Briefings in Bioinformatics
doi.10.1093/bib/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Facilitating Prediction of Adverse Drug Reactions by Using Knowledge Graphs and Multi-Label Learning Models

Emir Muñoz [1,2,*], Vít Nováček [2], Pierre-Yves Vandenbussche [1]

[1] Fujitsu Ireland Ltd., Co. Dublin, Ireland and
[2] Insight Centre for Data Analytics at NUI Galway, Co. Galway, Ireland.

[*] To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Timely identification of adverse drug reactions (ADRs) is highly important in the domains of public health and pharmacology. Early discovery of potential ADRs can limit their effect on patient lives and also make drug development pipelines more robust and efficient. Reliable *in-silico* prediction of ADRs can be very helpful in this context and thus it has been intensely studied. Recent works achieved promising results using machine learning. The presented work focuses on machine learning methods that use drug profiles for making predictions and utilise features from multiple data sources. We argue that despite promising results, existing works have limitations, especially regarding flexibility in experimenting with different data sets and/or predictive models. We suggest to address these limitations by generalisation of the key principles employed by the state-of-the-art. Namely, we explore effects of: (1) using knowledge graphs—machine-readable interlinked representations of biomedical knowledge—as a convenient uniform representation of heterogeneous data; and (2) casting ADR prediction as a multi-label ranking problem. We present a specific way of using knowledge graphs to generate different feature sets and demonstrate favourable performance of selected off-the-shelf multi-label learning models in comparison to existing works. Our experiments suggest better suitability of certain multi-label learning methods for applications where ranking is preferred. The presented approach can be easily extended to other feature sources or machine learning methods, making it flexible for experiments tuned towards specific requirements of end users. Our work also provides a clearly defined and reproducible baseline for any future related experiments.

Key words: Adverse Drug Reactions (ADR); Drug Similarity; Knowledge Graphs; Multi-Label Learning

1 Introduction

Adverse Drug Reactions (ADRs) can cause significant clinical problems and represent a major challenge for public health and the pharmaceutical industry. During a drug development process, pharmacology profiling leads to the identification of potential drug-induced biological system perturbations including primary effects (intended drug target interactions) as well as secondary effects (off-targets drug interactions) mainly responsible for ADRs [1]. Many ADRs are discovered during pre-clinical and clinical trials before a drug is released on the market. However, the use of a registered drug within a large population (demonstrating a wider range of clinical genotypes and phenotypes than considered in the clinical trials) can result in serious ADRs that have not been identified before. This has a large impact on patient safety and quality of life, and also has significant financial consequences for the pharmaceutical industry [2].

The result of a recent review of epidemiological studies in Europe states that 3.5% of hospital admissions are due to ADRs and 10% of patients experience an ADR during their hospitalisation [3]. ADRs are a major cause of morbidity (and associated reduction of quality of life) and mortality [2, 4]. Recent estimates set the number of yearly drug-induced fatalities to 100,000 in the USA and almost 200,000 in Europe, making it the fourth cause of death before pulmonary diseases or diabetes [3, 5]. In addition to the significance for the public health, ADRs are associated with an important economic burden imposed for public health systems and pharmaceutical industry. The extra costs are caused mainly by the withdrawal of dangerous drugs from the market, litigations and further hospitalisations to treat the adverse effects. The annual cost of ADRs in the USA is estimated at $136 billion [6].

Any improvements in the early identification of Adverse Drug Reactions can decrease the high attrition rate in the drug discovery and development process. After the drug registration, better prediction of ADRs can alleviate associated clinical problems and decrease the adverse effect-induced extra costs. *In silico* approaches to predict ADRs of candidate drugs are now commonly used to complement costly and time consuming *in vitro* methods [7]. Computational methods differ by the drug development/deployment stage they are applied at, and by the features used for the prediction of ADRs. Pharmacovigilance systems (monitoring the effects of drugs after they have been licensed for use) mine statistical evidence of ADRs from spontaneous reports by physicians such as the FDA's Adverse Event Reporting System (FAERS) [8–10]; from patient records [11]; or more recently from non-traditional sources such as logs of search engine activity or social media [12, 13]. While these methods limit the risk of serious public health issues by identifying early occurrences of ADRs, they assume that such adverse effects are already demonstrated within a population.

Computational prediction of ADRs during the development cycle of a drug (before the drug is licensed for use) can reduce the cost of drug development and provide a safer therapy for patients [14]. Most state-of-the-art techniques adopt a drug-centric approach and rely on the assumption that similar drugs share the same properties, mechanism of action and therefore also ADRs [15, 16].[1] Predictions of new adverse drug reactions are then based on a drug-drug similarity network. In most of the early works, this network was based on the similarity of the substructures within the active ingredients of drugs [17–20]. More recent approaches combine data covering both chemical space of drugs and biological interaction-based features such as drug target, pathways, enzymes, transporters or protein-protein interactions [21–23]. Lately, integrative methods take into account also phenotypic observation-based features such as drug indications [24–27]. The availability of multi-source structured data has allowed for integration of complementary aspects of drugs and their links to side effects leading to higher accuracy [28].

The scope of this review is given by recent state-of-the-art methods (from 2011 on) that satisfy two key requirements. First, we consider methods that take advantage of multi-source structured data. Second, we focus on techniques that use machine learning to predict the likelihood of a side effect being caused by a given drug (drug-centric approach). Table 1 lists the reviewed approaches along with the features they use.

While many of the state-of-the-art approaches produce results that have great potential for being used in drug development pipelines, there are still things to improve. A limitation that is most relevant as a motivation for the presented review is the lack of flexibility that prevents users who are not proficient in machine learning from easily using the predictive models. This makes it difficult for people like biologists, pharmacologists or clinicians to experiment with data and models fine-tuned towards their specific requirements on the ADR prediction (such as increasing the sensitivity or specificity of the model). The main issue of existing models is that they typically work with data sets that have been manually pre-processed, and the particular prediction methods are adapted to the experimental data in a very focused manner.

We review the key points and limitations of existing approaches and introduce their generalisation based on: 1) Tapping into many diverse interlinked knowledge bases (i.e., knowledge graphs) related to drugs and adverse effects that substantially limits the manual effort required for data integration and feature engineering. 2) Rigorous formulation of the adverse drug reaction prediction problem as a multi-label learning-to-rank problem that allows for easy experimentation with many off-the-shelf machine learning models.

We show that specific applications of these two principles can lead to performance comparable to existing methods. Moreover, the proposed approach produces ranked predictions by default, with many relevant predictions present at the top of the ranked result lists. This is potentially very useful in scenarios where experts (e.g., pharmaceutical researchers or public health officials) have limited time and/or resources, and thus they can only process a few prediction candidates out of many possibilities (there can often be hundreds of predictions for a single drug).

The main contributions of this work are as follows. We propose a specific way of using knowledge graphs to generate different feature sets for ADR prediction and demonstrate the favourable performance of selected off-the-shelf multi-label learning models in comparison to existing works. In addition to that, we show how the approach can be easily extended to other feature sources or machine learning methods. This makes the method flexible for experiments tuned towards specific requirements of end users. Our results and data also provide a clearly defined and reproducible baseline for any future related experiments.

Table 1. Multi-source feature sets used by state-of-the-art methods.

	Atias and Sharan (2011) [17]	Pauwels *et al.* (2011) [18]	Mizutani *et al.* (2012) [21]	Yamanishi *et al.* (2012) [22]	Liu *et al.* (2012) [24]	Bresso *et al.* (2013) [19]	Huang *et al.* (2013) [23]	Jahid and Ruan (2013) [20]	Zhang *et al.* (2015) [25, 26, 28]	Rahmani *et al.* (2016) [29]	Muñoz *et al.* (2016) [27]
Chemical space											
Drug compound substructure	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Biological space											
Drug target				✓	✓		✓		✓	✓	✓
Pathway			✓		✓				✓		✓
Enzymes					✓				✓		✓
Transporters					✓				✓		✓
PPi							✓			✓	
Phenotypic space											
Indication					✓				✓		✓
Cell line response	✓										

2 Materials

Various publicly available data sources can be used to define similarity between drugs [14]. Each data source describes a specific aspect of the pharmacological space of a drug such as its chemical, biological or phenotypic properties. For instance, SIDER database [30] presents information of side effects and indication for marketed drugs. PubChem Compound data [31] contains chemical structure description of drugs. DrugBank [32] provides detailed information about drugs such as their binding proteins and targets, enzymes or transporters, thus informing on drugs' mechanism of action and metabolism. KEGG Genes, Drug, Compound and Disease databases [33] describe further information about molecular interaction of drugs and their signalling pathways.

[1] There are also methods that focus on the ADR information to overcome certain specific problems like drugs with little or no features at all, or ADRs with low number of related drugs [9, 15]. The methods are, however, less numerous and also harder to evaluate in a comparative manner.

Table 2. The data sets characteristics.

Data set	# Drugs	# Side effects
Liu's data set	832	1,385
Bio2RDF data set	1,824	5,880
SIDER 4 data set	1,080	5,579
Aeolus data set	750	181

Table 3. Number of $\langle s, p, o\rangle$ triples in the Bio2RDF data set used in our experiments.

Data set	Type of information	# Triples
DrugBank	Drug types, chemical information	5,151,999
SIDER	Side effects of drugs	5,578,286
KEGG	Drugs, genes and pathway maps	4,387,541

In the following we review the materials—results of data integration using multiple data sources, provided by the authors of the state-of-the-art methods. Because previous data integration activities were expensive and mostly carried out manually, here, we propose a different data source and representation which can be considered a superset of all previous data sets used. This data source is represented using a graph database, a model in which it is simpler to integrate different data sources such as the ones already mentioned. We also provide an algorithm to generate the required drugs' profile, similarly to the ones provided by the reviewed methods (supplemental material Section D). For comparisons, we use Liu's data set [24] and Zhang et al. data set termed "SIDER 4" [25] as benchmarks. As presented in Table 1, Liu's data set contains six types of features covering the chemical, biological and phenotypic spaces of drugs combined with information on their associated ADRs (cf. Table 2). We use this data set as primary means to compare the reviewed methods. SIDER 4 data set introduced by Zhang et al. [25] is an update of Liu's data set integrating the fourth version of SIDER. This data set is interesting as it introduces newly approved drugs for which fewer post-market ADR have been detected. We use the SIDER 4 data set as secondary means to compare the methods.

A new alternative multi-source graph data has recently become via the Bio2RDF project [34]. Bio2RDF publishes the pharmacological databases used in many ADR prediction experiments in the form of a knowledge graph—a standardised, interlinked knowledge representation based on labelled relationships between entities of interest. Bio2RDF data was first used for the prediction of ADRs by Muñoz et al. [27], where drug similarities were computed by measuring the shared connections between drugs in the graph. Here, we build on top of that and evaluate the use of the BioRDF knowledge graph as a means to facilitate the generation of more expressive features for computing similarity between drugs. Such automatically generated data can be used to replace or enrich existing manually integrated feature sets, and be used to evaluate prediction methods as per normal machine learning pipelines.

Finally, to get another perspective for interpreting the evaluation results, we use the Food and Drug Administration (FDA) Adverse Event Reporting System (FAERS) [8, 10]. FAERS publishes recent ADR reports coming from population-wide post-marketing drug effect surveillance activities. Extracting the most recent ADRs for newly marketed drugs helps us to evaluate the ability of various methods to predict ADRs of drugs after their release on the market. We extract this information from the Aeolus data set [35], which is a curated and annotated, machine-readable version of the FAERS database. We use Aeolus to generate an updated version of the SIDER 4 data set that includes also the latest ADRs as observed in the population.

