Structural Characteristics of Knowledge Graphs Determine the Quality of Knowledge Graph Embeddings Across Model and Hyperparameter Choices

Jeffrey Sardina

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science in the Data Science Strand

2021

Declaration

Signed: Jeffrey Sardina
Date: 15 August 2021

Acknowledgements

I would like to thank my supervisor Dr. Declan O’Sullivan for helping me with this research from the start to the end. In addition, I would like to thank Fabrizio Orlandi for pointing me to several references regarding Knowledge Graph structure, and Jennifer Ní Chiara for help in proofing the Irish-language version of the abstract. Finally, I would like to thank my family and all my friends who helped me get here.

Ba mhaith liom buíochas a thabhairt do mo stiúrthóir Dr. Declan O’Sullivan as cúnamh a thabhairt dom leis an taighde seo ó thús go deireadh. Leis sin, ba mhaith liom buíochas a thabhairt do Fabrizio Orlandi as tagairtí maidir le structúr Graf Eolais a chur ar fáil dom, agus do Jennifer Ní Chiara as an leagan Gaeilge den achoimre a phrofáil. Ar deireadh, ba mhaith liom buíochas a thabhairt do mo mhuintir agus do mo chairde ar fad a chabhraigh liom é seo a bhaint amach.

Table Of Contents

Tables and Figures

Abbreviations

Abbreviation	Full Form
AUC	Area Under the Curve
KG	Knowledge Graph
KGE	Knowledge Graph Embedding
LOD	Linked Open Data
MSE	Mean Squared Error
PBG	PyTorch-BigGraph

1 Introduction

1.1 Outline of the Introduction

Section 1.2 provides a general overview of all Knowledge Graph, Linked Open Data, and machine learning topics necessary to understand the context of this work. Section 1.3 provides an overview of the motivations, objectives, methods, and results of this work. Finally, Section 1.4 provides an outline of the remainder of the sections in the dissertation.

Since this work was completed in the context of biomedical research, all examples contained within will be made in reference to well-known biological facts and processes. The intent of this is to clearly demonstrate how these techniques and results are immediately applicable to biomedical research.

1.2 General Background

This section serves to give a general context of the work. Section 1.2.1 introduces the problem this work seeks to address in the field of Knowledge Graphs and biomedical research. Section 1.2.2 explains the background concepts of Linked Data and Knowledge Graphs in the larger context of biological Linked Open Data datasets. Section 1.2.3 summarizes the mechanisms and motivation for using Knowledge Graph Embeddings to learn from Linked Open Data data sources. Section 1.2.4 continues to explain the process of learning Knowledge Graph Embeddings using a technique known as negative sampling. Section 1.2.5 considers the details of biological and cancer data that lend themselves to Linked Data representations and to Knowledge-Graph Embedding techniques.

1.2.1 Problem Statement

The fields of cancer biology and biomedical sciences have been revolutionized by Big Data. From projects such as Bio2RDF [1], the International Cancer Genome Consortium [2], the 1,000 Genomes Project [3], and TumorMap [4], big data has become a centerpiece of biomedical research. With the ever-increasing magnitude of these datasets, several approaches have been taken to analyze and utilize the full breadth of these resources. Some projects, such as TumorMap, have focused on transforming the available data into a simpler format through dimensional reduction mechanisms, accepting a degree of information loss in exchange for easier usability [4]. On the other hand, recent Linked Open Data (LOD) systems have attempted to represent the entirety of the data in an easily-queryable graph-based format [1, 7, 13, 15]. Among the projects that have taken this approach is Bio2RDF, a LOD data store that incorporates data from many different biological and biomedical datasets into a graph format [1]. Several groups have followed up upon such projects with graph-based machine learning techniques called Knowledge Graph Embeddings (KGEs) in an attempt to process entire LOD datasets at once [5-7].

Using Knowledge Graph Embeddings requires the selection of hyperparameters to the models, and proper selection of hyperparameters is critical to enabling the model to best learn the data at hand [8, 28]. This research addresses the question of whether biological LOD datasets could be characterized with a set of general dataset features. This characterization leads to an exploration of the possibility of allowing relational learning algorithms to operate on these datasets using a consistent set of known hyperparameters. Such a system would allow biological LOD datasets to be analyzed much more quickly, without the need for running a full search for optimal hyperparameters on every new dataset.

Finally, relating optimal hyperparameters and KGE performance to structure would allow predicting and explaining why certain models perform better than others, and would allow these results to be generalized to similarly structured graphs from radically different, non-biological fields. While in this work the cancer biology and biomedical fields are used--due to their importance and due to the author’s previous experience in both--the goal of explaining results in terms of KG structure allows for a generalization beyond that domain.

Please note that herein, references to “hyperparameters” refer not only the parameters to a KGE model (such as the learning rate), but also to the choice of the models themselves; while this use is not strictly in keeping with the formal definition of hyperparameters, it results in a simpler phrasing of the model choices made.

1.2.2 Background on LOD and KGEs

Linked Data is a system by which heterogeneous data from a variety of distributed data sources can be connected, creating a larger web of data not dissimilar to the Internet [9]. Linked Open Data, by extension, is the subset of Linked Data that is accessible under an open-source license, allowing it to be used by anyone [9].

The most popular Linked Data technology is RDF, the Resource Description Framework, in which all data is represented in a graphical format [9]. The smallest unit of information in an

RDF graph is the triple, a set of two entities and one predicate (or relationship) that links them [9]. The first entity is called the subject or the head, and the second entity the object or the tail [9]. In this configuration, RDF triples mirror, to a basic degree, linguistic statements. For example, the statement “P53 is a protein” could be modelled in RDF as P53 (subject) is-a (predicate) protein (object).

In the RDF format, subject, predicates, and objects are often represented by URLs which allows for entities and predicates to be easily reused and dereferenced, either within one data source or between different data sources [9]. The ability to reuse entities and relationships means that various triples can be linked and connected logically to each other, either by sharing a head, a tail, or both. Moreover, the RDF specification also allows for objects to be represented by literals, such as integers or strings, rather than URLs [10]. Such literals can only appear at the tail of triples, and thus cannot be re-used as subjects the way URLs can be [10].

Data in the RDF format is most commonly queried using the SPARQL Query Language [48]. In SPARQL, queries are formulated as graph matching patterns, allowing the queries to take full advantage of the graphical structure of the data [48]. This allows the construction of queries that can return a tabular set of results or a subgraph of the queried graph that match the query [48]. Moreover, it is possible to take full advantage of the LOD ecosystem and use information (such as entities or relationships) defined in one KG to query another KG that contains references to those items [48]. Due to the power of this approach, many projects such as [7, 13, 23, 24, 39, 43] have made exposing data in a SPARQL-queryable form the end-goal of their work.

1.2.3 Learning RDF data: outline and goals

LOD graphs constructed in the RDF format are referred to as Knowledge Graphs (KGs). Due to the triple-based structure of KGs, the data contained within can be very easily related. This is seen simply in the following example:

Subject	**Predicate**	**Object**
protein	made-by	ribosome
amino-acids	monomers-of	protein

Repeating entities as both a subject and an object allows linking triples to each other. While this can be used in many cases for logical reasoning using formal rules to extract more information from the graph [9], this property also allows machine learning techniques to operate on the graph and learn to distinguish and related entities based on the triples they are involved in [11, 45]. These machine learning techniques are referred to as relational machine learning since they learn based on these relations [11]. The output of this relational machine learning is a set of Knowledge Graph Embeddings, where entities are typically represented as vectors in R^n and relationships are represented by transformations on those entities [11, 45]. These relationships are structured as functions that map R^n onto R^n, allowing them to convert from subjects to the expected objects of that relationship [11, 45]. The choice of the dimension into which the entities are placed, as well as the choice of what sort of transformation is used to model the relationship (such as vector addition or matrix multiplication) are model design choices that must be investigated by developers to find the optimal combination for a given data set [11, 45].

In most cases, given a triple (*s,p,o*), applying the relationship that represents *p* to *s* should result in the vector embedding for the object *o* [45]. It should be noted that some of the more advanced models depart from this rule and allow other matching systems; however, this embedding method is the most representative and remains a commonly used one [45]. It can be expressed simply in mathematical terms as the following objective function:

$f_p(s) \approx o$

where f_p is the function for the relationship *p*.

For example, in the triple (*protein, created-by, ribosomes*), applying the transformation corresponding to the relationship *created-by* to the subject *protein* should result in a vector close to or equal to the embedding of *ribosomes*.

This formulation of knowledge graph embeddings suggests a very clear use of KGEs: given a subject and a predicate, predict the associated object [11, 45]. In this way, the KGE system can be understood as a question-answering system, where the subject and predicate make up the question and the object is the answer.

This formulation of relational learning works due to the fact that KGEs can learn to identify true facts without having explicitly seen them, based on how the entities and relationships of those facts are used elsewhere in the knowledge graph. For example, suppose a KG consists of a variety of facts about proteins, such as the following:

Subject	**Predicate**	**Object**
protein	made-by	ribosome
amino-acid	monomer-of	protein
P53	is-a	protein
P53	made-by	ribosome
SHH	is-a	protein

and then is asked to predict the object given the subject “SSH” and the predicate “made-by”, it should be able to correctly return the answer “ribosome”, despite never having seen this fact before [11, 45].

In the context of biomedical and cancer research, this could amount to predicting the result of loss-of-function of a gene, overexpression of a protein, or environment-stimulated release of various intra- or inter- cellular messenger molecules. Moreover, since KGEs take into account the relationships between various elements in the graph, this approach provides an intuitive way to make use of entire datasets, rather than only subsets thereof.

1.2.4 Learning KGEs: Negative Sampling

In order for the KGE model to effectively learn to predict true triples and reject false ones, it must be trained not only on the known-true triples but also on known-false triples [11, 45]. This is done using a technique called negative sampling [11, 45]. Negative (or false) triples are typically generated either under the Closed-World Assumption, the Open-World Assumption, or under the Local Closed-World Assumption [11, 45]. The Closed-World Assumption states that all true triples are contained within the knowledge graph, and that there are no true statements that do not exist within it [11, 45]. Under the Closed-World Assumption, negative samples can be generated very easily. Since all triples not in the training set are assumed to be false, simply inventing a new triple not previously seen will generate a valid negative. However, in real-world datasets where knowledge is constantly expanding and growing, this assumption practically never holds, and is thus rarely useful for training on real-world data [11, 45].

The Open-Word Assumption, by contrast, assumes that there may be arbitrarily many true statements which are not contained in the KG [11, 45]. However, it gives no way of predicting which, if any, triples that are not in the KG are true and which are false [11, 45].

The Local Closed-World Assumption is a more practical adaptation of the Open- and Closed-World Assumptions that claims that if a given subject and predicate are observed with a certain set of objects, then that subject and predicate only ever match to objects of those types [11, 45]. In essence, this means that a Closed-World can only be assumed in the local context of subject-predicate pairs, rather than in the unrestricted context of the whole graph. One major benefit of this approach is that it strikes a balance between overconfidence in the dataset (seen in the Closed-World Assumption) and ability to identify likely-false statements (which is not possible under the Open-World Assumption).

As a result, training is most often done under the Local Closed-World Assumption. Under this assumption, negative sampling is typically done by corrupting the subject or object of a known-true triple to another entity that is not in the training set [11, 45]. This also has the effect of making negatives appear more realistic, since they more closely resemble triples in the KG [11, 45].

1.2.5 Biomedical LOD

LOD and KGs have become increasingly popular in biological and biomedical research. The Bio2RDF project focused on converting popular biological databases--relating various fields of biology such as genetics and molecular biology. It was first announced in 2008 and now contains 35 distinct datasets that have been converted into RDF format [1, 12]. The Linked Cancer Genome Atlas (Linked TCGA or LTCGA), published first in 2013, was created to convert data from the Cancer Genome Atlas (TCGA) into RDF format [13, 14]. The project was later vastly expanded in 2014 to include data from PubMed, a large biomedical database maintained by the National Center of Biotechnology Information in the United States [15, 16].

The success of these approaches comes from the naturally linked form of biological and biomedical data. Biological research and knowledge in most cases is understood in a cause-effect manner, linking various events such as mutations, protein interactions, pathway activation, and external stressors to effects such as protein dysfunction, catalysis, cellular division, or apoptosis; this sort or cause-effect based knowledge can be seen very clearly in [17, 18, 20]. Moreover, within cellular biology most events are understood in terms of pathways rather than solitary actions, making the sequential nature of events and interactions fundamental to understanding the system as a whole. It is for this reason that biopathways are presented themselves as graphs, as seen again in [17, 18, 20].

Biomedical data thus has a fundamentally graphical structure, which is easily represented in an RDF format. Yet on top of that, the ability to link disparate datasets, combine structured and unstructured data, and to easily extend this data by adding new triples as more data is discovered make RDF a very natural representational format for big biomedical data. As mentioned above, the use of KGEs to learn from this data allows taking advantage of the fundamental structure of this data, rather than only a subset or select portion of it.

1.3 Overview of this Research

This section contains a summary of the objectives, methods, and results of this work. Section 1.3.1 discusses the goals of this work in the formal context of Knowledge Graph Embeddings and Linked Data. Section 1.3.2 presents an overview of the methodology used, and Section 1.3.3 discusses the major results and outlines the expected contributions of this work.

1.3.1 Research Objectives

There have to date been many attempts to use KGs and KGEs to understand and characterise big biomedical data [5, 7, 13, 15, 21-24, 27, 39-43]. Almost all of these have focused upon using queries on KGs directly without machine learning models to help researchers understand and learn from the data [5, 7, 13, 15, 21-24, 39-42], while many fewer have made use of KGEs a principal goal of their work [27, 43]. However, to date, multiple massive biomedical LOD sources and projects exist [1, 13, 15], which provide a strong basis for the application of KGEs to the field.

The selection of the proper model and hyperparameters are fundamental to the creation and the success of any KGE system [8, 11, 28]. It has been established that KGs are very sensitive to hyperparameter choice, but that the grid search needed to select them can be very time consuming [43]. Moreover, several elements of KG structure have been identified not only as defining characteristics of the KG type [46], but as critical to producing optimal embeddings in neural-network based KGEs [47]. Many of the KGs examined in this work had a similar structure, which motivated an attempt to find a common set of hyperparameters that could be broadly applicable to various biomedical datasets. Thus, the methodology and analysis were aimed at characterizing the nature of the structural similarity between biological KGs, and exploiting this similarity to find a set of hyperparameters that could operate effectively across datasets of similar structure.

As such, the objective of this research was guided by a single central question: **To what extent can the structural properties of Knowledge Graphs explain their performance under different hyperparameter configurations, and can this be exploited to find a single, cross-dataset optimal hyperparameter set for biomedical LOD?** The specific structural features examined were the centralities of each node (measured by degree) and the ratios of sinks, sources, and repeated entities in the KG relative to the total number of triples in the KG.

1.3.2 Methodological Overview

In order to address the goal, a set of 5 datasets contained within Bio2RDF were selected to be analyzed by their structure and optimal hyperparameters. The selected datasets were chosen from the domains of cancer and biomedical research; a full description of the metrics by which these were selected will be given in the methodology in Chapter 3. The 5 datasets chosen were: BioPortal [30], DBSNP (Database of Single-Nucleotide Polymorphisms) [31], DrugBank [32], OMIM (Online Mendelian Inheritance in Man) [33], and PharmGKB [34], all of which are contained within Bio2RDF [12].

Two searches were conducted for the optimal algorithms and hyperparameters: in the first case, the AUC score of the model was used to gauge its success. In this instance, the AUC score is a measure of the probability that a true triple will be chosen over any negative triple [29]. The second hyperparameter search attempted to maximise the “r1” score, which is the probability that a true triple would outrank all of the negatives created samples for it specifically under the Local Closed-World Assumption, rather than any possible negative sample [29]. During both hyperparameter search rounds, the datasets were subsetted randomly to ensure they would be able to train sufficiently quickly to enable analysis and to enable proceeding to further rounds of analysis.

In both searches, hyperparameters were initialized with arbitrary default values, and then updated in three main rounds. In the first round, hyperparameters relating to how the data was modelled were varied and the optimal value was selected in a grid-like search. In the second round, hyperparameters for batching and negative sampling were varied and selected; in the third and final round, hyperparameters for embedding dimension and number of epochs were selected. The choice of three rounds was based upon two key principles: focusing on the most determinant model features first and reducing the number of hyperparameter combinations in any step to a relatively small value measured in the tens or low hundreds rather than thousands or beyond.

In terms of the first principle, all of the model choices that defined the algorithm were taken as the most determinant of how the model represents knowledge. Every different method of embedding, comparing, and learning (through the loss function) entities is the very base upon which KGEs are able to model knowledge, and the most important distinguishing elements between different approaches [45]. Thus, these were selected first. Negative sampling and batching methods were sampled next as representative of how the model learns to distinguish truth and falsehood after it has a knowledge model structure [45]. Thus, the remaining hyperparameters (embedding dimension and epochs) were left to the end once the way the model represents knowledge, and learns to identify truth were established.

In the case of the second principle, reducing the number of combinations to consider at any one step was particularly important since time available was a very critical factor, and since every additional combination to search increased multiplied the running time of the search several-fold. In the final form after this level of time optimisation, the searches took between 1 to 3 days to finish in all cases.

Each search was run three times and the results were averaged and analyzed. This led to the determination that the r1-based search yielded much better KGE models than the AUC based-search, and as thus the results of that search were carried forwards. Once a final pool of validated hyperparameters was obtained, the full datasets were trained and evaluated using those hyperparameters. In the end, two sets were curated with all datasets using either one set or the other. Specifically, one was hypothesized to be better on BioPortal alone, and the other hypothesized to be better on the other four datasets (DBSNP, DrugBank, OMIM, and PharmGKB).

Metrics on the structure of those KGs--notably measures of entity degrees and the numbers of source / sink nodes--were calculated and analyzed to determine patterns and to understand what similarities in their structures were most indicative of being well-suited to a given set of hyperparameters. Four additional datasets were selected for this round of analysis, which were not used at any phase of the hyperparameter searches. They were similarly chosen based on relevance to cellular and biomedical data. These datasets were GOA (Gene Ontology Annotations) [35], HGNC (HUGO Gene Nomenclature Committee) [36], KEGG (Kyoto Encyclopedia of Genes and Genomes) [25], and the LSR (the Life Sciences Resource Registry) [37], all of which are also contained within Bio2RDF [12]. Note that since evaluation by r1 was preferred, only the two hyperparameter sets from that evaluation round were used.

While several different structural metrics were examined, the distribution of the degrees of nodes (also referred to as ‘centrality’) was found to be a particularly relevant feature for predicting relative performance under the selected models. The set of results of this level of analysis was then used to analyze the relative performance of each dataset under different sets of hyperparameters.

The findings from this final round, in tandem with the performances of the datasets used in the hyperparameter search, suggest a very strong influence of structure on KGE hyperparameter choice, and on the performance of any given dataset under a fixed hyperparameter configuration. Taking this into account, several final conclusions on the interaction between structure and performance were drawn, and a final list of major contributions and directions for future research was compiled.

1.3.3 Results and Contribution

The datasets examined in this work were found to have a very high proportion of sinks (nodes that only ever were objects) relative to entities that appeared as both subjects and objects. It was similarly found that the distribution of degrees of nodes was highly skewed left. In total, the ratio of sink and repeats to the total number of triples in the KG, as well as the extent of the skewness in the centrality distribution, were found to be very powerful predictors for KGE performance by all metrics measured under two different hyperparameter sets. Moreover, these features were found to have moderate predictive power in determining the difference in efficacy between the two hyperparameter sets, which allows the construction of a model to predict the best hyperparameter set given KG structure.

However, this analysis also revealed that neither of the two candidate hyperparameter sets could be considered optimal with confidence despite the clear effect of structure on them. This sub-optimality was particularly noted even in the context of the datasets that the hyperparameter configurations were designed for. In addition, in the context of the selected hyperparameters for optimizing the r1 score, the r1 scores for the trained models tended to be quite low with none of the datasets exceeding an r1 of 0.4. KGE AUC scores fared somewhat better, but none exceeded 0.85 in the final models.

Overall, these results show a strong effect of structure on optimal hyperparameter choice and emphasize that similar structured datasets can be placed into classes that operate optimally under the same hyperparameter configurations. However, the relative weakness of the predictive power of these models indicates that much work remains to be done regarding optimizing predictive power in datasets that exhibit such a strong left skewness of entity degree values. In addition, the poor fit of the hyperparameter search to finding ideal hyperparameters means that the results of this work should be interpreted in terms of structural and predictive power of the KGEs, not in terms of having found optimal hyperparameter sets.

The main contribution of this dissertation to the field is twofold. For bioinformaticians, it is intended to create an easily-applicable KGE framework that can be applied to narrow the hyperparameter search space when designing KGE models. By doing so, it will also reduce the technical expertise needed to run KGE models and open up KGE use to less-technically oriented biological researchers. For relational learning researchers, it is intended to highlight the importance of KG structure on KGE models by demonstrating that similarly-structured KGs perform similarly under identical hyperparameter choices. In addition to that, this work further suggests to relational learning researchers that graph structure and embedding models are fundamentally related and should be understood in context of each other.

1.4 Outline of the Remainder of the Dissertation

The remainder of this dissertation is structured as follows. Chapter 2 provides a literature review of related works in the field of KGEs for biomedical data, with a focus on the gap in current literature. Chapter 3 explains in detail the methodology used, and is followed by Chapters 4 and 5, which give an analysis of the hyperparameter searches based on both the AUC and r1 statistics respectively. Chapter 6 presents the results of these models and explains how they relate to the structure of the various knowledge graphs investigated. Chapter 7 concludes the work with a discussion of these results and their significance both to biomedical research and to relational learning. It also describes important future directions and research questions opened up by this work.

2 Literature Review

2.1 Outline of the Literature Review

This chapter will focus on the details of biomedical LOD, KGs, KGEs, and graph structural analysis needed to provide full context and background for the research at hand. Section 2.2 describes two of the largest, most comprehensive biological LOD datasets with an eye to their structure, motivations, and uses. Section 2.3 reviews how various groups have used these and other datasets, as well as the overall RDF framework, to draw conclusions about biopathways, drugs, and treatments. Section 2.4 transitions into the details of KGEs, including a summary of common algorithms and open-source libraries for running them. Section 2.5 discusses graph structure metrics and features and provides a background of research in the area as it relates to how KGEs interact with various graph structural features. Section 2.6 concludes the chapter with a formal identification of the gap in modern research this dissertation aims to fill, as well as the perceived benefits of addressing this gap.

2.2 Major Biological LOD Projects

Many biological and biomedical LOD projects have begun and been extended throughout the last 15 years. Among these are Bio2RDF [1] and the LTCGA [13, 15]. Both of these projects are very notable for incorporating within them a great number of datasets; Bio2RDF currently contains 35 datasets on various fields of the life sciences [12]. The LTCGA contains the National Center for Biotechnology Information (NCBI) Cancer Genome Atlas (TCGA) [13] and was later expanded with a large aggregation of data from PubMed, an unstructured repository of biomedical research articles [15]. Both of these projects were created to help bring the power of Linked Data--notably, being able to integrate structured and unstructured data from distributed data sources--to life science research [1, 13, 15].

2.2.1 LTCGA

The Linked TCGA, or LTCGA, was originally announced in 2013 as a conversion of TCGA data to RDF format [13]. The project focused on ‘type 3’ data in the TCGA--data that had been processed by researchers after being submitted [13]. It ignored type 1 and 2 data, which were respectively raw data and normalized data, since the vast majority of biomedical analysis using TCGA occurs only on type 3 TCGA data [13].

The principal motivation for the LTCGA was twofold. The TCGA is a huge data repository, yet the various data files within it are not linked meaningfully, which means that discovering relationships and following links was a manual process without an immediately clear method for automation [13]. Saleem et al. also identified that while over 350 academic articles had cited the TCGA as of 2013, the majority of these only used raw TCGA data without linking to the wider context of the entire TCGA [13]. Moreover, they aimed to link the TCGA not only to itself, but to other large biological ontologies and KGs such as HGNC (HUGO Gene Nomenclature Committee), OMIM (Online Mendelian Inheritance In Man), and Homologene [13]. Doing so allowed them to take advantage of the full power of Linked Data--being able to incorporate many different datasets into a single LOD knowledgebase.

The LTCGA was later expanded in 2014 to include unstructured text data from research articles on PubMed, which deal with a large variety of biomedical concerns [15]. The motivation for this was to take full advantage of the ever-growing scientific literature on cancer, to link it with other sources of known data in the LTCGA, and to allow easier discovery of article contents through LTCGA queries [15].

The LTCGA was created at first by scraping text files available on the TCGA, cleaning and processing them, and then passing them through a RDFizer to convert them to a Knowledge-Graph format [13]. These RDF files were loaded onto a publicly-available SPARQL endpoint that allowed users to run queries on the data contained within the LTCGA [13]. Moreover, Saleem et al. took advantage of link-prediction tools to automatically link the LTCGA to HGNC, OMIM, and Homologene after its construction [13].

In order to expand the LTCGA to include PubMed, they scanned the text of all PubMed articles and selected those containing keywords related to cancer [15]. The article meta-data, as well as any named genes or diseases in the abstract, were added as records to the LTCGA to allow easy article discovery on topics contained in the LTCGA [15]. Moreover, since the LTCGA had by that point expanded to 20 billion triples and was projected to reach 30 billion, Saleem et al. created a federated query mechanism across 17 different SPARQL endpoints by which all of this data could be accessed [15].

Finally, Saleem at el. created a dashboard and various visualization systems to sit on top of the expanded LTCGA, with the aim of making data access simple for clinicians and researchers [15]. However, in all their work their focus remained principally on how to enable human-based analysis and querying, rather than also taking advantage of automated learning and information extraction from the information available [13, 15].

With all of these features, the expanded LTCGA is clearly a very powerful tool for biomedical research, particularly with regards to cancer. However, at the time this research was done, almost all of their endpoints were down and the main links contained within their articles were broken, which prevented taking advantage of this work. While some data was later found to be accessible in backup dumps on their website, recovery of the entire project was not possible through their website, especially as many of the download links contained there were broken. As a result, it was impractical to perform this research with the LTCGA system.

2.2.2 Bio2RDF

Bio2RDF was created in 2008 with similar intent to the LTCGA: to create a large repository of linked biological data that could be easily accessed and used with LOD and KG technologies [1]. However, its vision encompassed more than just one topic alone, and aimed to aggregate as many sources as possible related to the life sciences into an RDF format, and to publish it as fully-accessible LOD [1]. Their goal was not to allow the data-system to only answer specific problem-oriented questions, but to allow a more exploratory analysis of many fields using the collected data [1]. Belleau et al. described their work as a “mashup” of many different biological databases, which now includes 35 datasets, notably including BioPortal, DBSNP, DrugBank, KEGG, OMIM, PharmGKB, GOA, HGNC, LSR, and several from the National Center for Biotechnology Information [1, 12]. A full list of the datasets contained within Bio2RDF, as well as their provenance and basic contents details, can be found at [12].

Bio2RDF explicitly designed an ontology for the datasets collected to label them and assign various entities and relationships to classes; this was done using the Web Ontology Language (OWL) framework [1]. They created their own namespace and identification method to disambiguate various references, and RDF-ized all target datasets into this final format [1].

All of the data they collected was made available on the Internet both in HTML and RDF formats, and their raw RDF files were maintained and made available for downloading [1, 12]. Much like the LTCGA, the original intent of Bio2RDF was to facilitate querying these different databases by human experts and researchers [1]. In one case, Belleau et al. describe the power of the SPARQL query engine that sat on top of their RDF resources, but did not address automated relational learning or KGEs [1].

However, the project's focus on comparability, extensibility, and long-term maintenance [1, 12] have resulted in their collected data still being very readily accessible on the Web [12]. Moreover, Bio2RDF currently contains a large variety of Linked Data from various sources [12], making it an attractive source for data on automated relational learning of biomedical and biological data. As a result, this large data store is a very natural choice for use in this research. In the methodology section, further details will be provided on the use of Bio2RDf as the source of RDF files to be used to produce predictive KGE models.

2.2.3 Other Projects

Many other biomedical and biological LOD projects exist. Many, as previously mentioned, have since been incorporated into Bio2RDF or other datasets-of-datasets. However, a great many have not, or have been linked in a data mashup and also maintain their own independent Knowledge Graphs. A brief summary of several other common biological RDF datasets is listed here, both to give a complete overview of the state-of-the-art resources as well as to provide context for the later discussion on future directions of extending this work to a larger range of datasets.

2.2.3.1 NCI Thesaurus

The NCI Thesaurus is a large collection of terminologies relating to cancer, pharmaceutical drugs, anatomy, and biomedical, cellular, and molecular concepts [38]. It also contains several terms on common experimental organisms used in research [38]. It was created by the National Cancer Institute in the United States to serve as a common, controlled source of terminology for research and bioinformatics publications to ensure that data was consistent and easily accessible [38].

The NCI Thesaurus is based on the Web Ontology Language (OWL), which is a part of the RDF ecosystem [38, 50]. OWL serves to identify synonymous terms and establish how they relate to each other--such as if two relationships are inverses of each other, or if a relationship has the transitive property [50]. The vocabulary was designed to support all the common statements that a Knowledge Graph may contain about biomedical data, such as for expressing protein interactions or purported effects of diseases [38]. However, it was not created as a database but rather as a base of knowledge on how terms relate [38]. As a result, it is a very powerful tool for determining how concepts relate, rather than for predicting new facts about a given disease or about biological pathways.

2.2.3.2 miRBase

miRBase is a LOD adaptation of two databases focused on microRNAs and their biological functions [39]. They wrote their own RDF-izer to convert the data contained therein to the RDF format, with an end goal of aiding research specifically regarding the functions and inter-relationships of microRNAs [39]. Moreover, the group used text analysis on PubMed articles to find associations between miRNAs in their databases with various diseases [39].

Much like LTCGA and Bio2RDF, they made the data publicly available through a SPARQL endpoint [39]. In their analysis, they focused on the use of SPARQL queries over the RDF data to extract information and guide research [39]. They limited their examples and case studies to the query-ability of the data rather than to higher-order learning tasks upon the data, but even within that context were able to demonstrate that the resulting database would be a useful research tool [39].

At the time of writing, the links to the miRBase servers in [39] were broken, preventing access to any of the data or query systems created by the authors.

2.2.3.3 BIOOPENER

First presented in 2017, BIOOPENER was created by the authors of the LTCGA and with many of the same goals [7]. The authors describe its goals as being largely similar to Bio2RDF in that they aimed to create a mashup of many different data sources, linked using RDF, and to enable researchers to take advantage of this linking to discover new facts [7]. BIOOPENER was specifically created in the context of evaluation of cancer data and focused on collecting data in RDF format from sources highly relevant to cancer research such as the TCGA [7].

They created novel links between the member datasets of BIOOPENER to explicitly model identical elements from different datasets [7]. This was done using properties of the datasets--such as various unique gene IDs--that allowed matching elements with high certainty [7]. Since the resulting combination of these datasets was very large, they separated it into multiple components and employed a federated query engine to allow access to all the data hosted at different endpoints [7].

