Supporting Drug Prescription via Predictive and Personalized Query System

Samamon Khemmarat and Lixin Gao
Department of Electrical and Computer Engineering
University of Massachusetts Amherst, Massachusetts, USA
Email: {khemmarat,lgao}@ecs.umass.edu

***Abstract*—Drug prescription requires consideration of several factors, such as drug interactions and side effects. The process is further complicated by the fact that the presence of some drug properties, such as side effects, depends on patient characteristics, such as age and gender. Our goal is to provide a tool to assist medical practitioners in prescribing drugs. We develop an approach to query for drugs that satisfy a set of conditions based on drug properties. Furthermore, the approach tailors the answers to a given patient profile. We utilize drug information from multiple sources. However, data from these sources are usually noisy and incomplete as they are either manually curated or automatically extracted from text resources. To cope with incomplete and noisy data, our approach considers both the answers that exactly match and those that closely match the query. We represent drug information as a heterogeneous graph and model answering a query as a subgraph matching problem. To rank answers, our approach leverages the structure and the heterogeneity of the drug graph to quantify the likelihood of missing edges and integrates the likelihood into the scores of the answers. Our evaluation shows that for quantifying the edge likelihood, our graph-based approach can improve the AUROC (Area under Receiver Operating Characteristic) [1] by up to 40%, comparing to a baseline approach. We demonstrate the benefits of our system through several query examples.**

Keywords*—*query system, drug prescription, personalization

I. Introduction

For medical practitioners, prescribing drugs requires careful considerations of several factors, such as interactions among the prescribed drugs, interactions with the patient's current medication, and contraindications. In some cases, according to patients' conditions and lifestyles, there are particular side effects that should be avoided as they could cause serious health conditions or injuries. The process is further complicated by the fact that the presence of some drug properties, such as side effects, depends on characteristics of the patients, such as age, gender, and genetic profiles. Having to consider all these complicated factors can be a huge burden to medical practitioners.

In this work, our goal is to provide a tool to assist practitioners in the process of drug prescription. To achieve this goal, we develop an approach that allows a user to query for drugs that satisfy a set of conditions based on drug properties, such as drug indications, side effects, and drug interactions. For example, for a patient whose occupation is a driver, a doctor may want to issue a query: *Find a drug for fever and allergy that does not cause drowsiness.* Suppose a patient is currently taking some medicines, Enoxaparin and Aspirin, a query can be: *Find a drug for diabetes and a drug for epilepsy that do not interact with Enoxaparin and Aspirin and do not interact with each other*. Furthermore, the approach allows users to specify patient profiles and tailors the answers for the given profile. For example, to find a schizophrenia drug for an elder female patient who has a heart disease, a query can be: *Find a drug for schizophrenia without the side effect heart failure for a female patient, age 60.*

In order to answer these queries, it is important that we have comprehensive drug information. There are currently several drug information sources that are open to public, such as DrugBank [2], SIDER2 [3], and KEGG Drug [4]. As these data sources offer different facets of drug information and have different coverage of drugs, we combine the data from these data sources into a unified knowledge base. However, it is non-trivial to use these data to answer drug queries effectively. Many of the databases are manually curated and therefore have limited coverage. Some databases contain information automatically extracted from text resources, such as drug labels and published articles, which are prone to errors. Our challenge is to be able to provide answers to the queries based on data aggregated from these noisy and incomplete data sources.

Considering the incompleteness and the noisiness of data, traditional query systems that provide only the answers that exactly match the queries have several disadvantages. First, these systems can miss some answers that in fact can satisfy a query but do not exactly match the query due to the imperfection of the data. Second, by disregarding the possibility of missing data, the answers returned can be misleading. For example, if a query asks for a drug that does not interact with a particular drug, some of the drugs that interact with the given drug may also be given as answers because their interaction data is incomplete.

To cope with incomplete and noisy data, our approach considers not only the answers that exactly match the query but also the answers that closely match the query. We model the problem of answering query as a subgraph matching problem, in which drug information is represented as a heterogeneous graph and a query is represented as a query graph. To rank answers, we propose a score function for evaluating the quality of answers based on how well they match with the query graph. Our score function considers the likelihood that there would be an edge between two nodes in the drug graph; thus, both exact and inexact matches are included in our answer space. We utilize the structure of the drug graph to quantify the edge likelihood. As it has been shown that the structure of networks that represent relationships among drugs and drug properties can effectively help to discover novel drug properties [5]–[11], we quantify edge likelihood based on the number of paths between two nodes. However, in contrast to previous works in which the networks contain limited types of nodes, our drug graph contains various node types, and our approach takes into account the types of nodes along the paths in order to leverage the semantics in the heterogeneous drug graph.

The key contributions of this paper are as follows.

- We propose an approach for answering drug-centric queries, which finds a drug or a set of drugs that satisfy a set of conditions based on several types of drug properties. The approach personalizes the answers based on a given patient profile.
- To cope with incomplete data, our approach provides both exact and close-match answers. We propose a score function for ranking the answers. The score function leverages the network structure and the heterogeneity of the drug information graph to estimate the likelihood of missing edges and integrates the likelihood into the score of an answer.
- We evaluate our network-based approach of quantifying the missing edge likelihood. The result shows that our approach outperforms an existing approach that does not utilize network structure, achieving up to 40% increase of AUROC (Area under Receiver Operating Characteristic) [1].
- We develop a prototype of our approach, integrating data from DrugBank [2], SIDER [3], KEGG Drug [4], and FDA (U.S. Food and Drug Administration) drug adverse event reports. We demonstrate the benefits provided by our system through several example queries.