For details on the generation of Liu's data set [24] and the SIDER 4 data set [25], we refer the readers to the original articles. We will now detail the construction of the "Bio2RDF data set" and the "Aeolus data set".

2.1 Bio2RDF data set

The Bio2RDF project (`http://bio2rdf.org/`) aims at simplifying the use of publicly available biomedical databases by representing them in a form of an interconnected multi-graph [34, 36]. The project provides a set of scripts (`https://github.com/bio2rdf`) to convert from the typically relational or tabular databases (e.g., DrugBank, SIDER) to a more flexible triple-based RDF (Resource Description Framework) format. The project also makes available the output of their conversions, and in its version 4, published in December 2015, Bio2RDF represented 30 databases including PubChem, DrugBank, SIDER and KEGG, among others with valuable drug related information. Each information such as drug, protein or side effect is represented as a node entity with a unique identifier, and each relation between them as an edge with a qualified label, generating a network of great value for bioinformatics [37]. Here, we use the release 4 of Bio2RDF and represent its data using a *knowledge graph* $\mathcal{G} = \{\langle s, p, o\rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, which is a set of $\langle s, p, o\rangle$ triples each consisting of a *subject* s, a *predicate* p, and an *object* o, and encoding the statement "s has a relationship p with o". The subject and object $s, o \in \mathcal{E}$ are entities, $p \in \mathcal{R}$ is a relation type, and $\mathcal{E}$, $\mathcal{R}$ denote the sets of all entities and relation types in the knowledge graph, respectively. Fig. 1 shows a fragment of the Bio2RDF knowledge graph that integrates three colour-coded databases, namely, DrugBank (yellow), SIDER (green), and KEGG (blue). Usually, connections between databases are made using identifiers such as PubChem compound or Chemical Abstracts Service (CAS) number. Notice that a knowledge graph can also be built from the original databases using different methods or scripts, and here we select Bio2RDF because it already contains mappings for most of the relevant databases.

A knowledge graph $\mathcal{G}$ can contain biomedical facts[2] such as:

⟨ `http://bio2rdf.org/drugbank:DB00531`, *label*, "Cyclophosphamide"⟩

or

⟨`http://bio2rdf.org/drugbank:DB00531`, *enzyme*, `http://bio2rdf.org/kegg:1.14.13.-`⟩.

This format allows for easy representation of equivalences or external links between data sources as an additional relation/edge. For instance, a relationship between a DrugBank drug and a PubChem compound can be expressed as:

⟨`http://bio2rdf.org/drugbank:DB00531`, *x-pubchemcompound*, `http://bio2rdf.org/pubchem.compound:2907`⟩.

By simply merging multiple data sources from Bio2RDF, we are able to build an integrated knowledge graph with links between databases materialised. During the integration, the PubChem compound of a drug is used to link DrugBank and SIDER, while the CAS number is used to link DrugBank and KEGG. This flexibility for generating training and testing data is currently impossible with the manual integration pipelines used by the reviewed methods. In our experiments we shall use a knowledge graph integrating the DrugBank, SIDER and KEGG databases (cf. Table 3).

2.2 Aeolus data set

Aeolus [35] is a curated and standardised adverse drug events resource meant to facilitate research in drug safety. The data in Aeolus comes from

[2] Note that the URIs in the examples are used as unique identifiers; the availability of corresponding records through an HTTP request (such as in a browser) depends on a third-party service (Bio2RDF.org).

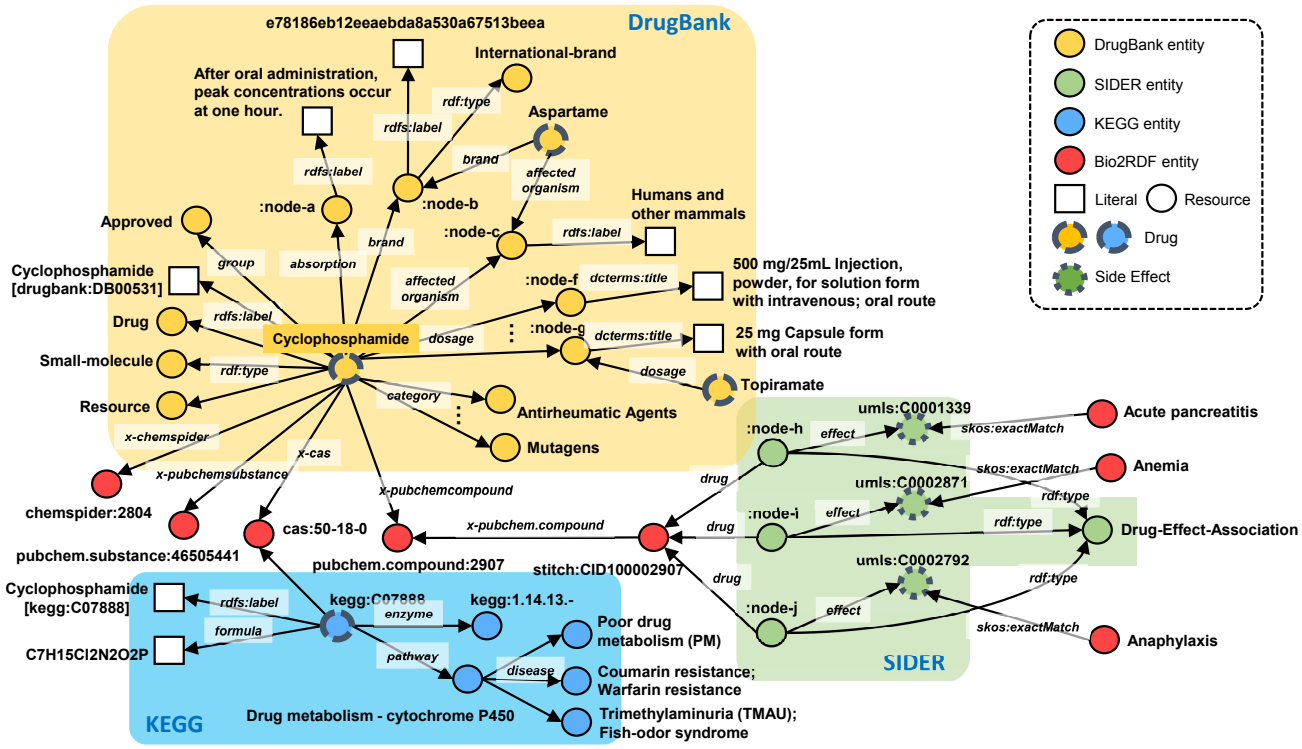

Fig. 1. A Bio2RDF fragment around the Cyclophosphamide drug, showing the connections between three main databases: DrugBank, SIDER, and KEGG.

the publicly available US Food and Drug Administration (FDA) Adverse Event Reporting System (FAERS), but is extensively processed in order to allow for easy use in experiments. In particular, the cases (i.e., adverse drug reaction events) in the FAERS reports are deduplicated and the drug and outcome (i.e., effect) concepts are mapped to standard vocabulary identifiers (RxNorm and SNOMED-CT, respectively). A similar approach for extracting ADR terms from FDA approved drug labels was applied in [38] to group similar drugs by topics. However, Aeolus is preferred due to its curated status.

The Aeolus data set is presented in a convenient comma separated values (CSV) format, from which we can easily extract pairs of drugs and their adverse effects ranked by the statistical significance of their occurrences within the FAERS reports. We map the identifiers for drugs and for adverse effects in Aeolus to the ones in DrugBank which are used in our experiments. This means that we are able to use the FDA FAERS data as an additional manually curated resource for validating any adverse effect prediction method, as detailed later on in the description of our experiments.

3 Methods

In this section, we present details of the reviewed approaches for adverse drug reaction prediction, on the basis of a multi-label learning setting.

3.1 Multi-label Learning Framework

As a drug can generally have multiple adverse reactions, the ADR prediction can be very naturally formulated as a multi-label learning problem [39]. Multi-label learning addresses a special variant of the classification problem in machine learning where multiple labels (i.e., ADRs) are assigned to each example (i.e., drug). The problem can be solved either by transforming the multi-label problem into a set of binary classification problems, or by adapting existing machine learning techniques to the full multi-label problem[3].

Most of the current ADR prediction methods, however, do not fully exploit the convenient multi-label formulation, as they simply convert the main problem into a set of binary classification problems [40]. This is problematic for two main reasons. Firstly, transforming the multi-label problem into a set of binary classification problems is typically very computationally expensive for large numbers of labels (which is the case in predicting thousands of ADRs). Secondly, using binary classifiers does not accurately model the inherently multi-label nature of the main problem. We validate these two points empirically in Section 4. Here, we follow the philosophy of algorithm adaptation: fit algorithms to data [40].

Yet there are exceptions, such as the work in [25], presenting the multi-label learning method FS-MLKNN that integrates feature selection and k-nearest neighbours (kNN). Unlike most previous works, Zhang et al. [25] proposes a method which does not generate binary classifiers per label, but uses an ensemble learning instead. (We shall provide more details of this and other methods in Section 3.2.) Also Muñoz et al. [27] proposed a multi-label solution for the prediction problem using a constrained propagation of ADRs between neighbouring drugs, making clear the benefits of a graph structure of data (cf. supplemental material Section F).

In the following, we formalise the learning framework with Q-labels as in [41, 42]. Let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be the instance space of N different data points (i.e., drugs) in $\mathbb{R}^d$, and let $\mathcal{Y} = \{y_1, y_2, \ldots, y_Q\}$ be the finite set of labels (i.e., ADRs). Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq N\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is a d-dimensional drug feature vector $[x_{i1}, x_{i2}, \ldots, x_{id}]^\top$ and $Y_i \in 2^{\mathcal{Y}}$ is a vector of labels associated with $\boldsymbol{x}_i$, the goal of the learning system is to output a multi-label classifier $h : \mathcal{X} \to 2^{\mathcal{Y}}$ which optimises some specific evaluation metric. In most

[3] See https://en.wikipedia.org/wiki/Multi-label_classification for more details and a list of examples.

cases however, the learning system will not output a multi-label classifier, but instead will produce a real-valued function (aka. regressor) of the form $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $f(\boldsymbol{x}, y)$ can be regarded as the confidence of $y \in \mathcal{Y}$ being a proper label of $\boldsymbol{x}$. It is expected that for a given instance $\boldsymbol{x}$ and its associated label set Y, a successful learning system will tend to output larger values for labels in Y than those not in Y, i.e., $f(\boldsymbol{x}, y_1) > f(\boldsymbol{x}, y_2)$ for any $y_1 \in Y$ and $y_2 \notin Y$. In other words, the model should consistently be more "confident" about true positives (actual ADRs) than about false positives. Intuitively, the regressor $f(\cdot, \cdot)$ can be transformed into a ranking function $rank_f(\cdot, \cdot)$, which maps the outputs of $f(\boldsymbol{x}, y)$ for any $y \in \mathcal{Y}$ to $\{y_1, y_2, \ldots, y_Q\}$ such that if $f(\boldsymbol{x}, y_1) > f(\boldsymbol{x}, y_2)$ then $rank_f(\boldsymbol{x}, y_1) < rank_f(\boldsymbol{x}, y_2)$. The ranking function can naturally be used for instance for selecting top-k predictions for any given drug, which can be very useful in cases where only limited numbers of prediction candidates can be further analysed by the experts.