Unfortunately, much like LTCGA, many of the links to the project given in [7] were broken at the time of writing, making taking advantage of what otherwise seems like an invaluable resource impossible to use for the research at hand.

2.3 Biomedical Research based upon KGs

To date, there have been many studies attempting to use the power of LOD and KGs to produce new biological insights [5, 7, 13, 15, 21-24, 27, 39-43]. The vast majority of these have focused on using the properties of linked data alone to query information, retrieve results and facilitate manual data exploitation and analysis without using KGEs or relational learning techniques [5, 7, 13, 15, 21-24, 39-42]. However, some articles did find that KGEs can be a very natural perspective for understanding and processing large-scale biomedical KGs and were able to produce clinically relevant results using those techniques [27, 43].

This section gives an overview of the most common approaches to manual knowledge graph exploratory analysis and use with an eye to the benefits and drawbacks of the manual elements. From there, it proceeds to compare these results with automated KGE-based methods and the insights those methods generate.

2.3.1 Manual Exploratory Methods based on KGs

The most common manual exploratory methods can be divided into three approaches: those based on database queries alone, those that build visualization and automation tools on top of queryable endpoints to facilitate end-user access, and those that are not based upon database queries. These three approaches are outlined in sections 2.3.1.1, 2.3.1.2, and 2.3.1.3 respectively.

2.3.1.1 Approaches that Analyzed KGs based on Queries Alone

The work of Zhao et al. was centered around using a KG approach to understand and summarize large amounts of clinical data [23]. Zhao et al. used a natural-language processing pipeline on genetic reports from cancer patients to identify cancer-related terms from unstructured textual data [23]. Relevant terms, such as those relating to pathways, genes, and drugs, were identified and their co-occurrence within sentences was used to establish a network of relationships among the terms in a graphical format [23]. They proceeded to demonstrate--both through review by medical professionals and through direct analysis of the output graph--that the resultant knowledge graph was able to effectively identify gene mutations patterns in cancer patients, such as that the tumor-suppressor gene TP53 was mutated in 96% of observed ovarian cancers [23]. The main goal of this system was to enable future re-use of this data in clinical decisions, which would be significantly faster than manually reviewing all of the genetic reports; however, the resultant Knowledge-Graph was not intended to be expanded to cover a larger knowledgebase of the broader cancer or biomedical context [23].

In [24], Hasan et al. created a knowledge graph to contain data from the Louisiana Tumor Registry, particularly with the aim of overcoming the challenges of integrating heterogeneous data, linking disparate data sources, and facilitating the execution of complex queries on the knowledgebase [24]. They demonstrated that several higher-order tasks, such as constructing queries to identify common treatment sequences or to identify population-level occurrences of different types of cancer, were feasible using queries on their graph [24]. Since they built the graph to be easily extensible, they were able to link other datasets, and left the linking of other major biological datasets such as Bio2RDF as future work [24].

Dalamagas et al. created miRBase, a knowledgebase directed specifically at understanding the clinical and pathological functions of microRNAs (miRNAs) [39]. Like the work of Zhao et al., the goals of this KG were very specific and not immediately intended to be generalized across the wider fields of cancer biology, biomedicine, or molecular biology as a whole [23, 39]. The major contribution of miRBase was the ability to quickly search for lists of miRNAs based on type and function [39]. Very notably, miRBase was the only KG system examined that was built from the ground-up with the intent of having very easy version tracking and with the ability to explicitly query older or current versions of the KG very simply [39]. This ability in itself is critical to establishing reproducibility between studies on these knowledgebases, and is one of the most notable and important features of miRBase [39].

The LTCGA, as described before, was designed to link various datasets about cancer into a single large Knowledge Graph [13]. BIOOPENER, a tool based on the TCGA and various other genetics datasets, was a later iteration of this concept build by the same group to create a queryable interface [7]. In fact, BIOOPENER was developed using the results of a similar study by Zehra et al. as a part of the same research project which aimed to convert various biological datasets to RDF and to expose them with an easily-queryable interface [41]. Much like the work of Zhao et al., the principal function of both these systems was in relating and querying data in the LOD format, and all results were presented as query outputs [7]. Among use-cases given are queries for finding methylated gene promoters in DNA sequences and finding methylation changes between healthy and cancerous genes [7], as well as identifying patients who would be most likely responsive to cocktail-based cancer treatment or estimating survivorship of patients based on their clinical data [13]. However, unlike Zhao et al. and Dalamagas et al. the knowledge bases constructed in both these works was significantly larger and more general, containing several major biological knowledge bases (converted to RDF format) as well as patient and clinical data [23, 39, 7, 13].

Detwiler et al. expanded the concept of query-based exploratory analysis by constructing an extended query interface in which variable paths could be more constructed and used [42]. Their goal was to make querying paths as simple as constructing a regular expression, and they extended the default SPARQL query language for RDF datasets to form GLEEN, a language that supports regular-expression-like constructs as part of these queries [42]. This work was done in the context of the NCI Thesaurus [38], and specifically aimed to allow easy access to subgraphs of massive ontologies through regular path exploitation [42]. While the group did not apply this in a clinical context, this work (especially in the light of the query-based applications aforementioned) has great potential to expand the ability to analyze clinically relevant RDF data [42].

These approaches were all based heavily upon SPARQL queries or, more generally, KG queries [7, 13, 23, 24, 39, 42]. As thus, while they were accessible to knowledge engineers and RDF experts, their usability by non-technical biomedical researchers and clinicians was generally limited. Moreover, their dependence on manual queries meant that they were most applicable when researchers could formulate a research question directly in terms of the entities in the graph in the form SPARQL accepts [48]. This in turn meant that researchers would have difficulty taking the entire set of data in the graph into account when running individual queries, requiring multiple queries to get an overall understanding of the data at hand.

2.3.1.2 Approaches that Analyzed KGs based on Higher-order Tools

Many different groups observed that providing a UI, rather than simply a query endpoint, could greatly boost the usability of their LOD products, which are detailed below [22, 15].

In [22], Deus et al. created an endpoint through which SPARQL queries could be run on the TCGA; however, unlike the LTCGA project, they only allowed querying the hierarchy of the TCGA rather than the data contained within its files [22]. In order to make the endpoint accessible to researchers not familiar with SPARQL, they created a HTML-form like interface for constructing queries based on selectable components, essentially guiding the user through query construction [22]. It should be noted that this system still required the construction of queries even though it was facilitated; while this made it more accessible, it still was limited by being a predominantly query-based UI [22]. They made the endpoint public on the Web; however, as of the time of writing, the backend server supporting the endpoint was unreachable [22].

In [15], Saleem et al. expanded on the LTCGA project with a large, multi-faceted data visualization tool to allow exploratory analysis of the data contained in the Knowledge Graph [15]. They incorporated information from the LTCGA and PubMed articles, created a distributed data storage and query system, and then built a network exploration visualization tool and a genomic information browser on top of their knowledgebase [15]. In the network exploitation tool, they used a force-directed layout to display various tumor types linked to publications characterizing them [15]. The visualization was created to allow the user to expand on sub-graphs of interest, and to automatically run SPARQL queries to retrieve basic information on nodes, such as publication metadata [15]. This system allowed human users to more rapidly access and process the information, enabling learning to be done at a human level on the data [15].

The genome browser view they created similarly was designed to take full advantage of automated queries on the KG backend. Information from the Human Genome Project was linked to clinical data and known cancer markers [15]. As data from each clinical case was selected, data would be retrieved from the KG using automated SPARQL queries and displayed on a coordinate grid to facilitate human analysis [15]. As this, this project managed to overcome many of the limitations of query-based formats: using graphical layouts, they allowed human analysis to gain an intuitive understanding of the data as a whole, as well as to explore certain parts in much greater detail [15].

2.3.1.3 Approaches that used non-Query-based Analysis Methods

Some research and development in the area of KGs and biomedical LOD focused on construction of KGs not to run SPARQL queries, but to allow downstream analysis based directly on the graph format of the data.

McCusker et al. [5] constructed a KG containing information on known melanoma cancer pathways as well as on the mechanism of action of pharmaceutical drugs intended to treat a variety of diseases, not only melanoma [5]. They then manually annotated each statement in the graph with a reliability score reflecting how probable it was that the demonstrated relationship was true [5]. Once this graph had been constructed, they searched for drugs that affected known melanoma pathways but that had not been used previously for melanoma and selected only those that had an overall probability over the cutoff value [5]. By doing so, they were able to predict known drugs that could be repurposed to treat melanoma [5]. Moreover, they created an Augmented-Reality-based visualization of the graph structure to facilitate human analysis and pattern finding in the graph as a whole [5]. A web portal to a demonstration of the project, as well as to a callable API, were created but as of the time of writing are no longer accessible on the internet [5].

In [21], Kim et al. created a KG specifically to help prediction of cancer outcomes [21]. The KG was created on a comprehensive set of “multi-omics” data (such as genomic and proteomic data) that could effectively characterize the larger context of a cancer cell rather than only one piece of it [21]. In order to assess the system, they tested it on ovarian cancer data from the TCGA using a graph-based semi-supervised learning approach to predict cancer outcomes for patients based on their data and data contained in the graph [21]. Their model yielded an AUC statistic of 0.7866 [21]. When they integrated data from more sources, such as Gene Ontology, they were able to increase the model AUC to up to above 0.8 depending on the data integrated [21].

While Kim et al. did perform machine learning on their KG, the approach did not use KGEs, nor did it attempt to predict new edges given information already present in the graph [21]. In this sense, the approach was a more problem-specific method built to answer a particular question rather than to gain a general understanding of all data contained in the KG; however, it is still notable for having used machine learning to take into account the entire dataset in its graph format [21].

Overall, these systems provided a greater level of automation and used the entire graph as a whole in its native structure [5, 21]. The generally strong results produced in terms of predicting drug re-use ability [5] and predicting cancer outcomes [21] indicate that such automated methods can yield very medically relevant results even in the absence of traditional lab- or clinical-based experimentation [5, 21].

2.3.2 Automated analysis of KGs via KGEs

In contrast to the large number of groups that attempted to use query-based methods to assist human analysis of the dataset or that constructed problem-specific KG-based models, relatively few attempted to use KGEs to gain a comprehensive learned model of all the data contained in the KG. Notably, Celebi et al. [27] and Mohamed et al. [43] both consider the application of KGEs in the context of analysis of drug interactions and side-effects, an issue which has very high complexity and can only be fully understood by considering the full network of biological pathways and interactions at play.

Celebi et al. analyzed several different KGE algorithms to predict drug-drug interactions for pairs of drugs with no known interaction data [27]. In particular, they examined the RDF2Vec, TransE, and TransD algorithms for embedding knowledge graph entities and relationships, and used the DrugBank, PharmGKB, and KERG datasets as their knowledgebases [27]. Interestingly, their goal in constructing KGEs was not for link prediction, but to use the embeddings as input feature vectors to various classification algorithms (Random Forest, Logistic Regression, and Naive Bayes) [27], a task for which the KGEs were not directly trained [11, 45]. Notwithstanding, it is expected that a good KGE model will learn latent features of the dataset and gain some form of intuition-like understanding of the data [11, 45], which means that the embeddings can be expected to be applicable in other, non-link-prediction tasks. Indeed, Celebi et al. found that the RDF2Vec algorithm, in combination with a Random Forest Classifier, was most effective at predicting whether two drugs would have off-target interactions; the AUC of this prediction score was 0.93, indicating that the model was largely successful in predicting the presence or lack of interactions [27].

In [43], Mohamed et al. apply KGE models to two use-cases: predicting the targets of pharmaceutical drugs (modelled as a link prediction task) and predicting side-effects caused by the co-use of two drugs as a cocktail; in both cases, training and evaluation was based on link-prediction accuracy [43]. Their KG data sources for these tasks were DrugBank,

DrugBank_FDA, KEGG, and UNIPROT. Examining a variety of KGE algorithms, they found that TriModel, ComplEx, and DistMult were the most effective at both predicting drug targets and predicting side effects of drug cocktails [43]. Moreover, they found that these knowledge-graph embedding models produced results comparable to, or better than, the current state of the art predictors for both tasks [43].

Interestingly, Mohamed et al. also suggested several other applications of KGEs: measuring similarity between different entities in the graph by using distance measures between the various embedded nodes and allowing clustering algorithms to be run on embedded biological entities such as proteins that require transformation to vector space to be used as an input vector to clustering algorithms [43]. They also found that KGEs can be very sensitive to changes in hyperparameters, especially embedding size dimension, whose optimal value tended to be proportional to the overall size of the KG being learned [43]. They also noted that finding optimal hyperparameters is often a very time-consuming task, requiring in most cases a brute-force grid search of all possible values [43].

These findings are particularly interesting in the context of the work of McCusker et al., who also focused on predicting drug-use patterns, albeit in the different context of drug repurposing [5]. All three studies yielded very relevant pharmaceutical results, which is especially notable since none of their research involved any lab or clinical work [5, 27, 43]. However, while the work of McCusker et al. required very large amounts of manual work to rate various interactions [5], the works of Celebi et al. and Mohamed et al., being based on more KGEs, were much more automatable and as a result were more automated [27, 43]. Of particular note is Mohamed et al.'s reference to the high time cost of running a grid search to determine proper hyperparameters to new biomedical datasets [43], the exact problem that this research seeks to address.

2.3.3 Key Take-aways from the Literature

The current literature in KGs and KGEs has established that both human-based KG analysis and automated learning through KGEs or semi-supervised learning can be very effective in allowing the discovery of new facts about biological and biomedical systems [5, 7, 13, 15, 21-24, 27, 39-43]. There is a particular divide between those approaches that are designed to augment human reasoning and understanding, such as query-based interfaces and higher-order visualization methods [7, 13, 15, 22-24, 39-42], and those that are designed to take advantage of automated learning techniques with less analytical or explanatory input from humans [5, 21, 27, 43].

Many of the articles focused on creating exploratory visualizations noted the difficulty for those not familiar with SPARQL to use SPARQL endpoints to their full extent [15, 22]. These approaches have the benefit of greatly enhancing human understanding of the data, opening it up to easy use by clinicians, doctors, and researchers not familiar with KGs. Particularly in the case of BIOOPENER, in which the visualization efficacy was directly measured, it has been demonstrated that these approaches can be beneficial to end-use [15]. Unfortunately, there is a common trend that many of the endpoints and visualizations created by these groups become unavailable a few years after publication, as happened with [7], [15], [22], and [39].

Overall, the integration of KGs, KGEs, and biomedical research is still growing. However, as many of these recent projects show, it promises a wealth of benefits both to theoretical research, clinical practice, and to our understanding of machine learning [7, 13, 15, 27, 43]. The issue, however, of finding optimal hyperparameters to new KG datasets is noted particularly as a high cost of such research [43], and thus provides further motivation for this work.

The next two sections will explain in further detail the inner-workings of KGE algorithms (section 2.4), as well as a background in graph structural characteristics and how those have been shown to interact with graph-based machine learning (section 2.5).

2.4 Knowledge Graph Embeddings: Algorithms and Implementations

2.4.1 Knowledge Graph Embedding Algorithms

There are many different approaches to producing KGEs given a KG, which vary not only on how they represent data, but on how they use the embeddings to predict new data and on what extra data, if any, they incorporate [45].

The most basic conceptualization of KGEs is the TransE model [11, 45]. Under the TransE model, nodes are embedded as vectors, and relationships as vector displacements between those nodes. If s_i is the embedding of a subject node, p_i the embedding of a relationship, and o_i the embedding of an object node, then TransE attempts to enforce the objective function: $s_i + r_i = o_i$ [11, 45]. In essence, this gives a very intuitive definition of embeddings: the subject (entity) plus the predicate (how the subject relates to the predicate) should be close or equal to the object [11, 45].

While this model is very simple and intuitive, it fails to account for the full complexity that can be observed in many knowledge graphs, particularly in that it fails when relationships may be 1-to-many, many-to-many, or many-to-1 [45]. For example, suppose TransE learns the following triples which have a many-to-1 relationship:

Subject	**Predicate**	**Object**
P53	is-a	Protein
RNA Polymerase	is-a	Protein
RAS	is-a	Protein

It will model P53, RNA Polymerase, and RAS in a similar region of vector space so that the embedding of the “is-a” relationship maps them all to the single node for the embedding of “protein.” However, these entities are nothing like each other: P53 is well-known to be a tumor suppressor gene (protecting against cancer), RNA Polymerase a protein involved in the expression of genes (in both cancer and healthy cells), and RAS a known oncogenic protein (being able to drive cancer cell growth when over-activated). As thus, TransE in this case would be learning a representation of these entities that work only in the context of understanding that they are proteins, but that will reduce the model's performance when the functions of the proteins must be considered.

Most other common KGE models are attempts to evolve TransE into a form that retains its simplicity (as much as is possible) while allowing for them to model complex and high-cardinality relationships that TransE cannot perfectly describe [45]. TransH expands TransE by modelling the subjects and objects differently for each relationship [45]. It does this by projecting them onto a hyperplane in which the relationship lives, and then attempting to co-localize s_i + t_i a and o_i on that hyperplane [45]. Since head and tail embeddings are dependent on the relationship, this overcomes the cardinality issue faced by TransE [45]. TransR follows a very similar concept as TransH, except that it uses matrix multiplication to project head and tail embeddings into a vector space specific to each relationship, rather than a hyperplane specific to each relationship [45]. However, TransR must train many more parameters due to the use of a whole matrix multiply, and so modifications to TransR, such as TransD and TranSparse, typically attempt to reduce the space and time complexity of the algorithm [45].

Other variations of TransE attempt to allow matching subjects and predicates to their associated objects in embedding space without requiring that $s_i + r_i = o_i$ hold in any space (original, projected, or otherwise) [45]. For example, TransM introduces a relation-specific weight θ_i that is used to weight how close to the object embedding the value of $s_i + r_i$ must be, with lower values meaning it may lie farther away [45]. TransF only enforces that $s_i + r_i$ be in the same direction as o_i, regardless of location in space. Many other algorithms exist along this line, all of which attempt to allow inexact colocalization in space such that the cardinality of relationships does not affect the model negatively [45].

An entirely different set of models, called “Semantic Matching Models”, use similarities and dissimilarities rather than distances to match subject-predicate combinations with object embeddings [45]. These dissimilarities need not be measures of distance in space, so long as they are generally lower for embeddings that are more similar and higher for those that are more different. Among the algorithms in this class are RESCAL, which represents relationships with matrices and entities with vectors [45]. In the case of RESCAL, pairwise multiplication of all elements in the matrix with elements in both vectors is used as a dissimilarity measure. However, due to the use of a matrix, the number of parameters needed for each extra dimension is quadratic rather than linear [45].

The DistMult model attempts to correct the high time complexity of RESCAL by using only a diagonal matrix to represent relationships [45]. However, in its base implementation the multiplication is symmetric, which means that it cannot model asymmetric relationships as RESCAL (and the Trans model family) can [45]. The ComplEx model was created as a reaction to this; it uses embeddings in complex space rather than real space [45]. However, since the objective function extracts the real-values components only after performing transformations on the embeddings, it results in a non-symmetric function that can operate effectively on asymmetric relationships [45]. Thus, ComplEx can often model a variety of relationships that DistMult cannot [45].

Beyond these models, others have proposed to use probabilistic embeddings or neural networks to attempt to improve embedding quality [11, 45]. However, such algorithms are outside of the scope of this research, which focuses on the traditional distance and dissimilarity models of KGEs.

2.4.2 Knowledge Graph Embedding Algorithm Implementations

Many different implementations of KGEs have been made available as Open-Source software. However one such project, BioKEEN, was developed specifically to assist with running KGE models on biological datasets, coming with automated default hyperparameter selection to make it simple to use for researchers with little background in machine learning [28]. BioKEEN was later succeeded by PyKEEN, a project developed by the same group that aimed to generalize BioKEEN and expose the ability to use more advanced configuration options when running a KGE algorithm [44]. However, both BioKEEN and PyKEEN had a major limitation: they were limited by the RAM of the host computer, unable to train KGEs on KGs of any greater size [28, 44]. However, many biological datasets far exceed the RAM capacity of commercial computers; KGs such as the LTCGA or BIOOPENER contain billions of triples [7, 13], and from the subsets of LTCGA project that was still accessible online, the it alone contained datafiles measuring on the order of 13 TB. Running a KGE model on such a dataset--even with distributed memory management--would be an infeasible task on even some of the best modern computers. However, even subsets of this data would far exceed the 16 or 32 GB present in most commercial computers. Even smaller datasets, such as the Bio2RDF version of BioPortal, measured in excess of 18 GB when in their uncompressed forms.

A separate implementation of KGEs, PyTorch-BigGraph (PBG), was created to address this problem [8]. Also released as Open-Source software, PBG allows partitioning the graph into subgraphs, where each partition can be of a size that fits in system memory [8]. Each such partition is used for training, and then different partitions are swapped until all data has been used in training, which represents the end of a single epoch [8]. While such an approach clearly will only approximate the more-optimal model result obtained through non-partitioned training, Lerer et al. found that these differences were minimal for large datasets [8]. Moreover, PBG allows training KGEs on KGs of any theoretical size, so long as the entire graph can fit on the computer’s hard disk [8], at which point processing power limitations would almost certainly be more relevant than memory limits.

As described later in the methodology in Chapter 3, this research was conducted using PBG both to allow training on larger datasets such as BioPortal as well as to investigate the interactions between partitioning and KGE results in the context of optimal hyperparameter choices and graph structure.

2.5 Graph Structure Metrics and KGEs

Since KGEs are fundamentally connected to the structure of a graph, such as with how TransE can be negatively affected by high-cardinality relationships [45], searching for a set of hyperparameters also attentive to graph structure is a natural direction for research. However, to date relatively little work has been done in this context.

Ellefi et al. profiled a variety of Knowledge Graphs based on key features such as provenance, statistical and structural features, licensing, and dynamics [46]. Among the statistical and structural characteristics they noted as being most relevant were sizes of the different classes of nodes, class and predicate use per subject, depth of type and predicate hierarchies, URI usage per entity, and the number of triples containing literals [46]. However, the group did not aim to connect these features to machine learning or KGE models, but only to note that they were among the more significant features by which KGs could be classified and understood [46].

Sadeghi et al. observed that some structural quantities, such as centrality, could impact how well various relational learning systems performed [47]. However, they focused on creating a relational learning model that represented nodes not only as embeddings, but with knowledge of their centrality and hop-distance from other reference nodes [47]. In doing so, they were able to create a neural network model that accounted not only for the immediate context of nodes (such as the object matching a subject and a predicate) but to allow it to take into account some features of the overall structure and form of the graph [47]. They concluded that this yielded much better results than methods that did not take overall graph structure into account, beating even former state-of-the-art methods [47].

From the work of Sadeghi et al. came two facts that are very relevant to the work at hand; first, that traditional relational learning via KGEs is very attentive to the local (one-hop) area of nodes, but unable to detect major patterns outside that region [47]. Second, that centrality as a feature is critical to understanding and optimizing graph-based learning [47]. In this context, centrality is a measure of how connected a node is to the rest of the graph; many measures of centrality exist, including degree of the node [47]. Their observation regarding the relevance of centrality in particular suggests that finding hyperparameters for graphs based on centrality as a statistic for graph structure might result in reliable cross-dataset performance. A similar trend of counting the number of occurrences of nodes given properties (or vice versa) was also evident in the profiling system developed by Ellefi et al. [46], suggesting that using degree measure distributions could be a valuable approach for summarizing graph structure. As mentioned in methodology, node degree (as a measure of centrality) was the one of the most prominent structural statistics used in this work to estimate optimal cross-dataset hyperparameters.

2.6 Key Gap in the State-of-the-Art

The KGE models that have been applied have focused on producing a set of embeddings either for use as feature-vector inputs to a different machine learning application [27, 43] or directly for novel link prediction within the learned dataset [43]. While both of these approaches are very valuable, both run into the inevitable issue of finding the correct hyperparameters, which often required a brute-force search [43]. These methods also tend to be focused on specific applications [22, 43], although making more general link predictions using the KGEs produced is also possible [11, 45]. KGE-based methods, however, have a greater up-front barrier to entry than human-based visualization tools do, since finding ideal hyperparameters and training the model can both be very time-consuming processes [43].

The current literature on KGs and KGEs overall shows a field that is still maturing, but one that also has much potential for biomedical application even now. The recent trend towards both more advanced visualization, as in the transition from the LTCGA to BIOOPENER [7, 13, 15], as well as interest in automated KGE models [27, 43] are producing results that commonly match or exceed other modern approaches in data visualization and prediction [7, 43].

In the case of this dissertation, focus was given specifically at the automated KGE approach to modelling data, particularly with an eye to how to reduce the up-front barrier to entry posed by hyperparameter selection. By identifying the extent to which common hyperparameters can be applied to similarly structured KGs, and by understanding what elements of structure are most relevant to hyperparameters choice and ultimate model performance, it creates a framework that could allow automated learning of Knowledge Graphs to be performed more quickly and with less need for machine learning expertise.

While various articles have established that KGEs are very sensitive to good hyperparameter choice [43], characterized the most important meta-elements of graphs such as structure and provenance [46], and noted the effect of graph structure on embeddings [47], no previous attempt has been found to identify a common set of optimal or near-optimal hyperparameters for KGE models that could be applied across datasets. Furthermore, no attempts have been made to relate these hyperparameters to KG structural features for use in model design, model choice, or final performance prediction tasks. Determining whether such a set of hyperparameters exist, and what they would be if so, would thus be a contribution both to the field of knowledge-graph machine learning and to the field of bioinformatics. Moreover, relating these hyperparameters to structure would expand the field's understanding of knowledge representation systems and why hyperparameters' optimal values vary as they do. As thus, this research has been aligned with the end goal of answering the question of to what extent hyperparameters can work optimally or near-optimally across biomedical datasets, and what structural characteristics can explain this effect.

3 Methodology

3.1 Outline of the Methodology

Section 3.2 provides an outline of data sources and KGs selected for use in this work, with a detailed analysis of how and why they were chosen. Section 3.3 continues to explain how the data was obtained from online repositories. and preprocessed. Section 3.4 discusses the choice of KGE algorithm implementations and the specific metrics considered when evaluating the available options. Section 3.5 describes the methods used to select hyperparameters to the chosen datasets, and contains sub-sections outlining the two major hyperparameter searches conducted based on optimizing the AUC or r1 statistics respectively. Section 3.6 concludes the methodology with an explanation of how structural analysis of the KGs was carried out.

Please note, all code used to collect data, preprocess it, perform analysis, and identify structural patterns is made available in a zip file submitted with the dissertation, as well as on GitHub at the following link: https://github.com/Jeffrey-Sardina/cod-trachtais.

3.2 Selection of LOD Data Sources

3.2.1 Selection of a Biological LOD Mashup

Selection of data sources occurred in two steps: selection of a multi-dataset LOD mashup, and selecting datasets from within that mashup. Selection of data from a single mashup rather than from solitary KGs was done for four reasons: simplicity, ease of reproducibility, relevance, and consistency.

Simplicity and ease of reproducibility for LOD-based projects go hand in hand. Mashup systems such as the LTCGA and Bio2RDF are intended to be used in many different contexts and by different applications [1, 13, 15]. Moreover, they are designed for easy access of their components by researchers [1, 13, 15]. Both of these attributes make the datasets very attractive in terms of simplicity: all the data is easy to obtain from a single place. Moreover, ease of access to the data is a great boost to ease of reproducibility, since other groups who wish to reproduce the results of this work need only reference data from a single location rather than many. Since there is an unfortunately common trend for biological LOD endpoints and data availability to fall offline, as noted in the literature review, a single point of access to a reliable data source reduces the chance that some of the relevant datasets later become inaccessible.

In addition, these larger data mashups tend to have much higher overall relevance. Bio2RDF, for example, was constructed from the most commonly used biological datasets [1], and now contains a total of 35 datasets [12]. Similarly, the LTCGA was built upon the TCGA [13] and PubMed [15], which are among the most commonly used and critical resources in cancer bioinformatics. This makes these projects highly relevant, since they contain the most in-demand modern biological data. Other projects, such as miRBase [39] and the NCI Thesaurus [38], while very useful for their given domains, are not as broadly applicable and thus have a lower relevance to the biomedical LOD community as a whole.

Finally, consistency between datasets was a critical factor for selection. Since mashups are intended to link all the data contained within them [1, 7, 13, 15], all their graphs are naturally linked together and need little to no modification to be used with each other if required. However, projects such as miRBase [39] would have to be computationally linked to other, newer datasets, which increases the time needed to get the full data system operational. The fact that Bio2RDF and the LTCGA had already established links and namespaces means that data consistency comes built-in, rather than being another pre-processing step [1, 13].

As mentioned in the literature review, the largest and most relevant biomedical dataset mashups in this domain are LTCGA [13, 15], BIOOPENER [7], and Bio2RDF [1]. The original intent of this research had been to apply this work specifically in a cancer biomedical context, which naturally suggested the LTCGA and BIOOPENER projects. However, the afore-mentioned inaccessibility of data from the LTCGA and from BIOOPENER left only a single feasible choice for a biological LOD mashup: Bio2RDF. As such, all data sources were taken from the Bio2RDF project.

3.2.2 Selection of datasets from Bio2RDF

Selection of datasets from within Bio2RDF was based upon two key principles: relevance to cancer and biomedical research and size of the whole dataset.

In terms of the first criterion, datasets from Bio2RDF that were immediately relevant to cancer and biomedical research were selected. These domains, of course, are very broad and can encompass a variety of types and sources of data. Specifically, in order to select the datasets most relevant to these categories, datasets containing information on drugs, molecular biology, clinical data, and genetics were selected as the most highly relevant. Datasets containing data exclusively from non-human animals were excluded. This left a list of 15 potential datasets. The sizes of all raw data, after being downloaded and uncompressed as described in Section 3.3, were measured and linked to each dataset. This information, as well as the purpose of each dataset as described in [12], are detailed in Table 3.1.

Dataset	**Contents**	**Uncompressed RDF Size**
BioPortal	A variety of biomedical ontologies useful in bioinformatics and biomedical research [12]	18.3 GB
Clinical Trials	Results of human-based clinical studies from public and private sources [12]	46.2 GB
Comparative Toxicogenomics Database (ctd)	Information on chemical-gene and chemical-drug interactions from multiple species [12]	97.5 GB
Database of single nucleotide polymorphism (DBSNP)	Information on common single nucleotide polymorphisms and other important short DNA mutations [12]	2.9 GB
DrugBank	Data on drugs and their targets [12]	1.6 GB
Gene Ontology Annotation (GOA)	Ontologies of genes sourced from the UniProt Knowledgebase and the International Protein Index [12]	17.1 GB
HUGO Gene Nomenclature Committee (HGNC)	Nomenclature and ontology of human genes [12]	844.8 MB
Interaction Reference Index (iRefIndex)	Protein interactions sourced from over 10 different protein interaction databases [12]	67.9 GB
Kyoto Encyclopedia of Genes and Genomes (KEGG)	A large genetic, biosystems, and chemical knowledgebase integrating information from a 16 different databases [12]	18.1 GB
The Life Science Resource Registry (LSR)	Common biological datasets and vocabularies [12]	12.1 MB

NCBI Gene	A broad range of genetic information from multiple species [12]	68.2 GB
Online Mendelian Inheritance in Man (OMIM)	Information on human genetics, including links between genotypes and phenotypes [12]	2.2 GB
Orphanet	Information on rare diseases and less-commonly used drugs [12]	29.0 KB
Pharmacogenomics Knowledge Base (PharmGKB)	Interaction between gene products, protein, drugs, genetics, and disease [12]	1.5 GB
PubMed	Metadata and some textual data from PubMed [12]	415.7 GB

Table 3.1. A listing of biomedically relevant databases contained within Bio2RDF. Datasets were selected based on containing pharmaceutical, molecular, clinical, or genetic information. Datasets containing data exclusive from non-human animals were excluded.

