# Comparing Link Prediction Approaches for Polypharmacy Side Effect Modelling

## Abstract

## Introduction

The comforts of modern life are causing a demographic shift in populations globally. By 2050, the number of people aged 60+ is projected to more than double [cite [WHO](https://www.who.int/news-room/fact-sheets/detail/ageing-and-health)], compared to an overall population increase of around just 21% [cite [UN](https://www.un.org/en/global-issues/population)]. With this, medical events that primarily affect older persons will become increasingly important areas of research. Multimorbidity is one such phenomenon, whose prevalence among elderly populations may range from 55-98% [cite [marengoni et al](https://www.sciencedirect.com/science/article/pii/S1568163711000249?via%3Dihub)]. Though it does also impact young people, it does so less frequently due to the increased exposure to risk factors that naturally occur as a person’s life progresses. Closely associated with multimorbidity is another issue: polypharmacy, the taking of two or more medications simultaneously by the same individual. Polypharmacy can be usefully employed in some contexts, for example to achieve drug synergism [cite [tallarida](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3379564/)], where two medications are combined to produce an effect greater than the sum of their parts. However, the practise of polypharmacy can also lead to the emergence, via chemical interactions, of adverse drug reactions (ADRs) that are not associated with either drug individually [cite [ahmed et al](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4234513/)]. ADRs put a large strain on healthcare systems, with a systematic review from 2002 putting their annual cost to the NHS at £380 million [cite [bandolier 2002](•%09http:/www.bandolier.org.uk/Extraforbando/ADRPM.pdf)]. Since ADR incidence has remained relatively unchanged over time [cite [Coleman and Pontefract, 2016](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6297296/)], we can estimate current day costs, using inflation and population increase, to be £763 million per year.

Correct anticipation of ADRs related to polypharmacy is a challenging task, owing mainly to the exponential nature of the problem. In England an estimated 8.4 million people are taking at least 5 prescribed medications, and a quarter of those are taking 10 or more [cite [National overprescribing review report](https://www.gov.uk/government/publications/national-overprescribing-review-report)]. With complexity given by *n(n-1)/2*, testing every possible *n*-combination of the thousands of commercially available drugs quickly becomes infeasible in wet-lab experiments or clinical trials, even with smaller values of *n* than are commonly found in populations. As a result, pre-clinical screening with statistical/computational methods is a necessity for identifying drug combinations of interest [cite [ryall and tan 2015](https://jcheminf.biomedcentral.com/articles/10.1186/s13321-015-0055-9#Sec2)]. Natural language processing (NLP) techniques have been applied with some success to identify promising candidates from unstructured text data. Early such work extracted facts about drug-targets from the literature, using these to detect possibly interacting drugs from a given set [cite [tari et al 2010](https://academic.oup.com/bioinformatics/article/26/18/i547/206034#393571806), [percha et al 2012](https://pubmed.ncbi.nlm.nih.gov/22174296/)]. Other research took advantage of the information available in electronic health records (EHRs) to enrich prediction models with text data from multiple sources [cite [duke et al 2012](https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002614)]. Though focusing more on the monopharmacy case, several studies have demonstrated the viability of scraping ADRs from text on health forums [[Sarker and Gonzalez, 2015](https://www.sciencedirect.com/science/article/pii/S1532046414002317)] as well as general social media websites such as *Instagram* [cite [correia et al](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4720984/)]and *Twitter* [cite [Nikfarjam 2014](https://www.semanticscholar.org/paper/Mining-Twitter-for-Adverse-Drug-Reaction-Mentions-%3A-Nikfarjam/7da014a71967131bc0123f687cc403293d6c8067)]. Non-NLP methods can broadly be categorized based upon whether they use graph data or not. Those that don’t, tend to use drug feature vectors as their inputs, with values representing drug-target data, single drug side effects, drug indications, and more. In a crude but surprisingly effective example of this, Zhao *et al.* view pairs of these features as patterns, calculating the probability of a positive drug-drug interaction (DDI) as the ratio of occurrences of this patten in ‘effective’ drug combinations to occurrences in a background set of all known drug combinations [cite [zhao et al 2011](https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002323#abstract0)]. This is an unusual method, however, and many other studies opt for a more supervised approach, where they perform one or more logistic regressions on input features [cite [huang et al 2014](https://www.nature.com/articles/srep07160#Sec9)a, [shi et al 2017](https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1818-2#Sec2)]. Graphical approaches are perhaps more diverse, with success being found by techniques including distance-based metrics [cite [chen et al 2016](https://pubs.rsc.org/en/content/articlehtml/2016/mb/c5mb00599j)], clustering [[huang et al 2014](https://academic.oup.com/bioinformatics/article/30/12/i228/388003" \l "393812843)b], and structural similarity [cite [Takeda et al 2017](https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0200-8)].