Our learning problem can now be formally stated as: Given a drug $\boldsymbol{x}$ and a finite-size vector Y with its initially known adverse reactions (i.e., labels), seek to find a discriminant function $f(\boldsymbol{x}, Y) = \hat{Y}$, where $\hat{Y}$ is a finite-size vector representation of the labelling function $\hat{Y} = [f(\boldsymbol{x}, y_1), \ldots, f(\boldsymbol{x}, y_Q)]^\top$ for $y_i \in \mathcal{Y}$. For instance, *headache (C0018681)* and *vomiting (C0042963)* are very common adverse reactions of *Atomoxetine (DB00289)*, a drug used for the treatment of attention deficit hyperactivity disorder (ADHD), and they should be ranked higher than *conjunctivitis (C0009763)* or *colitis (C0009319)*, which are rare or unregistered ADRs for Atomoxetine (cf. supplemental material Section E for features manipulation guidelines).

3.2 Learning Models

To complement most previous works, we formulate ADR prediction as a multi-label ranking problem, and train different machine learning models. This allows for approaching the problem more naturally in many practical use cases, such as when one prefers to explore only a few, i.e., the most relevant adverse effect candidates out of many possible for a certain drug. Multi-label learning models learn how to assign sets of ADRs/labels to each drug/example. The main motivation for our model choices was to have a representative sample of the different multi-label learning families described in the machine learning literature (ranging from decision trees through instance-based learning or regression to neural networks). Such an approach demonstrates the broad range of possibilities when adopting off-the-shelf models. We investigate state-of-the-art multi-label learning models, namely, Decision Trees, Random Forests, Nearest Neighbours (kNN), and Multi-Layer Perceptron. We also investigate the use of Logistic Regression binary classifiers for multi-label following the *one-vs-all* strategy in which the system builds as many binary classifiers as input labels, where samples having label y are considered as positive, and negative otherwise (cf. supplemental material Section B for a description of each model).

Among the methods for predicting ADRs that accept multi-source data are: Liu's method, FS-MLKNN (feature selection-based multi-label k-nearest neighbour) [25], the linear neighbourhood similarity methods (LNSM) with two different data integration approaches, similarity matrix integration (LNSM-SMI) and cost minimisation integration (LNSM-CMI) [28], and finally KG-SIM-PROP (knowledge graph similarity propagation) [27]. Liu et al. [24] proposed a multi-source method using chemical, biological and phenotypic information about drugs and built an SVM classifier for each ADR. FS-MLKNN is a method that simultaneously determines critical feature dimensions and builds multi-label prediction models. A FS-MLKNN model is composed of five MLKNN models constructed from a selected subset of features selected using a genetic algorithm. In the learning step, the LNSM-SMI method generates K similarity matrices from K different data sources and combines them using θ_i weights (for all $1 \leq i \leq K$), while the LNSM-CMI learns the LNSM independently on each data source. LNSM is itself a method that can train models and make predictions based on single-source data, and takes the assumption that a data point (i.e., drug) can be optimally reconstructed by using a linear combination of its neighbours. Because of this, LNSM methods usually require a large number of neighbours to deliver better results. Both LNSM-SMI and LNSM-CMI are formulated as convex optimisation problems using the similarity between drugs to later make predictions. On the other hand, KG-SIM-PROP [27] proposes to exploit a graph structure built from the similarity matrix of drugs to propagate ADR labels from one drug to other drugs in its neighbourhood. Later we will see that such propagation, unlike LNSM-based methods, requires a smaller number of neighbours to deliver efficient predictions. KG-SIM-PROP has been modified to not limit the number of predictions as stated in [27], and adopt the evaluation protocol defined for this review, ensuring a fair comparison with the other models.

A comparative review of existing multi-source machine learning models and selected off-the-shelf multi-label learning models trained on knowledge graphs allows for assessing not only the performance, but also the flexibility of the various approaches. The performance of the off-the-shelf methods can also be used as a baseline for more focused experiments in ADR prediction, which is something that has been missing before. An additional contribution of this review is the analysis of the model performance not only on the hand-crafted feature sets employed by existing approaches, but also on drug features automatically extracted from knowledge graphs (cf. supplemental material Section E). This is to demonstrate the feasibility of this particular approach to increasing practical applicability of automated ADR prediction pipelines.

We perform a comparison of the above models (Liu's method, FS-MLKNN, LNSM-SMI, LNSM-CMI, KG-SIM-PROP, Decision Trees, Random Forests, Multi-Layer Perceptron, Linear Regression) in terms of performance based on several multi-label ranking evaluation metrics. All models are given a design matrix $\mathbf{X}$ with binary features as input, where the row i of $\mathbf{X}$ represents drug-i using a vector $\boldsymbol{x}_i$ (for $1 \leq i \leq N$) with a 1 or 0 in column j to indicate whether drug-i has feature j (for $1 \leq j \leq d$) or not, respectively. In the same way, labels are represented using a binary matrix $\mathbf{Y}$, where row i contains either 1 or 0 in column j indicating whether drug-i has ADR-j or not, respectively. For instance, considering the following three features: ($j = 0$) *enzyme* P05181, ($j = 1$) *indication* abdominal cramp (C0000729), and ($j = 2$) *pathway* hsa00010, we can have the vector $\boldsymbol{x}_1 = [1, 0, 1]$ for the drug *Fomepizole (DB01213)*, meaning that Fomepizole interacts with enzyme P05181; is not used to treat abdominal cramps; and is part of pathway hsa00010.

In Fig. 2 we show a typical flowchart for the processes of training, testing, and evaluating machine learning models. For a given model, its output is used to generate ADR predictions. These predictions are evaluated using Liu's, SIDER 4, and Aeolus data sets as gold standards.

Most of the reviewed models work directly with the drug feature matrices. However, two models, namely, KG-SIM-PROP and kNN, require a similarity graph as input, which in this case is generated from the similarity between drugs using either the original data sets features or the Bio2RDF knowledge graph. Such a similarity graph encodes the similarity relations between drugs, where the value of the i-th row with the j-th column is the similarity score between drug-i and drug-j. In supplemental material Section A we describe a method to generate such similarity network of drugs from a knowledge graph.

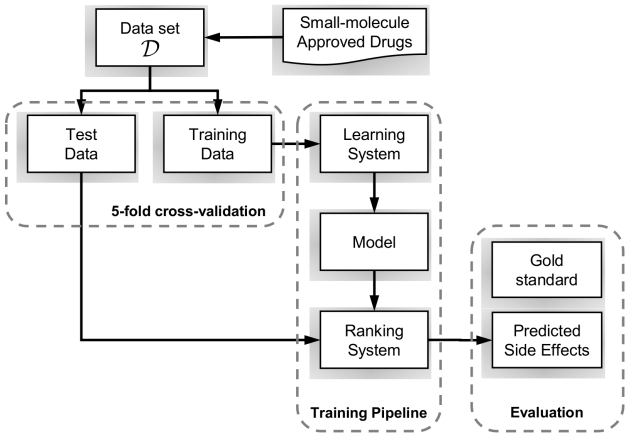

Fig. 2. Machine learning flowchart for training and testing of a model.

4 Results and Discussion

4.1 Experimental Configuration and Evaluation Metrics

All five multi-label learning models plus KG-SIM-PROP were implemented using the Scikit-Learn Python package [43] (`http://scikit-learn.org/stable/`), whereas, when available, we use the implementations provided by the reviewed methods. (General details on training and using the models are provided in the supplemental material Section B.) In many cases, we used the default hyper-parameters values as our main focus was to compare the performance of different models and not to find the optimal hyper-parameter settings for each of them. Some specific hyper-parameters, however, proved to have an obvious impact on the model results, and therefore we changed some of the default values and performed limited hyper-parameter optimisation via grid search. In particular: (a) The KG-SIM-PROP model uses the 3w-Jaccard similarity metric [44], with 10 up to 100 neighbours size; (b) The kNN model is tested with 10 up to 100 neighbours, with uniform and distance based weights using the Minkowski, Manhattan, Chebyshev, Jaccard, and Rogers Tanimoto distance metrics [44]; (c) The decision trees and random forests models use the mean squared error criterion (MSE), which is the only one supporting multi-label learning; (d) The multi-layer perceptron model is set with a unique hidden layer with 64, 128, 256, and 512 hidden units, a batch size equals to the 20% of drugs (which was chosen from an independent grid search), a logistic sigmoid activation, and the Adam solver; (e) The logistic regression model uses a L_2 penalty function, $C = 1.0$, Stochastic Average Gradient as solver, and 200 maximum iterations.

To compare all the models we adopt common metrics for ranking in multi-label learning [40]. We also compute example-based ranking metrics [40] used in related works, namely, One-error, Coverage (Cov-error), Ranking Loss (R-loss), and Average Precision (AP). Summary and details of all metrics we use are given in the supplemental material Section C. The performance of all models is evaluated using a 5-fold cross-validation. First, all drugs are randomly split into five equal sized subsets. Then, for each of the k folds, one part is held-out for testing and the learning algorithm is trained on the remaining four parts. In this way, all parts are used exactly once as validation data. The selection of the best hyper-parameters for each model is performed in each fold on the training set during the 5-fold cross-validation, and the best model is applied over the test set for validation (cf. supplemental material Section G). The five validation results are then averaged over all rounds. We also use common evaluation metrics for ranking systems, the Area Under the Receive Operator Curve (AUC-ROC) and the Area Under the Precision-Recall Curve (AUC-PR)[4] to evaluate the models, because they can be used to evaluate models regardless of any threshold. However, because of the existing unbalance of the labels (i.e., an ADR is more commonly found as a negative value than as a positive one among drugs), the AUC-PR gives a more informative picture of the model's performance [45]. Thus, we set the AUC-PR as our target metric (in the grid searches) for each of the rounds. Additionally, we compute other example-based metrics [40, 46], namely, Average Precision, One Error, Coverage Error, and Ranking Loss. The last type of measures we use are the general ranking evaluation metrics Hits at K (Hits@K), and Precision at K (P@K). Among the measures we used, the Hits@K and P@K are arguably the most accurate scores in terms of evaluating the benefit of ADR discovery for certain types of end-users like clinical practitioners. As explained in [47], these scores are easily grasped by non-informaticians and are therefore apt for explaining the reliability of the system to them. Moreover, in settings where quick decisions are needed, like in clinical practice, users do not tend to perform comprehensive search among many possible alternatives to find the relevant ones [47]. The Hits@K and P@K scores reflect the likelihood that such users will find relevant results very quickly at the top of the list of possibly relevant results.

4.2 Comparison on Liu's data set

In this section, we present the evaluation of all methods using Liu's data set, which includes multi-source data with different types of features about drugs. Specifically, we compare the methods considering the features and labels in Liu's data set, which was introduced in [24] and has been considered as a benchmark in [25, 28].