Once this list of suitable datasets was obtained, a subset of datasets to be used in analysis was selected based on the size of the dataset. This restriction was introduced for purely practical reasons: larger datasets take significantly longer to preprocess and train even with low epochs, and even on a memory-efficient system such as PyTorch-BigGraph processing power remained a significant barrier to dataset use.

In order to strike a balance between including enough datasets in the pool for analysis and minimizing the overall computational time and power spent, any datasets in excess of 20GB were removed from consideration. Moreover, datasets measuring under 1 MB were removed for containing too little information, since the goal of this work is to focus on big data rather than learning from small KGs. This resulted in a list of 9 datasets; PubMed, Orphanet, NCBI Gene, iRefIndex, the Comparative Toxicogenomics Database, and Clinical Trials having been removed. The remaining datasets are shown in Table 3.2.

Dataset	**Contents**	**Uncompressed RDF Size**
BioPortal	A variety of biomedical ontologies useful in bioinformatics and biomedical research [12]	18.3 GB

Database of single nucleotide polymorphism (DBSNP)	Information on common single nucleotide polymorphisms and other important short DNA mutations [12]	2.9 GB
DrugBank	Data on drugs and their targets [12]	1.6 GB
Gene Ontology Annotation (GOA)	Ontologies of genes sourced from the UniProt Knowledgebase and the International Protein Index [12]	17.1 GB
HUGO Gene Nomenclature Committee (HGNC)	Nomenclature and ontology of human genes [12]	844.8 MB
Kyoto Encyclopedia of Genes and Genomes (KEGG)	A large genetic, biosystems, and chemical knowledgebase integrating information from a 16 different databases [12]	18.1 GB
The Life Science Resource Registry (LSR)	Common biological datasets and vocabularies [12]	12.1 MB
Online Mendelian Inheritance in Man (OMIM)	Information on human genetics, including links between genotypes and phenotypes [12]	2.2 GB
Pharmacogenomics Knowledge Base (PharmGKB)	Interaction between gene products, protein, drugs, genetics, and disease [12]	1.5 GB

Table 3.2. The 9 datasets remaining, as well as their description and size, after removing those that were in excess of 20 GB or under 1 MB.

These 9 remaining datasets were all used in analysis, either in the original two hyperparameter searches for optimising AUC or r1, or in a hold-out set used to validate how good these hyperparameter sets were on different KGs not involved in hyperparameter selection. The allotment and use of these two sets is described later in Section 3.5.

3.3 Obtaining and Preprocessing the Datasets

3.3.1 Downloading from Bio2RDF

Datasets were all downloaded from the Bio2RDF website. However, link-based access to the datasets directly resulted in “file not found” errors due to limitations of their web-based file viewer. In order to avoid this error, a Python script was written to crawl the Bio2RDF website and automatically download all the datasets contained therein. All files were downloaded from the Bio2RDF release 4, which contained all of the selected datasets listed above. The code used to do this download is contained within the GitHub repository referenced at the start of chapter 3.

Once datasets were downloaded to local folders, they were manually checked for completeness, and any files that had not fully downloaded were re-downloaded manually from the Bio2RDF website. The datasets were then manually uncompressed from their archived form, and stored in a permanent location on a local hard drive.

3.3.2 Structure of the downloaded data

It is important to note that each downloaded dataset was represented as a set of files, rather than just a single file. For example, the RDF data or BioPortal was contained in a total of 365 individual files rather than a single one. All of this data was stored in the N-Quads RDF format, which represents data as 4-element quads rather than 3-element triples. The first three elements of a quad are the subject, predicate, and object. A fourth element is introduced to represent to which subgraph a specific datapoint belongs, thus allowing different statements to be optionally localised into different graphs rather than the same one.

3.3.3 Pre-processing the datasets

The uncompressed datasets were not immediately usable; both the aforementioned KGE technologies PyTorch-BigGraph and PyKEEN require input to be in a tab-delimited subject-predicate-object format [8, 44] while the downloaded data was stored in the N-Quads RDF format as mentioned above, including subgraph membership data that cannot be fully expressed in a triples-only format. As a result, converting into a triples-based format by removing the fourth element in the quad results in some degree of data loss.

However, the lost data is not of biological relevance; all biomedically relevant data is contained within the subject, predicate, and object rather than within the fourth element of the quad. Moreover, inspection of the datasets revealed that the different subgraphs were used on different files, but that within each file all triples belong to the same subgraph. As a result of these two observations, an n-Quads to tab-delimited triples conversion based upon removing the fourth element in each quad was deemed feasible and valid.

However, this conversion process was made difficult by a variety of inconsistencies in the base dataset. Some RDF subjects contained literal string quotes within them, which itself is not standard RDF syntax [10]. Moreover, the “<” character, typically used to denote the beginning of a new element of the quads in the n-Quads format, could be found within those strings. Finally, some of the strings were opened with open-quotes, but never closed, which made the process of identifying which “<” signs were less-than signs, and which defined the start of a new triple, much more complex.

In order to overcome the inconsistencies of this syntax and manage the conversion to a tab-delimited triples-only format, a custom parser was developed. The parser combined regular expressions, quoting pattern analysis, and extraction of consistent positional reference tokens to identify where each element began and ended. It then wrote only the first three elements (the subject predicate, and object) into a tab-delimited file. All triples were written to a single file, rather than to different files, for ease of use and analysis later in the pipeline.

This tab-delimited triples file, however, tended to be very large, since all the triples’ data was written in expanded form of URLs. However, PyTorch-BigGraph’s pre-processor and PyKEEN training algorithm both have upper-bound memory limits proportional to computer memory, even though PyTorch-BigGraph is significantly more resilient to large-memory datasets [8, 44]. In the cases of both PyTorch-BigGraph and PyKEEN, however, the actual textual contents of each element are irrelevant; it is only the relationships between elements that are needed to produce embeddings [8, 44].

This property immediately suggests a way to reduce the memory footprint of processing and preparing the dataset--replacing the large URLs with smaller unique character sequences. To implement this idea, a custom compression system was developed that replaced each URL or literal with a much shorter sequence of characters. The most common URLs and literals were prioritised to have the shortest character representations, and the least common ones were given the longer leftover representations. In order to allow the compression to run on larger files, a sharded version was created in which the original file was split into sub-files, which were fed through the compression pipeline. The results of both systems were identical in terms of final compression size; however, the second method was more resilient in the face of larger files that could not fit entirely in system memory.

This resulted in a very significant reduction of space needed to represent these datasets, which both facilitated running them with constrained memory and, for several structural analysis tasks, greatly reducing the running time of algorithms that analysed the contents of these files. Table 3.3 below shows a list of the original (uncompressed n-Quads) dataset size, the size of the tab-delimited triples file before compression, and its size after compression.

Dataset	**Size of uncompressed n-Quads files**	**Size of uncompressed tab-delimited triples file**	**Size of compressed tab-delimited triples file**
BioPortal	18.3 GB	12.8 GB	907.6 MB
DBSNP	2.9 GB	2.1 GB	114.7 MB
DrugBank	1.6 GB	1.2 GB	59.3 MB
GOA	17.1 GB	12.0 GB	779.3 MB
HGNC	844.8 MB	584.2 MB	40.3 MB
KEGG	18.1 GB	12.7 GB	918.5 MB
LSR	12.1 MB	8.6 MB	481.8 KB
OMIM	2.2 GB	1.6 GB	102.5 MB
PharmGKB	1.5 GB	1.0 GB	67.2 MB

Table 3.3. The size of the data in n-Quads, tab-delimited triples, and compressed tab-delimited triples files. While many of the n-Quads files take up close to 20GB of space, all of the final compressed files are smaller than 1GB.

Since these compressed tab-delimited triples files were structurally identical to the KGs from which they came, and since all embedding systems considered only take into account these relative relationships [8, 44], these files were used in all further steps of the KGE pipeline and analysis.

3.4 Selection of KGE implementation and hyperparameters for study

The selection of which KGE implementation to use (PyKEEN or PyTorch-BigGraph) fundamentally affects which hyperparameters would be selected for study, since the two systems support different KGE algorithms and express KGE training workflows slightly differently [8, 44]. The KGE implementation to use was first chosen as outlined in Section 3.4.1, and the set of hyperparameters to study was selected as described in Section 3.4.2.

3.4.1 Selection of KGE implementation

As mentioned in the literature review in Chapter 2, two major open-source KGE implementations were considered for use in this work: PyKEEN [44] and PyTorch-BigGraph [8].

PyKEEN is an open-source KGE implementation that aims to provide a simple, Python-based pipeline for running KGEs on default or custom datasets [44]. As of the time of writing, it contains 31 different KGE models and 3 different negative samplers, as well as a large variety of other algorithmic options that can be mixed-and-matched with each other to create an embedding system [49]. However, as noted in the literature review, it was, at the time this research was conducted, unable to run KGEs on KGs that required more space than was available in system memory [44]. Unfortunately, early experiments with PyKEEN showed that even on the compressed datasets, it was unable to process the larger-end datasets such as BioPortal and GOA on a system with 16 GB RAM. As a result, it was not a feasible choice for use in this research.

The second alternative considered was PyTorch-BigGraph (PBG) [8]. Like PyKEEN, PyTorch-BigGraph is based on the idea of modularity: it aims to allow mixing and matching of various components and parameters that determine what KGE model will be run [44. 8]. However, it does not provide implementations of standard KGEs the way PyKEEN does; instead, it allows the user to define their own KGE algorithm by choosing an operator (to transform subject embeddings to object embeddings), a comparator (to define how to measure closeness of two embeddings) and a loss function (to be used to optimize the model) [8]. However, the authors demonstrated that several of the common KGE algorithms, such as TransE and CompleX, can be expressed in this format [8].

The most important facet of PBG is that it allows the main graph to be split into partitions. Each partition is loaded into memory one at a time, and each one represents the fraction of the graph that can fit in system memory at once [8]. PBG then handles communicating the results of training on different partitions between partitions and uses this to create an overall KGE model [8]. The authors showed that, for large KGs, this system approximates the results that would be obtained using an identical configuration on non-partitioned graphs [8]. However, they did note that creating partitions on smaller graphs could have some negative effects on embedding quality [8].

In the case of this work, many of the datasets considered could only be run under the system limit of 16 GB with multiple partitions in use. As thus, PBG was employed to allow embedding and analysing all of these datasets.

3.4.2 Selection of hyperparameters to study

Once PBG was selected as the KGE implementation, the types of models, hyperparameters, and hyperparameter values to study and optimise were selected. The hyperparameters chosen were those that applied to the KGE models at hand. In PBG, KGE models all require these parameters to be defined as a part of the algorithm description. Since testing each value was preferred to accepting defaults, as those could be arbitrarily poor, all of these hyperparameters were examined. The hyperparameters chosen are listed in Table 3.4.

Model-related Hyperparameters	Function
Comparator	Defines the distance or dissimilarity metric to be used to determine how close two embeddings are to each other [29]
Learning rate	Defines the learning rate value to be used by the optimizer [29]
Loss function	Defines the function that should be optimized by the optimizer to find optimal embedding values [29]
Operator	Defines the transformation used to represent predicates, which is applied to subjects to map from subjects to predicted objects of that predicate [29]
regularisation coefficient	Specifies the coefficient to the regularization (penalty) term added to the cost function [29]
Batch-related Hyperparameters	
Batch size	The number of edges to include in each minibatch for the optimizer [29]
Number of batch negatives	The number of negatives to sample, under the Local Closed-World Assumption, with entities sampled proportional to their degree when constructing negatives [29].
Number of uniform negatives	The number of negatives to sample, under the Local Closed-World Assumption, with entities sampled with equal probability regardless of their degree when constructing negatives [29].

Epochs and Dimensions	
Embedding dimension	The dimension into which subject and object embeddings will be placed [29]. This also affects the size of the operator, but different operators may scale linearly or quadratically with entity embedding dimension [29].
Number of epochs	The number of full passes over the entire dataset to perform during training [29]

Table 3.4. A list of relevant hyperparameters to KGE models in PBG, as well as a description of their function in the KGE algorithms.

Once the relevant hyperparameters to examine were identified, a set of values to use for those hyperparameters was chosen. Choosing these values was guided by four main goals: choosing as few values as reasonable for each hyperparameter, maximising coverage of the search (for continuous-valued hyperparameters), choosing the most distinct hyperparameters among the possible values (for discrete-valued hyperparameters), and simplicity (for discrete-valued hyperparameters).

This first requirement was implemented since the hyperparameter used search was based off of a grid search, with some modification as explained later in this section. In a grid search, a ‘grid’ of all combinations of all hyperparameter values are tested, and the combination leading to the best results is selected. As a result, adding many values to search leads to a combinatorial explosion in the number of combinations that must be searched; as a result, using as few values as is reasonable greatly speeds up search time.

The next two goals were used to increase the breadth of different models searched without increasing the search space. For example, examining two different models equal in all respects, except that one has a learning rate of 0.01 and the other a learning rate of 0.011, is largely redundant. Since the two values are so similar, it is unlikely that there would be a significant difference, especially compared to a choice of parameters that covers a wider range of values such as 0.1 and 0.01. As thus, values chosen were selected for maximal difference from each other. In the discrete case, this argument cannot be made numerically. However, as described below, an understanding of the implementations and algorithmic effects of the discrete-valued option did reveal that some were more similar to, or different from, others. Thus, the more different values were selected for testing.

Finally, for discrete-valued hyperparameters, it was common that many of the choices presented would be similar, but that one would be of a more simple form that the other more complex. Herein, simpler forms were generally preferred when a choice between two similar options was presented, both to make the resultant model more simple to understand, and to ensure consistent design choices when selecting hyperparameters. The set of values selected for each hyperparameter is shown below in Table 3.5, which is followed by a more detailed explanation as to why the given values for each hyperparameter were chosen.

Model-related Hyperparameters	**Values Searched**
Comparator	`translation, diagonal, affine`
Learning rate	`1e-2, 1e-3, 1e-4`
Loss function	`ranking, softmax`
Operator	`dot product,` L_2 `distance`
regularisation coefficient	`1e-1, 1e-3, 1e-5`
Batch-related Hyperparameters	
Batch size	`500, 1000, 1500, 2000`
Number of batch negatives	`10, 50, 100, 250, 500`
Number of uniform negatives	`10, 50, 100, 250, 500`
Epochs and Dimensions	
Embedding dimension	`50, 100, 200, 400`
Number of epochs	`50, 100, 200, 400`

Table 3.5. The values searched for each hyperparameter that was cross-validated.

Choice of comparator, loss function, and operator were unique since the values they take are discrete rather than continuous. In the case of the comparator, there were four options: dot product, L_2 distance, cosine distance, and squared L_2 distance [29]. Of these four, dot and L_2 distance (not squared) were selected. The reasoning for this was of non-redundancy: both L_2 and squared L_2 distance are ultimately based upon the same metric: Euclidean distance. Since having fewer values greatly speeds up the search, the simpler of the two--L_2 distance--was chosen.

The choice of dot product rather than cosine distance was similar; both dot products and cosine distances can be interpreted as measures of the angles between the two vectors. As noted in [29], the cosine distance is in fact the dot product divided by the product of the lengths of the vectors. Thus the simpler form--the dot product--was once again chosen.

In terms of loss functions, there were a total of three choices: ranking, logistic, and softmax [29]. Of these, ranking was chosen because it is based fundamentally on the task at hand. In KGEs, a ranking loss function is calculated based on the score of a positive (true) triple against all of the negative (false) triples generated specifically for it [29]. The difference between these is used as a metric for the goodness of fit of the model [29]. This loss function, however, mirrors exactly what is done when KGEs are used: when trying to predict new triples, the ideal situation is that the true result is preferred over all possible false results. Since this loss function bijects so well to the goal of using KGEs, it was selected to be used in the hyperparameter search.

In order to reduce the total number of values to examine, only one other loss function was chosen. This was softmax, which is a commonly used loss function within the realm of machine learning. This choice, however, was somewhat arbitrary and motivated more by lack of computational power to examine all three than by a belief in any fault in a logistic loss function.

As a final note on loss function: the PBG implementation of the ranking loss function by default does not penalize the scores of a positive triple that are very close to the negatives, only those that are relatively far away. The width of this interval is determined by a separate hyperparameter, which takes a default value of 0.1 but was not included in hyperparameter search [29]. The reasons for not considering this parameter were twofold: for one, it only applies when the ranking operator is used [29]. In addition, adding in the values of this hyperparameter would have greatly increased the time to run the grid search, which was already very computationally expensive. As thus, the effect of this hyperparameter, and its ideal settings, are left as a future direction.

In terms of the operator, there were 5 choices: translation, linear, affine, diagonal, and complex diagonal [29]. However, since Lerer et al. found that PBG runs sub-optimally with some graphs on ComplEx [8], the complex diagonal operator (used to implement the ComplEx model) was not considered. Looking at the remaining four, affine was chosen over linear since it was a more general form of the same operation: linear is a simple matrix multiply, while affine combines a matrix multiply with a translation operation [29]. The translation operator was chosen to examine the TransE model, which is one of the most used KGE models despite its limitations [45]. Finally, the diagonal operator, which multiplies by a diagonal matrix [29] was chosen as a third option that was significantly different from the both affine and translation operators.

The other parameters were all continuous or defined on the domain of all positive integers. As such, they were selected using values that increased roughly 2 or 10 times between each item in the sequence. This was done to give a wide coverage of possible values while using relatively fewer values in total. Notably, epochs and dimensions did not exceed 400, since the amount of time required for computing KGEs at 400 epochs or 400 dimensions made using anything beyond that infeasible.

3.5 Searching for optimal hyperparameters

This section outlines the methods by which the search for optimal hyperparameters was conducted. Moreover, the key points of this search are summarized in Figure 3.1.

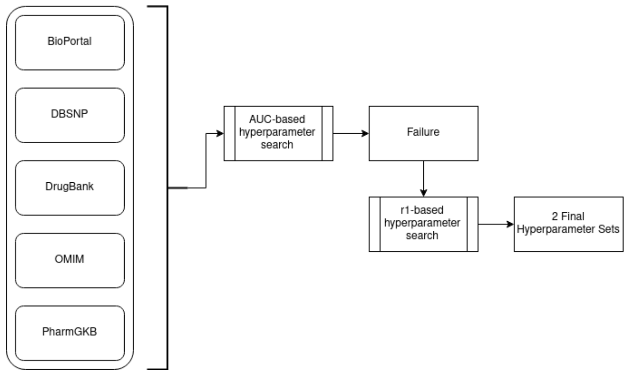

BioPortal
DBSNP
DrugBank
OMIM
PharmGKB
AUC-based hyperparameter search
Failure
r1-based hyperparameter search
2 Final Hyperparameter Sets

Figure 3.1. A summary of the hyperparameter searches and their results. Five datasets (BioPortal, DBSNP, DrugBank, OMIM, and PharmGKB) were examined in two searches. The first, using AUC to select hyperparameters, failed as was stopped in preference of one optimizing for higher r1 scores. This resulted in a total of 2 hyperparameter sets, which were carried forwards for analysis.

3.5.1 Modified grid search protocol

In order to determine the optimal hyperparameters, a modified version of the grid search was used. In a traditional grid search, all values of all hyperparameters in question are varied over a grid, and the best choice from among them is chosen. However, the vast number of KGE models and hyperparameters listed above made such an approach infeasible. Thus, a set of default parameter values given by [29] were used to initialize the model. Three grid searches were then carried out: in the first, model-related hyperparameters were varied. These were, specifically: comparator, learning rate, loss function, operator, and regularisation coefficient. It should be noted that, due to the design of PBG, a regularization coefficient hyperparameter is not given to KGE models using the affine operator [29]. As such, this combination of operator and regularization coefficient was not allowed when searching for optimal hyper parameter calculations.

In the second round, hyperparameters relating to batching were varied; these were batch size, the number of batch negatives to use, and the number of uniformly sampled negatives to use. Finally, in the third round, the number of epochs and embedding dimensions were varied. A summary of this modified grid search structure is given below in Table 3.6.

Grid search round	**Hyperparameters varied**
1 (Model-related hyperparameters)	comparator, learning rate, loss function, operator, regularisation coefficient
2 (Batch-related Hyperparameters)	Batch size, number of batch negatives, number of uniform negatives
3 (Epochs and Dimensions)	Embedding dimension, number of epochs

Table 3.6. A summary of the hyperparameters searched in each of the three search rounds.

In addition, rather than conducting the hyperparameter search on the entire dataset, datasets were subsetted randomly in order to make the search feasible in the available time. In order to do this, the decision of how large to make the subsets was critical to ensuring that they could well represent the data from which they were drawn. As a note, all subsets taken were taken in a single-pass traversal of the graph in which a triple was randomly chosen with probability equal to the desired number of triples divided by the total number of triples in the graph. This was done to reduce the time and memory complexity of the subsetting step, as well as due to the fact that at large subsets the variance from the target number of triples would be negligibly low due to the large sample sizes.

3.5.1.1 Selecting the proper subset size

In order to determine the correct size of the subsets, as well as to verify that these subsets captured the structural elements of the original graph, two experiments were performed: one based on timing and output statistics, and one investigating the structure of the subgraphs as compared to the overall graph. These experiments were done on a subset of the BioPortal dataset, since carrying them out on all available datasets required significant time in terms of training, and since the time taken, which was one of the most important factors for choosing a subset size, is more dependent upon the subset size itself than upon the dataset from which the data was drawn.

In the first case, a timing experiment was performed. Subsets of the BioPortal dataset were taken with sizes of 200, 2000, 4000, and 8000 triples. This subset was then used for a sample round of search iteration 1 to estimate the computation load of the grid search. In each case, the time taken to perform the search was recorded. The results of this are shown in Figure 3.2.

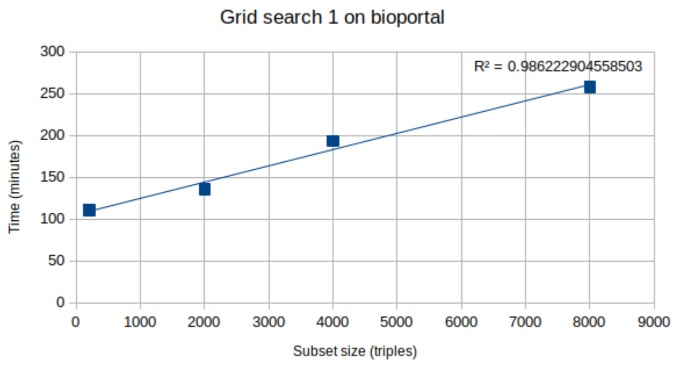


Figure 3.2. The relationship between subset size (measured by the number of triples included) and time taken to run the search (in minutes).

The relationship is very clearly linear and has an overall R^2 value of 0.9862. However, subsets of BioPortal sized at 8,000 triples took over 4 hours to train, while those sized at 4,000 triples took around 3 hours. During this research, the difference of one hour per round was quite significant--conducting the search, correcting errors, and re-running the search in different iterations resulted in a large amplification of time. However, it was critical that the size of the dataset not be allowed to compromise the quality of the search data.

Thus, the similarity of the AUC and the r1 scores for each hyperparameter combination in the grid search were recorded to ensure that choosing a smaller subset size (below 8000) would not result in dramatic loss of performance that could negatively impact the quality of hyperparameters found in the search. For each size and the one immediately following (i.e., 200 and 2000, 2000 and 4000, etc.) the difference between the AUC and r1 statistics for each of 84 total hyperparameter combinations was taken. This difference was calculated as the Euclidean distance between the r1 or AUC scores of each respective pair. The results of this are shown in Table 3.7.

Subset size pair	**Euclidean distance between r1 scores; Total | Average**	**Euclidean distance between AUC scores; Total | Average**
200 vs 2,000	0.3347 | 0.0040	0.4831 | 0.0058
2,000 vs 4,000	0.2175 | 0.0026	1.5024 | 0.0179
4,000 vs 8,000	0.2724 | 0.0032	1.3731 | 0.0163

Table 3.7. A listing of the Euclidean distances between r1 and AUC scores of different subgraph sizes. The first number is the total distance, and the second number is the average distance equal to the total distance divided by the number of observations, 84.

In all cases, the difference between the two elements was minimal, especially in reference to the average differences expected between two elements from the pairs. However, there is also much data contained in the graphs not captured in this analysis, such as structural patterns. However, in order to facilitate running the search, the subset size of 8,000 was not preferred. As a middle ground, the subset size of 4,000 was selected for further evaluation to help ensure that it matched the key structural characteristics of the overall graph and did not exclude too many small trends. Particularly, it was important to ensure that this random subsetting method did not over-select for sinks and sources by chance of not picking up on triples that connected to each other within the domain of the subgraph.

As thus, the ratios of sinks and sources to the total number of entities in the overall BioPortal graph and the size-4000 subset were collected. The results of this analysis are shown in Table 3.8.

	Size-4000 Subset	**Full BioPortal Graph**
Ratio of sinks to triples	0.3844	0.6032
Ratio of sources to triples	0.5180	0.1942
Ratio that were neither sinks nor sources to triples	0.0976	0.2026

Table 3.8. The ratios of sinks, sources, and repeated entities to the total number of entities in the 4000-triple subset and in the full BioPortal graph.

From this data, it is clear that there is a difference between the subsetted data and the original data. In particular, the number of sources is overestimated, likely due to an under-estimation of the number of repeated (neither source nor sink elements. However, the fact that many repeated elements were still captured indicates that some of the original graph structure is capable of being replicated in the subsetted graph. Moreover, while running on the full dataset would clearly be preferred, the processing power needed to run such a hyperparameter search far exceeded that which was available for datasets over several MB. As such, a middle line was drawn by using the 4000-triples subset, balancing time taken with the ability to faithfully represent the full graph. It is likely that this sub-optimal choice resulted in the lower-performances of some of the models and sub-optimality of the resultant hyperparameters, as noted in Chapter 6. However, it is important to note that, even in the absence of a perfect hyperparameter search, patterns of overall graph structure versus KGE performance under various hyperparameters are still indicative of the effects of KG structure on KGE models; the only difference is that the hyperparameters found here cannot be claimed to be entirely optimal.

3.5.1.2 Running the modified grid search

When run, the modified grid search was run using two partitions. This was done because PBG and KGEs generally behave somewhat differently when partitioned [29], and since in the case of almost all the KGs examined herein, it was necessary to use at least 2 partitions to run KGE models on them. 15% of the data was placed in a held-out evaluation dataset that was not used during training, and AUC and r1 estimations were performed from this held-out set rather than on the training set.

This grid search was performed twice: once to optimise the model's performance as estimated by the AUC statistic, and once to estimate its performance as measured by the r1 statistic. As mentioned in the introduction, the AUC statistic represents the probability that a positive triple will be preferred over any negative, not only the negatives created for it; r1 is a measure of the probability that a triple will be preferred over all triples created specifically for it under the Local Closed-World Assumption [29].

In the searches for both AUC and r1, when multiple hyperparameter values worked similarly well, preference was given to those that were more generally applicable across the datasets in question. This was done to produce one, or very few, hyperparameter sets that could apply optimally to multiple datasets in question. The process by which this was done is explained in detail in the two following chapters that detail the steps of the AUC and r1 based searches respectively.

In both rounds, hyperparameters were selected in three rounds since varying all of the hyperparameters at once led to a combinatorial explosion and excessive time cost. This had two main effects: some hyperparameters had to be varied and selected before others, and yet unselected hyperparameters had to be given arbitrary starting values. In response to this, hyperparameters were selected in order of how impactful they were to the KGE's knowledge representation ability as a whole, with the most impactful features coming first. As such, the hyperparameters regarding the embedding operation, comparison calculation between embeddings, and the learning (loss) function were varied first since they are the defining elements of how KGE's model and learns from data [45]. The regularisation coefficient and learning rate were also included in this round since they fundamentally affect how the algorithm and loss function interact with the data.

The second round focused on batching and negative sampling, as these are representative of how the model learns to distinguish truth from falsehood using its knowledge model [45]. This left the number of epochs and the embedding dimension to be determined last, in the third round.

Moreover, to address the need to have values for the hyperparameters not yet validated, it was necessary to initialize the hyperparameters that had not been validated yet to starting values.

This was done arbitrarily based on a default example set given in PBG [51], as shown below in Table 3.9.

Batch-related Hyperparameters (validated in round 2)	**Arbitrary Initial Value**
Batch size	`1000`
Number of batch negatives	`50`
Number of uniform negatives	`50`
Epochs and Dimensions (validated in round 3)	
Embedding dimension	`100`
Number of epochs	`100`

Table 3.9. The initial values taken by hyperparameters. Note that the hyperparameters validated in rounds 2 and 3 were taken as the defaults in round 1, and that only those validated in round 3 were used as the defaults in round 2. In round three, all hyperparameters had either been selected, or were being varied, and thus none had arbitrary values anymore.

Following the searches, their results, both in terms of the hyperparameter sets produced as well as the performance of the output models, were finally analysed. The conclusions of this analysis are given in Chapter 6.

3.5.2 The AUC-based search

The AUC-based search was run using a round of the modified grid search to directly predict the best hyperparameters for the next round. Default values were taken and used exactly as in the AUC-based search, as described in Table 3.9. This was then repeated two more times to gain a total sample size of 3 to relate hyperparameter assignments to AUC results.

However, as described in Chapter 4, validating by AUC led to high AUC scores but still very low r1 scores. As mentioned before, AUC is the probability that a true statement will be preferred over any false one, whereas r1 the probability that a true statement be preferred over its most believable negatives. Thus, this effect generally of high AUC and low r1 indicates the model lacks specificity and could not well distinguish a true statement and a corrupted true statement that was made false.

As a result of this observation, the AUC-based search is not expanded upon beyond the first round, and attention is given to the r1-based search which could produce more robust KGE models in terms of r1 without excessive loss of AUC-score-based fitness.

3.5.3 The r1-based Search

In the r1-based search, each of the three search rounds these rounds were run three times, and their results were averaged. These results were used to estimate the best hyperparameters from the current round, which then were used in the next round. Once again, default values were taken and used exactly as in the AUC-based search, as described in Table 3.9. By the end, all hyperparameters had been replaced by their optimised values. The full details and steps of this search are explained in Chapter 5.