The rest of this paper is organized as follows. We discuss related work in Section II. Section III provides our problem definition. Our methodology is presented in Section IV. In Section V, we describe how our approach personalizes answers for specific patient profiles. Section VI provides information about our data sources for the prototype system. Section VII presents the evaluation of our approach. We summarize our work in Section VIII.

II. Related Work

There are currently multiple drug information databases available for public access, such as DrugBank [2], KEGG DRUG [4], and PharmGKB [12]. Our work leverages these existing databases in order to provide a decision support tool for medical practitioners and drug consumers. There are a few studies that aim to answer medical questions, which include drug-related questions, and provide decision support on drug prescription [13]–[15]. These works model drug information as an RDF (Resource Description Framework) knowledge base and focus on the problem of how to convert raw data into RDF triplets and how to translate questions in natural languages to SPARQL, a query language for RDF. In contrast, our focus is to provide high quality answers to users despite the incompleteness and noisiness of available data.

There are recent works on predicting novel drug properties (including drug targets, indications, and adverse effects), which are related to our problem of quantifying the likelihood of missing associations between drugs and drug properties. Methods that predict drug properties based on chemical structures [16]–[19] have been proposed. Some approaches analyze the correlation between drug targets, their corresponding biological pathways, adverse effects, and indications to perform prediction [20], [21]. These existing works suggest that various types of data are potentially useful in predicting drug properties. Network-based approaches have been proposed to discover novel drug-target interactions [5]–[7], drug-drug interactions [8], [9], [11], and drug adverse reactions [10]. In these approaches, networks containing drugs and drug property entities are created, and the network features, such as common neighbors or the number of paths, are used to predict drug properties. These existing studies have demonstrated that network structures can be used to effectively predict drug properties. However, in these approaches, the networks usually contain limited types of drug property entities. Inspired by these early works, in our work, the likelihood of missing drug properties is quantified based on the structure of a heterogeneous network, containing various types of drug property entities, including targets, pathways, side effects, indications, and drug interactions.

III. Problem Description

We represent drug information aggregated from multiple data sources as a heterogeneous graph, i.e., a graph that has nodes of multiple types. Using the drug graph as a basis, we model the problem of answering queries as a subgraph matching problem. In this section, we describe the schema of the drug graph, the query graph, and our problem.

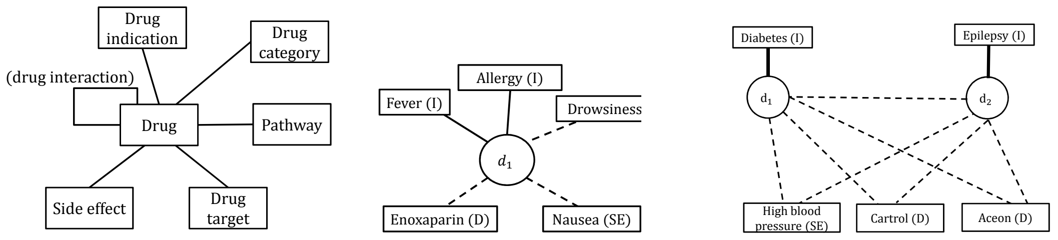
A. Drug Graph Schema

Drug information is represented by a graph $G(V_G, E_G, type_G, key_G)$, where V_G is a set of nodes and E_G is a set of edges. $type_G$ is a function that maps a node to a node type, which is either a *drug* (D) or a type of *drug properties*, such as a MeSH pharmacological actions category of drug (C), a pathway associated with a drug (P), a protein that is a target of a drug (T), an indication of a drug (I), and a side effect of a drug (SE). key_G is a function that maps a node to a set of keywords that are identifiers or descriptions of the node. For example, a node v that represents a drug has $type_G(v) = "Drug"$, and $key_G(v)$ includes the drug's generic names and brand names.

The schema of the drug graph is shown in Figure 1a. An edge from a drug to a drug property node (type C, P, T, I, SE) represents the fact that the drug has a particular property. An edge between two drug nodes represents the fact that the two drugs can interact. Notice that the drug graph includes not only the drug properties expected to be in the queries but also drug chemical/biological information, such as targets and pathways. These nodes allow us to establish relationships among drugs, which are useful for inferring potential associations between drugs and drug properties.

B. Query Expression

Now we describe queries on the drug graph. A query is represented as a graph $Q(V_Q, E_Q, type_Q, key_Q)$. We refer to nodes in the query graph as *query nodes*. The query graph follows the same schema as the drug graph, but in contrast to the drug graph, some nodes may not be assigned a keyword as they represent the information the user is looking for.



(a) Schema of drug graph (b) Example query graph (c) Example query graph 2

Fig. 1. Schema of the drug graph and example query graphs. (solid edges: positive, dashed edges: negative)

Accordingly, we can divide the query nodes into two groups. (1) **Variable node**: A variable node represents the information the user wants to find. The keywords of the variable nodes are not given in the query. (2) **Reference node**: A reference node serves as a reference for identifying the variable nodes. Each reference node has a keyword specified by a user.

Each edge in the query graph has a **sign**, which can either be positive or negative. A positive edge means that the user wants the connection to exist between the two nodes, while a negative edge means there should be no connection between the two nodes.