We compare the results reported in [28] for four existing methods (Liu's method, FS-MLKNN, LNSM-SMI, and LNSM-CMI) with the KG-SIM-PROP [27] and the five off-the-shelf multi-label learning models selected by us. Table 4 shows the values of evaluation metrics for each model, highlighting the best performing methods per metric in bold. We found out that the methods FS-MLKNN, LNSM-SMI and LNSM-CMI proposed by Zhang et al. very recently [25, 28] perform best on Liu's data set . The multi-layer perceptron, comes second by a rather small margin in all but one metric. The methods FS-MLKNN [25], LNSM-SMI and LNSM-CMI [28] exploit the notion of drug-drug similarity for propagating side effects from a drug to its neighbours. A similar approach is followed by the KG-SIM-PROP and kNN models, which can be considered a simplified version of the ones presented in [28]. The difference between the KG-SIM-PROP and kNN methods and the FS-MLKNN, LNSM-SMI and LNSM-CMI methods is that the last three require large numbers of neighbours to work properly (400 as reported in [25, 28]). while the KG-SIM-PROP and kNN methods can work with as few as 30 neighbours. This makes them more applicable to sparse data sets. As hypothesised by the authors [28], the better results of LNSM-SMI and LNSM-CMI may be attributed to their consideration of neighbourhood as an optimisation problem via the linear neighbourhood similarity used. This is confirmed by the observed results and leads to better accuracy in the similarity computation, but at the cost of efficiency because of the neighbourhoods generation. The benefits of treating the similarity as an optimisation problem are also shown in the competitive results of multi-layer perceptron, where a logistic sigmoid function was used as kernel. On the other hand, KG-SIM-PROP and kNN employ widely used off-the-shelf similarity metrics between feature vectors to determine the neighbourhoods. Methods that do not consider a similarity, namely, decision trees, random forests, and linear regression, are among the worst-performing methods. In terms of efficiency, we report that FS-MLKNN was the slowest method with more than

[4] As defined by Davis and Goadrich in [45], when dealing with highly skewed data sets.

two weeks running time on a single machine with commodity hardware. This is mainly due to its multiple feature selection steps based on genetic algorithms. From the multi-label ranking methods, the slowest was kNN with 13 hours and 18 minutes, followed by linear regression with 9 hours and 26 minutes. Both multi-layer perceptron and KG-SIM-PROP took ca. 2 hours and 16 minutes, while the decision trees were the fastest with only 16 minutes. We can see that even the slowest among multi-label learning models we have tested is orders of magnitude faster than the best performing previously published method. This is important information in the context of applicability of different models that is not obvious from previously published work. In cases where quick experimentation with different data sets or model parameters is required, the multi-layer perceptron may very well be the best choice as its results are very close to the best performance of existing tools.

In addition to the metrics reported in previous works, we report the ranking performance of the multi-label learning to rank methods in Table 5. Results show that multi-layer perceptron gives the best rankings across all metrics. This may indicate that non-linear methods (such as deep neural nets) are better suited to the ADR prediction problem. Deep learning methods have shown to excel in applications where there is an abundance of training data, and sources such as Bio2RDF could serve for this purpose. The use of deep learning methods for the prediction of ADRs is still an open problem. Further studies in this area may lead to significant performance improvements as indicated by the preliminary results presented in this review.

4.3 Comparison on the Bio2RDF data set

Several authors have found that combining information from different sources can lead to improved performance of computational approaches in bioinformatics (see [48, 49] among others). In Section 2 we introduced the Bio2RDF data set, which is a multi-source knowledge graph. An important aspect of increasing the practicality of ADR prediction we suggest in this review is automation of the feature extraction process. A possible way of doing it is to use heterogeneous knowledge graphs to represent entities such as drugs. This makes experimentation with different feature sets easier than with the existing reviewed works. To show the benefits of combining diverse data sources in terms of performance, we tested the multi-label learning models against two versions of the Bio2RDF data set: (v1) containing DrugBank and SIDER, and (v2) containing DrugBank, SIDER and KEGG. Table 6 shows the performance of six multi-label learning methods[5] using the set of 832 drugs and 1,385 side effects from Liu's data set, but replacing the feature vectors of drugs with those extracted from the Bio2RDF v1 (resp. Bio2RDF v2) data set. Originally, Liu's data set contained a set of 2,892 manually integrated features coming from six sources. These are replaced by 30,161 and 37,368 features in Bio2RDF v1 and v2, respectively. Both sets are automatically generated using the method described in supplemental material Section A, and represent a drug according to its incoming and outgoing relations with other entities in the knowledge graph.

Results show that in both cases (Bio2RDF v1 and v2) the methods perform better with the Bio2RDF features than with Liu's original data set features, confirming our assumption that combination of various feature sources may increase the performance. This can be explained by the fact that Bio2RDF provides a richer representation of drugs and their relationships than the traditional feature sets. This is an important finding, as the Bio2RDF features can be constructed automatically, while the features in the Liu's and Zhang's data sets require non-trivial manual efforts. Furthermore, our results also indicate that having extra information about pathways provides better performance as shown in Table 7, where Bio2RDF v2 is built by adding KEGG data set [33] to Bio2RDF v1. To further explore the influence of possible feature set combinations on the results, we integrated the original Liu's data set [24] features with Bio2RDF v2, leading to 40,260 features in total. Table 8 shows the performance results obtained when combining feature sets from Liu's and Bio2RDF v2 data sets. This yields slightly better results in terms of the Average Precision and AUC-PR metrics.

4.4 Comparison on the SIDER 4 data set

To further evaluate the practical applicability of the multi-label learning models, we performed an experiment using SIDER 4 [25]. The intuition behind this experiment is to test the predictive power of the models under a simple train and test setup. SIDER 4 data set contains 771 drugs used for training, which are also present in Liu's data set, and 309 newly added drugs used for testing. First, we run all methods on the original SIDER 4 data set features and labels, and compare them against the results provided by Zhang et al. [28]. Table 9 shows the results of the different methods over the SIDER 4 data set. The state-of-the-art method LNSM-SMI gives the best Average Precision and AUC-PR while LNSM-CMI produces the best Coverage Error. However, multi-layer perceptron is the best performing model in the AUC-ROC, Ranking Loss, and One-Error metrics. These results suggest better relative suitability of some multi-label learning methods for applications where a ranking function is preferred over classification. Examples of such applications are use cases where experts can only review a few prediction candidates and need the relevant ones to appear at the top of the list. Such use cases are indeed quite realistic, as there are often hundreds of predictions for every single drug. The results of multi-layer perceptron show some improvements when using features coming from the Bio2RDF v2 data set (cf. Table 10).

4.5 Comparison on the SIDER4 and Aeolus data sets

We further evaluate the models considering both the SIDER 4 and Aeolus data sets [35]. Aeolus data set provides us with relations between drugs and ADRs that were not previously known during the training or testing steps. The reason for the experiments using the SIDER 4 and Aeolus data sets is the evolving nature of the knowledge about drugs – generally, new ADRs can always be discovered for a drug, either by new studies or via pharmacovigilance in the post-market stage. The classic approach for validating ADR predictions follows the Closed World Assumption (i.e., missing predictions are false), but the actual problem follows the Open World Assumption (i.e., missing predictions may be just unknown). Therefore it is always possible that predictions that are currently deemed false positives can be considered true positives if more knowledge becomes available in the future. We hope to reflect this phenomenon by using the complementary Aeolus data that is very frequently updated and contains information based on manually validated reports. For these reasons, we believe it will be beneficial to use this data set for complementary validations also in future studies in this domain.

To test this point, we updated the SIDER 4 matrix $\mathbf{Y}$ of ADRs of the test set using a version of Aeolus data set generated after the release of the SIDER 4 data set. We found 142 drugs in the intersection of the SIDER 4 test set and Aeolus. Whenever a new drug-ADR relationship is reported in the Aeolus data set for any of the 309 drugs in the test set, this is reflected by modifying the SIDER 4 data set. Aeolus introduces 615 new ADR relations in total with an average of 4.3 per drug. For example, Aeolus provides two new ADRs for *Triclosan (DB08604)*, an aromatic ether widely used as a preservative and antimicrobial agent in personal care products: *odynophagia* and *paraesthesia oral*. While these changes due to the Aeolus data set are not crucial for drugs with many previously known

[5] Unfortunately, there were no implementations available for LNSM-SMI, LNSM-CMI [28] for comparison at the time of this writing. And FS-MLKNN was discarded due to its intractability on larger feature sets.

ADRs[6], they can have high impact on drugs with very few known ADRs (such as *Triclosan* or *Mepyramine* both with only 1 ADR). In total, Aeolus provides at least one new ADR for 46% of drugs in the SIDER 4 test set. Interestingly, most of the new ADRs added by Aeolus data set are related to the digestive system (e.g., intestinal obstruction, gastric ulcer, etc.), which we believe is due to the disproportionate FAERS reporting [8, 10] frequency for this type of events.

We ran the models once more and evaluated them against the new gold standard with the updates provided by the Aeolus data set. Table 11 shows the results of the updated data set using the Aeolus data for the four best performing multi-label models, and when compared against values in Table 9 results are marginally lower across all metrics. For instance, the Average Precision of multi-layer perceptron drops by 0.92%, and AUC-ROC by 1.85%. This observation is not consistent with our assumption that new knowledge about relations between drugs and ADRs can increase the true positive rate by confirming some of the previous false positives as being actually true. We believe that this could be due to two reasons. (A) The added ADRs are under-represented across drugs. We observed this in SIDER 4 where 37.5% (2,093 out of 5,579) of ADRs are present at most once in either the training or test set. This makes those ADRs hard to predict. (B) There is a "weak" relation between the drugs and the introduced ADRs. This weak relation comes from the original split in training and test set provided in SIDER 4 data set; we found out that 50.15% (2,798 out of 5,579) ADRs are only present in the training set and not in the test set, compared to a 7% (392 out of 5,579) of ADRs that are only present in the test set.

Advantages of using Aeolus data set are illustrated for example by the drug *Eribulin (DB08871)* that contains 123 ADRs in SIDER 4, most of which have been discovered in the post-marketing stage. Aeolus introduced 7 new ADRs for *Eribulin*, where one of them, namely, *Pharyngitis (C0031350)*, was ranked number 36 among all 5,579 ADRs, which is a high ranking considering the total of 123 ADRs. This means the models are able to perform well for reactions that are true based on the recent data in Aeolus, but not present among positives in the primary validation data like SIDER (and thus they could only be interpreted as false positives during the primary evaluation). Such encouraging results were observed on several of the analysed drugs for which predictions previously considered as false positives were indeed shown to be true by Aeolus.

All analysed methods consider a static view over the data, and do not consider the changes in data, e.g., new ADRs discovered in a post-marketing stage. Therefore, a future research direction could study the effects of learning under evolving data sets (i.e., new drug-ADR relations), which is known as incremental learning (see [50–52] among others).

4.6 Comments on the Behaviour of the Models

To illustrate the flexibility and robustness of the approach we suggest to complement the existing predictive models, we enriched the Liu's data set using Bio2RDF data set features, which in general are numerous. Intuitively, by having more features for a drug, we can achieve a better representation of it, which should lead to better performance results. However, we observed mixed small positive and negative changes in the results shown in Table 8 when compared to the performance previously reported in Tables 6 and 7. This can be attributed to the famous curse of dimensionality, where the performance degrades as a function of dimensionality. This issue may have large impact on models like multi-layer perceptron where the large number of inputs hampers the training performance if the first hidden layer is too small. This is the case of our experiments, as we limit the size of the first hidden layer for the multi-layer perceptron. However, it is possible to cope with the curse of dimensionality, using methods such as embeddings into low rank feature spaces. Embedding models aim to find a dimensionality reduction, generating latent representations of the data that preserve structural details as much as possible [53]. This is something that represents a new research direction, by considering learning of drug representations for tasks such as comparison. We believe this could substantially improve the performance of some of the models here reviewed.

We also observed that when merging Liu's data set with Bio2RDF, some features can be considered as duplicated features. Certain models deal with this situation better, and others would apparently require a filtering of duplicated features. During our experiments, we did not filter features, and assumed that deduplication is performed by the models.

Regarding scalability, despite the substantial increase of the feature space (up to almost 13-fold), we only noticed up to double execution times of the multi-label learning methods. All running times are still far better than the time required by the previously existing methods, which is another argument for higher practical applicability of the suggested approach.