Recently, there has been an explosion of interest in graph-embedding methods to solving the problem of polypharmacy side effect prediction. This can largely be traced back to the Decagon paper [cite decagon], in which the author’s construct a knowledge graph from the following data: drug-target, protein-protein interaction, monopharmacy side effects, and drug-pair side effects. A portion of the drug-pair side effects are left out to enable out-of-sample prediction and assessment. By constructing the data in this way, the problem is cast as a multirelational link prediction problem, which, the authors claim, makes this the first technique that allows prediction of the *type* of side effect that will occur rather than a simple binary categorisation or magnitude of effect. The Decagon model itself consists of two components. Firstly, node embeddings for the network are encoded using a graph convolutional model. These embeddings are then passed to a tensor factorisation decoder which produces a score for a particular side effect *r* between a drug pair *(vi, vj).* Finally, that score is passed to a sigmoid function which outputs a probability that the given triple *(vi, r, vj)* is true*.*

The data used by Decagon is publicly available [cite biosnap] and dozens of papers have been published since which claim to match or improve upon the model’s performance. Some studies have noted Decagon’s inefficient performance, and have presented models which are substantially simpler and therefore faster to run while achieving similar scores as measured by area under the Receiver-Operating Characteristic (AUROC) and area under the Precision-Recall Curve (AUPRC) [cite [burkhardt et al](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7153048/) and [masumshah et al](https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-021-04298-y)]. Reduced model complexity also lends itself to improved interpretability, which has been a frequently documented criticism of Decagon [cite [Malone](https://link.springer.com/chapter/10.1007/978-3-030-06016-9_14), [bang](https://academic.oup.com/bioinformatics/article/37/18/2955/6170654), [liu](https://link-springer-com.bris.idm.oclc.org/article/10.1007/s10489-021-02296-4), [dai](https://academic.oup.com/bib/article/22/4/bbaa256/5943784), [vaida](https://ieeexplore.ieee.org/document/8999197/authors#authors)]. In a similar vein, Decagon’s inability to generalise to unseen nodes has also drawn flack [cite [lukashina](https://jcheminf.biomedcentral.com/articles/10.1186/s13321-022-00632-5), [huang](https://ojs.aaai.org/index.php/AAAI/article/view/5412), [yang](https://pubs.rsc.org/en/content/articlelanding/2022/SC/D2SC02023H)]. While this is a valid and relevant appraisal, it is worth noting that all of these papers bring in additional data in the form of chemical substructure information, making any direct comparisons against Decagon’s *performance* unfair. If Decagon had this additional training data, it may in fact outperform these more externalisable models. This is not an uncommon theme - of the 20 papers we found that directly use Decagon’s data as a point of comparison, half of them either bring in extra data for their new models or perform large transformations on the base data. As a result, it is often unclear whether any performance gains are due to changes in model architecture or caused by the differential inputs.