We show two example query graphs in Figure 1. In Figure 1b, the query graph corresponds to the query: *Find a **drug** for **fever** and **allergy** that does not interact with **Enoxaparin** and does not cause **drowsiness** and **nausea***. In this query graph, there is one variable node, d_1, representing the drug to find. The drug d_1 is connected to the two indications with positive edges. To avoid side effects and interactions, negative edges are used to connect d_1 to the side effect nodes and another drug node. The query graph in Figure 1c is for the query: *Find a **drug** for **diabetes** and a **drug** for **epilepsy** that do not interact with **Cartrol** and **Aceon**, do not interact with each other, and do not cause **high blood pressure***. Notice that there are now two variable drug nodes, d_1 and d_2, and there is a negative edge between them as we want to avoid drug interaction between d_1 and d_2.

In this work, to allow flexibility in query expression, we assume the signs are given in the query graph. However, for specific applications, the signs may be inferred based on the types of the nodes adjacent to the edges. For example, for drug prescription, an edge between a drug node and an indication node should always be positive.

C. Answering Drug Queries

Given a drug graph and a query graph, answering query is to find a subgraph in the drug graph that matches the query graph. We formally define an answer to a query as follows. An *answer* of a query $Q(V_Q, E_Q, type_Q, key_Q)$ is in the form of a mapping function f that maps each query node to a node in the drug graph such that: (1) For each variable node q_v, $type_G(f(q_v)) = type_Q(q_v)$. (2) For each reference node q_r, $type_G(f(q_r)) = type_Q(q_r)$ and $key_G(f(q_r)) \cap key_Q(q_r) \neq \emptyset$.

Traditional query systems find exact answers for a given query, which are defined as follows. An *exact answer* is an answer f that satisfies the following properties: (1) For each positive edge $e(q_i, q_j)$ in E_Q, there is an edge $e(f(q_i), f(q_j))$ in the drug graph. (2) For each negative edge $e(q_i, q_j)$ in E_Q, there is no edge between $f(q_i)$ and $f(q_j)$ in the drug graph. In this work, to cope with data incompleteness, we consider both the answers that exactly match the query graph and those that *closely* match the query graph. Therefore, the main problems we address are as follows: (1) How to assign a score to an answer to quantify how well it matches the query? (2) How to find the top-k answers with the highest scores to present to the users? We present our solutions to these problems in the next section.

IV. Methodology

In this section, we describe how we design a score function to evaluate the quality of the answers and present our algorithm for finding the top-k answers according to the score function. To simplify the explanation, the solution presented in this section does not take into account patient profiles (i.e., no personalization). We discuss how to extend the approach to provide personalization in Section V.

A. Design of the Score Function

Because the drug information graph is incomplete, having no edge between a drug node and a drug property node does not necessarily mean that there is no association between the node pair. Taking this fact into consideration, our score function quantifies for each node pair the likelihood that there is an edge between the node pair and integrates the likelihood into the score of an answer. We first describe how to quantify the likelihood of a single edge and then present the score function.

1) Quantifying Edge Likelihood: For each pair of nodes, our approach quantifies the likelihood of having an edge between the two nodes based on the existing connections between the two nodes. Intuitively, two nodes that have many paths in-between should be closely related, and thus it is more likely that there is an edge between them. However, simply using the number of paths may not be effective. Because the drug graph contains several types of nodes, the paths in the graph are representing different types of complex relationships. These complex relationships can have different importance as an indicator on whether there is an edge between a given node pair. Therefore, we differentiate the paths by *path types* and assign different importance levels to different path types.

The type of a path is defined from the types of nodes along the path. For example, the path d_1(D)-t_1(T)-d_2(D)-se_1(SE) has the path type D-T-D-SE, which represents the relationship where a drug shares a target with another drug that has a particular side effect. The likelihood of an edge is quantified as a function of the number of paths of different types. Formally, let $M = \{m_1, ..., m_k\}$ be the set of path types. Let $c_{m_l}(v_i, v_j)$ be the number of paths between v_i and v_j that have type m_l. The likelihood of an edge between v_i and v_j, denoted by $p(v_i, v_j)$, is based on a logistic regression model as follows.

$$p(v_i, v_j) = 1/(1 + e^{-(\beta_0 + \beta_1 c_{m_1}(v_i, v_j) + ... + \beta_l c_{m_l}(v_i, v_j))}), \quad (1)$$

where $\beta_0, ..., \beta_l$ are the model parameters reflecting the importance of each path type.

Learning the model parameters. We apply the maximum likelihood estimation (MLE) techniques [22] to learn the parameters. The learning process uses the known associations between drugs and drug properties to fit the parameters. Since MLE is a well-studied technique, we omit the details of the approach due to space limitation. For different types of edges (e.g., drug-side effect edge or drug-drug edge), the importance of each path type can be different. Furthermore, for the same edge type, the importance of path types could also be different for each node. Therefore, we consider two schemes for parameter learning. **(1) Global parameter learning**: Learn only one set of parameters for each edge type. **(2) Local parameter learning**: For each edge type, for each node, learn one set of parameters. For example, with global learning, a single set of parameters is learned for the edge type drug-side effect; with local learning, one set of parameters is learned *for each side effect*. While local learning may provide the parameters that better fit each node, it can suffer from the fact that there are only a few positive samples for each node. On the other hand, if the importance of the paths varies greatly among the nodes, global learning may not yield good performance. We compare the performance of the two schemes in our experiments.

Selecting path types. As a general rule, the path types considered should correspond to the relationships that can signify edge existence. Although the weight of the path types would be adjusted through the parameter learning, it is computationally inefficient to include irrelevant path types. In the following, we discuss our selection of path types for drug-side effect edges as an example.