5 Conclusion

We have shown that using knowledge graphs for automated feature extraction and casting the problem of ADR prediction as multi-label ranking learning can be used for building models that are comparable to or better than existing related methods. Moreover, the off-the-shelf models are orders of magnitude faster than existing related ADR prediction systems. We argue that due to the demonstrated speed-up and automation of most of the steps in building the prediction pipelines, this review provides a broad range of possibilities for biomedical experts to build their own ADR prediction systems more easily than before. This is supported by extensive documentation of all necessary steps provided in the article (cf. supplemental material).

The applicability of some of the reviewed models is further supported by very good results in ranking metrics. This can be useful in many practical scenarios where experts cannot possibly explore all computed predictions, but require ranked results and highly relevant candidates appearing at the top of the list. Last but not least, the review presents results of the off-the-shelf machine learning modules in a way that can be used as a well-documented baseline for future experiments in this area.

In our future work, we want to investigate the influence of embeddings (i.e., latent feature models and feature extractors) on the performance of multi-label learning models for the ADR prediction. We also want to analyse the influence of various hyper-parameters on the prediction results more thoroughly. This will bring more insight into the most promising directions for further improvements of the performance of ADR prediction models. Another area we want to target is development of more stratified and comprehensive benchmark data sets that could increase the interpretability of ADR prediction results in future studies. Last but not least, we would like to perform not only quantitative validation, but also qualitative trials with actual sample users. This will let us assess the real-world usability of the reviewed approaches and gain valuable feedback for further developments in this field.

[6] For instance, *Nilotinib (DB04868)* has 333 ADRs in SIDER 4 and Aeolus only adds 3 new ADRs.

Table 4. Predictive power of the models using Liu's data set. For each metric, we report the standard deviation values (when available). The values for the first four models were taken from [28]. The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
Liu's method [24]	0.2610	0.2514	0.8850	0.0927	0.9291	837.4579
FS-MLKNN [28]	0.5134	0.4802	**0.9034**	0.0703	0.1202	795.9435
LNSM-SMI [28]	**0.5476**	**0.5053**	0.8986	0.0670	**0.1154**	789.8486
LNSM-CMI [28]	0.5329	0.4909	0.9091	**0.0652**	0.1250	**776.3053**
KG-SIM-PROP [27]	0.4895±0.0058	0.4295±0.0078	0.8860±0.0075	0.1120±0.0139	0.1610±0.0164	1100.9985±65.8834
kNN	0.5020±0.0078	0.4417±0.0081	0.8892±0.0085	0.1073±0.0053	0.1538±0.0181	1102.3548±41.4641
Decision Trees	0.2252±0.0137	0.1989±0.0181	0.6634±0.0316	0.6519±0.0242	0.5493±0.0374	1377.1316±8.3936
Random Forests	0.4626±0.0163	0.4331±0.0261	0.8342±0.0218	0.2525±0.0176	0.2007±0.0154	1284.3111±27.0454
Multi-Layer Perceptron	0.5196±0.0069	0.4967±0.0204	0.9003±0.0057	0.0874±0.0009	0.1454±0.0166	954.0372±22.2870
Linear Regression	0.2854±0.0088	0.2595±0.0196	0.6724±0.0232	0.6209±0.0137	0.4267±0.0103	1380.0763±4.0209

Table 5. Ranking performance of the models using Liu's data set. The evaluation metrics are P@X (precision at 3, 5, and 10), and HITS@X (hits at 1, 3, 5, and 10). (For all metrics, the higher the value of the metric the better.)

Model	Evaluation Criterion						
	P@3	P@5	P@10	HITS@1	HITS@3	HITS@5	HITS@10
KG-SIM-PROP [27]	0.9333±0.1333	0.8400±0.2332	0.9200±0.1166	0.8390±0.0164	2.4351±0.0240	3.8691±0.0671	7.0734±0.0746
kNN	0.9333±0.1333	0.9200±0.0980	0.9400±0.0800	0.8450±0.0173	2.4568±0.0316	3.9027±0.0452	7.1744±0.0581
Decision Trees	0.4667±0.2667	0.4400±0.2653	0.4800±0.1470	0.4171±0.0176	1.1971±0.0570	1.9651±0.0940	3.8076±0.1941
Random Forests	0.9333±0.1333	0.9200±0.0400	0.9200±0.0400	0.8101±0.0088	2.3353±0.0594	3.7451±0.0779	6.9434±0.0982
Multi-Layer Perceptron	**1.0000±0.0000**	**0.9600±0.0800**	**0.9600±0.0490**	**0.8546±0.0166**	**2.4676±0.0295**	**3.9773±0.0544**	**7.3633±0.1451**
Linear Regression	0.3333±0.2981	0.4000±0.1265	0.4400±0.1347	0.5745±0.0469	1.6262±0.0716	2.6394±0.0782	5.1851±0.0823

Table 6. Predictive power of the models using drugs in Liu's data set with features from Bio2RDF v1 (DrugBank + SIDER). The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
KG-SIM-PROP [27]	0.5011±0.0106	0.4485±0.0115	0.8935±0.0096	0.1058±0.0122	0.1586±0.0177	1095.3082±55.47904
kNN	0.4977±0.0107	0.4210±0.0228	0.8848±0.0062	0.1211±0.0113	0.1658±0.0206	1127.7254±45.6342
Decision Trees	0.1964±0.0116	0.1710±0.0138	0.6301±0.0250	0.7220±0.0194	0.5673±0.0144	1377.2001±6.9189
Random Forests	0.4317±0.0107	0.3843±0.0143	0.8097±0.0102	0.3037±0.0088	0.2212±0.0139	1314.5006±17.6714
Multi-Layer Perceptron	**0.5099±0.0159**	**0.4546±0.0169**	**0.9010±0.0061**	**0.0791±0.0022**	**0.1430±0.0160**	**892.8340±20.4758**
Linear Regression	0.2847±0.0083	0.2482±0.0137	0.6404±0.0248	0.6726±0.0141	0.3467±0.0238	1383.3808±3.2383

Table 7. Predictive power of the models using drugs in Liu's data set with features from Bio2RDF v2 (DrugBank + SIDER + KEGG). The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
KG-SIM-PROP [27]	0.5118±0.0101	0.4604±0.0097	0.8954±0.0054	0.1051±0.0109	0.1466±0.0214	1091.9749±51.4537
kNN	0.5083±0.0124	0.4341±0.0277	0.8835±0.0086	0.1281±0.0031	0.1478±0.0027	1155.2053±36.5165
Decision Trees	0.2069±0.0176	0.1742±0.0266	0.6258±0.0242	0.7140±0.0233	0.5469±0.0385	1370.7402±7.5913
Random Forests	0.4438±0.0162	0.3993±0.0256	0.8153±0.0171	0.2883±0.0225	0.2103±0.0169	1295.7516±20.2287
Multi-Layer Perceptron	**0.5278±0.0106**	**0.4725±0.0284**	**0.9002±0.0074**	**0.0795±0.0028**	**0.1322±0.0298**	**909.7297±19.7920**
Linear Regression	0.2919±0.0109	0.2587±0.0165	0.6441±0.0261	0.6665±0.0166	0.3557±0.0306	1383.3796±3.2407

Table 8. Predictive power of the models using a combination of features from both Liu's data set and Bio2RDF v2 data set. The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
KG-SIM-PROP [27]	0.5012±0.0079	0.4471±0.0097	0.8882±0.0089	0.1184±0.0139	0.1526±0.0177	1127.3234±51.2769
kNN	0.5020±0.0808	0.4482±0.0101	0.8883±0.0089	0.1184±0.0139	0.1502±0.0208	1127.1279±51.3701
Decision Trees	0.2080±0.0190	0.1728±0.0149	0.6306±0.0239	0.6944±0.0215	0.5444±0.0289	1372.1095±9.6089
Random Forests	0.4609±0.0174	0.4331±0.0127	0.8357±0.0117	0.2627±0.0134	0.1995±0.0241	1308.7285±24.9798
Multi-Layer Perceptron	**0.5281±0.0088**	**0.4870±0.0269**	**0.8946±0.0067**	**0.0835±0.0034**	**0.1418±0.0158**	**937.8773±36.9387**
Linear Regression	0.3031±0.0108	0.2681±0.0169	0.6578±0.02424	0.6431±0.0147	0.3617±0.0273	1381.7218±4.0156

Table 9. Predictive power of the models using SIDER 4 data set. The values for the first four models were taken from [28]. The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
Liu's method [24]	0.1816	0.1766	0.8772	0.1150	0.9870	1587.5663
FS-MLKNN [28]	0.3649	0.3109	0.8722	0.1038	0.1851	1535.9223
LNSM-SMI [28]	**0.3906**	**0.3465**	0.8786	0.0969	0.2013	1488.2977
LNSM-CMI [28]	0.3804	0.3332	0.8852	0.0952	0.1916	**1452.7184**
KG-SIM-PROP [27]	0.3375	0.2855	0.8892	0.1398	0.2233	4808.3689
*k*NN	0.3430	0.2898	0.8905	0.1392	0.2168	4086.0777
Random Forests	0.3004	0.2599	0.8235	0.3318	0.2848	5362.6117
Multi-Layer Perceptron	0.3546	0.2899	**0.8943**	**0.0922**	**0.1309**	4054.0356

Table 10. Predictive power of the models using drugs in SIDER 4 data set and Bio2RDF v2 data set features. The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
KG-SIM-PROP [27]	0.3438	0.2876	0.8764	0.17460	0.2427	4969.0647
*k*NN	0.3416	0.2835	0.8728	0.1777	0.2395	5002.6084
Random Forests	0.2384	0.2061	0.7651	0.4567	0.4304	5440.0712
Multi-Layer Perceptron	0.3529	0.2857	0.9043	0.0852	0.1909	3896.3625

Table 11. Predictive power of the models using SIDER 4 data set, and updating the ADRs with Aeolus data set. The evaluation metrics are AP (Average Precision), AUC-PR (Area Under the Precision-Recall Curve), AUC-ROC (Area Under the Receive Operator Curve), R-loss (Ranking Loss), One-Error (One Error), and Cov-Error (Coverage Error). ("↑" indicates that the higher the metric value the better, and "↓" indicates that the lower the metric value the better.)

Model	Evaluation Criterion					
	AP ↑	AUC-PR ↑	AUC-ROC ↑	R-Loss ↓	One-Error ↓	Cov-Error ↓
KG-SIM-PROP [27]	0.3272	0.2791	0.8796	0.1619	0.2233	5040.06149
*k*NN	0.3324	0.2834	0.8808	0.1613	0.2168	5038.6570
Random Forests	0.2883	0.2447	0.8059	0.3717	0.3366	5478.8479
Multi-Layer Perceptron	0.3437	0.2836	0.8858	0.1050	0.1909	4339.7540

Key Points

- Knowledge graphs allow for easy, automated integration of multiple diverse datasets in order to extract features for ADR prediction.
- Approaching the ADR prediction as a multi-label learning problem facilitates easy experimentation with a diverse range of off-the-shelf algorithms. It also produces results that can be used as a well-documented baseline for future, more sophisticated experiments.
- Applying these two principles (i.e., knowledge graphs and multi-label learning) leads to results that are comparable to or better than existing related approaches, while the training is orders of magnitude faster on the same data. Also, the resulting models provide ranked predictions by default, which further contributes to their practical applicability.
- Interested stakeholders can straightforwardly use the review for building their own ADR prediction pipelines and fine-tuning them based on their specific requirements (such as increasing particular classification or ranking performances).

Acknowledgements

The authors kindly acknowledge Pasquale Minervini for contributing to the Python implementation in an early stage of this work. The authors thank the three anonymous reviewers for their valuable comments and suggestions that helped us improve the manuscript.

