al. 2021. Biological Insights Knowledge Graph: an integrated knowledge graph to support drug development. *bioRxiv* (2021).

[12] Anna Gogleva, Dimitris Polychronopoulos, Matthias Pfeifer, Vladimir Poroshin, Michaël Ughetto, Matthew J Martin, Hannah Thorpe, Aurelie Bornot, Paul D Smith, Ben Sidders, et al. 2022. Knowledge graph-based recommendation framework identifies drivers of resistance in EGFR mutant non-small cell lung cancer. *Nature Communications* 13, 1 (2022), 1–14.

[13] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[16] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. 2017. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife* 6 (2017), e26726.

[17] Lisa E Hines and John E Murphy. 2011. Potentially harmful drug–drug interactions in the elderly: a review. *The American journal of geriatric pharmacotherapy* 9, 6 (2011), 364–377.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[19] Hyunsoo Kim and Haesun Park. 2008. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications* 30, 2 (2008), 713–730.

[20] Thomas N. Kipf and Max Welling. [n.d.]. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[21] Hiroaki Kitano. 2002. Systems biology: a brief overview. *science* 295, 5560 (2002), 1662–1664.

[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. [n.d.]. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[23] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. 2005. Entrez Gene: gene-centered information at NCBI. *Nucleic acids research* 33, suppl_1 (2005), D54–D58.

[24] Zitnik Marinka, Sosic Rok, Maheshwari Sagar, and Leskovec Jure. 2018. BioSNAP Datasets: Stanford Biomedical Network Dataset Collection. http://snap.stanford.edu/biodata.

[25] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *International Conference on Machine Learning*. PMLR.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[28] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability Methods for Graph Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10772–10781.

[29] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying Deepwalk, LINE, PTE, and Node2Vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 459–467.

[30] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. 2020. On Proximity and Structural Role-Based Embeddings in Networks: Misconceptions, Techniques, and Applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 5 (2020), 1–37.

[31] Benedek Rozemberczki, Stephen Bonner, Andriy Nikolov, Michael Ughetto, Sebastian Nilsson, and Eliseo Papa. 2021. A Unified View of Relational Deep Learning for Polypharmacy Side Effect, Combination Synergy, and Drug-Drug Interaction Prediction. *arXiv preprint arXiv:2111.02916* (2021).

[32] Benedek Rozemberczki, Peter Englert, Amol Kapoor, Martin Blais, and Bryan Perozzi. 2021. Pathfinder Discovery Networks for Neural Message Passing. In *Proceedings of The Web Conference 2021*. ACM.

[33] Benedek Rozemberczki, Anna Gogleva, Sebastian Nilsson, Gavin Edwards, Andriy Nikolov, and Eliseo Papa. 2021. MOOMIN: Deep Molecular Omics Network for Anti-Cancer Drug Combination Therapy. *arXiv preprint arXiv:2110.15087* (2021).

[34] Benedek Rozemberczki, Charles Tapley Hoyt, Anna Gogleva, Piotr Grabowski, Klas Karis, Andrej Lamov, Andriy Nikolov, Sebastian Nilsson, Michael Ughetto, Yu Wang, et al. 2022. ChemicalX: A Deep Learning Library for Drug Pair Scoring. *arXiv preprint arXiv:2202.05240* (2022).

[35] Benedek Rozemberczki and Rik Sarkar. 2018. Fast Sequence-Based Embedding with Diffusion Graphs. In *International Workshop on Complex Networks*. Springer, 99–107.

[36] Jae Yong Ryu, Hyun Uk Kim, and Sang Yup Lee. 2018. Deep learning improves prediction of drug–drug and drug–food interactions. *Proceedings of the National Academy of Sciences* 115, 18 (2018), E4304–E4311.

[37] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2021. Interpreting Graph Neural Networks for {NLP} With Differentiable Edge Masking. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WznmQa42ZAx

[38] Suvrit Sra and Inderjit Dhillon. 2005. Generalized nonnegative matrix approximations with Bregman divergences. *Advances in neural information processing systems* 18 (2005).

[39] Damian Szklarczyk, Alberto Santos, Christian Von Mering, Lars Juhl Jensen, Peer Bork, and Michael Kuhn. 2016. STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data. *Nucleic acids research* 44, D1 (2016), D380–D384.

[40] T Trouillon, CR Dance, E Gaussier, J Welbl, S Riedel, and G Bouchard. 2017. Knowledge Graph Completion via Complex Tensor Factorization. *Journal of Machine Learning Research* 18, 130 (2017), 1–38.

[41] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. [n.d.]. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[42] Brian Walsh, Sameh K Mohamed, and Vít Nováček. 2020. Biokg: A knowledge graph for relational learning on biological data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3173–3180.

[43] David S Wishart, Craig Knox, An Chi Guo, Dean Cheng, Savita Shrivastava, Dan Tzur, Bijaya Gautam, and Murtaza Hassanali. 2008. DrugBank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research* 36, suppl_1 (2008), D901–D906.

[44] David S Wishart, Craig Knox, An Chi Guo, Savita Shrivastava, Murtaza Hassanali, Paul Stothard, Zhan Chang, and Jennifer Woolsey. 2006. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic acids research* 34, suppl_1 (2006), D668–D672.

[45] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. *Advances in neural information processing systems* 32 (2019), 9240.

[46] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On Explainability of Graph Neural Networks via Subgraph Explorations. (18–24 Jul 2021).

[47] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 34, 13 (2018), i457–i466.