3.6 Structural analysis of KGs and connection to KGEs

Structural analysis of the KGs was performed using three methods. In the first, the numbers of sinks (i.e., nodes that were always objects), and sources (nodes that were always subjects), and the number of repeat nodes that were neither sources nor sinks (those that appeared as both subjects and objects) were calculated. All of these values were collected both in raw form and normalized to the total number of triples in the graph.

In the second method, the centrality of each node was calculated. Since Sadeghi et al. established that KGE models typically only can capture graph structural information one degree out from a given node [47], the nodes' centrality was calculated simply as their degree. The distribution of centralities was taken, both unnormalized and normalized to the number of triples in the graph.

In terms of the KGE models themselves, all datasets considered were run on both of the hyperparameter sets. When doing so, a baseline of 2 partitions was used, except in the case of LSR and HGNC which were very small and ran easily on 1 partition. If the baseline number of partitions resulted in a memory error, then the number of partitions was increased by one until the model ran without error. The overall pipeline that was followed in this process is summarised below in Figure 3.3.

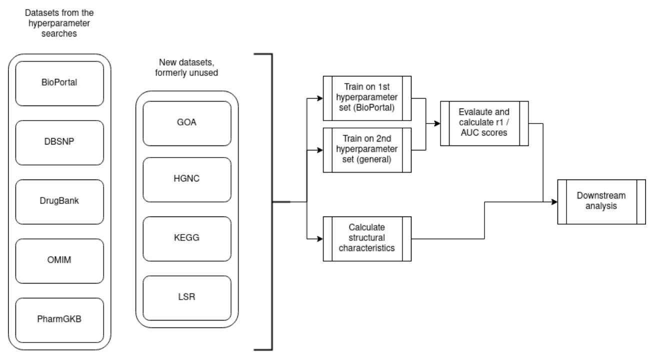

Datasets from the hyperparameter searches
BioPortal
DBSNP
DrugBank
OMIM
PharmGKB
New datasets, formerly unused
GOA
HGNC
KEGG
LSR
Train on 1st hyperparameter set (BioPortal)
Train on 2nd hyperparameter set (general)
Calculate structural characteristics
Evalaute and calculate r1 / AUC scores
Downstream analysis

Figure 3.3. An overview of the methods by which analysis was performed. All datasets were run under both hyper parameter configurations, and their resultant r1 and AUC scores were recorded. Structural characteristics were calculated, and these results were pooled for downstream analysis.

The selection of hyperparameters is detailed in Chapters 4 and 5, and a complete characterization of how the datasets were run under these hyperparameters, as well as the results, is detailed in Chapter 6, since it depends on hyperparameter search information covered in the next two chapters.

The result of both of these approaches were tables containing information in the distribution of these various statistics in the graphs and their corresponding r1 and AUC scores. The best-fit hyperparameter set for the given dataset, as well as the AUC and r1 scores for that model, were then connected to the structural information for each dataset. Patterns between these various characteristics and KGE model performance were then analysed through regression analysis. In the regression analysis, structural features were used to predict r1 or AUC for each hyperparameter configuration found in the previous hyperparameter searches. In addition, those features were used to predict the difference between the effectiveness of both hyperparameter sets when run on the same KG. The full details of this methodology are dependent both upon the results of the hyperparameter searches (in Chapters 4 and 5) and the structural features (presented in Chapter 6); as thus, the methodology by which this analysis was done is included in greater detail in Chapter 6. Moreover, the results of this analysis, and a discussion thereof, are given in Chapters 6 and 7.

4 AUC-Based Search

4.1 Chapter Outline

This section explains the methodology and results of the AUC-based hyperparameter search. Section 4.2 covers the search for model-related hyperparameters. Section 4.3 conducts an analysis of the overall predictive power of the KGE model after this round of searches in terms of both AUC and r1 scores. Since this results in a conclusion that focusing on AUC is a poor hyperparameter validation decision, Section 4.3 also concludes this section, rather than moving on to the later rounds of an AUC-based hyperparameter search.

Note again that the term 'hyperparameters' in this section is used broadly to reference both the parameters to models, as well as model choice decisions such as embedding operators.

4.2 Selecting model-related hyperparameters

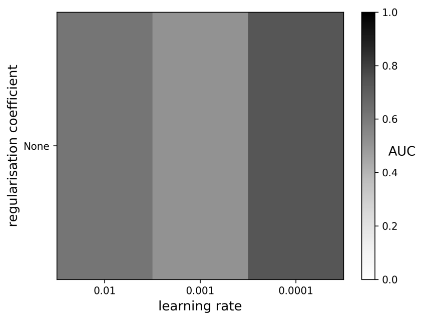
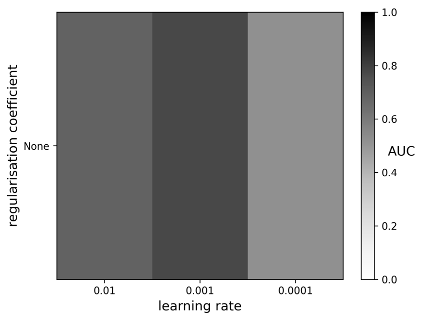
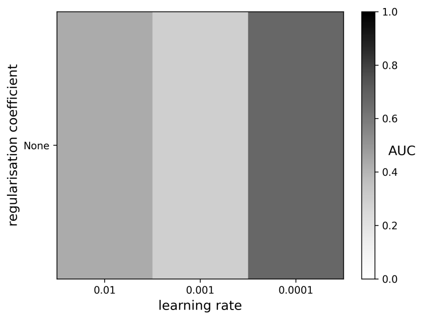
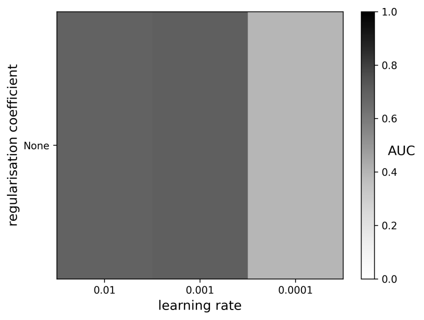
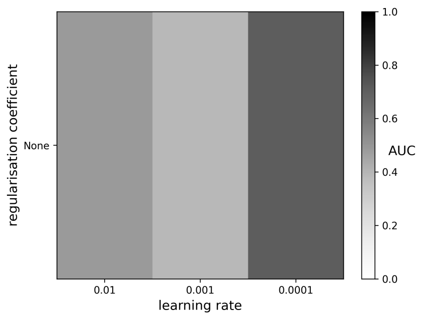
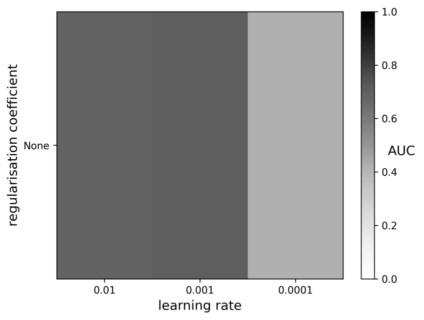
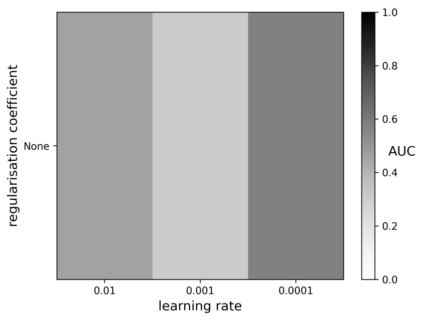
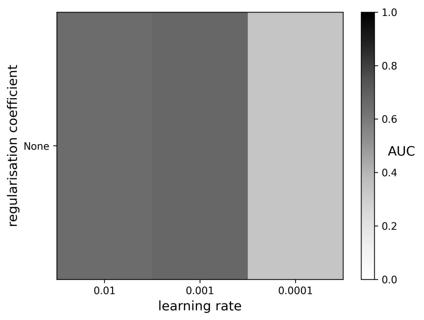
In the first round, model-related hyperparameters and model choices were considered. The full list of considered hyperparameters for this round and the values examined for each of them are given in Tables 3.4 and 3.5 in Chapter 3.

When running the grid search as described in the methodology, hyperparameters were selected such that they would most-reliably lead to the highest AUC score for the KGE model. This search was run in triplicate on the five test datasets: BioPortal, DBSNP, DrugBank, OMIM, and PharmGKB.

Since a total of 5 hyperparameters were examined, visualizing them all on the same graph was not feasible. In order to account for this, AUC was plotted against learning rate and the regularization coefficient for each value of the other parameters.

The averaged results of the three iterations on each of the datasets are displayed below in Figures 4.1 to 4.6. **Please note that darker colors correspond to higher AUCs, which in turn correspond to better model performance.**

Ranking loss function and Affine operator:

Dataset	L_2 Comparator	Dot Comparator
BioPortal		
DBSNP		
DrugBank		
OMIM		

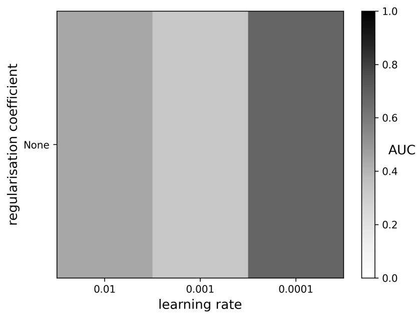
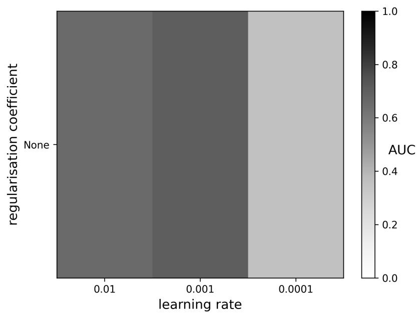
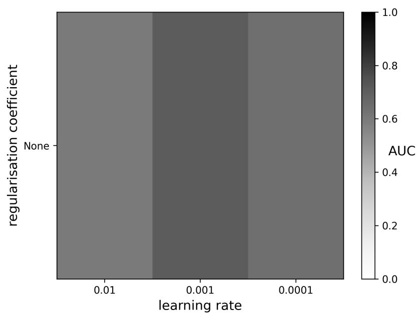
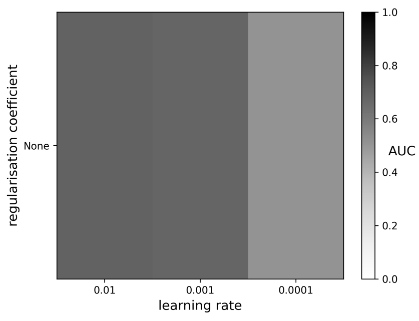
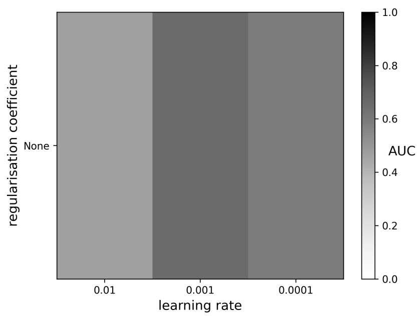
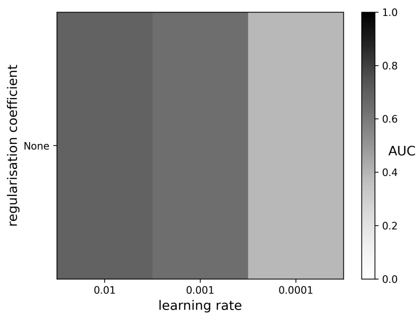
PharmGKB		

Figure 4.1. The average effect of hyperparameters on AUC over 3 iterations, when the ranking loss function and affine operator are used.

Softmax loss function and Affine operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		

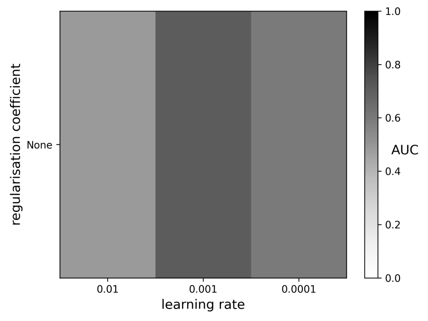
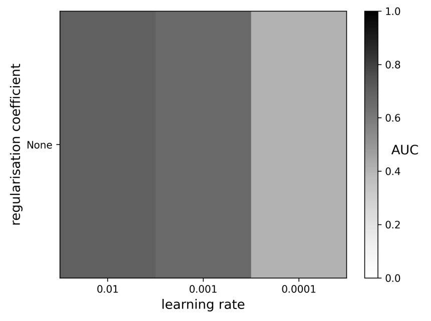
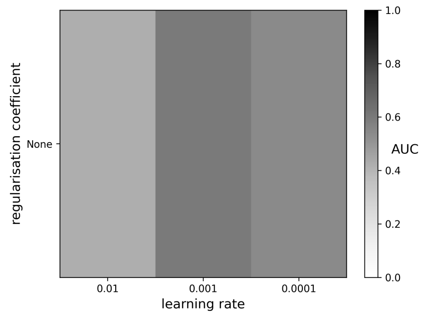
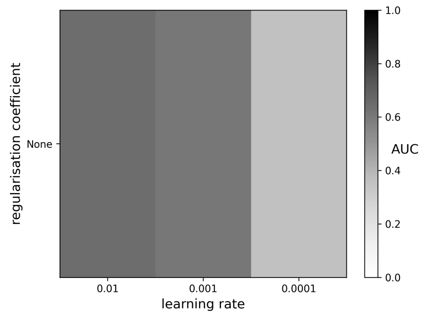
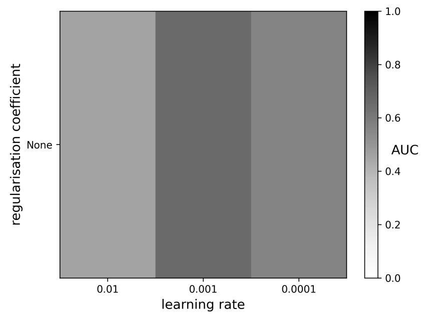
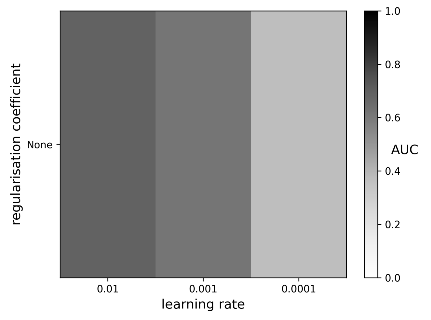
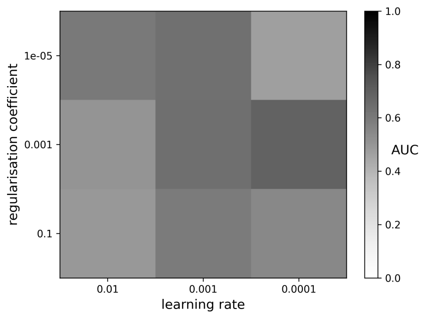
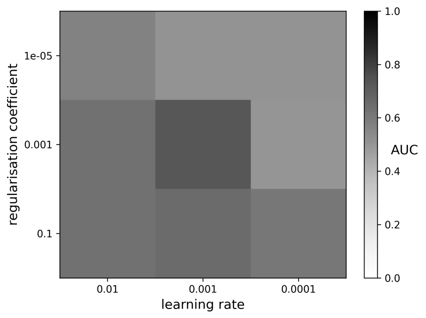
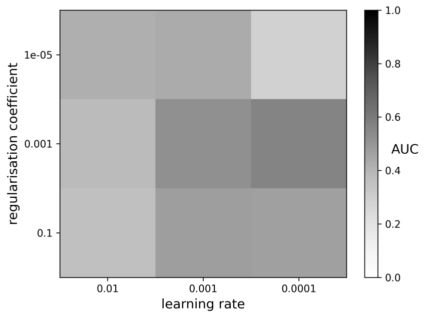
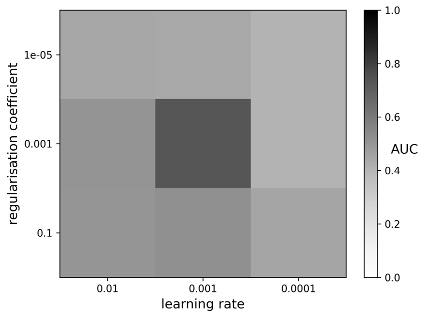
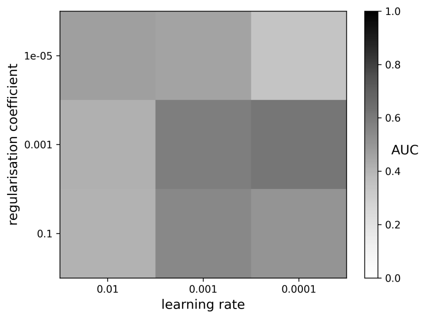
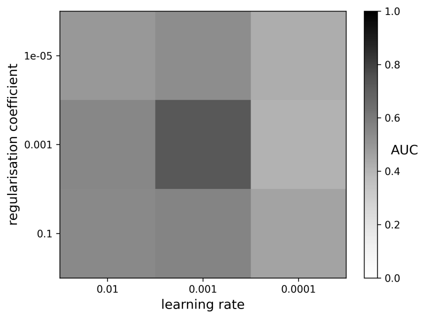
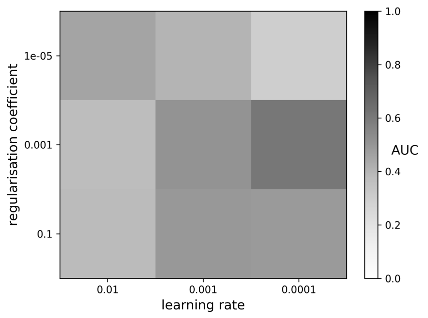
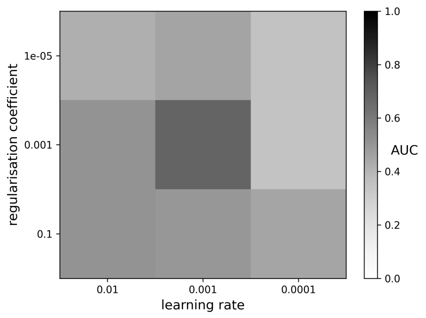
DrugBank		
OMIM		
PharmGKB		

Figure 4.2. The average effect of hyperparameters on AUC over 3 iterations, when the softmax loss function and affine operator are used.

Ranking loss function and Diagonal operator:

Dataset	**L_2 Comparator**	**Dot Comparator**

BioPortal		
DBSNP		
DrugBank		
OMIM		

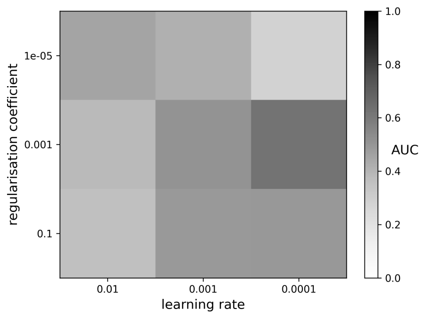
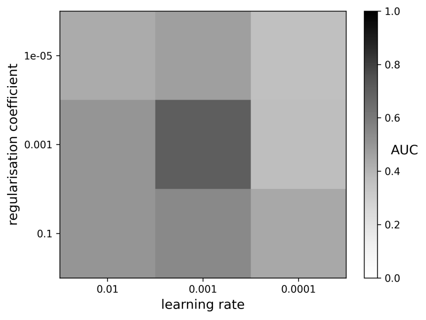
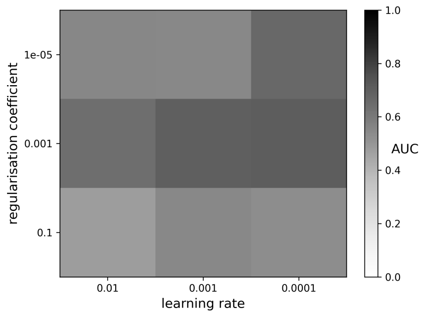
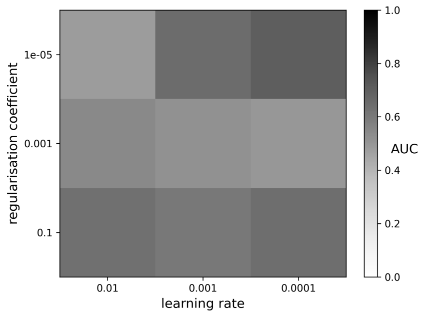
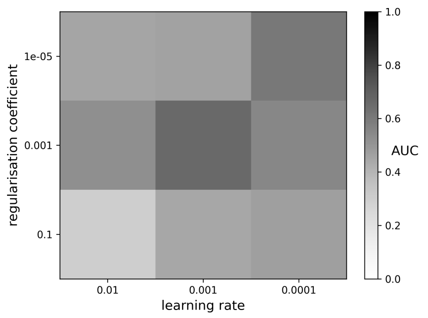
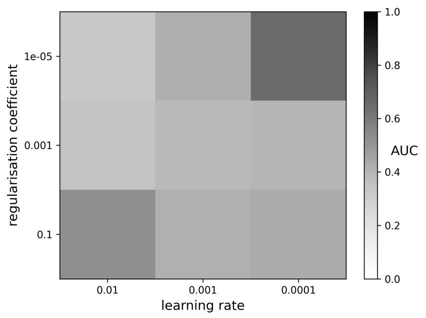
PharmGKB		

Figure 4.3. The average effect of hyperparameters on AUC over 3 iterations, when the ranking loss function and diagonal operator are used.

Softmax loss function and Diagonal operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		

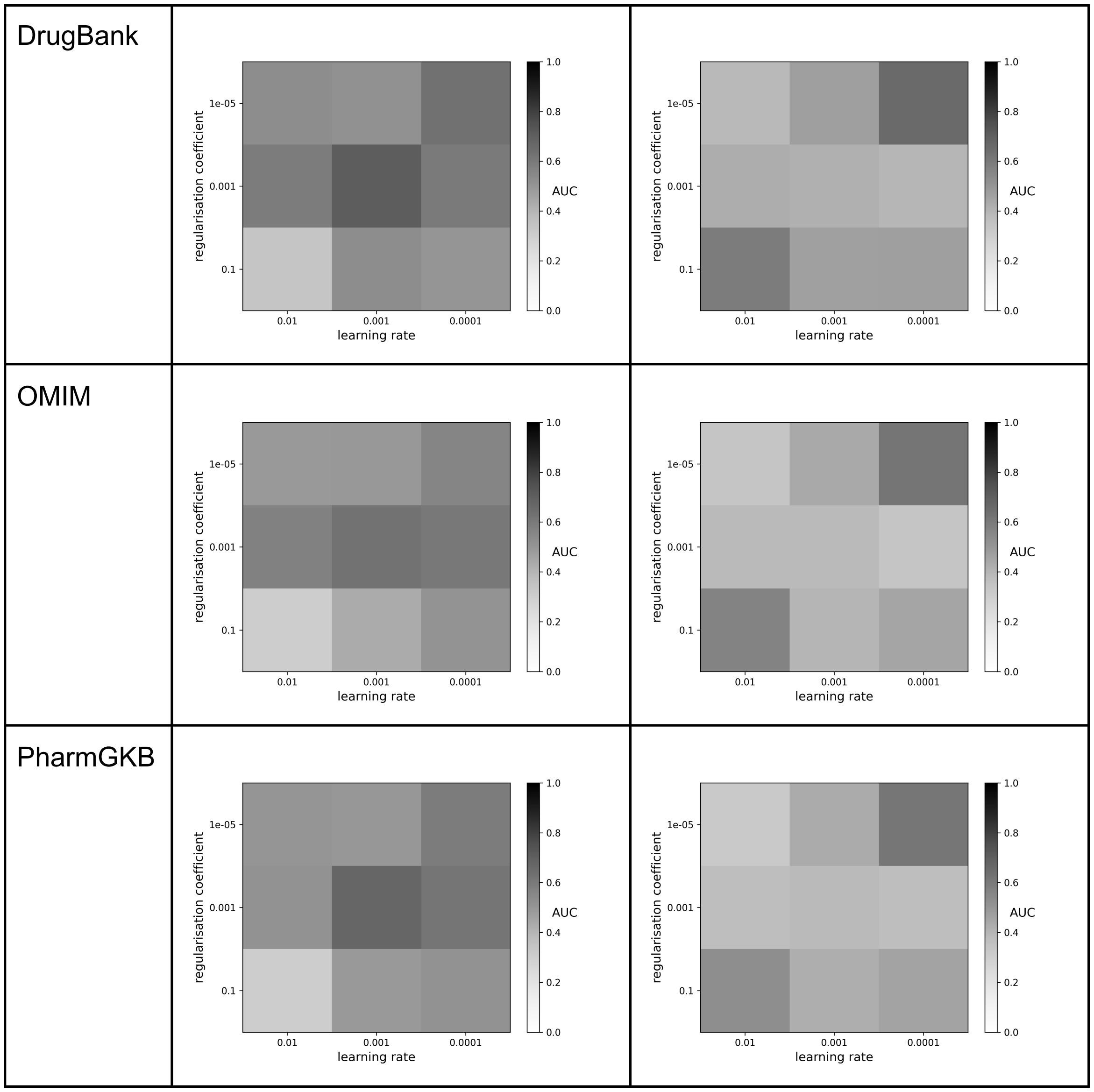
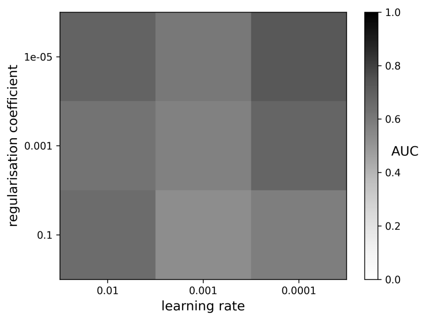
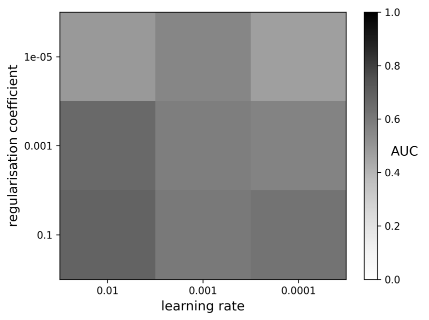
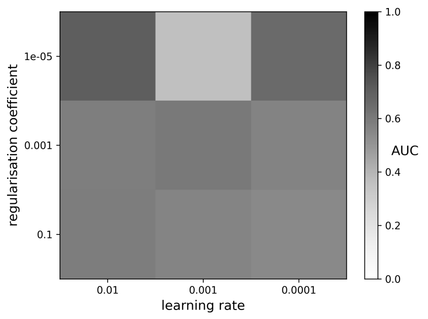
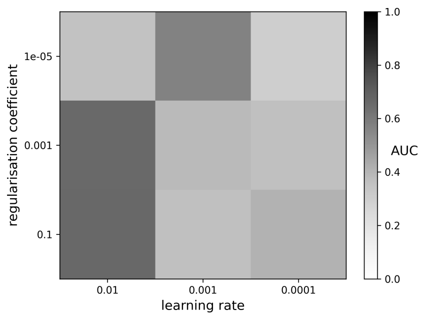
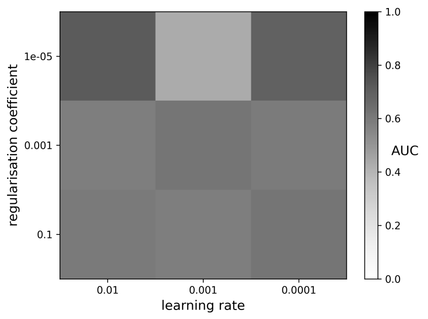
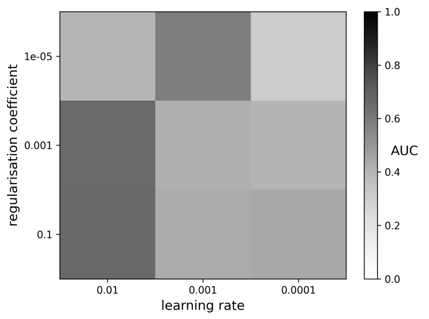
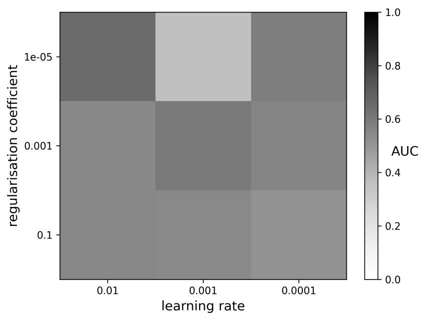
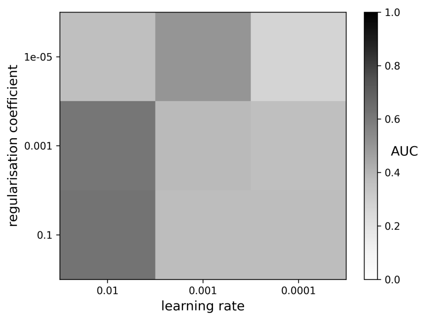


Figure 4.4. The average effect of hyperparameters on AUC over 3 iterations, when the softmax loss function and diagonal operator are used.

Ranking loss function and Translation operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		
DrugBank		
OMIM		

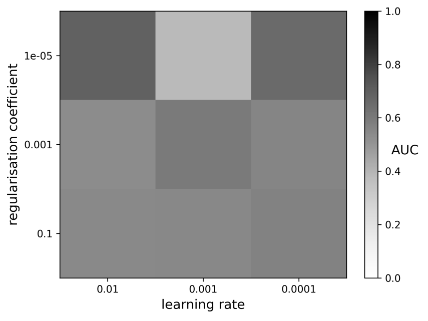
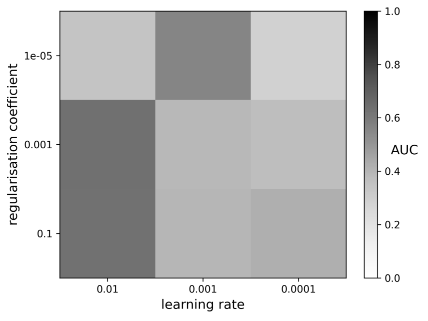
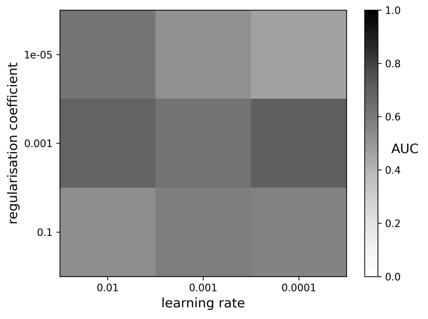
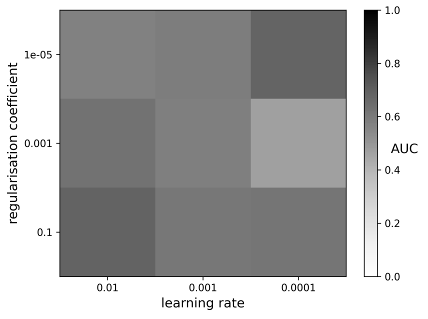
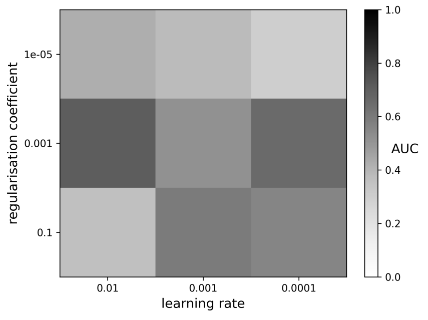
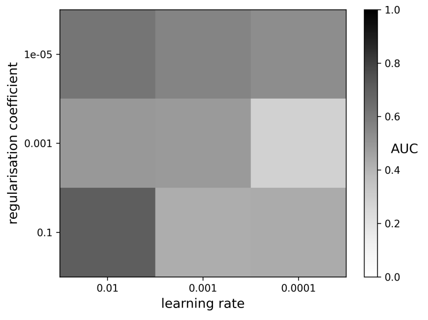
PharmGKB		

Figure 4.5. The average effect of hyperparameters on AUC over 3 iterations, when the ranking loss function and translation operator are used.