The path types used for the drug-side effect edges are shown in the second column of Table I. The path type D-D-SE is based on the hypothesis that drugs that interact tend to have similar side effects. The other path types are selected based on the hypothesis that if two drugs share a *property*, such as an indication or a target, they tend to have similar side effects; therefore, these paths are in the form of D-*propType*-D-SE, where *propType* is a drug property node type, including I, P, T, and SE. The path types used for drug-indication edges and drug-drug edges are selected with similar ideas, as shown in Table I.

TABLE I. PATH TYPES USED FOR COMPUTING EDGE LIKELIHOOD.

Path ID	D-SE edge	D-I edge	D-D edge
P1	D-I-D-SE	D-I-D-I	D-I-D-D
P2	D-P-D-SE	D-P-D-I	D-P-D-D
P3	D-T-D-SE	D-T-D-I	D-T-D-D
P4	D-D-SE	D-D-I	D-D-D
P5	D-SE-D-SE	D-SE-D-I	D-SE-D-D

2) Score Function for Query Answers: Based on our approach for quantifying the edge likelihood, we now define the score of an answer for a given query. Intuitively, in a good answer, the edges among the matches of the query nodes should correspond to the edges in the query graph. More specifically, if there is a positive edge between q_i and q_j, then there should be an edge between $f(q_i)$ and $f(q_j)$, the matches of q_i and q_j in an answer f. If there is a negative edge between q_i and q_j, then there should be no edges between $f(q_i)$ and $f(q_j)$. Our scoring function is defined based on this concept, as follows. For a given query graph Q, let E_Q^+ and E_Q^- denote the set of positive edges and the set of negative edges in the query graph, respectively. Let $w_G(v_i, v_j)$ be the function indicating whether there is an edge between v_i and v_j in the drug graph. That is, $w_G(v_i, v_j) = 1$ if there is an edge in the graph. Otherwise, $w_G(v_i, v_j) = 0$. The score of an answer f, denoted by $S(f)$, is defined as

$$S(f) = \prod_{e(q_i,q_j) \in E_Q^+} p'(f(q_i), f(q_j)) \prod_{e(q_i,q_j) \in E_Q^-} (1 - p'(f(q_i), f(q_j))), \quad (2)$$

where $p'(v_i, v_j) = 1$ if $w_G(v_i, v_j) = 1$; otherwise, $p'(v_i, v_j) = p(v_i, v_j)$ (defined in Eq. 1). The function p' is used to modify the likelihood score so that the likelihood is equal to 1 if an edge already exists in the graph. The first product in $S(f)$ considers the edge likelihood between the node pairs connected by positive edges. The second product considers the complement of the edge likelihood, i.e., $1 - p(v_i, v_j)$, for the node pairs connected by negative edges. The value of $S(f)$ ranges from 0 to 1. An answer f having $S(f)$ equal to 1 is an exact answer.

B. Finding the Top-k Answers

Having defined the score function, now we describe the algorithm for finding the top-k answers that have the highest scores. The algorithm consists of three main steps as follows.

Step 1: Find candidates matches of each query node. Based on the definition of an answer in Section III-C, the candidates of a reference node, q_r, are the nodes that have the same type as q_r and have at least one keyword that matches with $key_Q(q_r)$. The candidates of a variable node, q_v, are the nodes that have the same type as q_v. We denote the set containing all the candidate matches of a query node q_i as $can(q_i)$.

Step 2: Compute edge likelihood among candidate matches. The edge likelihood scores are used for computing the scores of the answers. For each edge in the query graph, $e(q_i, q_j)$, we compute the edge likelihood between the nodes in $can(q_i)$ and $can(q_j)$, which requires counting paths of different types among the candidate nodes. To count the paths of a specific type T, we use a modified breadth-first search (BFS) algorithm, where in each level of the search, only the nodes having the correct type according to T are visited. With a BFS starting from a source node v, we can obtain the number of T-paths from v to every node in the graph. Therefore, for an edge $e(q_i, q_j)$, for each node in $can(q_i)$, we perform m BFSes, where m is the number of path types being used. In total, for an edge $e(q_i, q_j)$, we perform at most $m \cdot |can(q_i)|$ BFSes.

Step 3: Search for top-k answers. In this step, we search for k answers that have the highest scores among all the answers, which are all the combinations of the query nodes' candidate matches. We apply the branch-and-bound technique to obtain the top answers quickly. As the technique is a classic solution for combinatorial optimization, we refer readers to [23] for more details.

V. Personalizing Answers Based on Patient Profiles

It is not uncommon that some drug properties are more common or present only in a patient with a specific profile. For example, the side effect *vomiting* for the drug *Tamiflu* is more common in children than adults [24]. In this section, we describe how to extend our approach to personalize the answers according to a patient profile. We discuss the extension in terms of side effect personalization, but the approach can be applied to other types of drug properties.

To provide personalization, we first define a collection of patient states that users can use to describe a patient in a query. The patient states include the characteristics of patients that can affect the side effects of drugs such as age, gender, genetic markers, and lifestyles. A *patient profile* is defined as a set of patient states, e.g., $\{female, elder\}$. Then, we modify the graph model and the score function as follows.

The drug graph model is extended so that the existence of each edge is *conditioned based on patient profiles*. We define a weight function $w_G(v_i, v_j, A)$ to represent the association strength between node v_i and node v_j for a patient with profile A. The value of $w_G(v_i, v_j, A)$ ranges from 0 and 1. In a basic scheme, for a drug d_i and side effect se_j, if there is a chance that the side effect will occur for a patient profile A, we let $w_G(d_i, se_j, A)$ be equal to 1; otherwise, let $w_G(d_i, se_j, A)$ be equal to 0. However, if the data sources allow quantifying the likelihood of side effects for each profile, the weight can also be set to reflect such likelihood.