Funding

This work was supported by the TOMOE project funded by Fujitsu Laboratories Ltd., Japan and Insight Centre for Data Analytics at National University of Ireland Galway (supported by the Science Foundation Ireland grant 12/RC/2289). The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

Availability of Data and Material

We make available all design matrices with drug features and labels (ADRs) as MATLAB files. All data files are available for download at `http://purl.com/bib-adr-prediction`. Further details on the feature extraction step and manipulation of data sets are provided in the supplemental material.

References

1. Bowes J, Brown AJ, Hamon J, *et al.* Reducing safety-related drug attrition: the use of in vitro pharmacological profiling. Nature Reviews Drug discovery. 2012;11:909–922.
2. Sultana J, Cutroneo P, Trifiró G, *et al.* Clinical and economic burden of adverse drug reactions. Journal of Pharmacology and Pharmacotherapeutics. 2013;4:73–77.
3. Bouvy JC, De Bruin ML, Koopmanschap MA. Epidemiology of adverse drug reactions in Europe: a review of recent observational studies. Drug Safety. 2015;38:437–453.

4. Edwards IR, Aronson JK. Adverse drug reactions: definitions, diagnosis, and management. The Lancet. 2000;356:1255 – 1259.
5. Giacomini KM, Krauss RM, Roden DM, *et al.* When good drugs go bad. Nature. 2007;446:975–977.
6. Johnson J, Booman L. Drug-related morbidity and mortality. Journal of Managed Care Pharmacy. 1996;2:39–47.
7. Kola I, Landis J. Can the pharmaceutical industry reduce attrition rates? Nature Reviews Drug discovery. 2004;3:711–716.
8. Szarfman A, Machado SG, O'Neill RT. Use of screening algorithms and computer systems to efficiently signal higher-than-expected combinations of drugs and events in the US FDA's spontaneous reports database. Drug Safety. 2002;25:381–392.
9. Mammadov MA, Rubinov AM, Yearwood J. The Study of Drug-reaction Relationships Using Global Optimization Techniques. Optimization Methods and Software. 2007 Feb;22:99–126.
10. Harpaz R, Vilar S, DuMouchel W, *et al.* Combing signals from spontaneous reports and electronic health records for detection of adverse drug reactions. Journal of the American Medical Informatics Association. 2013;20:413–419.
11. Karimi S, Wang C, Metke-Jimenez A, *et al.* Text and data mining techniques in adverse drug reaction detection. ACM Computing Surveys. 2015 May;47:56:1–56:39.
12. Ginsberg J, Mohebbi MH, Patel RS, *et al.* Detecting influenza epidemics using search engine query data. Nature. 2009;457:1012–1014.
13. White RW, Wang S, Pant A, *et al.* Early identification of adverse drug reactions from search log data. Journal of Biomedical Informatics. 2016;59:42–48.
14. Tan Y, Hu Y, Liu X, *et al.* Improving drug safety: From adverse drug reaction knowledge discovery to clinical implementation. Methods. 2016;110:14 – 25. Protein-Protein Interactions Bioinformatics.
15. Campillos M, Kuhn M, Gavin AC, *et al.* Drug target identification using side-effect similarity. Science. 2008;321:263–266.
16. Vilar S, Hripcsak G. The role of drug profiles as similarity metrics: applications to repurposing, adverse effects detection and drug–drug interactions. Briefings in Bioinformatics. 2016;p. bbw048.
17. Atias N, Sharan R. An algorithmic framework for predicting side effects of drugs. Journal of Computational Biology. 2011;18:207–218.
18. Pauwels E, Stoven V, Yamanishi Y. Predicting drug side-effect profiles: a chemical fragment-based approach. BMC Bioinformatics. 2011;12:169.
19. Bresso E, Grisoni R, Marchetti G, *et al.* Integrative relational machine-learning for understanding drug side-effect profiles. BMC Bioinformatics. 2013;14:1.
20. Jahid MJ, Ruan J. An ensemble approach for drug side effect prediction. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine. IEEE; 2013. p. 440–445.
21. Mizutani S, Pauwels E, Stoven V, *et al.* Relating drug–protein interaction network with drug side effects. Bioinformatics. 2012;28:i522–i528.
22. Yamanishi Y, Pauwels E, Kotera M. Drug side-effect prediction based on the integration of chemical and biological spaces. Journal of Chemical Information and Modeling. 2012;52:3284–3292.
23. Huang LC, Wu X, Chen JY. Predicting adverse drug reaction profiles by integrating protein interaction networks with drug structures. Proteomics. 2013;13:313–324.
24. Liu M, Wu Y, Chen Y, *et al.* Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs. Journal of the American Medical Informatics Association. 2012;19:e28–e35.
25. Zhang W, Liu F, Luo L, *et al.* Predicting drug side effects by multi-label learning and ensemble learning. BMC Bioinformatics. 2015;16:1.
26. Zhang W, Zou H, Luo L, *et al.* Predicting potential side effects of drugs by recommender methods and ensemble learning. Neurocomputing. 2016;173:979–987.
27. Muñoz E, Novacek V, Vandenbussche PY. Using drug similarities for discovery of possible adverse reactions. In: AMIA 2016, American Medical Informatics Association Annual Symposium. American Medical Informatics Association; 2016. p. 924–933.
28. Zhang W, Chen Y, Tu S, *et al.* Drug side effect prediction through linear neighborhoods and multiple data source integration. In: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM); 2016. p. 427–434.
29. Rahmani H, Weiss G, Méndez-Lucio O, *et al.* ARWAR: A network approach for predicting adverse drug reactions. Computers in Biology and Medicine. 2016;68:101–108.
30. Kuhn M, Letunic I, Jensen LJ, *et al.* The SIDER database of drugs and side effects. Nucleic Acids Research. 2016;44:D1075.
31. Kim S, Thiessen PA, Bolton EE, *et al.* PubChem substance and compound databases. Nucleic Acids Research. 2016;44:D1202.
32. Law V, Knox C, Djoumbou Y, *et al.* DrugBank 4.0: shedding new light on drug metabolism. Nucleic Acids Research. 2014;42:D1091–D1097.
33. Kanehisa M, Furumichi M, Tanabe M, *et al.* KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Research. 2017;45:D353–D361.
34. Belleau F, Nolin MA, Tourigny N, *et al.* Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. Journal of Biomedical Informatics. 2008 Oct;41:706–716.
35. Banda JM, Evans L, Vanguri RS, *et al.* A curated and standardized adverse drug event resource to accelerate drug safety research. Scientific Data. 2016 May;3.
36. Dumontier M, Callahan A, Cruz-Toledo J, *et al.* Bio2RDF release 3: A larger, more connected network of Linked Data for the Life Sciences. In: International Semantic Web Conference (Posters & Demos). vol. 1272 of CEUR Workshop Proceedings. CEUR-WS.org; 2014. p. 401–404.
37. Ritz A, Tegge AN, Kim H, *et al.* Signaling hypergraphs. Trends in Biotechnology. 2014;32:356–362.
38. Bisgin H, Liu Z, Fang H, *et al.* Mining FDA drug labels using an unsupervised learning technique - topic modeling. BMC Bioinformatics. 2011;12:S11.
39. Tsoumakas G, Katakis I. Multi-label classification: An overview. International Journal of Data Warehousing and Mining. 2006;3.
40. Zhang ML, Zhou ZH. A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering. 2014 Aug;26:1819–1837.
41. Zhang ML, Zhou ZH. Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering. 2006 Oct;18:1338–1351.
42. Zha ZJ, Mei T, Wang J, *et al.* Graph-based semi-supervised learning with multiple labels. Journal of Visual Communication and Image Representation. 2009;20:97 – 103.
43. Pedregosa F, Varoquaux G, Gramfort A, *et al.* Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research. 2011 Nov;12:2825–2830.
44. Choi Ss, Cha Sh, Tappert CC. A survey of binary similarity and distance measures. Journal on Systemics, Cybernetics and Informatics. 2010;0:43–48.
45. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning. ICML '06. New York, NY, USA: ACM; 2006. p. 233–240.

46. Tsoumakas G, Katakis I, Vlahavas I. In: Maimon O, Rokach L, editors. Mining multi-label data. Boston, MA: Springer US; 2010. p. 667–685.
47. Manning CD, Raghavan P, Schütze H. 8. In: Chapter 8: Evaluation in information retrieval. Cambridge University Press Cambridge; 2008. p. 151–175.
48. Polikar R. Ensemble based systems in decision making. IEEE Circuits and Systems Magazine. 2006 Third;6:21–45.
49. Yang R, Zhang C, Gao R, *et al.* An ensemble method with hybrid features to identify extracellular matrix proteins. PLOS ONE. 2015 02;10:1–21.
50. Schlimmer JC, Granger RH. Incremental Learning from Noisy Data. Machine Learning. 1986;1:317–354.
51. Rüping S. Incremental Learning with Support Vector Machines. In: ICDM. IEEE Computer Society; 2001. p. 641–642.
52. Raway T, Schaffer DJ, Kurtz KJ, *et al.* Evolving data sets to highlight the performance differences between machine learning classifiers. In: GECCO (Companion). ACM; 2012. p. 657–658.
53. Dai G, Yeung DY. Tensor embedding methods. In: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1. AAAI'06. AAAI Press; 2006. p. 330–335.
54. Barabási AL. Network science. Cambridge University Press; 2016.
55. Liu TY. Learning to rank for information retrieval. Springer Science & Business Media; 2011.

Supplemental Material

A Similarity Graph Construction

Intuitively, there are different ways to obtain a similarity graph, and here we explore a graph-based approach.

Bio2RDF is a directed graph where each node (e.g. a drug) can be characterised by its incoming and outgoing edges. In network sciences [54], the size of such sets determine the *out-degree* and *in-degree* of a node, that is, the number of links that point from the focus node to others, and the number of other nodes pointing to the focus node, respectively. In a knowledge graph, the sets of incoming and outgoing edges for a node u can be generated from the edges that match `(?,?,`u`)` and `(`u`,?,?)`, respectively, where `?` can take any value. Thus, the edge and source/target node labels matching the `?` spaces become the features of a given node. More formally, we represent the set of features of a drug in a knowledge graph $\mathcal{G}$ using two functions: Let $\Phi^{\downarrow}(\mathcal{G}, u) = \{(r, v, \text{'}in\text{'}) \mid \exists r, v(v, r, u) \in \mathcal{G}\}$ be the function that extracts the finite set of incoming relations r from v to u, and let $\Phi^{\uparrow}(\mathcal{G}, u) = \{(r, v, \text{'}out\text{'}) \mid \exists r, v(u, r, v) \in \mathcal{G}\}$ be the function that extracts the finite set of outgoing relations r from u to v. Hence, the function to extract features for nodes in the knowledge graph $\mathcal{G}$ is defined as:

$$\Phi(\mathcal{G}, u) = \Phi^{\downarrow}(\mathcal{G}, u) \cup \Phi^{\uparrow}(\mathcal{G}, u). \tag{1}$$

Example 1. *Consider the example knowledge graph in Fig. 3 with ten resources and relations ℓ_i, for $1 \leq i \leq 5$. Then resulting feature set for $\boldsymbol{a}$, i.e., $\Phi(\boldsymbol{a})$, consists of the following elements:*

$$\Phi(\mathcal{G}, \boldsymbol{a}) = \big\{(\ell_4, \boldsymbol{b}, \text{'}out\text{'}), (\ell_3, \boldsymbol{d}, \text{'}out\text{'}), (\ell_5, \boldsymbol{y}, \text{'}out\text{'}), (\ell_1, \boldsymbol{e}, \text{'}in\text{'}), (\ell_2, \boldsymbol{z}, \text{'}in\text{'}), (\ell_3, \boldsymbol{x}, \text{'}in\text{'}), (\ell_2, \boldsymbol{c}, \text{'}in\text{'})\big\}.$$