Knowledge graph embedding (KGE) methods are a broad family of techniques that take heterogenous graphs as input and cast the comprising nodes into a low-dimensional vector space, usually having one such space per type of edge in the KG. They have proved useful in several real-world tasks, such as item recommendation [cite [Catherine and cohen 2016](https://www.cs.cmu.edu/~wcohen/postscript/recsys-2016.pdf)], drug development and repurposing [cite [Geleta et al 2021](https://www.biorxiv.org/content/10.1101/2021.10.28.466262v1.full)], and knowledge base completion [cite transE]. The application of these techniques to the polypharmacy side effect modelling problem has been somewhat scarce considering their natural affiliation for this type of data. In the original paper, the RESCAL method [cite rescal] is employed as a baseline against which the Decagon model is compared. RESCAL is reported as the worst performing technique, being outshone even by simple embedding methods such as random-walks and feature vectors. In stark contrast, Malone *et al.* applied the DistMult [cite distmult] method to the Decagon data as a comparator, finding it outperformed not only their presented model but also Decagon itself [cite [Malone et al](https://link.springer.com/chapter/10.1007/978-3-030-06016-9_14)]. Other studies have reported similar results for DistMult on Decagon data [cite [kim and shin 2023](https://www.mdpi.com/2076-3417/13/5/2842), [Liu et al 2022](https://link.springer.com/article/10.1007/s10489-022-03839-z#Sec10)], but Dai *et al*. report substantially worse performance under equivalent conditions [cite [Dai et al](https://academic.oup.com/bib/article/22/4/bbaa256/5943784#273493426)]. In that study, the KGE models ComplEx [cite], SimplE [cite], and RotatE [cite] all achieved much better results than DistMult, as measured by hits@k and rank-based metrics. A modified version of ConvE [cite conve], a deep learning KGE technique, was used by Wang *et al.*, who once more report significant improvements over Decagon [cite wang et al].

In this paper we perform a comparative study with the goal of achieving some clarity on the performance of standard KGE models on the task of polypharmacy side effect prediction. As we convert the Decagon graph into binary triple format, we also explore the impact of different methods for doing so. By using the LibKGE platform [cite], we present our work as easily reproducible experiments.

## Methods

A diagram of a training program

Description automatically generated

Figure X. Flowchart giving an overview of the analysis. Blue rectangles represent raw data, green rounded rectangles are processed data, yellow diamonds are scripts/softwares, and grey circles are instantiated KGE methods.

Raw data, processed in [cite decagon], was downloaded from the Stanford Network Analysis Project (SNAP) [cite snap]. The task then was to prepare the data for LibKGE [cite libkge], which reads graphs as a list of edges in triple (head, relation, tail) format. Multiple different approaches could be taken to perform this conversion. In our pilot work we focused on three such methods, which we have named ‘selfloops’, ‘non-naïve’, and ‘multidrug’ – the difference between them being the way in which mono-/polypharmacy side effect data is structured in the resulting graph. The selfloops approach treats side effects as edges, either between pairs of drug nodes for polypharmacy, or from one drug back to itself (hence the name) in the monopharmacy case. The non-naïve construction method is similar, but monopharmacy data is instead modelled as n-hot node feature-vectors, with ‘hot’ columns indicating which side effects are associated with a given drug. Dimensionality reduction via principal component analysis (PCA) was performed on these features to create a smaller matrix for each possible dimensional size of embeddings. Then LibKGE was modified to load these vectors from disk, using them as the starting point for learning node embeddings for this dataset rather than any of its usual stochastic initialisation techniques. Finally, the multidrug approach was inspired by work in [cite kim and shin 23] and involves the creation of drug-pair nodes which occupy the same embedding space as regular drug nodes. This way, side effects can be modelled as nodes rather than edges, allowing simple ‘monopharmacy side effect’ edges from drugs to side effects and ‘polypharmacy side effect’ from drug-pair nodes to side effects. To have LibKGE recognise the association between multidrugs and their constituent drugs, two ‘multidrug contains’ edges were created between every drug-pair node and their corresponding singular drugs. Graph statistics for the three networks are listed in [table X] below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Graph | Meta nodes | Nodes | Meta edges | Edges |
| Selfloops | 2 | 19734 | 11148 | 5485566 |
| Multidrug | 4 | 94353 | 5 | 5612510 |
| Non-naïve | 2 | 19734 | 964 | 5310589 |

To assess the broad viability of dataset/model combinations, a pilot study was conducted which employed one geometric model (TransE [cite]), one deep learning model (ConvE [cite]), and one matrix factorisation method (ComplEx [cite]) on each of the three datasets. Results indicated that the multidrug graph was a poor input for obtaining quality embeddings, so we decided to exclude it from the main analysis to reduce computational burden by approximately one third. Keeping computational efficiency in mind, LibKGE contains 11 KGE methods, many of which have been shown to produce comparable results when run with the same training methods and loss functions [cite old dog new tricks]. Therefore to avoid redundancy, we created some exclusion criteria which whittled down our method count to just 5. The criteria included such things as models being direct enhancements of one another, or inefficient performance in the pilot study. A list of all methods and reasons for their exclusion are given in [table X] below.