Softmax loss function and Translation operator:

Dataset	L_2 Comparator	Dot Comparator
BioPortal		
DBSNP		

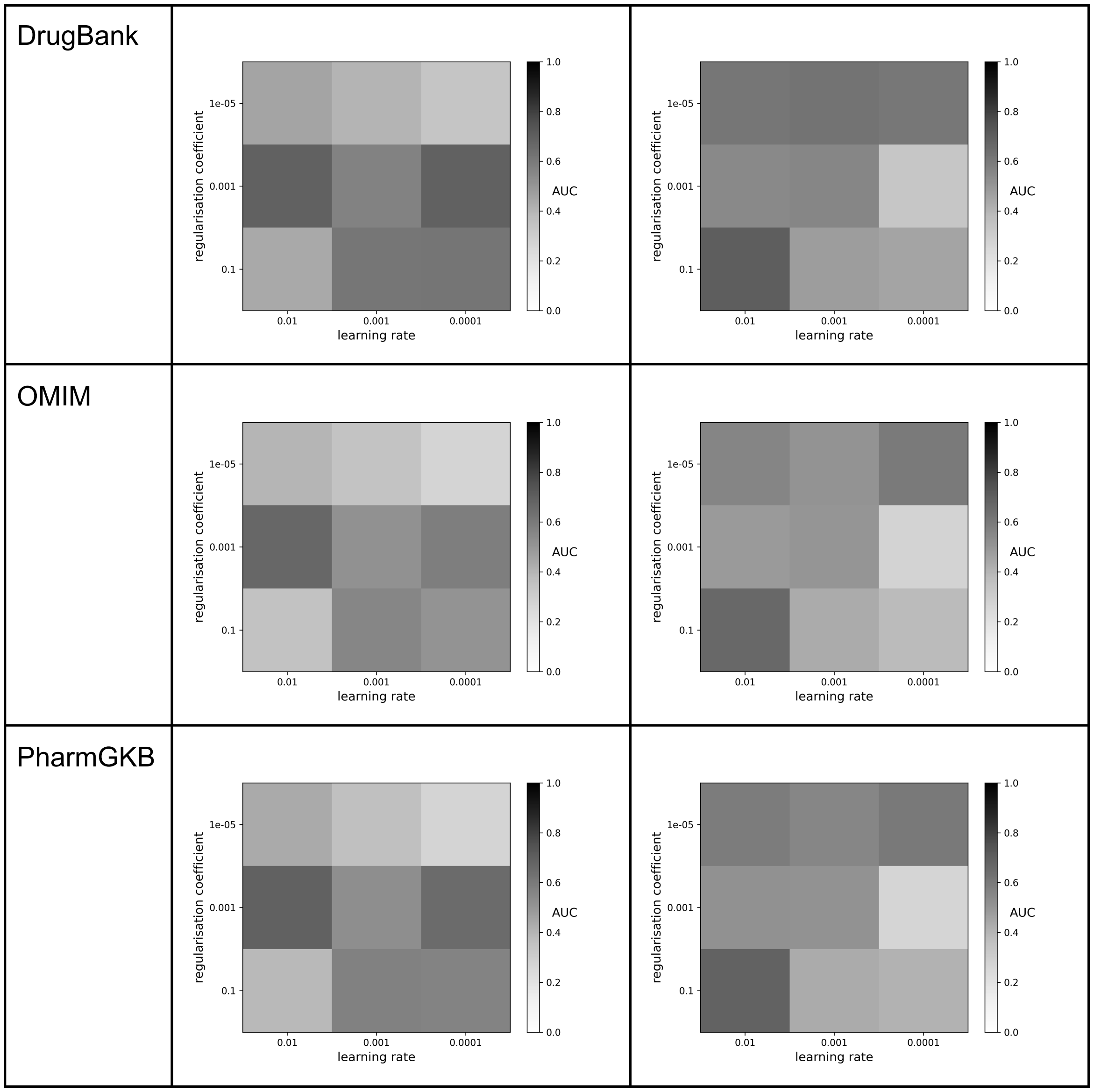


Figure 4.6. The average effect of hyperparameters on AUC over 3 iterations, when the softmax loss function and translation operator are used.

Several trends were noted from the three grid search iterations on these datasets. In most cases, there are general cross-dataset trends as to which combinations of learning rate and coefficient work best, although the optimal combination changes based on the operator, comparator, and loss function in question. When the ranking loss function is used with the affine operator, the dot comparator with a learning rate of 0.001 is optimal for all datasets. When softmax is used with the affine operator, the dot comparator still outperforms the L_2 comparator overall and works best with a higher learning rate of 0.01.

The combination of the diagonal operator with a ranking loss function has a very clear preference for a learning rate of 0.001 and a regularization coefficient of 0.001, and the performance of this combination is nearly identical across the five examined datasets. The dot product comparator once again yields better results than the L_2 comparator for this combination. Changing out the ranking loss function for softmax, but keeping the dot product comparator, changes this preference. Under those conditions much smaller learning rates and regularization coefficients of 0.0001 and of 1e-5 respectively are optimal for all five datasets. Interestingly, however, a learning rate of 0.001 and a regularization coefficient of 0.001 are optimal under softmax when the L_2 comparator is used.

When the dot product comparator and the translation operator are used with either ranking or softmax as a loss function, a learning rate of 0.01 and a regularization coefficient of 0.1 are preferred across all datasets. When the L_2 comparator is used with translation, the trend is less general: under ranking loss functions learning rates of 0.01 and regularization coefficients of 1e-5 are optimal, while under softmax the optima are 0.01 and 0.001 respectively. Once again, in both cases these optima function well across datasets, but it is worth noting that in this case these parameters sometimes are second-best choices rather than best choices for the given datasets.

It is worth noting that while many of the differences are decently strong--on the order of a change in AUC of 0.1--many of them are much smaller. Most of these smaller changes--and some of the larger ones, even--were well within one standard deviation of each other. Thus, determining with certainty which of two similarly-valued hyperparameter combinations leads to better results is not certain.

4.3 Analysis of predictive power

The high AUCs--many of the optimals being near 0.8--found in the previous section are initially encouraging. However, r1 scores were universally poor, the highest of them BioPortal) just under 0.24 and most of them under 0.1. Moreover, the highest AUCs never corresponded to the highest r1 values in any of the datasets, and the correlations between them were weak. Considering the r1 and AUC scores for all hyperparameter combinations, the correlations of the two sets were calculated and are shown in Table 4.1.

Dataset	**Correlation between AUC and r1 scores**	**Positively correlated?**
BioPortal	-0.5335	No
DBSNP	0.8675	Yes
DrugBank	0.8049	Yes
OMIM	0.7405	Yes
PharmGKB	0.8045	Yes

Table 4.1. Correlation values and directionality between the lists of AUC and r1 scores of each dataset.

As mentioned in the literature review, there is a very important difference between the two statistics: AUC reflects the ability to distinguish a true fact from any false one, whereas r1 reflects the ability to distinguish a true fact from a similar, more convincing wrong one made by corrupting the truth. As thus, AUC only measures a more general predictive modelling ability, whereas r1 reflects a very precise ability to distinguish truth from convincingly constructed false statements.

While the correlation values indicate a positive relationship between AUC and r1 for DBSNP, Drugbank, and PharmGKB, this relationship was not consistent across all datasets. For bioportal, they were anti-correlated, with higher AUC being linked to lower r1. The presence of this anticorrelation between the two statistics indicates that maximizing AUC does not necessarily maximize r1 in all these cases, nor vice versa.

For this reason, it was decided that focusing on AUC rather than r1 would lead to in general poorer KGE models, since only a more general form of knowledge representation was being selected for, rather than a specific ability to discern the difference between similar statements with different truth values. As thus, the AUC-based search was deprecated in favor of a search based on the r1 statistic, which is detailed in Chapter 5.

5 r1-Based Search

5.1 Chapter Outline

This section contains 3 main parts, one for each iteration of the search. Section 5.2 covers the search for model-related hyperparameters, Section 5.3 the search for batch-related Hyperparameters, and Section 5.4 the search for proper number of epochs and the embedding dimension. Section 5.5 ends with a discussion of the obtained hyperparameters in the context of the search methodology. Table 5.1, in Section 5.5, provides an overview of the hyperparameters chosen at each iteration and as thus is useful to refer to in sections 5.2 through 5.4.

Note again that the term 'hyperparameters' in this section is used broadly to reference both the parameters to models, as well as model choice decisions such as embedding operators. Also, **please note that darker colors correspond to higher AUCs, which in turn correspond to better model performance.**

5.2 Round 1: Model-related hyperparameters

In the first round, model-related hyperparameters and model choices were considered. The full list of considered hyperparameters for this round and the values examined for each of them, are given in Tables 3.4 and 3.5 in Chapter 3.

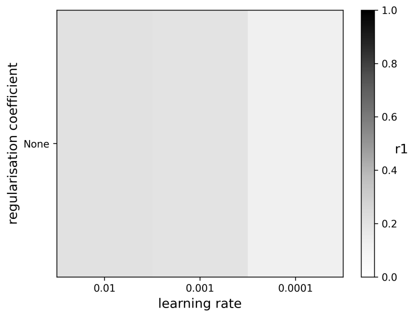
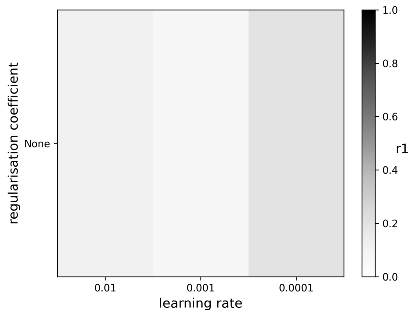
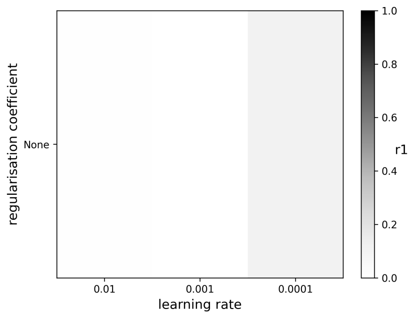
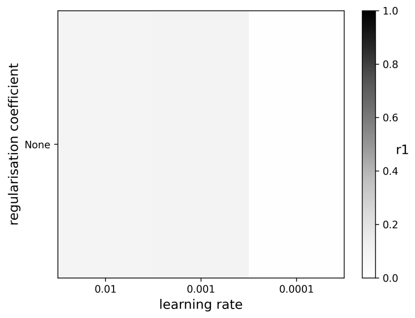
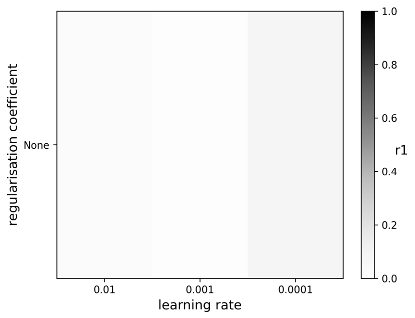
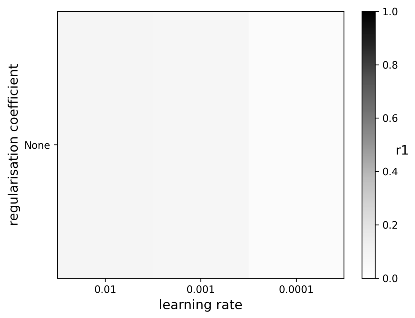
When running the grid search as described in the methodology, hyperparameters were selected such that they would most-reliably lead to the highest r1 score for the KGE model. This search was run in triplicate on the five test datasets: BioPortal, DBSNP, DrugBank, OMIM, and PharmGKB.

Since a total of 5 hyperparameters were examined, visualizing them all on the same graph was not feasible. In order to account for this, r1 was plotted against learning rate and the regularization coefficient for each value of the other parameters.

The averaged results of the three iterations on each of the datasets are displayed below in Figures 5.1 to 5.6. Please note that darker colors correspond to higher r1s, which in turn correspond to higher performance.

Even more so than in Chapter 4, it is worth noting that the differences here, being averages over three rounds, had relatively high standard deviations. Most of the differences observed--save those between the best and the worst combinations for any given dataset--were contained within one standard deviation of each other and as thus cannot be considered significant. While those with high values are tentatively taken as better, this is not certain.

Ranking loss function and Affine operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		
DrugBank		

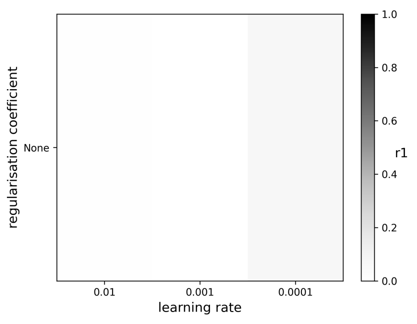
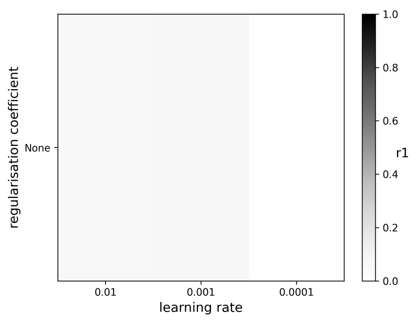
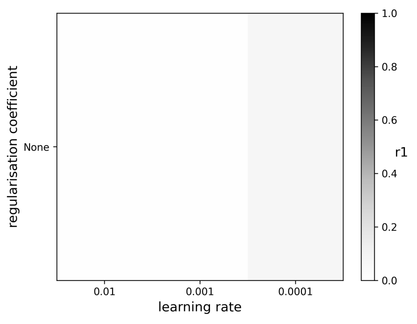
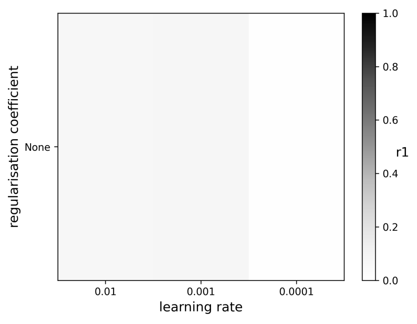
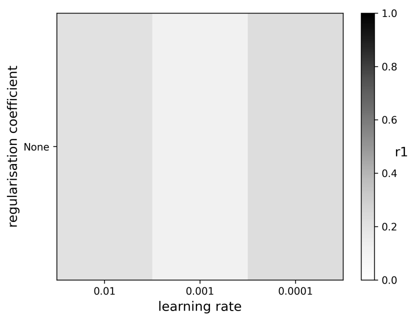
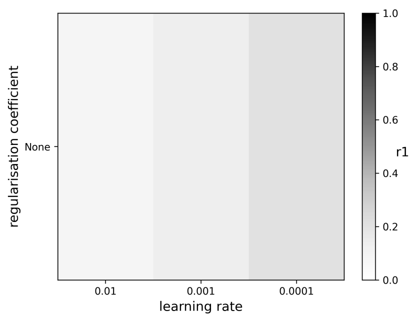
OMIM		
PharmGKB		

Figure 5.1. The average effect of the hyperparameters on r1 over 3 iterations, when the ranking loss function and affine operator are used.

Softmax loss function and Affine operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		

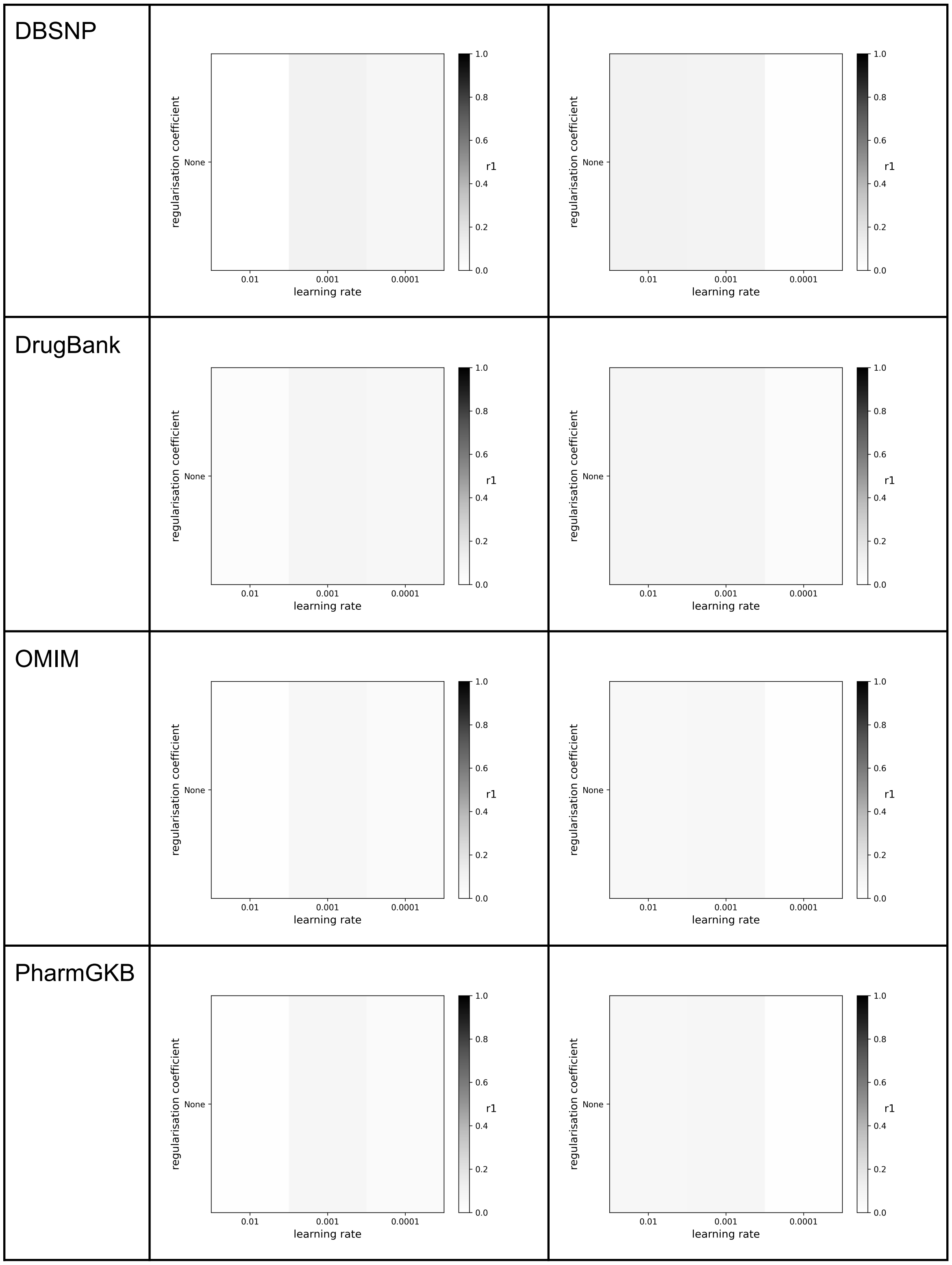
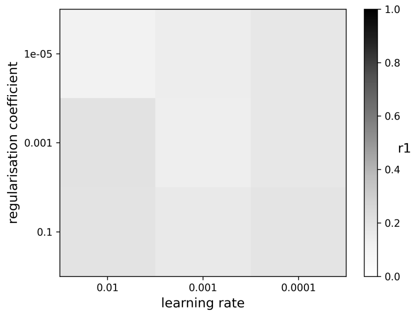
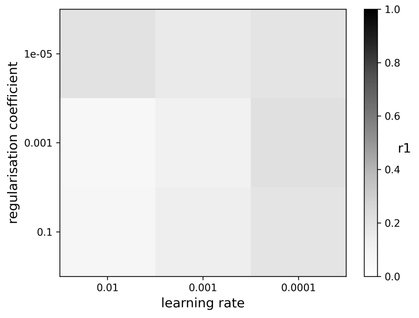
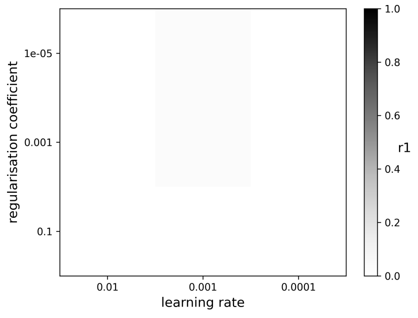
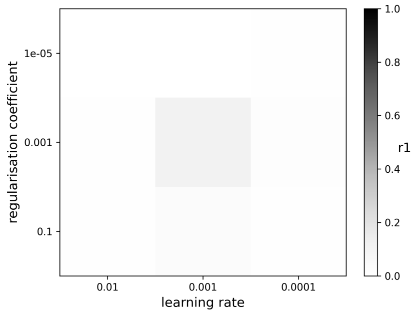
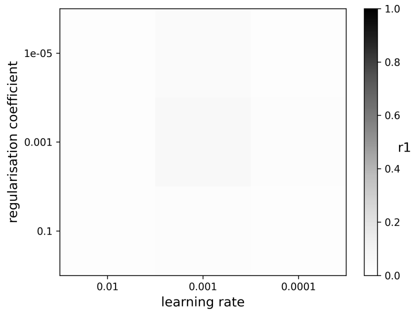
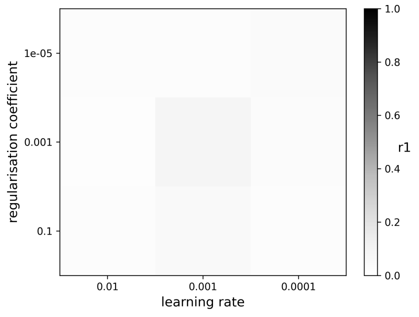
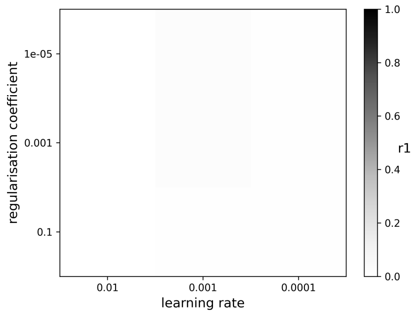
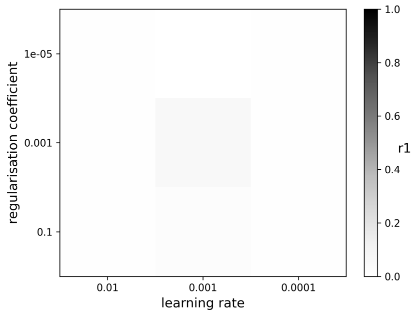


Figure 5.2. The average effect of the hyperparameters on r1 over 3 iterations, when the softmax loss function and affine operator are used.

Ranking loss function and Diagonal operator:

Dataset	L_2 Comparator	Dot Comparator
BioPortal		
DBSNP		
DrugBank		
OMIM		

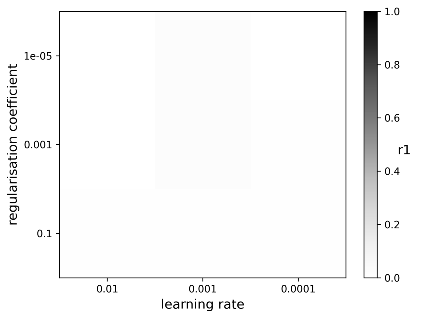
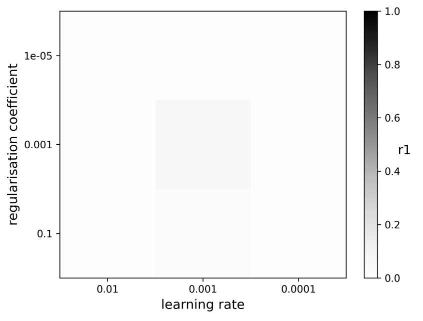
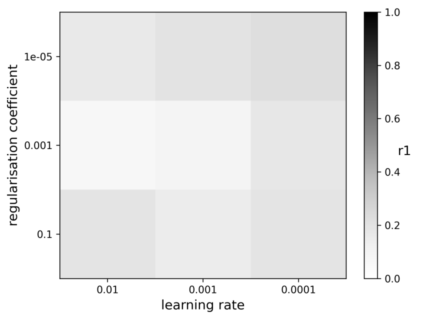
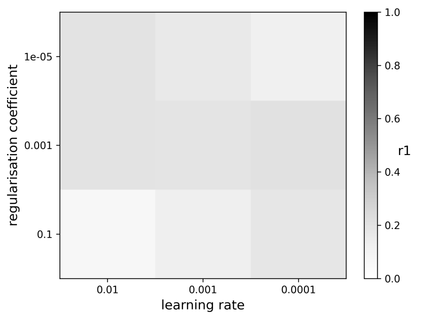
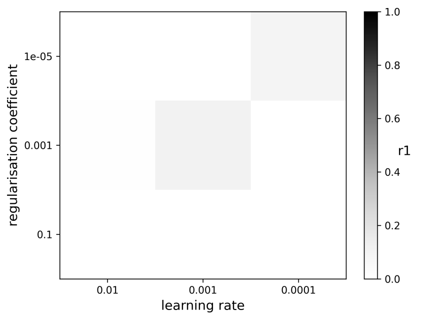
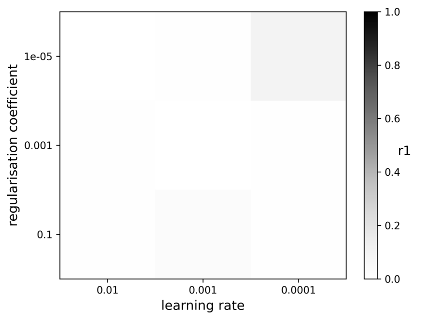
PharmGKB		

Figure 5.3. The average effect of the hyperparameters on r1 over 3 iterations, when the ranking loss function and diagonal operator are used.

Softmax loss function and Diagonal operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		

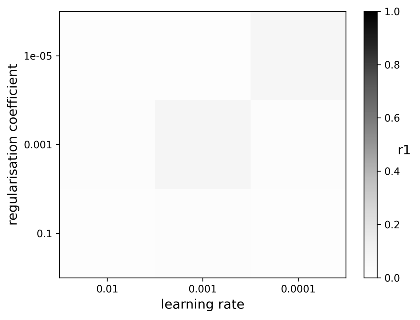
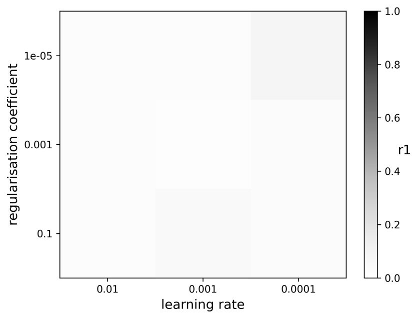
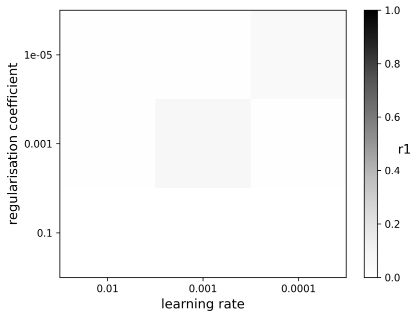
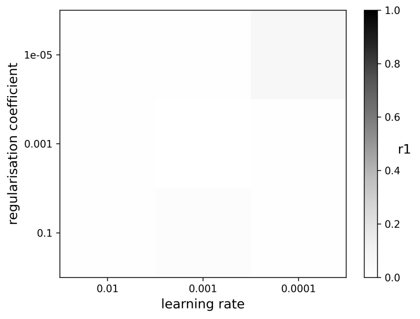
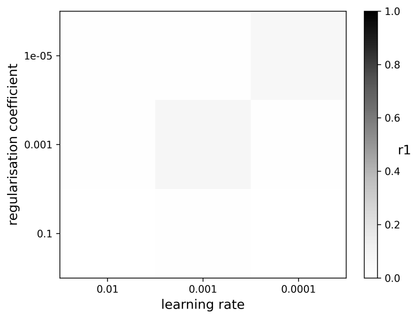
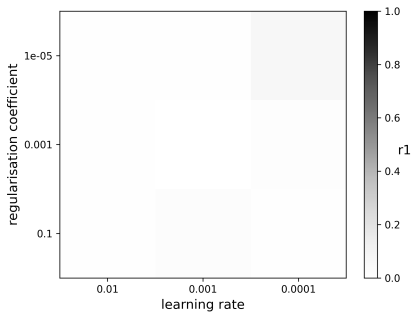
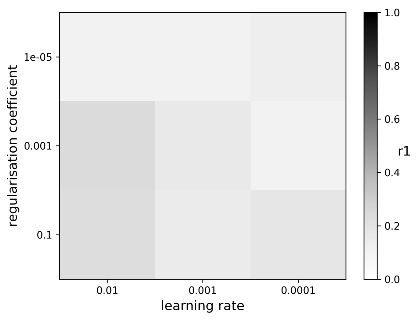
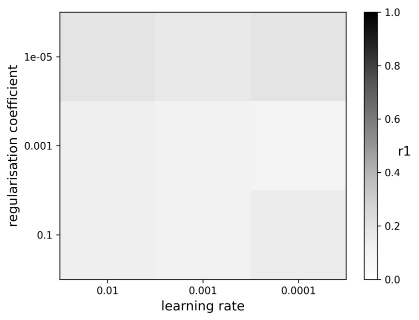
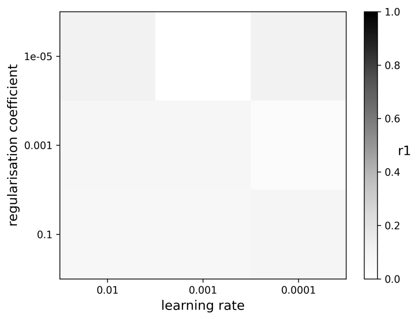
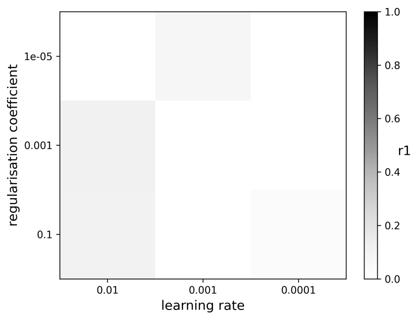
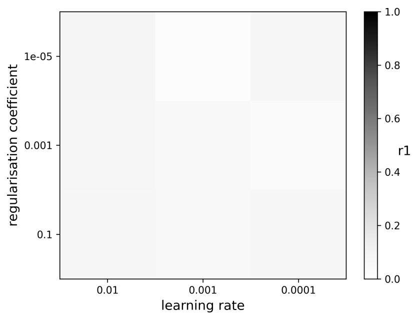
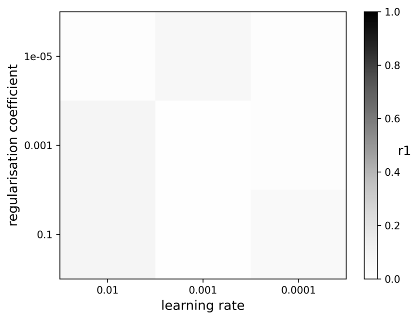
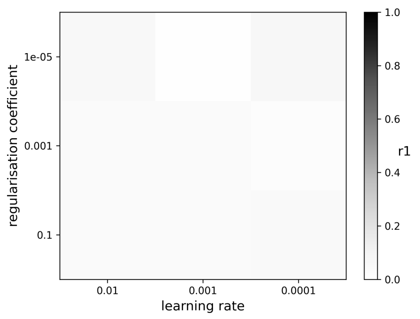
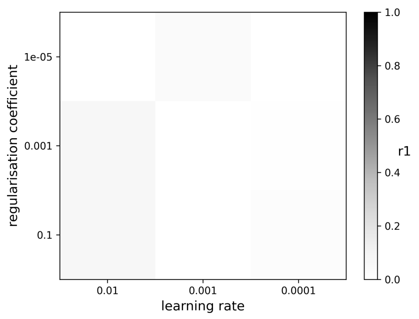
DrugBank		
OMIM		
PharmGKB		

Figure 5.4. The average effect of the hyperparameters on r1 over 3 iterations, when the softmax loss function and diagonal operator are used.