Similarly, we can extract the feature set for $\boldsymbol{x}$:

$$\Phi(\mathcal{G}, \boldsymbol{x}) = \big\{(\ell_2, \boldsymbol{z}, \text{'}in\text{'}), (\ell_4, \boldsymbol{v}, \text{'}out\text{'}), (\ell_3, \boldsymbol{w}, \text{'}out\text{'}), (\ell_5, \boldsymbol{y}, \text{'}out\text{'}), (\ell_3, \boldsymbol{a}, \text{'}out\text{'})\big\}.$$

Intuitively, the more features two nodes have in common, the more similar they are. Thus, we use the feature vectors of drugs to compute similarities. We use 3w-Jaccard [44] similarity to compute similarities between all drugs: assigning higher weight to common features, and lower weight to discriminant features, i.e. those only present in one drug. The 3w-Jaccard between two drugs u and v is defined as:

$$S_{\text{3W-JACCARD}}(u, v) = \frac{3x}{3x + y + z}, \tag{2}$$

where $x = |\Phi(u) \cap \Phi(v)|$, $y = |\Phi(u) - \Phi(v)|$, and $z = |\Phi(v) - \Phi(u)|$, with $0 \leq S_{\text{3W-JACCARD}} \leq 1$. The similarity metric takes into account all features returned by the feature extraction (see supplemental material Section D), including categorical (e.g., *"International-brand"/"Local-brand"*, *"Approved"/"Not Approved"/"Waiting for Approval"*) and non-categorical (e.g., numerical, free-text) values. (The method as it stands considers non-categorical values as a sequence of characters. Future research areas can go in the direction of studying another way to use non-categorical values.) In both cases, each feature becomes a component in the feature vectors used to compute similarity, where a feature takes value 1 or 0, indicating whether the drug contains or not that feature. Feature selection in a multi-label setting may ignore label dependencies, thus, we leave this decision up to each of the multi-label learning model.

Finally, for KG-SIM-PROP and kNN we construct the $\mathbf{W}$ similarity matrix using the 3w-Jaccard similarity between every pair of drugs: the similarity graph is represented as an adjacency matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ with $w_{ij} = w_{ji} = S_{\text{3W-JACCARD}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $w_{ii} = 0$. While for all the other methods, we use the extracted features to build the design matrix.

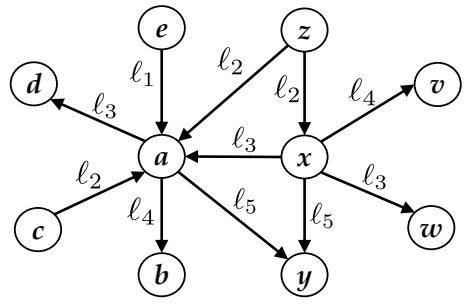

Fig. 3. Example knowledge graph, where labelled nodes $\boldsymbol{a}$ to $\boldsymbol{d}$ and $\boldsymbol{v}$ to $\boldsymbol{z}$, correspond to entities in the knowledge graph, and relationships are represented as edges between nodes with labels ℓ_i, $1 \leq i \leq 5$.

B Overview of the Reviewed Multi-Label Learning Models

In the following, we briefly describe the models studied in this article. All models learn a function $f(\boldsymbol{x}, Y)$ as the one introduced in Section 3.1. The only model that differs is Logistic Regression, in which we use binary classifiers to emulate the function $f(\boldsymbol{x}, Y)$.

Decision trees are a non-parametric supervised learning method used for classification and regression. In this paper, we use the regression capabilities. In both cases, the goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. In the ADRs prediction scenario, we want them to learn a drug's ADRs based on the biological features of the drug. Decision trees are interesting for their interpretability – the generated decision rules are represented as a tree, which can be visualised and can help understanding the decisions to assign ADRs to drugs. Another interesting feature of decision trees is the low cost required for training and prediction.

In our experiments, we used the implementation provided by Scikit-learn `sklearn.tree.DecisionTreeRegressor`.

Random forests is a ensemble method that combines the predictors of several base estimators built with decision trees algorithm. In random forests, a multitude of decision trees are build at training time and the output correspond to the mode of the classes (in the case of classification) or the mean prediction (in the case of regression) of the individual trees.

In our experiments, we used the implementation provided by Scikit-learn `sklearn.ensemble.RandomForestRegressor`.

Nearest Neighbours supports unsupervised and supervised learning functionality. Nearest neighbour methods find a predefined number of training samples closest in distance to the new point, and predict the label(s) from these. In our case, we use a user-defined constant k to select these neighbours, and the 3w-Jaccard distance metric measure. Neighbours-based methods are known as non-generalising machine learning methods, since they simply "remember" all training data.

For experimentation, we used the Scikit-learn implementation in `sklearn.neighbors.KNeighborsRegressor`.

Multi-layer perceptron is a supervised learning algorithm that learns a function by training on a data set. This function receives a number of dimensions for input, and returns a number of dimensions for output. Multi-layer perceptron is able to learn a non-linear function approximator for either classification or regression. It differs from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden-layers.

In our experiments, we used the Scikit-learn implementation in `sklearn.neural_network.MLPRegressor`.

KG-SIM-PROP is a similarity-based method that performs a constrained propagation of labels, based on the closest in similarity to the point. For one example, KG-SIM-PROP performs a truncated propagation of labels from the closest neighbours.

For experiments, we implemented the algorithm described in [27].

Logistic regression finds a linear combination of the input features, to perform a regression which divides the class space. In logistic regression, the probabilities describing the possible outcomes of a single example are modelled using a logistic function. It implements an optimisation problem, which we set to be solve using Stochastic Average Gradient descent.

In our experiments, we used the Scikit-learn implementation in `sklearn.linear_model.LogisticRegression`, and the strategy known as one-vs-all to generate one classifier per label, implemented in `sklearn.multiclass.OneVsRestClassifier`.

C Evaluation Metric Formulas

To evaluate the performance of the models we use specific evaluation metrics for multi-label learning, which are different from the ones used in traditional supervised learning [40]. Adopting the same notations used in Section 3.1, let $\mathcal{D}$ be our multi-label data set consisting of N multi-label examples (i.e., drugs) $(\boldsymbol{x}_i, Y_i)$, $1 \leq i \leq p$, $(\boldsymbol{x}_i \in \mathcal{X}, Y_i \in \mathcal{Y} = \{0,1\}^Q)$, with a label set $\mathcal{L}$, where $|\mathcal{L}| = Q$, i.e., the total number of side effects. Here, $f(\cdot,\cdot)$ represents the multi-label regressor and $f(\boldsymbol{x}_i, Y_i) = \{0,1\}^Q$ is the set of label (i.e., ADRs) memberships predicted by f for the example drug $\boldsymbol{x}_i$. We compute the following example-based ranking metrics for evaluating the results of the predictions [40, 46]:

One-Error: Evaluates the fraction of examples whose top-ranked label is not in the set of relevant labels of the instance.

$$One\text{-}Error(f) = \frac{1}{p}\sum_{i=1}^{p} I(\arg\max_{y\in\mathcal{Y}} f(\boldsymbol{x}_i, y) \notin Y_i),$$

where, I is an indicator function and $f(\boldsymbol{x}_i, y)$ is the score of label y for an instance $\boldsymbol{x}_i$.

Coverage Error: Evaluates how far we need, on average, to move down the ranked list of labels in order to cover all the relevant labels of the instance.

$$Cov\text{-}Error(f) = \frac{1}{p}\sum_{i=1}^{p} \max_{y\in Y_i} rank_f(\boldsymbol{x}_i, y) - 1.$$

Ranking Loss: Evaluates the fraction of reversely ordered label pairs, i.e. an irrelevant label is ranked higher than a relevant label.

$$R\text{-}Loss(f) = \frac{1}{p}\sum_{i=1}^{p} \frac{1}{|Y_i||\overline{Y_i}|}\big|\{(y', y'') : rank_f(\boldsymbol{x}_i, y') > rank_f(\boldsymbol{x}_i,\ y''), (y', y'') \in Y_i \times \overline{Y_i}\}\big|,$$

where $\overline{Y_i}$ is the complementary set of Y_i in $\mathcal{Y}$.

Average Precision: Evaluates the average fraction of relevant labels ranked higher than a particular label $y \in Y_i$ which actually are in Y_i.

$$AP(f) = \frac{1}{p}\sum_{i=1}^{p} \frac{1}{|Y_i|} \sum_{y\in Yi} \frac{\big|\{y' : rank_f(\boldsymbol{x}_i, y') \leq rank_f(\boldsymbol{x}_i, y),\ y' \in Y_i\}\big|}{rank_f(\boldsymbol{x}_i, y)}$$

For *One-Error*, *Coverage* and *Ranking Loss*, the smaller the metric value the better the system's performance, with optimal value of $\frac{1}{m}\sum_{i=1}^{m}|Y_i| - 1$ for *Coverage* and 0 for *One-Error* and *Ranking Loss*. For *Average Precision* metric, the larger the metric value the better the system's performance, with optimal value of 1.

Additionally, we compute three scores originally used in information retrieval (IR) systems [55] that reflect the prediction ranking and the extent to which the methods produce not only good, but also highly-ranked good results, namely:

Hits@K: Hits@K measures the number of elements retrieved among the K elements with the highest score. Since this metric is per example, we report its average dividing by the number of examples. We extract this score for K $\in \{1, 3, 5, 10\}$.

P@K: P@K stands for the precision at K, i.e. precision computed only among top K ranking side effects per drug. We extract this score for K $\in \{3, 5, 10\}$.

D Feature Extraction from a Knowledge Graph

In Section A, we described how to generate a feature set for an entity x in the knowledge graph based on its incoming and outgoing edges. In practice, this feature extraction method is implemented using an SPARQL query, where SPARQL is a semantic query language for graphs, and the standard for querying RDF graphs. A SPARQL query is submitted to an Apache Jena Fuseki (downloaded from `https://jena.apache.org/documentation/fuseki2/`) HTTP endpoint containing the Bio2RDF data files for DrugBank, SIDER, and KEGG (downloaded from `http://download.openbiocloud.org/release/4/`). Listing 1 shows the SPARQL query that generates the feature set for a given `DRUG_URI` (a URI identifying a drug in DrugBank). This query has three main parts: (a) the query in the DrugBank database; (b) the query in the KEGG database; and (c) the filtering of irrelevant relations.

First, in (a) all triples in the knowledge graph having the `DRUG_URI` in a subject or object position are matched, and its direction is set as "in" (resp. "out") if it is in a object (resp. subject) position. Second, in (b) a similar process is repeated but over KEGG database. Since KEGG uses a different URI from DrugBank to represent a drug, a matching triple is added to the sub-queries. A drug in KEGG is connected to a unique drug in DrugBank through the `kegg:x-drugbank` predicate. Thus, given a DrugBank URI, we can find its equivalent in KEGG using the triple: `?drug kegg:x-drugbank DRUG_URI`.

Finally, as shown in Fig. 1, a drug entity contains some functional relationships, i.e., that only occur for that entity. Such relations have no influence when comparing pairs of drugs (i.e., computing distance between them) and therefore they can safely be removed at the feature extraction stage. Examples of these relation include: identifiers, e.g., x-identifiers.org, cas number, x-drugbank; not biological, chemical or phenotypic properties, e.g., brand, packager, mixture, toxicity, clearance, patent; and meta-data relations, among others. Therefore, in (c) a manually selected subset of relations is filtered out from the final results of the SPARQL query.