Based on the extended drug graph, we modify the score function (Eq. 2) by changing the definition of the function p'. The function p' is originally used so that when there is no edge between nodes u and v in the drug graph, the edge likelihood is used to compute the score S. With patient profiles, we modify $p'(v_i, v_j)$ so that it selects the correct weight of an edge based on a given user profile as follows. Let Q_A be the patient profile provided by the user. Let $\mathcal{A}$ be the set of patient profiles X such that $X \subseteq Q_A$ and $w(v_i, v_j, X) > 0$. Here we consider any profile X that is a subset of Q_A because any patient who fits the profile Q_A also fits the profile X. For example, if the user specifies the profile $\{female, teenage\}$, the profile $\{female\}$ is also applicable to such patients. **(1) If $\mathcal{A}$ is not empty**, which means for the given patient profile, v_i is associated with v_j, we let $p'(v_i, v_j) = w(v_i, v_j, X^*)$, where $X^* = argmax_{X \in \mathcal{A}} |X \cap Q_A|$. Here the value $p'(v_i, v_j)$ is set to be the weight between v_i and v_j based on the profile X^*, which best fits with the given profile Q_A. **(2) If $\mathcal{A}$ is empty**, which means there is no association between v_i and v_j in the drug graph according to the given user profile, we fall back to using the edge likelihood. That is, we let $p'(v_i, v_j) = p(v_i, v_j)$ (as computed in Section IV-A1).

Now we provide an example of a strategy for assigning $w(d_i, se_j, A)$ for a drug d_i and a side effect se_j. The assignment is based on FDA adverse drug event (ADE) reports. Each ADE report contains the information about a patient (age, weight, gender), the list of drugs taken by the patient, and the side effects of the drugs. Although we cannot conclude that the drugs in each report are the causes of the side effects, the reports provide signals of the potential association between the side effects and drugs for different patient profiles.

With the information available from the reports, our patient states include characteristics based on age and gender. The age is divided into four ranges: child (age 0-12), teenage (age 13-19), adult (age 20-64), elder (age 65 and up). The full set of patient states is $\{male, female, child, teenage, adult, elder\}$. To assign the value of $w(d_i, se_j, A)$, we consider two supporting factors: (1) whether se_j is on the label of drug d_i (2) whether there is a report that contains d_i and se_j with patient profile A. Let $R(d_i, se_j, A)$ be the number of reports containing drug d_i and side effect se_j with the patient profiles that fit A. We assign $w(d_i, se_j, A)$ based on the four cases shown in the following table.

TABLE II. Weight assignment for drug-side effect edges.

se_j in label of d_j	$R(d_i, se_j, A) > 0$	$w(d_i, se_j, A)$
true	true	1
true	false	0.75
false	true	0.25
false	false	0

In this assignment, we assign different confidence levels to the two supporting factors, giving more confidence to the drug labels. The association between a drug and a side effect for a patient profile A is strongest if both supporting factors are available. There are other possible assignment schemes. For example, the number of reports and the bias of the reports can be taken into consideration. As the focus of this work is to provide a framework for supporting personalization, we leave the problem of determining the best weight assignment schemes as our future work.

VI. Data Sources and Drug Graph Characteristics

We consolidate drug information from multiple data sources to create the drug graph for our prototype query system. The details of each data source are given as follows.
DrugBank. DrugBank [2] is a database that contains chemical, pharmacological, and pharmaceutical data of drugs along with drug target information. For each drug, we obtain its category, targets, and drug interactions. The number of drugs in DrugBank is 7,682. 86% of the drugs have target information. 15% of the drugs have the drug interaction information.
SIDER2. SIDER2 [3] contains the information about drug indications and side effects extracted from drug labels. The side effects and indications are mapped to MedDRA[1] preferred terms. There are 2,021 drugs in SIDER2. 49% of the drugs have side effect information, and 90% of the drugs have indication information.
KEGG Drug. KEGG Drug [4] is a database containing information for approved drugs in Japan, USA, and Europe. For each drug, we obtain the information of its associated pathways. There are 9,354 drugs in the database. 27% of the drug have pathway information.

[1]http://www.meddra.org

FDA Adverse Drug Event (ADE) Reports. As described earlier, we use the ADE reports to personalize our answers according to patient profiles. We use the reports from the year 2012 and 2013. There are a total of 1.46 million reports.

From the mentioned data sources, we link the information of each drug by matching drug brand names and generic names. After linking the information and creating the drug graph, we remove the drug nodes that do not have any links to the other nodes in the graph. The resulting graph contains 17,842 nodes and 162,673 edges. The numbers of nodes and edges in each type are shown in Table III.

TABLE III. DRUG GRAPH CHARACTERISTICS.

Node type	#Nodes
Drug	7,395
Indication	3,032
Side effect	3,180
Pathway	132
Target	4,103
Total	17,842

Edge type	#Edges
Drug-Drug	24,224
Drug-Side effect	102,990
Drug-Indication	17,809
Drug-Target	15,105
Drug-Pathway	2,545
Total	162,673

VII. EVALUATION

Our evaluation consists of two parts. First, we evaluate the quality of edge likelihood scores. Then, we evaluate our query system by showing examples of query results and discuss the benefits provided by our system.

A. Evaluation of Edge Likelihood Quantification

Edge likelihood scores are the basis for computing the scores of the answers; therefore, it is important that the likelihood scores are good predictors of the existence of edges in reality.