Ranking loss function and Translation operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		
DrugBank		
OMIM		

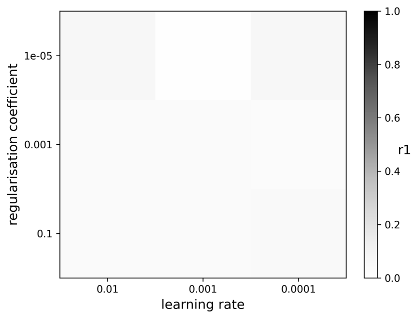
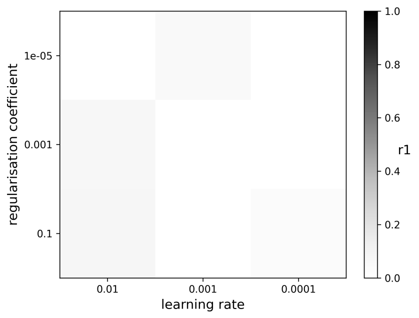
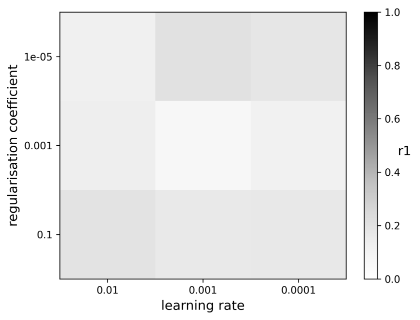
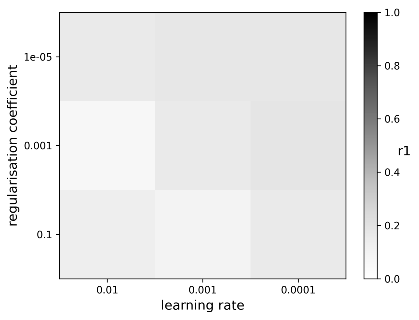
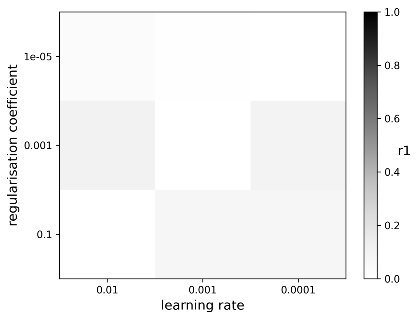
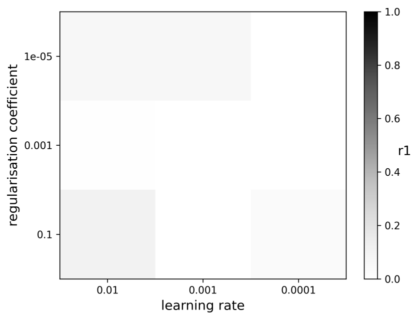
PharmGKB		

Figure 5.5. The average effect of the hyperparameters on r1 over 3 iterations, when the ranking loss function and translation operator are used.

Softmax loss function and Translation operator:

Dataset	**L_2 Comparator**	**Dot Comparator**
BioPortal		
DBSNP		

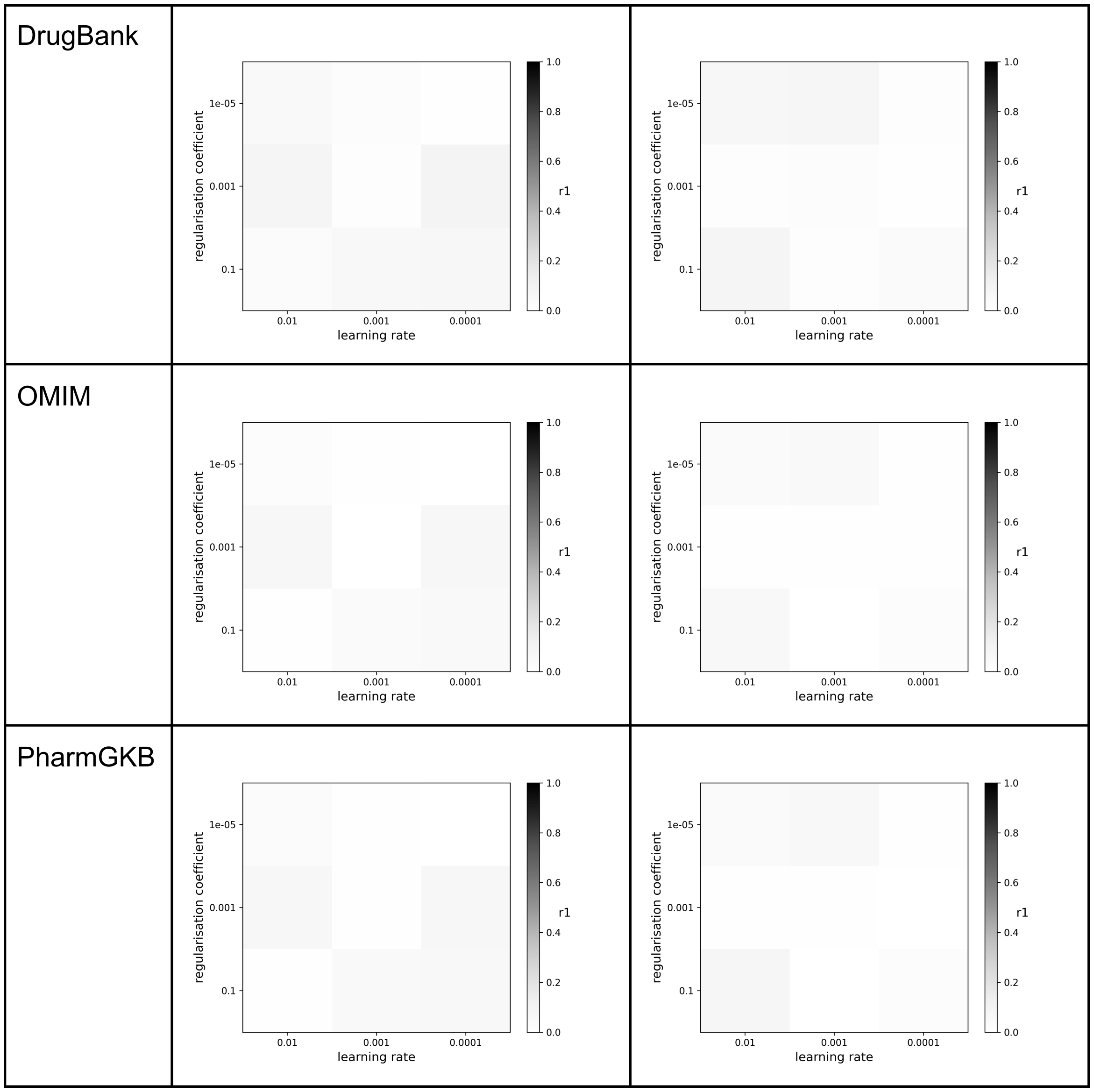


Figure 5.6. The average effect of the hyperparameters on r1 over 3 iterations, when the softmax loss function and translation operator are used.

Several trends were noted from the three grid search iterations on these datasets. First, BioPortal consistently performs better--over a wide range of hyperparameterisations--than any of DBSNP, DrugBank, OMIM, and PharmGKB. However, for all the datasets the translation operation was most enriched for relatively high r1 scores. Within the translation r1 scores, consistent patterns in hyperparameter combinations were seen across DBSNP, DrugBank, OMIM, and PharmGKB; however, BioPortal was an exception to these trends under both loss functions and under both comparators.

Quite notable, under softmax, the combination of a learning rate of 0.0001 (1e-4) and a regularization coefficient of 0.001 (1e-3) was the best or second best for all four of those datasets, BioPortal excluded. The softmax operator outperformed ranking in general for translation, and for these four datasets L_2 distance was similarly preferred over dot product as a comparator. As thus, these hyperparameters were chosen to be used by those four datasets.

As to BioPortal, a different configuration was clearly needed as the above-mentioned assignments of hyperparameters were suboptimal, resulting in r1 scores that were among the lowest rather than the highest. While many different configurations led to relatively high r1 scores for bioportal, the ranking loss function with a translation operator, L_2 as a comparator, a learning rate of 0.01, and a regularization coefficient of 0.1 was particularly good, yielding an average r1 of 0.23. Moreover, the standard deviation of that point was only 0.0067, reflecting a very high confidence in the given r1 score under those hyperparameters. Unfortunately, there were no other datasets that worked with those same hyperparameters, so it was not possible to estimate whether that assignment would be generalizable given the five testing datasets that were in use. In any case, these hyperparameters were selected for bioportal.

Thus, the output of this round was not one set of hyperparameters, but two: one for bioportal, and one for the other four datasets DBSNP, DrugBank, OMIM, and PharmGKB. In the next two rounds, BioPortal was thus evaluated independently of the other datasets, its configuration no longer being comparable to theirs.

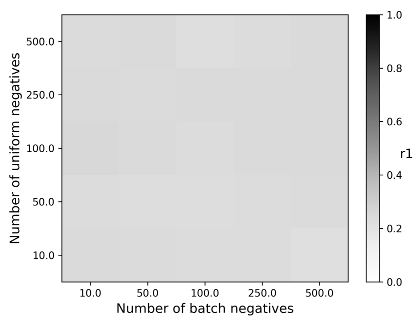
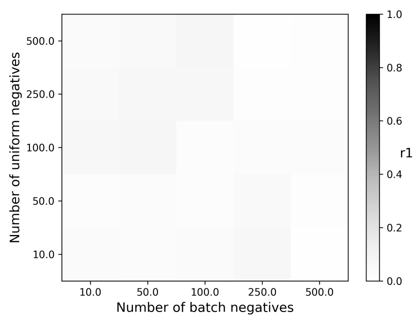
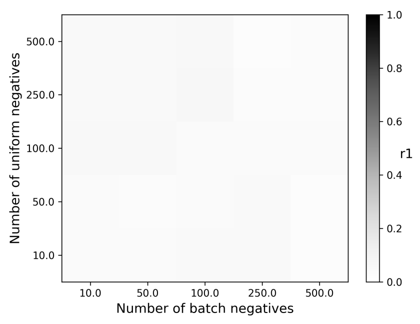
5.3 Round 2: Batch-related hyperparameters

In the second round, batch-related hyperparameters and model choices were considered. The full list of considered hyperparameters for this round and the values examined for each of them, are given in Tables 3.4 and 3.5 in Chapter 3.

When running the grid search as described in the methodology, hyperparameters were selected such that they would most-reliably lead to the highest r1 score for the KGE model. This search was run in triplicate on the five test datasets: BioPortal, DBSNP, DrugBank, OMIM, and PharmGKB.

Since a total of 3 hyperparameters were examined, visualizing them and the corresponding r1 scores on the same graph was not feasible. In order to account for this, r1 was plotted against the number of batch negatives and the number of uniform negatives for each value of the batch size. The averaged results of the three iterations on each of the datasets are displayed below in Figures 5.7 to 5.10. Note that, although BioPortal is displayed with the other datasets, it was run on a different hyperparameter configuration (as explained in Section 5.2) and thus cannot be compared with the four other datasets.

Batch size: 500

Dataset	**Graph**
BioPortal	
DBSNP	
DrugBank	

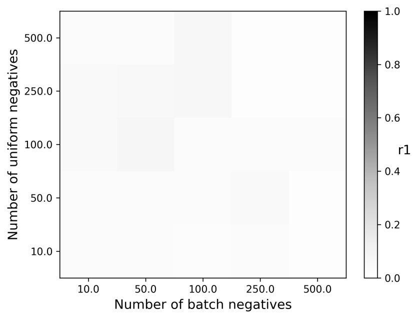
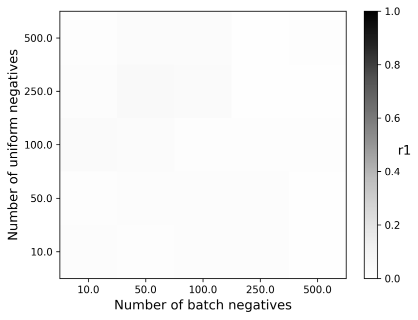
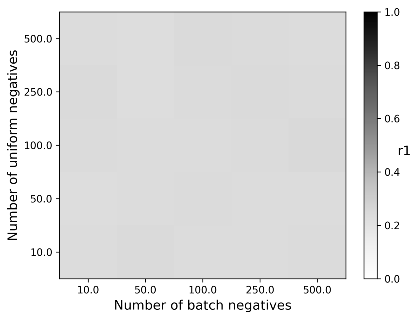
OMIM	
PharmGKB	

Figure 5.7. The average effect of the hyperparameters on r1 over 3 iterations, when the batch size is 500. Note that for bioportal the loss function used was ranking, while the other datasets used softmax.

Batch size: 1000

Dataset	**Graph**
BioPortal	

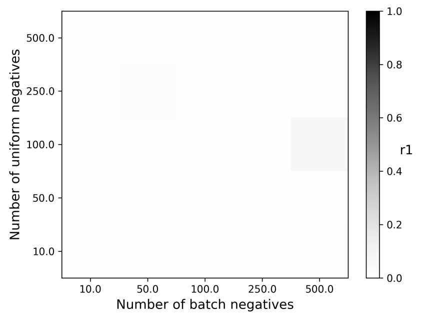
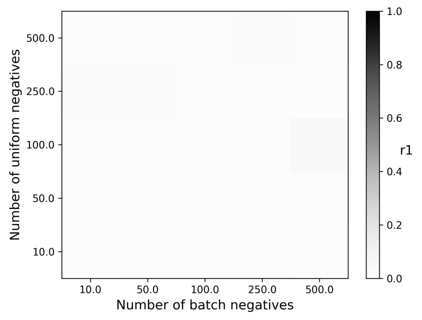
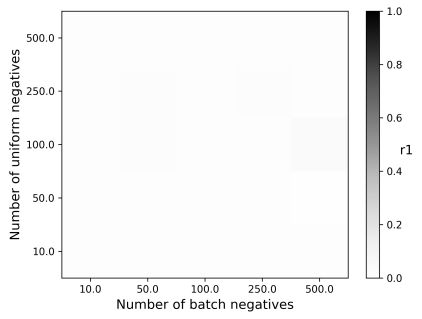
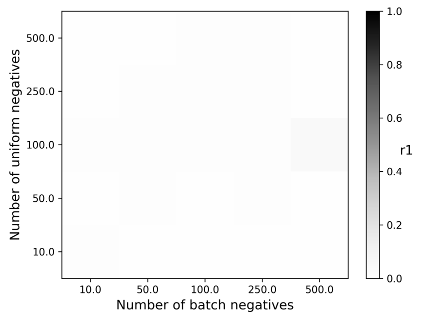
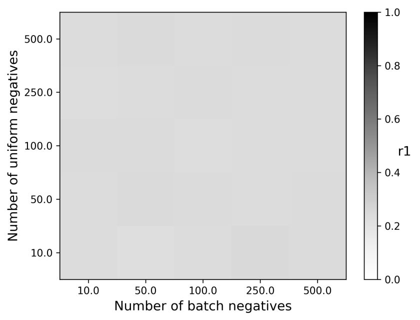
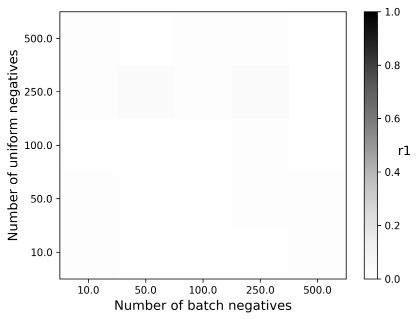
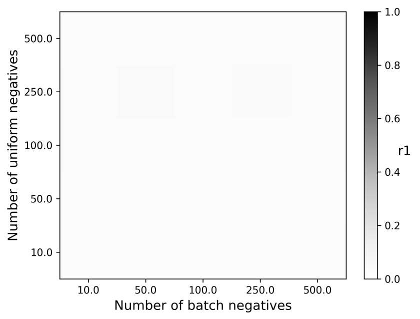
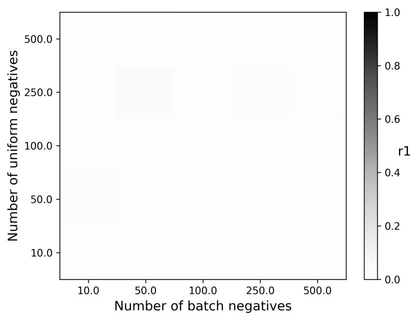
DBSNP	
DrugBank	
OMIM	
PharmGKB	

Figure 5.8. The average effect of the hyperparameters on r1 over 3 iterations, when the batch size is 1000. Note that for bioportal the loss function used was ranking, while the other datasets used softmax.

Batch size: 1500

Dataset	**Graph**
BioPortal	
DBSNP	
DrugBank	
OMIM	

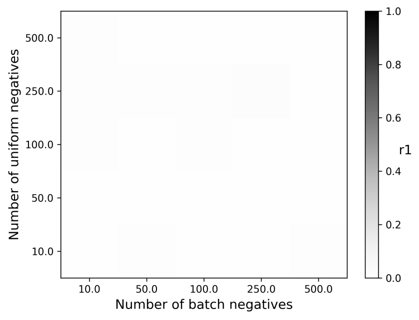
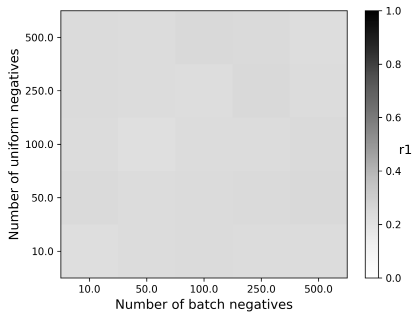
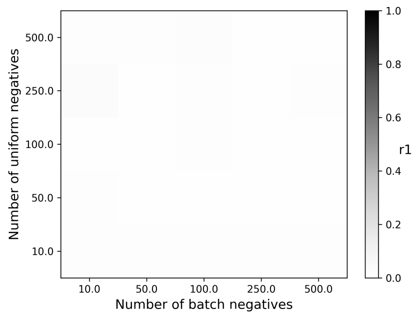
PharmGKB	

Figure 5.9. The average effect of the hyperparameters on r1 over 3 iterations, when the batch size is 1500. Note that for bioportal the loss function used was ranking, while the other datasets used softmax.

Batch size: 2000

Dataset	**Graph**
BioPortal	
DBSNP	

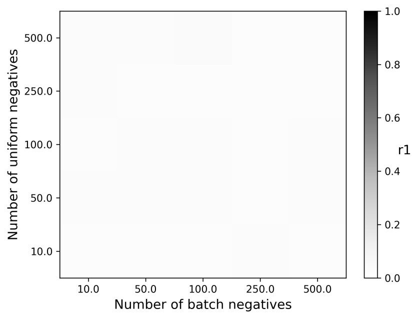
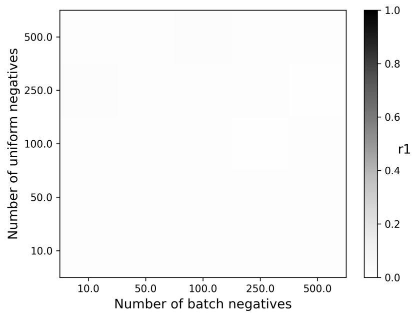
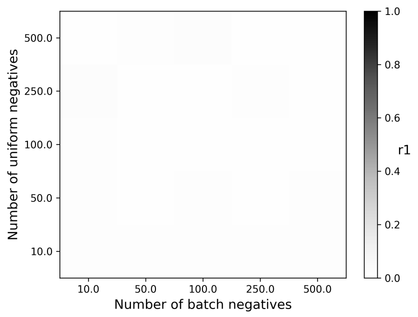
DrugBank	
OMIM	
PharmGKB	

Figure 5.10. The average effect of the hyperparameters on r1 over 3 iterations, when the batch size is 2000. Note that for bioportal the loss function used was ranking, while the other datasets used softmax.

Similar to what was seen in the first round, BioPortal had a much higher r1 values for nearly all parameterizations that did any of the other four datasets. However, in this case the differences between different parameterizations were very small, with the heatmap nearly uniform in most cases. A similar trend was seen among the other four datasets--the heatmaps nearly uniform, reflecting a lack of strong influence of any of the hyperparameters on the final r1 score.

In the case of BioPortal, the variation in the r1 scores for different hyperparameter combinations was sufficiently low that even an arbitrary choice would not have been poor. However, using 1000 as the batch size, 500 batch negatives, and 100 uniform negatives resulted in a relatively high r1 score of 0.253, and notably had a low standard deviation of 0.029. As thus, this parameterization was both reliable and high-scoring, and as a result was selected for BioPortal.

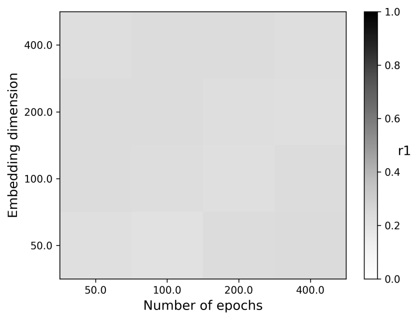
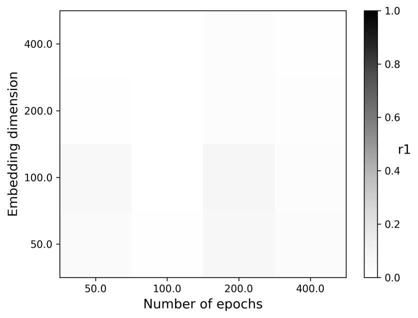
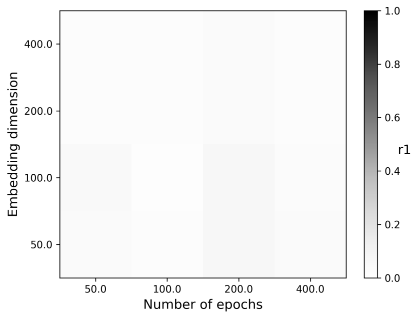
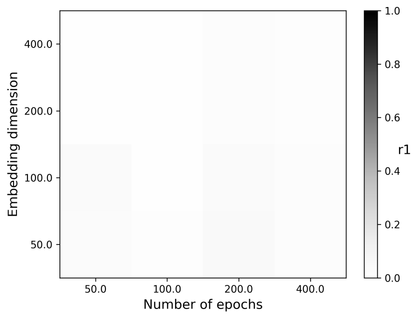
For the other four datasets, there was some patterning--as well as relatively high-scoring hyperparameter combinations under a batch size of 500. In this case, there was a slight preference towards lower numbers of batch negatives, and middling values for uniform negatives. In particular, using 100 batch negatives and 250 uniform negatives was optimal or near-optimal for all of these four datasets. As thus, these hyperparameters were chosen from the four datasets.

5.4 Round 3: Epochs and embedding dimension

In the third and final round, the embedding dimension and number of epochs to use, as explained in Tables 3.4 and 3.5 in Chapter 3.

When running the grid search as described in the methodology, hyperparameters were selected such that they would most-reliably lead to the highest r1 score for the KGE model. This search was run in triplicate on the five test datasets: BioPortal, DBSNP, DrugBank, OMIM, and PharmGKB. Note that, although BioPortal is displayed with the other datasets, it was run on a different hyperparameter configuration (as explained in Sections 5.2 and 5.3) and thus cannot be compared with the four other datasets.

The averaged results of the three iterations on each of the datasets are displayed below in Figure 5.11.

Dataset	Graph
BioPortal	
DBSNP	
DrugBank	
OMIM	

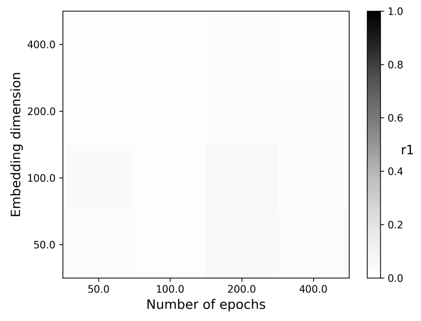
PharmGKB	

Figure 5.11. The average effect of the number of epochs and embedding dimension on r1 over 3 iterations.

The difference in r1 scores for different dimensions and epochs was relatively low for all datasets considered, similar to what was seen in the previous round. However, there were some clear patterns.

For bioportal, when epochs were under 200, there was a clear preference towards higher dimensions for higher r1 values. However, when the number of epochs was equal to or exceeded 200, lower embedding dimensions were preferred. The single best r1 score came from 400 dimensions and 50 epochs, with 200 dimensions and 50 epochs coming in second place. However, the combination of very high dimensions and low epochs was concerning--that sort of a parameterization would very easily lead to overfitting, and especially since the grid search was done on subsetted data it was difficult to be confident that the model was not using the higher dimensions to memorize data rather than understand it. This is doubly true since learning true patterns, rather than simply memorizing common mappings, would be expected to require more epochs to do. As thus, the second-best hyperparameterisation was chosen to reduce the chance that the model would overfit.

For the other four datasets, low numbers of epochs yielded the lowest scores in almost all cases, regardless of embedding dimension. However, interestingly, the use of 200 epochs and 100 dimensions was universally either the best or second best for these datasets. As thus, these two values were selected.

5.5 Analysis of the hyperparameters

Ultimately, two different sets of hyperparameters were created: one for BioPortal, and one for DBSNP, DrugBank, OMIM, and PharmGKB. Very notably, in almost all cases the distribution of r1 scores given different hyperparameter combinations for DBSNP, DrugBank, OMIM, and PharmGKB matched very closely. This reflects a similarity in these datasets, and a possibility of finding other datasets that may also be similarly described by the hyperparameters found for this set.

Since BioPortal was in a set of its own, it is less clear how many other datasets its parameters could be expected to match. Similarly, since BioPortal was significantly larger than the other four datasets, it is unclear if the difference in the optimal hyperparameters is an effect of size or of structure of the datasets. Chapter 6 will continue this discussion with an analysis of graph structure and how it relates to hyperparameter choices, taking into account as well other datasets of various sizes to determine whether there is an effect of size or just one of structure.

For reference, a summary of all the selected hyperparameters from both hyperparameter sets is given below in Table 5.1. Table 5.2, directly following, gives a summary of the final r1 scores from each hyperparameter validation round described above, as well as those scores obtained by running the full datasets with their respective hyperparameter sets. Note that, for ease of reference, the set of parameters that best fits KGEs for BioPortal will be referred to as the "BioPortal configuration" herein, and the hyperparameter set corresponding to the other datasets as the "general configuration."

BioPortal ("BioPortal Configuration")		**DBSNP, DrugBank, OMIM, and PharmGKB ("General Configuration")**	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
Model-related Hyperparameters		*Model-related Hyperparameters*	
Comparator	L_2	Comparator	L_2
Learning rate	1e-2	Learning rate	1e-4
Loss function	Ranking	Loss function	Softmax
Operator	Translation	Operator	Translation

regularisation coefficient	1e-1	regularisation coefficient	1e-3
Batch-related Hyperparameters		*Batch-related Hyperparameters*	
Batch size	1000	Batch size	500
Number of batch negatives	500	Number of batch negatives	100
Number of uniform negatives	100	Number of uniform negatives	250
Epochs and Dimensions		*Epochs and Dimensions*	
Embedding dimension	200	Embedding dimension	100
Number of epochs	50	Number of epochs	200

Table 5.1. A summary of the hyperparameter sets chosen for both BioPortal (individually) and for the remaining four datasets (together).

Dataset	**Round 1**	**Round 2**	**Round 3**	**Full datasets (post-round 3)**
BioPortal	0.2304	0.2525	0.2363	0.3894
DBSNP	0.1020	0.0737	0.0776	0.1306
DrugBank	0.0945	0.0627	0.0647	0.1067
OMIM	0.0684	0.0713	0.0463	0.1209
PharmGKB	0.0742	0.0443	0.0609	0.0757

Table 5.2. The r1 scores obtained in each of the three rounds listed above, as well as those obtained by running KGE models on the full datasets using the selected hyperparameters. Note that the values obtained for the three rounds are averages over a sample size of three, while data for the full datasets is a point-estimate since running the datasets in triplicate was not feasible within the time limitations of this work.

6 Results

6.1 Overview of the Results

Section 6.2 presents the base results obtained from the KGE models: number of partitions used and the r1 and AUC scores of each dataset run under the selected hyperparameter models. Section 6.3 continues to explain the structural metrics identified for each dataset. Section 6.4 concludes the sections with a synthesis of these two sets of results and proposes a framework by which variations in model performance and hyperparameter set choice can be understood.

6.2 KGE Model Results

When running the KGE models, different numbers of partitions were required for each. The number of partitions depended not only upon the size of the raw dataset, but on the configuration used and on the dataset itself. In fact, several similarly sized datasets such as BioPortal, GOA, and KEGG needed different numbers of partitions to run, indicating that the raw size of a dataset alone is not a perfect predictor of KGE model memory consumption. The data for the number of partitions used on each dataset is shown below in Table 6.1.

Dataset	**Number of partitions: BioPortal configuration**	**Number of Partitions: general configuration**
Datasets used in the hyperparameter searches		
BioPortal	5	5
DBSNP	2	2
DrugBank	2	2
OMIM	2	2
PharmGKB	2	2
Datasets not used in the searches		
GOA	6	4
HGNC	1	1
KEGG	9	5
LSR	1	1

Table 6.1. The number of partitions used for each dataset under both hyperparameter configuration options.