|  |  |  |  |
| --- | --- | --- | --- |
| LibKGE model | Selected | Mechanism | Citation |
| ComplEx | Yes | MF | [Trouillon et al., 2016](https://proceedings.mlr.press/v48/trouillon16.html) |
| ConvE | No | DL | [Dettmers et al., 2018](https://dl.acm.org/doi/10.5555/3504035.3504256) |
| Canonical Polyadic Decomposition (CP) | No | MF | [Lacroix, Usunier, and Obozinski, 2018](https://proceedings.mlr.press/v80/lacroix18a.html) |
| DistMult | Yes | MF | [Yang et al., 2014](https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nips2014_relation_semantics_arxiv_version.pdf) |
| Relational Tucker3  (RT3) | Yes | MF | [Wang, Broscheit, and Gemulla, 2019](https://aclanthology.org/W19-4313/) |
| Rescal | No | MF | [Nickel, Tresp, and Kriegel, 2011](https://paperswithcode.com/paper/a-three-way-model-for-collective-learning-on) |
| RotatE | No | G | [Sun et al., 2018](https://arxiv.org/abs/1902.10197) |
| SimplE | Yes | MF | [Kazemi and Poole, 2018](https://arxiv.org/abs/1802.04868) |
| TransE | No | G | [Bordes et al., 2013](https://papers.nips.cc/paper_files/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html) |
| Transformer | No | DL | [Chen et al., 2020](https://aclanthology.org/2021.emnlp-main.812.pdf) |
| TransH | No | G | [Wang et al., 2014](https://ojs.aaai.org/index.php/AAAI/article/view/8870) |

Table X. All models available in LibKGE, their citations, mechanism class, and whether they were included in this study. MF = matrix factorisation, DL = deep learning, G = geometric.

All the methods listed above are theoretically suitable for this analysis, so we felt it necessary to work backwards with reasons to *exclude* rather than *include* them. CP and RESCAL are both predecessors of others in this list (SimplE and RT3 respectively), and in fact RESCAL had already been directly compared to Decagon in the original paper, so running them would be wasteful. Four more models were excluded for inefficiency: ConvE, TransE, RotatE, and TransH – the former two failed, during the testing phase of this study, to embed the large datasets within computational limits, while the remaining two had faced similar issues on smaller datasets in previous work [cite chapter 1]. The final model to be excluded was Transformer, which cannot perform the ‘\_po’ scoring necessary to be trained under our experimental configuration.

All of the methods employed here can be classified as some form of matrix factorisation. ComplEx [cite complex] uses the Hermitian product to embed entities into the complex space. This ironically straightforward approach allows anti-symmetric relations to be modelled with a low-rank decomposition, which, in the real space, would only be possible for symmetric relations. DistMult [cite distmult], also referred to as ‘bilinear-diag’ by its authors, is similar to TransE but uses a *multiplicative* operation rather than an *additive* one to combine dyadic vectors. Relational Tucker3 [cite reltucker] uses the eponymous Tucker decomposition to reduce a KG (modelled as a 3-way tensor) into an entity embedding matrix, a relation embedding matrix, and a core tensor. Lastly, SimplE [cite simple] is an enhancement of the Canonical Polyadic (CP) decomposition for embedding KGs. CP itself was used often in early LP research but has a major shortfall in that it learns head and tail vectors for a given node independently – SimplE addresses this problem by also considering inverted relations to create dependency between said vectors.

The hyperparameter optimisation space, established in configuration files, was the same for all models. Hyperparameter ranges were based on those from the search performed by the LibKGE developers in their demonstrative paper [cite old dog new tricks]. We made a few adjustments to the space, notably expanding the number of available options for optimiser and loss function from 2 and 3 to 8 and 5 respectively. Other alterations included adding possible embedding sizes of 32 and 64, and reducing the aggressiveness of the learning rate scheduler. The grid searches took place over 100 trials, with the first 50 having values chosen by Sobol sequence [cite sobol] and the remaining 50 chosen by Bayesian optimisation through the Ax platform [cite [Ax](https://ax.dev/)]. All configuration files used to create the experiments are available in our GitHub repository [cite static repo].