1) Evaluation Method: We evaluate the edge likelihood score for three types of edges: drug-side effect (D-SE), drug-indication (D-I), and drug-drug (D-D). For each type of edge, we first learn the model parameters with training data. Then, for all the node pairs that are not connected by an edge in the training data, we compute their likelihood scores. Finally, we measure how much the likelihood scores correlate with the existence of the edges in testing data. We describe our training and testing data, the evaluation metric, and the baseline algorithm we compare with in the following.

Training and Testing Data. Drug-Side effect. The training data are based on the edges obtained from SIDER2. The testing data are based on the edges learned from the ADE reports. For each drug-side effect pair, if there are at least two reports that associate the drug with the side effect, we consider the drug-side effect pair to be positive in the testing data. Here we require at least two reports because the reports are submitted by consumers and practitioners, which means the associations between drugs and side effects obtained from the reports have not been scientifically confirmed. The number of new drug-side effect pairs obtained from the ADE reports is 33,146. Drug-Indication. The training data are based on the associations obtained from SIDER2. For the testing data, we extract associations between drugs and indications from the ADE reports, which contain for each drug an indication that the drug is prescribed for. We consider a drug-indication pair to be positive if we find at least one report associating the drug with the indication. The number of new drug-indication pairs obtained from the ADE reports is 11,903. Drug-Drug. Drug interactions extracted from DrugBank are used as the training data. We obtain additional drug interaction data from KEGG Drug to create the testing data. The number of new drug interaction pairs obtained from KEGG is 52,743.

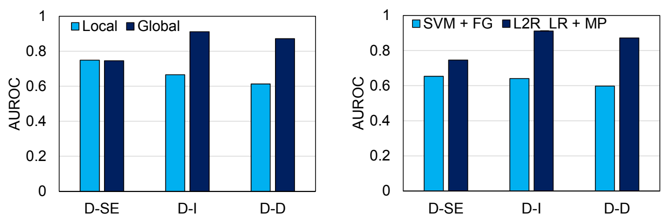
Evaluation Metric. Our evaluation metric is the AUROC [1]. The AUROC is computed as the area under the plot of the true positive rate against the false positive rate for different threshold values of the likelihood score. The true positive rate and the false positive rate are computed based on the testing data. The AUROC has the value between 0 and 1, and the higher the value the better.

Baseline Algorithm. We compare our approach with an approach adapted from [25], which we refer to as SVM+FG (SVM learning algorithm with fine-grained drug features). The approach was originally proposed for predicting drug-side effect associations. To predict the associations, SVM+FG represents each drug with a binary feature vector describing various types of drug properties, such as indications, side effects, targets, and chemical structures. For a drug d, the value in a dimension i is equal to 1 if drug d has the property associated with that dimension. Otherwise, the value is 0. Our goal here is to compare our approach, which uses network features, with the approach that uses fine-grained features of the drugs; therefore, we provide equivalent input data to both approaches. That is, the drug properties used to create the binary feature vectors include side effects, indications, targets, pathways, and drug interactions, corresponding to the path types used in our network-based approach. The SVM learning algorithm is used to train a classifier from the drug feature vectors. To improve performance, we apply a basic feature selection technique by retaining only the features that are present in at least one positive sample.

2) Evaluation Results: **Global vs. Local Parameter Learning.** First, we compare the performance of global and local parameter learning, as discussed in Section IV-A1. Figure 2a shows the performance of the two schemes for drug-side effect, drug-indication, and drug-drug edges. It can be seen that for all three types of edges, the global scheme performs better than the local scheme. Especially for drug-indication and drug-drug, the global parameter learning has the AUROC higher than the local learning by more than 35%. The fact that global parameter learning performs well even though the same set of parameters are used for all the nodes suggests that the importance of path types are quite consistent across the nodes. For all the remaining experiments, we use the parameters learned through the global parameter learning scheme.

Comparison with Baseline Algorithm. The performance comparison between our algorithm and SVM+FG is shown in Figure 2b. Our approach is denoted by L2R_LR+MP (L2-regularized logistic regression with multiple path types) in the figure. As can be seen from the figure, our network-based approach ourperforms SVM+FG for all three types of edges. The AUROC is improved by up to 40% compared to the baseline algorithm.

Benefits from Using Multiple Path Types. Next, we evaluate how much using multiple path types helps to improve the



(a) Global vs. local learning (b) Our approach vs. baseline

Fig. 2. Performance comparison.

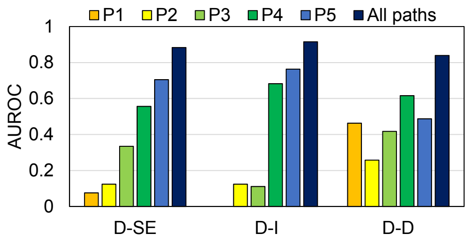


Fig. 3. Performance comparison when a single path type is used and when all the path types are used.

accuracy, in comparison to using only a single path type. Figure 3 shows the performance when each of the path type is used individually and when all the five path types (listed in Table I) are used. The results show that using multiple types of paths can significantly improve the performance. When a single path type is used, the maximum AUROC that can be achieved are 0.70, 0.77, and 0.62, for drug-side effect, drug-indication, and drug-drug, respectively. When all the five path types are used, we can obtain the AUROC of 0.89, 0.91, and 0.93. Additionally, it should be noted that the difference in the AUROC obtained from each path type supports our hypothesis that different path types have unequal importance in signifying the likelihood of edge existence.

B. Evaluation of Query Answering

In this section, we demonstrate the usefulness of our query system by showing examples of the query results returned from our system.