The results for the r1 and AUC scores for each dataset with each hyperparameter configuration are given below. All of these results were obtained after running the KGE model using the corresponding number of partitions given in Table 6.2.

Dataset	**r1**	**AUC**
Under BioPortal Configuration		
BioPortal	0.3894	0.8100
DBSNP	0.1907	0.6853
DrugBank	0.2068	0.7115
OMIM	0.1381	0.6634
PharmGKB	0.1312	0.6814
GOA	0.3099	0.7362
HGNC	0.1257	0.6629
KEGG	0.2948	0.7126
LSR	0.1873	0.6716
Under General Configuration		
BioPortal	0.3912	0.8103
DBSNP	0.1306	0.7013
DrugBank	0.1067	0.7190
OMIM	0.1209	0.6428
PharmGKB	0.0757	0.6584
GOA	0.2373	0.7622
HGNC	0.0711	0.6860
KEGG	0.0938	0.7841
LSR	0.14600	0.6976

Table 6.2. The r1 and AUC scores of each model-configuration pair that was tested.

Within the context of these results alone, there are a few trends worth noting. AUC scores universally exceed r1 scores, often by two to four times the raw numeric value. As mentioned in Chapter 4, this is indicative of the models being much better suited to general knowledge than to specific knowledge. While the models can distinguish a true statement from arbitrary false statements effectively (as measured by the relatively high AUCs) they cannot nearly as effectively distinguish a true statement from similar, but false, statements (as measured by r1). Even in the datasets with the highest AUCs (such as BioPortal and GOA), the AUC score approximately doubles the observed r1 score.

In Chapter 4, it was noted the AUC and r1 were not consistently correlated. This statement was revisited in the light of the final results. The r1 and AUCs scores of each model were plotted against each other for each of the two hyperparameter configurations used, as shown in Figure 6.1.

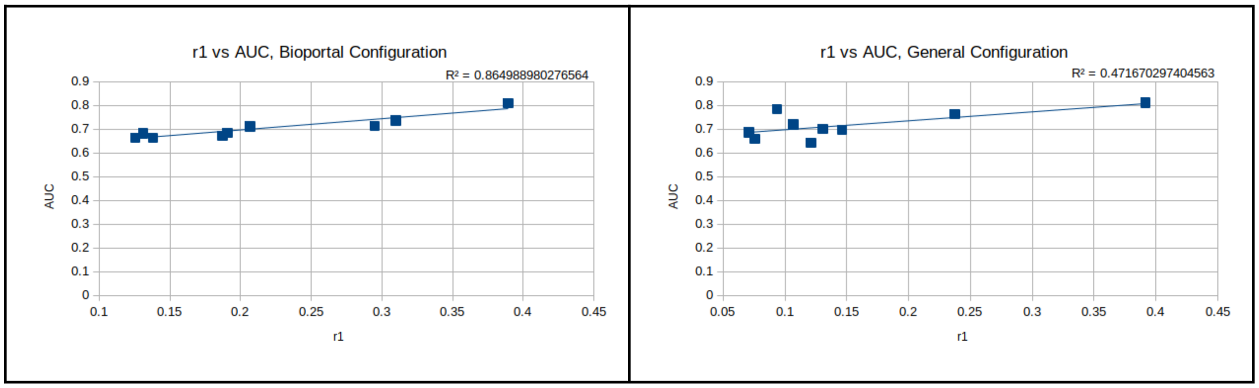


Figure 6.1. A plot of the r1 and AUC scores of each model tested against each other, for each of the two hyperparameter configurations. A trendline is given, and it's R^2 value is shown in the upper-right hand corner.

In the BioPortal configuration plot the trend is clearly linear, and the data points were spread out roughly evenly along the length of the line. The R^2 value of this correlation was 0.8650, meaning that roughly 86.5% of the variation in either r1 or AUC can be explained by understanding variation in the other one alone.

The fact that the correlation was positive, as well as its high value, indicate that in the final model general and specific knowledge (as measured in r1 and AUC) are more strongly correlated than was seen in the first round of the hyperparameter search. Overall, this reflects an increase in the predictive and understanding power of these models: the choice of better hyperparameters brought these two forms of understanding closer into alignment, reflecting an increase in the alignment of both generalized and specific knowledge. As thus, while the low r1 scores indicate that the models are far from perfect, the stronger correlation of r1 and AUC reflect a better ability to use and represent knowledge in the KG though the resultant KGEs. While exploring the cause of this effect is beyond the scope of this work, this result generally indicates a capacity to model the data at hand at both the specific and general levels.

Under the general configuration, the results were not as clear. While the overall R^2 value was 0.4717, the data points were almost all clustered in a single cluster at low values of r1. Only two points, at r1 values of 0.2373 and 0.3912 respectively, were significantly distanced from that cluster. These two data-points came from the KEGG and Bioportal datasets respectively. Moreover, by removing these points from analysis, the correlation between r1 and AUC became negative, with an R value of -0.0227 and a corresponding R^2 value that approached zero, at 0.0051. Thus, while the overall trend is linear, nearly all of the strength of the detected linearity comes from only two points pulling the correlation upwards to higher positive values, and a cluster of other points whose correlation is insignificant.

Overall, this reflects a failing of the general configuration, especially in comparison to the bioportal configuration. For it, increasing specific knowledge (as reflected by r1) did not necessarily correspond to an increase in general ability to discern true facts against any false ones (as reflected by AUC). Moreover, the deviation KEGG and BioPortal points from the trend visible in the rest of the points suggests that much of the supposed correlation may be an artefact of their high scores, rather than a reflection of the r1-AUC trend of any model under the general configuration. Overall, this casts a level of doubt on the fitness of the general configuration for all datasets.

Even more interestingly, the general configuration yielded its best scores on datasets it had not been created to accommodate (Bioportal and KEGG). Those datasets it was trained on were (DBSNP, DrugBank, OMIM, and PharmGKB) were all a part of the uncorrelated cluster at low r1 values. In fact, the BioPortal configuration outperformed the general configuration on all datasets except one: BioPortal. However, in the case of BioPortal, the observed r1 scores of 0.3894 under the BioPortal configuration and 0.3912 under the general configuration have a very small difference, and it is possible that this variation is due to chance. However, due to time limitations and due to the time expense of running the KGE models, obtaining secondary or tertiary estimates for the r1 values of BioPortal under both configurations was not feasible, so the statistical significance of this (and all other) differences could not be assessed beyond the level of point estimates.

The departure of these results from the expected ones--that the BioPortal configuration would be optimal for BioPortal by a large margin and the general configuration would be similarly superior for DBSNP, DrugBank, OMIM, and PharmGKB from which it was created--casts doubt on the hyperparameter choice. However, it must be noted that these are all point estimates--most of the models took on the order of 1 to 5 days to run, and as thus running them multiple times was not feasible in the time available. These are thus point estimates, and the degree of their variation is unknown.

Supposing the general trend outlined above is indeed generally true, however, this suggests three possible conclusions. For one, it is likely (as noted in Table 4.5.1.1 in Section 3.8) that the difference in structure between the subsetted data and the full datasets led to sub-optimal hyperparameter selection. Second, it is very possible that the iterative method by which hyperparameters were selected was improper, since it was biased by the initial, arbitrarily chosen values and thus could have failed to converge on optimal hyperparameters. For the third, it is possible that BioPortal and KEGG, the only two datasets to have relatively high r1 scores under the general configuration, are significantly structurally different in some manner from the rest of the datasets, which performed universally poorly under that configuration despite having varied levels of success under the Bioportal configuration.

In order to address these questions, the structure of all datasets in question was analysed, and the results of this are detailed in the following Section 6.3.

6.3 Structural Results

Several structural elements of the graphs were considered, as explained in Section 3.6 of the Methodology. The first and simplest was size, measured in the number of triples, of each dataset. The data for the size of each dataset is given in Table 6.3.

The numbers and ratios of sources, sinks, and repeated triples in the graph were also considered. In this case, the number of unique triples that were only subjects (i.e., sources), only objects (i.e., sinks) or were used on both sides of different predicates (repeats) were counted. These values were also considered in proportion to the number of triples in the graph, and all of these metrics are given in Table 6.4.

Finally, the distribution of the centralities of all nodes in the graph was measured. Since the range of these distributions was immense, and since the differences in each dataset could be effectively and comprehensively characterized with relatively few percentiles, the data is given in tabular form. The 1st quartile (or 25th percentile), 2nd quartile (median or 50th percentile), 3rd quartile (75th percentile), 90th percentile, maximum, and average centrality values were calculated for each graph. Moreover, the ratio of the maximum centrality value to the number of triples in the graph was calculated. These values are displayed in Table 6.5.

Dataset	**Number of Triples**
Datasets used in the searches	
BioPortal	91058582
DBSNP	12633560
DrugBank	6497964
OMIM	10504995
PharmGKB	7113312
Datasets not used in the searches	
GOA	86404416
HGNC	4267572
KEGG	89195261
LSR	60701

Table 6.3. The size in triples of each of the datasets considered.

Dataset	Number of Sinks	Number of Sources	Number of Repeats	Ratio of Sinks to Triples	Ratio of Sources to Triples	Ratio of Repeats to Triples
Datasets used in the searches						
BioPortal	13623972	4385915	4577592	0.1496	0.0482	0.0503
DBSNP	2981629	586130	550671	0.2360	0.0464	0.0436
DrugBank	2079154	414454	399251	0.3200	0.0638	0.0614
OMIM	4507308	1121071	1122401	0.4291	0.1067	0.1068
PharmGKB	2811465	641974	521171	0.3952	0.0902	0.0733
Datasets not used in the searches						
GOA	15674396	3073804	3091543	0.1814	0.0356	0.0358
HGNC	1897778	415958	416245	0.4447	0.0975	0.0975
KEGG	27216889	8622153	8504190	0.3051	0.0967	0.0953
LSR	26089	6137	6130	0.4298	0.1011	0.1010

Table 6.4. The count and ratio of sinks, sources, and repeated entries in each dataset. Ratios given are to the total number of triples in the graph.

Dataset	Avg	1st quartile	2nd quartile	3rd quartile	90th percentile	Max	Max to Triples Ratio
Datasets used in the searches							
BioPortal	9.2939	1	2	6	15	6504014	0.0714
DBSNP	7.0801	1	1	1	17	697049	0.0552

DrugBank	5.2104	1	1	1	12	428085	0.0659
OMIM	3.7322	1	1	1	7	1185612	0.1129
PharmGKB	4.1191	1	1	1	8	666924	0.0938
Datasets not used in the searches							
GOA	9.2163	1	1	1	16	6080694	0.0704
HGNC	3.6888	1	1	1	8	416264	0.0975
KEGG	4.9774	1	2	2	12	12115498	0.1358
LSR	3.7672	1	1	2	8	4359	0.0718

Table 6.5. The distribution of centralities among all entities in the graphs. All percentile values presented are all raw numbers, not normalised to the number of triples in the graph.

Within the scope of the structural results alone, several notable trends present themselves. With regard to the analysis of sources, sinks, and repeats, in all datasets considered there were many more sinks than there were either sources or repeats, often by a factor of 10. Moreover, sources and repeats tended to occur within similar numbers within a given dataset. It is particularly notable that these graphs contain many literal values, which in RDF can only be objects, making them sinks. Moreover, all subjects have to be URLs. As thus, this tendency to have far many more sinks than sources or repeats reflects a greater tendency towards single-use objects and literals.

Moreover, the proportions of sinks, sources, and repeats were similar across all datasets considered. The ratio of sinks varied from 0.15 to 0.44, with an average of 0.32. The ratio of sources and repeats both varied from 0.036 to 0.11 and averaged at 0.076 and 0.074 respectively. This similarity across datasets suggests that the biomedical datasets here are more similar than they are different, and that they all have a strong skew to having many sinks and relatively few items that are either only sources, or that are used on both sides of a predicate.

Turning to the centrality data, a second major bias of the datasets becomes clear: in all but two datasets (those being BioPortal and KEGG), the median centrality of entities is 1. In other words, seven of the nine considered datasets have at least half of their data stored in entities that appear in a triple only once in the entire KG. At an even higher extreme, six datasets (excluding BioPortal, KEGG, and LSR) have 1 as their third quartile as well, indicating that at least 75% of their entity data is contained in entities that only appear once in the entire KG. Even in the cases of BioPortal, KEGG, and LSR, their median and third quartile centralities are all very low--below 10 in all cases. Even at the 90th percentile, no graph has an entity with degree centrality exceeding 20.

Given this, as well as the aforementioned bias towards sinks over either sources or repeats, as well the trend of these datasets to contain literal strings as data storage mechanisms, it is clear that all these datasets are very biased towards few-use objects and object literals. Quite notably, however, are the extreme values of the maximum centralities observed in the graph. All of these graphs are marked by a tendency to have very few entities that are used on many occasions. In the case of bioportal, where the maximum centrality is 6504014, that single entity is present in approximately 7% of all triples in the KG. KEGG, at an even higher extreme, has a single entity present in approximately 13% of all triples. DBSNP, at the lowest extreme, has its most central entity present in approximately 5.5% of all triples. Note that this interpretation of the max centrality to number of triples ratio can only be approximate: while uncommon, it is possible that an entity could appear twice in the same triple as the subject and object, which would increase its centrality without increasing the number of triples it is involved in.

This sort of extreme bias towards many low-centrality sinks and few high-centrality nodes is a hallmark of all these biomedical datasets. This similarity already suggests why similar patterns of optimal hyper parameter combinations were seen in the r1-based search in Chapter 5, and also suggests that their datasets in total have more in common than they have apart. Moving into Section 6.4, this relationship will be investigated in full.

6.4 Synthesis of Results

6.4.1 Measuring correlation of structure with KGE performance

In order to ascertain how structure of the graph related to the optimal set of hyperparameters, regression analysis was performed to analyse the power of the structural elements to predict r1 and AUC scores. When doing so, the data for both hyperparameter configurations were run separately, so that the relationship between structure and KGE performance could be analysed in the context of the given hyperparameters. This also allows predicting the difference in scores for a dataset of the same structure under both models, and thus allows estimating the best-fit hyperparameter configuration using that data. This second direction is the topic of Section 6.4.2.

The structural features selected to measure the effect of the high prevalence of sinks versus sources and repeats were the ratios of sinks and repeats to triples. The ratio of sources to triples was not included, since in all cases it was nearly identical to the ratio of repeats to triples and thus was redundant. Moreover, adding in more features on datasets with few data points can lead to machine learning models overfitting by memorizing data rather than learning general trends, which would make interpretation of the results less clear.

The ratio of the maximum centrality to the number of triples, as well as median, 3rd quartile, and 90th percentile centralities were used to represent the effects of centrality. Notably, the first quartile was not included since it was identical in all datasets.

These features were used to predict the r1 and AUC scores of all models in two rounds. In the first round, they were kept separate from each other and used to predict both AUC and r1 scores given both hyperparameter configurations. This resulted in a total of 8 models, whose results are given in Section 6.4.1.1. In the second round the most important features from the first round were pooled, resulting in a smaller set of models whose results are given in Section 6.4.1.2.

All data values were normalised prior to running the regression models, and all regression models were run with 5-fold cross-validation using an L_1 (or “Lasso”) penalty. The Lasso regression penalty was chosen since it tends to drive the values of parameters that are not needed in the regression decision to zero, thus providing an easy tool for the detection of which structural elements are important and which are of no use for gaining predictive power. Since all input values were normalized, the parameter values themselves can be used as an estimate of their importance to the regression decision within the context of a single model: those with higher values represent structural features that have a created correlation to the final r1 or AUC score obtained by the KGE model. It must be noted, however, that care must still be taken in this interpretation of the parameter values, especially since interaction terms were not considered and thus some effects of the variance of each parameter on the model may not be fully contained in the given data.

For each case, the overall predictive value of the model was assessed using its R^2 score, and the relative importance of each of the structural features by the absolute value of the coefficients assigned to them.

6.4.1.1 Predictive Analysis Round 1

The cross-validation plots and selected hyperparameters to the regression models are given in Appendix I. The R^2 results thereof are included here in Table 6.6, and the parameters to each feature in the model are given in Tables 6.7 and 6.8. Note that since the input values were normalised, within the context of any one model these parameter values can be compared using their relative value as a metric for importance in the prediction.

Data used	BioPortal Configuration	General Configuration
Sink-repeat ratios	`Predicting AUC: 0.7735 Predicting r1: 0.9076`	`Predicting AUC: 0.8078 Predicting r1: 0.6278`
Centrality distribution	`Predicting AUC: 0.7663 Predicting r1: 0.8282`	`Predicting AUC: 0.7807 Predicting r1: 0.8129`

Table 6.6. The R^2 scores of each model, when run on the full datasets under the given configuration.

Model	Sinks:Triples ratio	Repeats:Triples ratio
Predicting r1, BioPortal configuration	`-0.1359`	`0.0674`
Predicting AUC, BioPortal configuration	`-0.0508`	`0.0139`

Predicting r1, general configuration	-0.1089	0.0375
Predicting AUC, general configuration	-0.0782	0.0399

Table 6.7. Coefficient values corresponding to each feature in the sinks-source-repeat ratio dataset, for each of the regression models created.

Model	**Median centrality**	**3rd quartile centrality**	**90th percentile centrality**	**Max centrality to Triples Ratio**
To r1, BioPortal configuration	0.0290	0.0237	0.0373	0
To AUC, BioPortal configuration	0	0.0238	0.0121	0
To r1, general configuration	0	0.0675	0.0199	-0.0115
To AUC, general configuration	0.0273	0.0009	0.0188	0

Table 6.8. Coefficient values corresponding to each feature in the centrality distribution dataset, for each of the regression models created.

Note that an R^2 value of zero represents a no correlation; this occurs when the model always predicts a constant value. As thus, regressors with an R^2 value of 0 can be interpreted as being no better than a baseline regressor that always predicted the average value, and those with an R^2 value above zero thus can be interpreted as outperforming such a baseline regressor.

All models examined achieved R^2 values of at least 0.60, with the best model yielding an R^2 score above 0.90. While all of the models based on sink and repeat frequencies used both features in the final regression model, none of the models using centrality distribution statistics used all of the available features; in all cases one or two were driven to zero.

To first consider the results of using sink and repeat frequencies to predict the obtained r1 and AUC scores, there is a clear correlation between these features and both r1 and AUC scores. Under the BioPortal configuration, both scores were predicted with high accuracy. As shown in the R^2 scores, the models were able to explain 77.35% and 90.76% of the variation in AUC and r1 scores respectively. The relative success at predicting both of these is expected, as it reflects the strong correlation observed between the AUC and r1 scores under the BioPortal configuration as noted in Section 6.2.

Under the general configuration, these features were able to explain 80.78% of variation in AUC scores, but only 62.78% of variation in r1 scores. The wider gap between these two R^2 scores is similarly expected, as it was noted in Section 6.2 that the correlation between AUC and r1 for the general configuration was weak.

In this case, both features (the ratio of sinks to triples and the ratio of repeats to triples) were used in all the models; none of them were eliminated by being assigned a coefficient value of zero. As such, it can be concluded that both of these structural features are useful for predicting AUC and r1 scores of the KGE models in the absence of other available predictors. Interestingly, in all cases the sinks to triples ratio was given higher weight than the repeats to triples ratio by the regressors, suggesting that the extremity of the sparsity of the graph as measured by the relative number of sinks is very important in determining how KGE models learn to score high on specific (r1) and general (AUC) knowledge metrics. The relatively lower weight of the repeats to triples ratio suggests that repeats, which may repeat many times or just once, are not as strongly indicative of model performance but still are necessary to gain a more complete picture of model success.

Turning to the centrality distribution-based model, the R^2 results were much closer to each other for all models. Of all the cases selected, predicting the model AUC score under the BioPortal distribution has the lowest R^2 score, which was still quite high at 0.7663. The highest score obtained was for predicting r1 under the BioPortal configuration, at an R^2 value of 0.8282. Overall, the closeness of the R^2 values for all these cases suggests that the centrality distribution is a more broadly applicable metric for predicting model performance at both the specific (r1) and general (AUC) knowledge levels.

Interestingly, despite the low correlation between AUC and r1 under the general configuration observed in Section 6.2, the R^2s for both of those models were closer than the R^2s for the BioPortal configuration, which had a much higher correlation between AUC and r1. However, unlike in the case of the sink and repeat ratio data, in this case not all features were used. This suggests, immediately, that predicting specific knowledge modelling ability (r1) and general knowledge modelling ability (AUC) of datasets under both configurations may be based upon different structural features of the KGE at hand.

However, if that is the case, it was not completely captured in the data obtained here. There were no clear patterns of preference to higher weight of any one feature to predicting r1 or AUC across hyperparameter configurations. In fact, the weights given to the features of each model differed significantly from the weights given to the features of all other models, which limited the generalisability of these results.

That notwithstanding, the 3rd quartile and 90th percentile of the centrality distributions were quite notable assigned non-zero weights in all cases, indicating that they all made a contribution to model performance. However, the parameter to the 3rd quartile feature of the regressor to AUC under the general configuration was nearly zero, at 0.0009. Median centrality was only included in two models and the ratio of the maximum centrality to the number of triples only included in one, as shown in Table 6.8. In tandem with the generally high parameter values assigned to these features, this suggests that the 3rd quartile and 90th percentile centralities are the most generally important to predictions of r1 and AUC scores of the KGE models of the features considered here.

Considering again the structural statistics displayed in Table 6.5, the relatively low use of the median centrality becomes clear. Of all the datasets examined, only two of them (BioPortal and KEGG) had a median centrality greater than 1. As thus, median centrality can be to an extent seen as a proxy for these datasets alone, as none of the others varied in that metric. This highlights that BioPortal and KEGG as datasets are very significantly structurally different from the others: they were the only two to have a median centrality greater than one, and this difference was enough that it was specifically modelled in the regressors even after cross-validating to control for overfitting.

While the 3rd quartile centrality value was only greater than one in three datasets (BioPortal, KEGG, and LSR), this wider use was enough to make it useful to predictions of all the models created. 90th percentile centrality, and the ratio of the maximum centrality to the number of triples, were both different across all datasets.

In considering why the ratio of the maximum centrality to the number of triples was relatively of little use to the models, there are two major explanations. In one case, it may encode redundant data to another one of the included features, thus making it superfluous. In the other case, it may not be of significant use to the prediction because of a lack of ability to correlate with r1 and AUC themselves. In order to investigate this, the R^2 values between each of the three other features (median, 3rd quartile, and 90th percentile centrality) were calculated and pairwise plots of them were constructed. This data is shown in Figure 6.2.

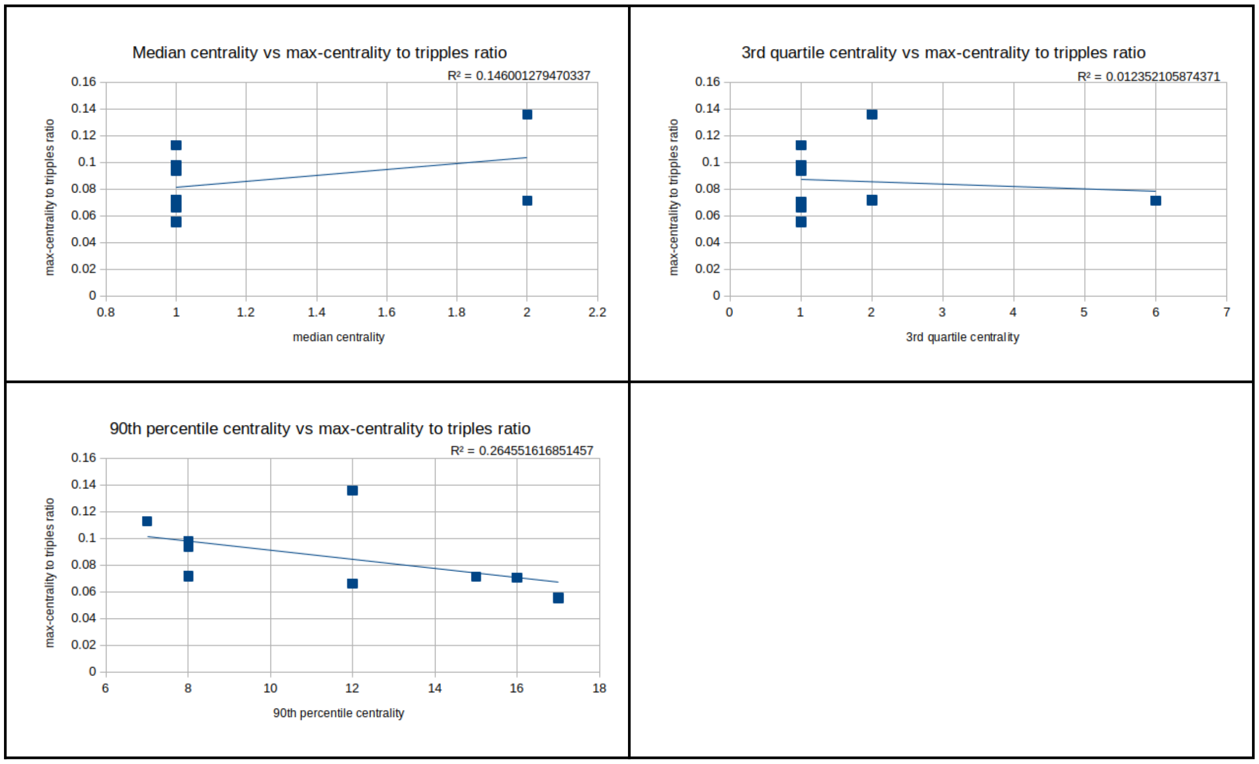


Figure 6.2. A plot of the median, 3rd quartile, and 90th percentile centralities of each model against the max-centrality to triples ratio. A trendline is given, and it's R^2 value is shown in the upper-right hand corner.

In the first two graphs using median and 3rd quartile centralities, the R^2 value is very low but also of little relevance: the data is clearly non-linear in those cases, and by inspection the trend line is clearly not a fit to the data. By inspection, both have no significant correlation, linear or otherwise, to the max-centrality to triples ratio value and thus could not be encoding redundant information that made this last feature unneeded.

The final graph does show a very roughly linear spread of points that is roughly fit by the trendline. As thus, the R^2 value can be interpreted as a measure of redundancy between the two features: the more correlated they are, the more redundancy information they encode. However, the R^2 value is somewhat low at 0.2646, indicating that variation in the 90th percentile centrality values can only explain 26.46% of the variation in the max-centrality to triples ratio values (and vice versa). To put this in other terms, however, this means that the two variables have a correlation of 0.5144, which while not particularly strong is still not particularly poor.

As such, it can be concluded that the two features do encode some amount of redundant information. However, the low-to-moderate correlation and R^2 values indicate that they do differ significantly and encode different aspects of the graph structure. This is largely as expected, since both do directly encode some information about the centrality distribution, though from different perspectives. Therefore, it is most likely that the reason that the max-centrality to triples ratio was driven to zero by the Lasso model is due to a combination of some redundant information being shared between it and the 90th percentile centrality feature and lack of relevance to the prediction task of the variation in it that is non-redundant.

Overall, these results suggest that the 3rd quartile and 90th percentile of the centrality distributions are particularly relevant to understanding both the specific (r1) and general (AUC) knowledge modelling abilities of KGE models across both configurations. It should be noted, however, that these results only hold in the context of this data, which has an extreme skewness in the centrality distribution. In other, less skewed KGs, it would not be reasonable to expect these same features to be the only, or even the most, important features relating to structure and performance.

However, the success of using these structural features to predict performance immediately yields a major finding of this work: that structure and performance are inherently related for KGEs in the context of a single hyperparameter configuration, and that this pattern extends even across different hyperparameter configurations. Moreover, it highlights that this high predictive power can be obtained in the absence of explicit modelling for graph size, suggesting that structure is more important than sheer size for understanding model performance under different hyperparameter configurations.

6.4.1.2 Predictive Analysis Round 2

As a second step, all features whose parameters were non-zero for predicting either r1 or AUC were combined into a single feature set, which was used to train a second model. The one exception was the 3rd quartile centrality feature for predicting AUC under the general configuration: it’s value of 0.0009 was nearly zero and very small compared to the values of the other parameters in that model, and as thus it was discarded so that the next model would be more simple in terms of the feature count.

It should be noted that this combined model was created only after examining the sink / repeat--based and the centrality-based data individually, since creating a regressor on all of the data immediately without verifying that it was relevant would almost certainly lead to overfitting and overconfidence in the results since the number of data points would be relatively small compared to the number of features.

Models were again created using 5-fold cross-validation on a Lasso regressor, and the full details on cross-validation and hyperparameter selection are given in Appendix II. The R^2 results thereof are included here in Table 6.9, and the parameters to each feature in the model are given in Table 6.10. Note that since the input values were normalised, within the context of any one model these parameter values can be compared using their absolute value as a metric for importance in the prediction relative to the others.

Regression Task	**BioPortal Configuration**	**General Configuration**
Predicting AUC	`0.9340`	`0.8517`
Predicting r1	`0.8804`	`0.9303`

Table 6.9. The R^2 scores of each model, when run on the full datasets under the given configuration.

Model	**Median centrality**	**3rd quartile centrality**	**90th percentile centrality**	**Max centrality to Triples Ratio**	**Sinks: Triples ratio**	**Repeats: Triples ratio**
To r1, BioPortal configuration	`0.0268`	`0.0105`	`0`	`NA`	`-0.0497`	`0`
To AUC, BioPortal configuration	`NA`	`0.0224`	`-0.0006`	`NA`	`-0.0322`	`-0.0008`

To r1, general configuration	NA	0.0406	-0.0692	-0.0322	-0.1248	0.0208
To AUC, general configuration	0.0308	NA	0.0058	NA	-0.0238	0

Table 6.10. Coefficient values corresponding to each feature in the centrality distribution dataset, for each of the regression models created. NAs were introduced when a given feature was not included in the model.

Looking at the results of the models using pooled feature sets, the R^2 values of the models are overall notably higher than those obtained in the previous round in which sink / repeat ratio data and centrality data were kept separate. The lowest R^2 value obtained was 0.8517, and the highest was 0.9430. All of the R^2s, as seen in the centrality-based models in the previous section, were close to each other for predicting both r1 and AUC.

Overall, this greater level of success could arise from two sources, which are not mutually exclusive: the use of a greater number of features in some models, which adds more information and may lead to some level of overfitting; or else the use of more relevant features to the task at hand that allow the model to focus on only the most important aspects of structure. In this case, the first source is possible, considering that only one of the models here used a greater total number of features (5) than had been used in previous rounds (2 and 4, from round 1). However, the best performing model used only 4 features, no more than had previously been considered, and still outperform all models from round 1 in terms of R^2. Moreover, many of the models drove the parameters of various features to 0, in one case resulting in only 3 features actually having any impact on predictions. As a result, it is reasonable to conclude that the increase in performance comes much more from the inclusion of more relevant features than from the simple inclusion of more features.

Of all the features considered, only the sinks to triples ratio was given a non-zero parameter value in all of the models. Moreover, in all but one case (the case of predicting r1 under the general configuration) its absolute parameter value was relatively high compared to the parameter values of the other features considered in the model.