The output of the SPARQL query in Listing 1 is a set of $(predicate, entity, direction)$ tuples, which are later used to generate the feature vector of an entity. All tuples in the output are considered for generating the feature set of a drug, and no further filtering is performed on the features despite the one mentioned within the SPARQL query.

Note that this approach can be easily applied also to other data sets from Bio2RDF, or even from other source, as long as they are represented using the RDF format. The data sets only need to be loaded into the endpoint and then the SPARQL query for generating features needs to be adapted to these data sets, replacing or augmenting the current list (i.e. Drugbank, SIDER and KEGG).

Listing 1. SPARQL query to extract the feature set for an entity in the knowledge graph.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX void: <http://rdfs.org/ns/void#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX bio2rdf: <http://bio2rdf.org/>
PREFIX drugbank: <http://bio2rdf.org/drugbank_vocabulary/>
PREFIX kegg: <http://bio2rdf.org/kegg_vocabulary/>

SELECT DISTINCT ?p ?o ?dir WHERE {

  { GRAPH bio2rdf:drugbank_resource:bio2rdf.dataset.drugbank.R4 {
    { ?o ?p < DRUG_URI > .
      BIND ("in" as ?dir)
    }
    UNION
    { < DRUG_URI > ?p ?o .
      BIND ("out" as ?dir)
    }
  }}

  UNION

  { GRAPH bio2rdf:kegg_resource:bio2rdf.dataset.kegg.R4 {
    { ?drug kegg:x-drugbank < DRUG_URI > .
      ?o ?p ?drug .
      BIND ("in" as ?dir)
    }
    UNION
    { ?drug kegg:x-drugbank < DRUG_URI > .
      ?drug ?p ?o .
      BIND ("out" as ?dir)
    }
  }}

  FILTER (?p != bio2rdf:bio2rdf_vocabulary:identifier>) .
  FILTER (?p != bio2rdf:bio2rdf_vocabulary:uri) .
  FILTER (?p != bio2rdf:bio2rdf_vocabulary:namespace) .
  FILTER (?p != bio2rdf:bio2rdf_vocabulary:x-identifiers.org) .
  FILTER (?p != drugbank:drugbank-id) .
  FILTER (?p != drugbank:brand) .
  FILTER (?p != drugbank:calculated-properties) .
  FILTER (?p != drugbank:ddi-interactor-in) .
  FILTER (?p != drugbank:packager) .
  FILTER (?p != drugbank:mixture) .
  FILTER (?p != drugbank:toxicity) .
  FILTER (?p != drugbank:half-life) .
  FILTER (?p != drugbank:food-interaction) .
  FILTER (?p != drugbank:half-life) .
  FILTER (?p != drugbank:route-of-elimination) .
  FILTER (?p != drugbank:ddi-interactor-in) .
  FILTER (?p != drugbank:product) .
  FILTER (?p != drugbank:clearance) .
  FILTER (?p != drugbank:experimental-properties) .
  FILTER (?p != drugbank:manufacturer) .
  FILTER (?p != drugbank:patent) .
  FILTER (?p != drugbank:dosage) .
  FILTER (?p != drugbank:absorption) .
  FILTER (?p != drugbank:absorption) .
  FILTER (?p != kegg:internal-id) .
  FILTER (?p != kegg:formula) .
  FILTER (?p != kegg:exact_mass) .
  FILTER (?p != kegg:mol_weight) .
  FILTER (?p != kegg:x-drugbank) .
  FILTER (?p != kegg:x-atc) .
  FILTER (?p != kegg:x-pubchem.compound) .
  FILTER (?p != kegg:x-cas) .
  FILTER (?p != kegg:type) .
  FILTER (?p != kegg:x-dailymed) .
  FILTER (?p != kegg:x-nikkaji) .
  FILTER (?p != kegg:x-ligandbox) .
  FILTER (?p != kegg:other_map) .
  FILTER (?p != kegg:generic) .
  FILTER (?p != kegg:x-ccd) .
  FILTER (?p != kegg:other) .
  FILTER (?p != kegg:product) .
  FILTER (?p != kegg:bracket) .
  FILTER (?p != kegg:original) .
  FILTER (?p != kegg:repeat) .
  FILTER (?p != kegg:source) .
  FILTER (?p != kegg:component) .
  FILTER (?p != rdf:type) .
  FILTER (?p != void:inDataset) .
  FILTER (?p != rdfs:seeAlso) .
  FILTER (?p != rdfs:label) .
  FILTER (?p != owl:sameAs) .
  FILTER (?p != dcterms:identifier) .
  FILTER (?p != dcterms:title) .
  FILTER (?p != dcterms:description) .
}
```

E Feature Set Manipulation

After the feature set for every drug entity is extracted from the knowledge graph Bio2RDF, the design matrix is build from them. Each feature set of a drug only contains the features for that drug, and it is not aware of the overall set of features. In order to build the design matrix, a vectorisation must first collect the full set of unique features to assign the final feature vectors. Listing 2 shows the Python 3 commands required to convert the feature sets generated by querying the knowledge graph (cf. Listing 1) into a NumPy (`http://www.numpy.org/`) 2D array using Scikit-Learn (`http://scikit-learn.org/stable/`) APIs.

Listing 2. Generating feature vectors for drugs from the KG.

```
import numpy as np
from sklearn.feature_extraction import DictVectorizer

kb = ... # instance of a knowledge graph
feature_vectorizer = DictVectorizer(sparse=False)
# Retrieving drugs' features
drug_to_features_list = [{feature: 1 for feature in
                          kg.get_features(drug)}
                         for drug in drugs]
kg_features = feature_vectorizer.fit_transform(drug_to_features_list)
```

More specifically, the class `DictVectorizer` (see `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html` for a full documentation) is used to convert the feature sets into a single design matrix $\mathbf{X}$ (2D Numpy array). In this matrix, a drug's features are represented by a row vector $\mathbf{X}_i$, and a single component $\mathbf{X}_{i,j}$ can take values 1 or 0 indicating whether drug i has feature j or not, respectively.

To load the Liu's data set in MATLAB `.mat` format, one can make use of the SciPy (`https://www.scipy.org/`) library, which provides an IO package for this. Listing 3 shows an example of this. Because Liu's data set provides a different matrix for each feature (i.e., enzymes, pathways, targets, transporters, treatments, and substructures), the design matrix can be build by concatenating all of them using Numpy `concatenate` function. Similarly, if we want to use different data sets together, e.g., Liu's and Bio2RDF, we can use the same `concatenate` function to generate a single design matrix.

Liu's data set also provides the labels (ADRs) for all drugs in a separate matrix termed `side_effect`, which is loaded into a `y` variable. (For the case of SIDER 4, we provide the design and target matrices for training and testing instead of by feature groups.)

We provide the different data sets for download and describe their characteristics in the Web page `http://purl.com/bib-adr-prediction` or access the GitHub mirror at `https://github.com/emir-munoz/adr-prediction`.

Listing 3. Loading Liu's data set and merging it with Bio2RDF data set.

```
import scipy.io as sio
import numpy as np

# Collecting features from the Liu's dataset
data = sio.loadmat(liu_dataset_path, struct_as_record=True)
X = np.concatenate((data['Enzymes'], data['Pathways'],
                    data['Targets'], data['Transporters'],
                    data['Treatment'], data['chemical']), axis=1)
y = data['side_effect']
logging.info('X: %s, y: %s' % (X.shape, y.shape))
```

F Algorithm for the KG-SIM-PROP Method

In this section we provide a pseudocode for the KG-SIM-PROP method [27]. KG-SIM-PROP is a similarity-based method that propagates the labels of the k nearest neighbours to a drug using the 3w-Jaccard similarity metric (cf. Section A). For that the pairwise distance between the feature vector of the input drug and the feature vectors of all other drugs is computed (this can be done in parallel per pair using Scikit-Learn APIs). For example, consider the k-fold cross-validation process where $(k-1)$ folds are used to train a model which is then evaluated on the remaining fold. Let `train_X` be the feature matrix for the $k-1$ folds and let `test_X` be the feature matrix for the remaining fold. Listing 4 shows the steps to obtain the similarity matrix between both sets of drugs. It also illustrates how to obtain the prediction scores for each drug using a weighted average of the labels coming from the nearest neighbours.

Listing 4. Pseudo code for KG-SIM-PROP model.

```
import numpy as np
from sklearn.metrics.pairwise import pairwise_distances

similarity_matrix = pairwise_distances(test_X, train_X,
                                       metric=similarity_fn, n_jobs=-1)

# Initialize the (#samples x #labels) result matrix
y_pred = np.zeros((test_X.shape[0], train_y.shape[1]))

for i in range(test_X.shape[0]):
    # Obtain the nearest neighbours for each sample
    neighbors = np.argsort(similarity_matrix[i, :],
    kind='mergesort')[::-1]
    nearest_neighbors = neighbors[:nb_neighbors]

    # Retrieve the similarities to the k nearest neighbors
    s = similarity_matrix[i, nearest_neighbors]
    # Retrieve the labels assigned to the neighbours
    t = train_y[nearest_neighbors]
    # Get a weighted average of the labels of the k nearest neighbors
    norm = np.sum(s)
    y_pred[i] = np.dot(s, t) / norm if norm != 0.0 else 0.0

return y_pred
```

G Algorithm for the Evaluation Protocol

The evaluation protocol followed to evaluate all the models presented in this manuscript is the same. This protocol considers the data, a model with its hyper-parameters, and runs a grid search to evaluate the model. Listing 5 shows the steps followed in this protocol in a pseudocode (enriched version with more details on the hyper parameters).

Listing 5. Algorithm of the evaluation pipeline for the multi-layer perceptron model.

```
# Import multi-label metrics
from sklearn import metrics

# Load data set
X, y = load_data()

# Define evaluation metrics
metrics = [metrics.coverage_error, metrics.label_ranking_loss, ...]

# Instantiate a multi-label learning model
pipeline = Pipeline([('MLP', MLPRegressor())])

# Define hyper-parameters space
parameter_space = {
    'MLP__hidden_layer_sizes': [(64,), (128,), (256,), (512,)],
    'MLP__activation': ['logistic'],
    'MLP__solver': ['adam'],
    'MLP__beta_1': [0.9],
    'MLP__beta_2': [0.999],
    'MLP__epsilon': [1e-8],
    'MLP__early_stopping': [True],
    'MLP__validation_fraction': [0.1],
    'MLP__alpha': [0.01],
    'MLP__batch_size': [int(np.ceil(x.shape[0] * .2))],
}

# Run grid search or train/test
run_grid_search(X, y, pipeline, parameter_space, metrics)
```

In Listing 5 one can use any of the metrics for multi-label learning available in Scikit-Learn (cf. `http://scikit-learn.org/stable/modules/model_evaluation.html` for details) in the `metrics` list parameter, and select one of them for tuning. The best combination of parameters for a model if found by using a k-fold cross-validation with grid search (cf. `http://scikit-learn.org/stable/modules/grid_search.html`). Also, new metrics can be defined by extending Scikit-Learn APIs. In Listing 6 we provide an example of how to define the One-Error metric not available from Scikit-Learn library.

Listing 6. Declare a new evaluation metric.

```
def one_error(y_true, y_pred, pos_label=1):
    assert y_true.shape[0] == y_pred.shape[0]
    assert y_true.shape[1] == y_pred.shape[1]

    p = y_true.shape[0]
    one_error = 0.0
    for i in range(p):
        y_i, p_i = y_true[i, :], y_pred[i, :]
        idx = np.argmax(p_i)
        one_error += (1.0 / p) if y_i[idx] != pos_label else 0.0

    return one_error
```