First, we show the top results for the query **[Query 1]** *Find a drug for peptic ulcer* in Table IV. Our system finds both exact matches and close matches for this query. The first 11 answers, which include Methscopolamine and Omeprazole, receive the highest possible score of 1. This means according to our data sources, these 11 drugs are indicated for peptic ulcer and thus they exactly match the query. The next three answers, Lansoprazole, Paroxetine, and Pantoprazole, are inexact matches. According to our data sources, these three matches are not indicated for peptic ulcer. However, according to RxList [2] and MedicineNet.com [3], Lansoprazole and Pantoprazole, which are ranked at the 12^{th} and 14^{th} places, can actually be used for treating peptic ulcer. These answers would be missed if we do not consider approximate matches. This example illustrates that our approach can provide answers that exactly match the query as well as potentially good answers that are inexact matches, which provides users with more alternatives.

[2]http://www.rxlist.com

[3]http://www.medicinenet.com

TABLE IV. Results for Query 1.

Rank	Drug	Score	Answer Quality
1	Methscopolamine	1	Exact match
..	...	...	...
11	Omeprazole	1	Exact match
12	Lansoprazole	0.01	Relevant according to external sources
13	Paroxetine	0.01	False positive
14	Pantoprazole	0.004	Relevant according to external sources

TABLE V. Results for Query 2.

(a) Traditional

Rank	Drug	Score
1	Clozapine	1
2	Ziprasidone	1
3	Quetiapine	1
...	...	...
16	Reserpine	1

(b) Our approach

Rank	Drug	Score
1	Fluphenazine	0.98
2	Lurasidone	0.98
3	Molindone	0.97
4	Reserpine	0.97
5	Haloperidol	0.96

However, it should be noted that our query system is not intended to replace experts. The system provide lists of drugs that can potentially fit users' requirements; answers obtained need to be further reviewed by users.

Next, we consider the following query: **[Query 2]** *Find a drug for schizophrenia for the patient who is taking Paroxetine*. In Table V, we compare the results from our approach and from the traditional approach that finds only exact matches. Using the traditional approach, we obtain 16 drugs. All the drugs have the same score of 1 because they do not interact with Paroxetine according to the data sources. Using our approach, the same 16 drugs are returned as answers, but their scores are now less than 1 and different from one another. The scores of less than 1 indicate that there is some possibility that the drugs will interact with Paroxetine, and the lower the score, the more likely the interaction. We manually checked drug interactions on Drugs.com[4] and found that 15 out of the 16 drugs can actually interact with Paroxetine. Additionally, Reserpine, the only drug that does not interact with Paroxetine according to Drugs.com, is ranked at the fourth place in our result list. This example demonstrates that by taking into account the likelihood of missing associations, our system is more informative and can better help users to discover drugs that best fit their needs.

In this following example, we illustrate how the answers are personalized according to a given patient profile. We use the query: **[Query 3]** *Find the drug for schizophrenia without the side effect cardiac arrest*. We compare the top 15 results obtained when a user profile is not given and when the user profile is specified as {*female, elder*} in Table VI. When the user profile is given, three of the drugs which are Haloperidol, Quetiapine, and Clozapine, receive lower scores and thus lower ranks. These three drugs were reported (via the FDA drug adverse event reporting system) as potential causes of cardiac arrest in elder female patients. Our approach takes into account this fact and adjusts the scores of the drugs accordingly.

Finally, we show an example of a query that asks for multiple drugs as follows: **[Query 4]** *Find a set of drugs for Parkinson's disease (I_1), epilepsy (I_2), and depression (I_3), that do not interact with one another*. We obtain 15 drug sets having the full score of 1, for example, {Carbidopa (for I_1), Carbamazepine (for I_2, I_3)} and {Carbidopa (for

[4]http://www.drugs.com

TABLE VI. RESULTS FOR QUERY 3.

(a) WITHOUT PROFILE

Rank	Drug	Score
1	fluphenazine	0.99
2	lurasidone	0.99
3	iloperidone	0.99
4	molindone	0.99
5	pimozide	0.99
6	reserpine	0.99
7	**haloperidol**	0.98
8	loxapine	0.97
9	asenapine	0.96
10	buspirone	0.87
11	paliperidone	0.87
12	**quetiapine**	0.86
13	ziprasidone	0.86
14	**clozapine**	0.25
15	thiothixene	0.25

(b) WITH PROFILE

Rank	Drug	Score
1	fluphenazine	0.99
2	lurasidone	0.99
3	iloperidone	0.99
4	molindone	0.99
5	pimozide	0.99
6	reserpine	0.99
7	loxapine	0.97
8	asenapine	0.96
9	buspirone	0.87
10	paliperidone	0.87
11	ziprasidone	0.86
12	**quetiapine**	0.75
13	**haloperidol**	0.75
14	thiothixene	0.25
15	aripiprazole	0.25

I_1), Gabapentin (for I_2), Bupropion (for I_3)}. If we use a traditional exact match approach, more than thousands of drug sets would be returned for this query; all of which have the same score of 1, which can make it difficult for users to select the best answer. Using our approach, some of these drug sets receive lower scores because of their potential interactions. For example, {Amantadine (for I_1), Fosphenytoin (for I_2), Aripiprazole (for I_3)} receives the score of 0.36. These drugs do not interact according to our data sources, but our approach considers the likelihood of their interactions and gives the drug set a lower score. According to Drugs.com, we found that Aripiprazole can indeed interact with the other two drugs.