We removed a portion of both test graphs, prior to the main analysis, for out-of-sample validation. Following the methodology used in the Decagon paper, the holdout data was created by randomly removing 10% of the edges belonging to each polypharmacy side effect. Since the graphs include the same polypharmacy data, the same edges were held out from both to enable fair comparison between the two datasets. This holdout set contains 458,061 edges, leaving 5,761,807 in selfloops and 5,586,830 in non-naïve. We measured performance using standard link prediction metrics employed in LibKGE, namely mean reciprocal rank (MRR) and hits@k. We also manually assess with the metrics used by the Decagon authors to allow direct comparison with their model - area under the receiver-operating characteristic (AUROC), area under the precision-recall curve (AUPRC), and average precision at 50 (AP@50). These are calculated individually per side effect type.

This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol—<http://www.bristol.ac.uk/acrc/>. The specific environment was CentOS-7 running Python 3.8.12 with PyTorch 1.7.1, accelerated with CUDA 11.4 on 4 × NVIDIA GeForce RTX 2080 Ti.

## Results

[if we’re not running decagon, why not compare against the reported scores of decagon AND the papers that claim to beat it? Just keep in mind that many of them have extra/transformed inputs]

## Discussion

## References

## Supplementary

Optimisation algorithms

Here we list and briefly describe the deep learning optimisers that are available to our embedding gridsearches. For further information please see the relevant citations.

**Adam** is an optimizer that is widely available across many deep learning frameworks, . Proposed in 2015 [cite [adam](https://arxiv.org/pdf/1412.6980.pdf)], the name is a contraction of ‘**Ada**ptive **m**oment estimation’, signifying it’s ability to calculate per-parameter learning rates using both the first and second moments of a given function’s gradient. These moments correspond to moving averages of gradients encountered at previous steps, the tracking of which allows the algorithm to increase the learning rate when gradients are gentle and vice versa. The same paper also proposed the **AdaMax** variation, which uses the infinity rather than L2 norm to scale gradients when updating weights.

**AdaGrad**, or ‘**Ada**ptive **Grad**ient algorithm’ [cite [adagrad](https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf)], improves on simple gradient descent methods by considering the geometry of the space encountered in earlier iterations of the algorithm when calculating the next step to take. This is achieved by tracking the squares of gradients found per dimension, the sum of which is then used to scale current gradients. The algorithm has been criticised for its poor running speed so is not used in this study, but is described anyway to provide context for AdaDelta and RMSprop.

**AdaDelta** [cite [adadelta](https://arxiv.org/pdf/1212.5701.pdf)] can be seen as an adaption of AdaGrad that improves descent speed. By adding a decay factor to the otherwise ever-growing sum of squared gradients, older gradients are rendered less influential when calculating current steps. In this way, the AdaGrad’s inclination to produce step sizes tending towards zero is avoided, leading to faster algorithm convergence especially when initial gradients are steep. Perhaps more importantly though, the algorithm is also less likely to find itself in an inescapable local minimum.

**AdamW** [cite [adamw](https://arxiv.org/abs/1711.05101)] is a variation of **Adam** that uses **W**eight decay to produce smaller weights, bringing the benefits of reduced overfitting and improved generalisation. Loss regularization achieves this for basic stochastic gradient descent, but the transplanting of this technique to adaptive gradient algorithms (i.e. the Adam family) does not produce equivalent results because weight reduction ends up being inversely proportional to the magnitude of the gradient. AdamW decouples the reduction of weight size from the step size calculation, thus allowing weights to be decayed independently of the geometry of the parameter-error space.

**RMSprop** is very similar to AdaDelta in that it improves upon AdaGrad by decaying the influence of temporally distant gradients. Despite its widespread use, RMSprop has never been officially published. For those interested in further reading, we refer you to PyTorch’s documentation for this algorithm as an informative starting point.