VIII. CONCLUSION

In this paper, we propose an approach for answering drug queries to support drug prescription. To cope with incomplete and noisy data, we allow both exact and close matches when answering queries. The answers are ranked by utilizing the structure of a drug information network to quantify the likelihood of associations between drug and drug properties in the case that the associations are missing. We demonstrate how our approach could assist practitioners to make informed decision when prescribing drugs through several examples.

While this work addresses one of the major problems in supporting prescription, which is how to obtain and rank the answers, further work is needed to achieve a complete prescription support system. First, our current system assumes a query graph is given by users. For better usability, a module that assists users in constructing the query graphs, such as a form-based user interface or a module that translates natural language queries to query graphs, should be developed. Second, as mentioned in our evaluation, our system is intended for assisting experts, not replacing them. The answers obtained need to be reviewed by experts. Thus, a module that provides users with supporting evidence for answers could be helpful. Additionally, there are many other factors apart from side effects and drug interactions that are important in prescribing drugs. For our future work, we would like to obtain evaluation and feedbacks from practitioners and improve our system to better support their needs according to real-world usage.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions. This work is supported by the NSF under Grant No. CNS-1217284 and CCF-1018114.

REFERENCES

[1] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[2] C. Knox *et al.*, "Drugbank 3.0: a comprehensive resource for omics research on drugs," *Nucleic acids research*, vol. 39, no. suppl 1, pp. D1035–D1041, 2011.

[3] M. Kuhn *et al.*, "A side effect resource to capture phenotypic effects of drugs," *Molecular systems biology*, vol. 6, no. 1, p. 343, 2010.

[4] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.

[5] J. Sun, H. Xu, and Z. Zhao, "Network-assisted investigation of antipsychotic drugs and their targets," *Chemistry & biodiversity*, vol. 9, no. 5, pp. 900–910, 2012.

[6] F. Cheng *et al.*, "Prediction of polypharmacological profiles of drugs by the integration of chemical, side effect, and therapeutic space," *Journal of chemical information and modeling*, vol. 53, no. 4, pp. 753–762, 2013.

[7] J. Sun *et al.*, "Network-assisted prediction of potential drugs for addiction," *BioMed research international*, vol. 2014, 2014.

[8] J. Huang *et al.*, "Systematic prediction of pharmacodynamic drug-drug interactions through protein-protein-interaction network," *PLoS computational biology*, vol. 9, no. 3, p. e1002998, 2013.

[9] F. Cheng and Z. Zhao, "Machine learning-based prediction of drug–drug interactions by integrating drug phenotypic, therapeutic, chemical, and genomic properties," *J. Am. Med. Inform. Assoc.*, vol. 21, no. e2, pp. e278–e286, 2014.

[10] H. Zheng *et al.*, "Linking biochemical pathways and networks to adverse drug reactions," *NanoBioscience, IEEE Transactions on*, vol. 13, no. 2, pp. 131–137, 2014.

[11] J. Sun *et al.*, "Characterization of schizophrenia adverse drug interactions through a network approach and drug classification," *BioMed research international*, vol. 2013, 2013.

[12] K. Sangkuhl *et al.*, "Pharmgkb: understanding the effects of individual genetic variants," *Drug metabolism reviews*, vol. 40, no. 4, pp. 539–551, 2008.

[13] C. Doulaverakis *et al.*, "Panacea, a semantic-enabled drug recommendations discovery framework," *J. Biomed. Semant.*, vol. 5, p. 13, 2014.

[14] M. Dumontier and N. Villanueva-Rosales, "Towards pharmacogenomics knowledge discovery with the semantic web," *Briefings in bioinformatics*, vol. 10, no. 2, pp. 153–163, 2009.

[15] A. Ben Abacha and P. Zweigenbaum, "Medical question answering: translating medical questions into sparql queries," in *Proceedings of the 2nd ACM SIGHIT*. ACM, 2012, pp. 41–50.

[16] J. Scheiber *et al.*, "Mapping adverse drug reactions in chemical space," *Journal of medicinal chemistry*, vol. 52, no. 9, pp. 3103–3107, 2009.

[17] A. Bender *et al.*, "Analysis of pharmacology data and the prediction of adverse drug reactions and off-target effects from chemical structure," *ChemMedChem*, vol. 2, no. 6, pp. 861–873, 2007.

[18] F. Hammann *et al.*, "Prediction of adverse drug reactions using decision tree modeling," *Clinical Pharmacology & Therapeutics*, vol. 88, no. 1, pp. 52–59, 2010.

[19] E. Pauwels, V. Stoven, and Y. Yamanishi, "Predicting drug side-effect profiles: a chemical fragment-based approach," *BMC bioinformatics*, vol. 12, no. 1, p. 169, 2011.

[20] M. Fukuzaki *et al.*, "Side effect prediction using cooperative pathways," in *BIBM'0*. IEEE, 2009, pp. 142–147.

[21] L.-C. Huang, X. Wu, and J. Y. Chen, "Predicting adverse side effects of drugs," *BMC genomics*, vol. 12, no. Suppl 5, p. S11, 2011.

[22] D. G. Kleinbaum and M. Klein, *Logistic regression: a self-learning text*. Springer, 2010.

[23] J. Clausen, "Branch and bound algorithms-principles and examples," *Dept of Comp. Sci., University of Copenhagen*, pp. 1–30, 1999.

[24] Genentech USA, "Tamiflu (oseltamivir phosphate) Prescribing Information." http://www.tamiflu.com/hcp/prescribing/hcp_prescribe.jsp.

[25] M. Liu *et al.*, "Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs," *J. Am. Med. Inform. Assoc.*, vol. 19, no. e1, pp. e28–e35, 2012.