Interestingly, all of the other parameters either take NA values (having not been of use in the previous round to the prediction) or zero (not being of use in this round despite having been of use in the previous round) for at least one of the four models. However, of the features that were included in these models (i.e., those that were not given NAs), only 2 in total were assigned parameters of 0 in the regression model: 90th percentile centrality was driven to zero once and the repeats to triples ratio was driven to zero twice. A third parameter value assigned to the ratio of repeats to triples in the model predicting AUC given the BioPortal configuration was assigned a value of nearly zero, at -0.0008, and in one other case a parameter value even smaller (0.0006) was assigned. These were far lower than the other parameter values for those models and as thus are likely of very little importance to those models. Similarly, when predicting r1 given the general configuration, the repeats to triples ratio had the lowest parameter value by a significant margin, even though it was decidedly non-zero.

This de-emphasis of the ratio of repeats to triples is much to be expected. Since this ratio also reflects the centrality distribution of the KG, it is likely to encode a significant amount of redundant information to that given by the quartiles and percentiles. To address this, once again plots and R^2 values were obtained between this feature and those suspected to contain redundant information: the 3rd quartile centrality, 90th percentile centrality, and max-centrality to triples ratio. This data is shown below in Figure 6.3.

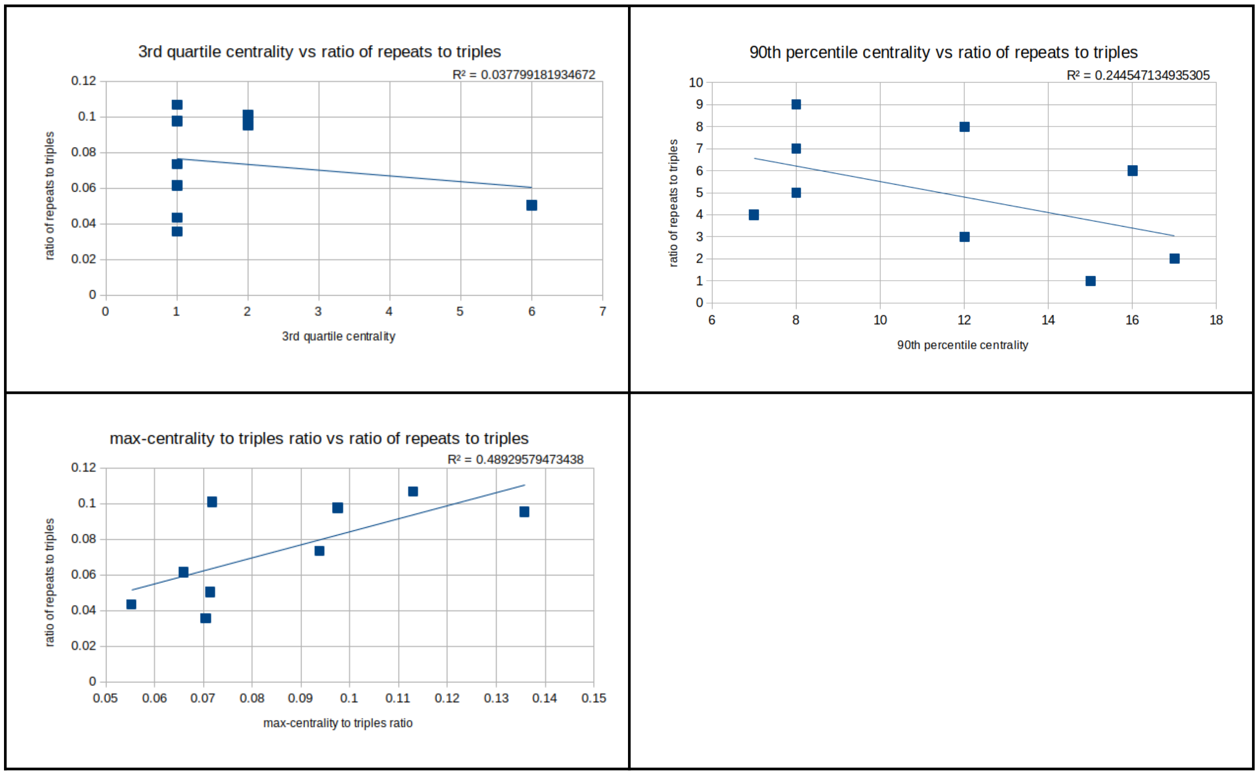


Figure 6.3. A plot of the 3rd quartile centrality, 90th percentile centrality and max-centrality to triples ratio of each model against the ratio of repeats to triples. A trendline is given, and it's R^2 value is shown in the upper-right hand corner.

In the case of the first plot, the 3rd quartile centrality is not only clearly not correlated to the ratio of repeats to triples, but the trend is clearly nonlinear. The trend line shown clearly does not capture the form of the data, for which most variation in the ratio of repeats to triples is not in the direction of the variation of the 3rd quartile centrality values. However, both the 90th percentile centrality and the max-centrality to triples ratio appear to be linearly correlated to the ratio of repeats to triples, with R^2 values of 0.2445 and 0.4893 respectively. This means that they both can explain a moderate or significant proportion (24.45% and 48.93%) respectively of the variation in the ratio of repeats to triples. However, it must be noted that these values are not additive. Moreover, the max-centrality to triples ratio was only used in one dataset, and that dataset was one of the only two to assign a non-zero parameter to the repeats to triples ratio feature. While the 90th percentile centrality was included in several models, in one case the parameter of both it and the ratio of repeats to triple were driven to zero, and in other it was driven to nearly zero (0.0006) whereas the ratio of repeats to triples was driven to zero.

However, the previous round did establish that the ratio of repeats to triples is of importance to models when only it and the ratio of sinks to triples are considered. As thus, it is likely that the inclusion of both sink / repeat ratio data and centrality distribution data made this feature superfluous and unnecessary to explain KGE performance given KG structure.

Overall, this data suggests two major findings. First, it is possible to use the structural characteristics of a KG to very accurately predict the ability of a given KGE configuration to model specific knowledge (reflected in r1) and general knowledge (reflected in AUC) with very high accuracy and in the absence of modelling for graph size explicitly. Second, the distribution of centrality, as well as understanding the prevalence of sinks vs repeats in the dataset, are both of particular value in this prediction even if the exact metrics of each of those features that are most useful does vary.

The context of this work must be noted, however. The reason, for example, that the ratio of sources to triples was not included was that it correlated very strongly with the ratio of repeats to triples. Had it not been so strongly correlated, it very well may have been an important predictive factor as well. Moreover, the use of only certain percentiles for measuring the distribution of centralities is an artefact of the data at hand and not a general rule: lower percentiles for centrality had little to no variation (often being 1 for all datasets) and thus would be of no use in a predictive task. In the context of other datasets where this rule would not hold, many other percentiles of the centrality distribution, as well as other centrality measures, may be of significant predictive power.

6.4.2 Identifying the best configuration based on structure

In terms of r1 scores, identifying the best configuration for each dataset could not be formulated as a classification problem, since in all cases except the BioPortal dataset, the BioPortal configuration outperformed the general configuration. As previously noted, since the two BioPortal r1 scores under both configurations are so similar, it is difficult to tell if the difference is significant or due to chance, and thus placing it into either class definitively is uncertain at best. In any case a simple classifier that always preferred the BioPortal configuration would already be achieving perfect or near-perfect results.

However, this fact alone is very interesting. While a determination of optimal hyperparameters could not be made, these results suggest a hierarchy of hyperparameter set optimality in the context of the highly left-skewed centrality distribution structure seen here. While the BioPortal configuration cannot be claimed to be optimal, in all cases it is approximately as optimal as, or often much more optimal than, the general configuration in terms of r1 scores. Taking this into account, in tandem with the very high accuracy with which r1 can be predicted given graph structure, suggest that there is a strong relationship between structure of a KG and KGE performance, and that similarly structured datasets will display perform similarly under the same hyperparameter configurations. This further suggests that finding an optimal or near-optimal hyper parameter set to match all of these similarly-structured datasets should be possible, even though such a set was not found here. Verifying that a similar trend holds for datasets of similar structure, but with vastly different structures than those seen here, is left as a future direction.

The optimal dataset for different AUC values did vary between the two configurations, and could be formulated as a classification problem. However, the inability to understand r1 results in this fashion, as well as the greater applicability of high-r1 classifiers to downstream tasks that require identifying a true statement from many plausible alternatives, suggested using a different approach that could accommodate the data from both approaches at once.

As a result, identifying the ideal dataset was formulated as a regression problem. In this formulation, the same structural characteristics discussed in Section 6.4.1 were used to regress to the difference in r1 and in AUC scores between the datasets. In this case, the difference was defined as the score under the BioPortal configuration minus that under the general configuration. In this way, the extent to which one dataset was better than the other was measured as a function of graph structure, allowing for a continuous understanding of structure and configuration suitability rather than a discrete one, and admitting the use of data that could not be effectively understood or modelled in a classification problem.

This problem was approached just as in Section 6.4.1: with a Lasso regressor, cross-validated with 5-fold cross-validation, run first on the sink / repeat ratios and the centrality distribution statistics individually. Pooling of the results was not done, since no significant predictors were found in the first round. The details by which this round of models was created are given in Appendix III.

The R^2 results of each of the models are included here in Table 6.11, and the parameters to each feature in the model are given in Tables 6.12 and 6.13.

Data used	**Predicting AUC**	**Predicting r1**
Sink-repeat ratios	0	0
Centrality distribution	0	0

Table 6.11. The R^2 scores of each model, when run on the full datasets under the given configuration.

Model	**Sinks:Triples ratio**	**Repeats:Triples ratio**
Predicting r1	0	0
Predicting AUC	0	0

Table 6.12. Coefficient values corresponding to each feature in the sinks-source-repeat ratio dataset, for each of the regression models created.

Model	Median centrality	3rd quartile centrality	90th percentile centrality	Max centrality to Triples Ratio
Predicting r1	0	0	0	0
Predicting AUC	0	0	0	0

Table 6.13. Coefficient values corresponding to each feature in the centrality distribution dataset, for each of the regression models created.

These results show that, in contrast to the former high predictive power of structure on model performance within the context of a given configuration, that prediction of the difference between the scores obtained between the two configurations above the baseline level is not feasible using Lasso linear regression on the structural metrics identified here.

Since this linear regression failed, a second attempt was made in which the predicted scores under the final models from section 6.4.1.2 (including all relevant structural characteristics) were used to estimate the difference in the score between the two hyperparameter sets. This was formulated as follows: letting f_{Bio} and f_{gen} respectively be the prediction models for the r1 or AUC scores under the BioPortal and the general configuration from Section 6.4.1.2, the difference between the models, f_{diff}, was modelled as:

$$f_{diff} = f_{Bio} - f_{gen}$$

This was done separately for the models to predict r1 and AUC scores. Plots of the predictions for the differences in r1 and AUC scores versus the true observed differences in those scores are shown below in Figure 6.4, along with a trend line and the associated R^2 value.

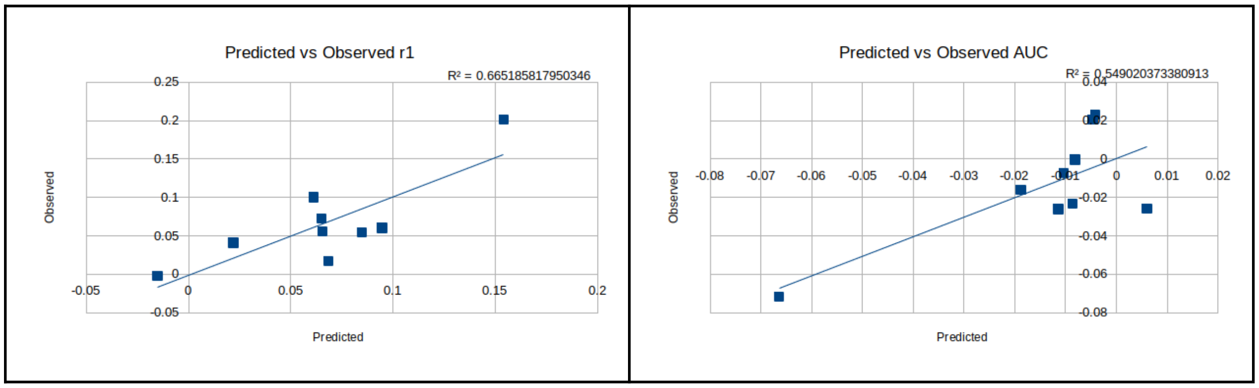


Figure 6.4. Plots of the predicted difference in r1 and AUC scores for both configurations versus the observed difference. A trendline is given, and it's R^2 value is shown in the upper-right hand corner.

Interestingly, this approach did yield results with explanatory power for the difference between predicted and observed scores. In the case of predicting the difference between r1 scores for the two models, the trend appears moderately linear although there is a notable cluster in the center that does not seem to fit the linear trend quite as well. Overall, the predicted values have an R^2 of 0.6652 with respect to the observed values, indicating that the difference can be predicted with moderate accuracy.

In the case of the AUC score difference, it appears that the trend line is being pulled towards a single distant point which is far separated from a cluster of points that do not have clear linearity. While the R^2 value for the entire dataset is 0.5490, which would seem to indicate a decently strong predictive power, removal of this one point reduces the R^2 to 0.0134, reminiscent of nearly no predictive power at all. While, therefore, the difference in r1 scores seem to be decently well modelled by the difference in the predicted r1 scores for both configurations, the ability to model the difference in AUC scores based on the predicted difference from the two configurations is dubious at best.

There are many possible explanations for the difficulty in predicting AUC accurately. At one level, there are many different factors that vary between the two hyperparameter configurations and the multiplicity of that variation could compound and obscure any trends that would have been seen had the two configurations differed in only one or two ways. In addition to that, it is possible that predicting the difference between the optimal score obtained under both datasets given graph structure is indeed possible, but that it requires different structural characteristics than those identified here. These two effects also likely explain why the R^2 score for predicting the difference in r1 scores between the two configurations is not higher, although the reason that r1 is predictable while AUC is not predictable is not clear.

Moreover, since interaction terms between structural features were not analyzed, it is possible that such a trend could only be explained by the interaction of various structural elements. However, as mentioned before, the scarcity of data-points means that adding in a large number of interaction features would almost certainly lead to data memorization and model overfitting. As thus, analysis of interaction terms is left as a future direction.

Overall, this suggests that while predicting model performance under a given configuration is very possible (as seen in Section 6.4.1) predicting the performance of different hyperparameter sets cannot be done with the same predictive power. Particularly, the issue of the hyperparameter configuration being a black box was not possible to overcome: since the two configurations differed in many ways, trying to estimate the difference in performance between different configurations without modelling all of these extra features was fundamentally limited. However, as previously noted, adding in all these features would likely produce more overfitting as the number of features would be much too high relative to the number of data points. Creating a model that combined KG structural data with hyperparameter choices to predict performance was thus left as a future direction, but one of great importance.

As a final note, the question of hyperparameter selection can be formulated as a classification or as a regression problem. While the former suggests a way to choose from a small number of disparate configurations, the latter if created would allow for choosing between a large number of untested configurations if both structural characteristics and hyperparameter values are included as features. As such, the latter allows for a search of optimal hyperparameters that have never been tested, even in the absence of a grid search. However, it is simultaneously less certain (possibly outputting a configuration that has never been used or tested before) and more complex (including a great many more features). The exploration of the pros and cons between the classification and regression approaches, as well as whether the regression approach is applicable across general datasets, is also left as a very relevant and important future direction.

7 Discussion

7.1 Overview of the Discussion

Section 7.2 presents a qualitative summary of the major findings and their significance. Section 7.3 continues to discuss the expected contribution of these findings to the field, and notes some of the most important future directions for research to build upon these findings. Section 7.4 discusses the limitations of these results and future directions that could address them. Section 7.5 concludes with a brief set of final observations from this work.

7.2 Overview of Major Findings

Overall, this work failed to determine a set of optimal cross-dataset hyperparameters for biological LOD datasets. However, it found a very clear and strong effect of KG structure on hyperparameter choice and model performance. The three major findings of this work which are described below.

First, despite the goal set out at the beginning of this work, a set of hyperparameters that could be applied across different biological LOD datasets was not found. However, it was found that in the context of similarly-structured KGs, hyperparameter configurations that led to high performance for one dataset led to higher performance for nearly all the others. Likewise, it was seen that hyperparameter configurations with low performance for one dataset tended to lead to worse performance on other, similarly structured datasets. This suggests that the optimal hyperparameter set should also be shared across similarly-structured datasets, although this hypothesis could not be verified in this work and is left as a future direction.

Second, it is demonstrated that, under two different hyperparameter configurations, the ability of two different KGE model configurations to model specific knowledge (as measured by the r1 score) and general knowledge (as measured by the AUC score) can be predicted with very high accuracy. In particular, at least 85% of the variation in both these scores can be explained using structural characteristics of the KG alone.

Third, it is demonstrated that although the r1 and AUC scores under a given hyperparameter configuration can be very easily determined from the structure, modelling the difference between predicted and observed r1 and AUC scores for the two configurations is notably less accurate. In particular, in the case of predicting the difference in AUC scores, it is unclear whether the relatively strong predictive power is due to a true trend or to an outlier point. Obtaining more data and re-analysing it in this context would help to elucidate this question, and examining whether other graph structural statistics and their interaction terms would help to create models with higher predictive power, is left as a future direction.

Very notably, the observed similarity of the DBSNP, DrugBank, OMIM, and PharmGKB datasets in the r1-based hyperparameter search suggests that the effects observed here are representative of a larger effect of KG structure on KGE embedding quality. For those datasets, during the hyperparameter search in almost every case they displayed almost exactly the same response to different combinations of hyperparameters. While the subsetting method used was found to be imperfect (not replicating the original KG structure perfectly), these results for the sub-graphs still reveal a very strong influence of graph structure on model performance and response to different hyperparameter choices.

The most important structural characteristics to predicting model performance and predicting which configuration would be ideal are twofold: the ratios of the number of sinks and repeated entities to the number of triples in the graph, and the descriptive statistics of the distribution of node centralities in the KG. In this case all of the datasets had an extreme left skew in their centrality distributions, and as a result the response of KGs with non-left-skewed centrality distributions to these or other structural features was not determined. However, it is hypothesized that various percentile markers of the centrality distribution would remain highly relevant in those cases.

7.3 Expected Contribution

It is expected that this work contributes to relational learning and bioinformatics in four key ways.

First, it demonstrates that KGE performance is very responsive to structure--particularly the ratio of sinks and repeated entities to the size of the KG and the distribution of the centrality of nodes in the KG.

It also demonstrates that the performance of a KGE in terms of specific and general knowledge modelling ability with a given set of hyperparameters can be predicted with very high accuracy considering only KG structure. This allows a very rapid prediction of what KGs may fit a given set of hyperparameters using only a linear regression model, rather than a time-intensive grid search.

Moreover, it highlights the ability to predict, with moderate accuracy, the difference in performance of a given KG dataset between different KGE model configurations / hyperparameters using only KG structural characteristics. This is also found to be possible using a simple linear regression model, and if the accuracy were improved would allow rapid detection of the best hyperparameter set to use without the need to run a grid search for new KGs.

The work also shows that many of the common biological LOD datasets have a very heavy left skew in terms of the distribution of their centralities, and that that has a very significant impact on model performance. Research into the effects of this on biological data will help to increase the accuracy of bioinformatics tools based on KGEs, as well as to facilitate the creation of new KGE models by determining if they would operate effectively on predetermined hyperparameter sets that had been curated for other KGs.

The work also leaves two important future directions, which if followed up could provide critical novel insights to the field. First, this research suggests that it would be possible to create a linear regression model that predicts model performance not only based off of the KG structure, but also upon discrete and continuous hyperparameters. If done, this would allow for the rapid detection of optimal hyperparameter configurations--even those never examined before--without the need to do a grid search. Moreover, since the linear regression model would be easily differentiable, finding all of its maxima for a given structure could be formatted as an equality- or inequality- constrained optimisation problem to set bounds on the graph structure in the problem space, and thus allow gradient-based optimisation methods to very rapidly find the ideal hyperparameter configuration for a KG with a given structure.

Secondly, exploring the pros and cons of predicting optimal hyperparameter sets using the above approach (based on regression) or a classification approach (classifying a dataset to known hyperparameter sets based on KG structure) is similarly merited as an enabler for faster, more efficient, and possibly more optimal hyperparameter selection.

7.4 Limitations of this Work

Since the method for selecting hyperparameters was found to be sub-optimal for the intended datasets (as outlined in Chapter 6), the hyperparameter sets produced are known to be imperfect. This is likely a result of having subsetted the graphs, producing subgraphs that could be easily grid-searched, but whose structure was notably different in some respects from the structure of the original KG. Unfortunately, the extent by which they vary from the optimum configurations was impossible to estimate, as the optimal configurations are not known. As a result, the results of this study are best seen as presenting data in the context of arbitrary hyperparameter configurations, rather than optimal or near optimal ones. However, the structural analysis of KGE scores under these configurations remains valid, because that analysis made no assumption of optimality of the configurations which it was predicting.

It should be noted that in this study, the contents of triples are entirely ignored. The contents of string literals were not scanned to attempt to learn what they said, everything was simply represented as an entity. Further analysis into the effect of reading the contents of literals, rather than simply treating them as black boxes, on hyperparameter choice is similarly merited. Interestingly, in that case the internal structure of nodes rather than just their relationship patterns could be expected to influence hyperparameter choice.

In addition, in this work only biomedical datasets were considered, and all of these datasets were observed to have an extremely strong skewness of centrality values. Given the findings of this research, examining how other datasets with very different centrality distributions than those seen here interact with optimal hyperparameters to KGE models is expected to be of benefit to the relational learning and KGE fields as a whole.

Finally, in this work even the final hyperparameter configurations identified yielded low r1 scores which never exceeded 0.4, and AUC scores that never exceeded 0.85. Scores observed in the hyperparameter search were similarly low, often significantly lower. However, the reason for these low scores was not directly identified. As outlined in Wang et al., KGs learn by understanding the relationships between entities [45]. This suggests, then, that entities with very few observed relationships would be relatively hard to learn to embed properly. Therefore, it stands to reason that an effect of structure would be observed here as well, very likely in terms of the number of sinks and sources in the graph relative to the number of triples in total. This effect may be partly agnostic to the hyperparameters involved, although this determination could not be made with the data available. Further research in this direction would be merited.

7.5 Final Observations

It is hoped that this work contributes to the understanding of hyperparameter choices not only in the realm of KGEs, but in the context of machine learning models generally. The finding that the structural elements of KGs are very highly predictive of model performance under different hyperparameter configurations suggests that data structure and model choice may be best understood in the context of each other.

Creating machine learning models by which structural elements could lead to optimal hyperparameter prediction and predictions of model performance, as mentioned in Section 7.2, is well merited. Moreover, it would be equally merited to extend this work to other machine learning domains, to understand if all hyperparameters and model performance--or only those for KGEs--can be modelled as a function of dataset structure.

The author of this work hypotheses that such dataset-structure based approaches would yield similarly fruitful results, advancing understanding of machine learning models and facilitating optimal hyperparameter selection in machine learning domains outside of KGEs alone.

References

[1] Belleau, François & Nolin, Marc-Alexandre & Tourigny, Nicole & Rigault, Philippe & Morissette, Jean. (2008). Bio2RDF: Towards A Mashup To Build Bioinformatics Knowledge System. Journal of biomedical informatics. 41. 706-16. 10.1016/j.jbi.2008.03.004.

[2] Zhang J, Baran J, Cros A, Guberman JM, Haider S, Hsu J, Liang Y, Rivkin E, Wang J, Whitty B, Wong-Erasmus M, Yao L, Kasprzyk A. International Cancer Genome Consortium Data Portal--a one-stop shop for cancer genomics data. Database (Oxford). 2011 Sep 19;2011:bar026. doi: 10.1093/database/bar026. PMID: 21930502; PMCID: PMC3263593.

[3] 1000 Genomes Project Consortium, Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM, Handsaker RE, Kang HM, Marth GT, McVean GA. An integrated map of genetic variation from 1,092 human genomes. Nature. 2012 Nov 1;491(7422):56-65. doi: 10.1038/nature11632. PMID: 23128226; PMCID: PMC3498066.

[4] Newton Y, Novak AM, Swatloski T, McColl DC, Chopra S, Graim K, Weinstein AS, Baertsch R, Salama SR, Ellrott K, Chopra M, Goldstein TC, Haussler D, Morozova O, Stuart JM. TumorMap: Exploring the Molecular Similarities of Cancer Samples in an Interactive Portal. Cancer Res. 2017 Nov 1;77(21):e111-e114. doi: 10.1158/0008-5472.CAN-17-0580. PMID: 29092953; PMCID: PMC5751940.

[5] McCusker, Jamie & Dumontier, Michel & Yan, Rui & He, Sylvia & Dordick, Jonathan & Mcguinness, Deborah. (2017). Finding melanoma drugs through a probabilistic knowledge graph. PeerJ Computer Science. 3. e106. 10.7717/peerj-cs.106.

[6] *S. M. S. Hasan, D. Rivera, X. -C. Wu, E. B. Durbin, J. B. Christian and G. Tourassi, "Knowledge Graph-Enabled Cancer Data Analytics," in IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 7, pp. 1952-1967, July 2020, doi: 10.1109/JBHI.2020.2990797.*

[7] Jha A, Khan Y, Mehdi M, Karim MR, Mehmood Q, Zappa A, Rebholz-Schuhmann D, Sahay R. Towards precision medicine: discovering novel gynecological cancer biomarkers and pathways using linked data. J Biomed Semantics. 2017 Sep 19;8(1):40. doi: 10.1186/s13326-017-0146-9. PMID: 28927463; PMCID: PMC5606033.

[8] Lerer, Adam & Wu, Ledell & Shen, Jiajun & Lacroix, Timothee & Wehrstedt, Luca & Bose, Abhijit & Peysakhovich, Alex. (2019). PyTorch-BigGraph: A Large-scale Graph Embedding System.

[9] Bizer, Christian & Heath, Tom & Berners-Lee, Tim. (2009). Linked Data: The Story so Far. International Journal on Semantic Web and Information Systems. 5. 1-22. 10.4018/jswis.2009081901.

[10] RDF/XML Syntax Specification (Revised). https://www.w3.org/TR/REC-rdf-syntax/. Accessed July 2021.

[11] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, "A Review of Relational Machine Learning for Knowledge Graphs," in *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11-33, Jan. 2016, doi: 10.1109/JPROC.2015.2483592.

[12] Bio2RDF Release 3. https://download.bio2rdf.org/files/release/3/release.html. Accessed July 2021.

[13] Saleem, Muhammad & Padmanabhuni, Shanmukha & Ngonga Ngomo, Axel-Cyrille & Almeida, Jonas & Decker, Stefan & Deus, Helena. (2013). Linked Cancer Genome Atlas Database. ACM International Conference Proceeding Series. 10.1145/2506182.2506200.

[14] The Cancer Genome Atlas https://www.genome.gov/Funded-Programs-Projects/Cancer-Genome-Atlas. Accessed July 2021.

[15] Saleem, Muhammad & Kamdar, Maulik & Iqbal, Aftab & Padmanabhuni, Shanmukha & Deus, Helena & Ngonga Ngomo, Axel-Cyrille. (2014). Big linked cancer data: Integrating linked TCGA and PubMed. Journal of Web Semantics. 10.1016/j.websem.2014.07.004.

[16] National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/. Accessed July 2021.

[17] Gomes AP, Price NL, Ling AJ, Moslehi JJ, Montgomery MK, Rajman L, White JP, Teodoro JS, Wrann CD, Hubbard BP, Mercken EM, Palmeira CM, de Cabo R, Rolo AP, Turner N, Bell EL, Sinclair DA. Declining NAD(+) induces a pseudohypoxic state disrupting nuclear-mitochondrial communication during aging. Cell. 2013 Dec 19;155(7):1624-38. doi: 10.1016/j.cell.2013.11.037. PMID: 24360282; PMCID: PMC4076149.

[18] Carballo GB, Honorato JR, de Lopes GPF, Spohr TCLSE. A highlight on Sonic hedgehog pathway. Cell Commun Signal. 2018 Mar 20;16(1):11. doi: 10.1186/s12964-018-0220-7. PMID: 29558958; PMCID: PMC5861627.

[19] McCusker JP, Dumontier M, Yan R, He S, Dordick JS, McGuinness DL. 2017. Finding melanoma drugs through a probabilistic knowledge graph. PeerJ Computer Science 3:e106 https://doi.org/10.7717/peerj-cs.106

[20] Lautrup S, Sinclair DA, Mattson MP, Fang EF. NAD$^+$ in Brain Aging and Neurodegenerative Disorders. Cell Metab. 2019 Oct 1;30(4):630-655. doi: 10.1016/j.cmet.2019.09.001. PMID: 31577933; PMCID: PMC6787556.

[21] Dokyoon Kim, Je-Gun Joung, Kyung-Ah Sohn, Hyunjung Shin, Yu Rang Park, Marylyn D Ritchie, Ju Han Kim, Knowledge boosting: a graph-based integration approach with multi-omics data and genomic knowledge for cancer clinical outcome prediction, *Journal of the American Medical Informatics Association*, Volume 22, Issue 1, January 2015, Pages 109–120, https://doi.org/10.1136/amiajnl-2013-002481

[22] Deus HF, Veiga DF, Freire PR, Weinstein JN, Mills GB, Almeida JS. Exposing the cancer genome atlas as a SPARQL endpoint. J Biomed Inform. 2010 Dec;43(6):998-1008. doi: 10.1016/j.jbi.2010.09.004. PMID: 20851208; PMCID: PMC3071752.

[23] Zhao Y, Yu H, Fu S, Shen F, Davila JI, Liu H, Wang C. Data-driven Sublanguage Analysis for Cancer Genomics Knowledge Modeling: Applications in Mining Oncological Genetics Information from Patients' Genetic Reports. AMIA Jt Summits Transl Sci Proc. 2020 May 30;2020:720-729. PMID: 32477695; PMCID: PMC7233104.

[24] Hasan, S M Shamimul, Rivera, Donna R., Wu, Xiao-Cheng, Christian, Blair, and Tourassi, Georgia. Wed . "A Knowledge Graph Approach for the Secondary Use of Cancer Registry Data". United States. https://www.osti.gov/servlets/purl/1558464.

[25] Kanehisa M, Goto S. KEGG: kyoto encyclopedia of genes and genomes. Nucleic Acids Res. 2000 Jan 1;28(1):27-30. doi: 10.1093/nar/28.1.27. PMID: 10592173; PMCID: PMC102409.

[26] Dingerdissen HM, Bastian F, Vijay-Shanker K, Robinson-Rechavi M, Bell A, Gogate N, Gupta S, Holmes E, Kahsay R, Keeney J, Kincaid H, King CH, Liu D, Crichton DJ, Mazumder R. OncoMX: A Knowledgebase for Exploring Cancer Biomarkers in the Context of Related Cancer and Healthy Data. JCO Clin Cancer Inform. 2020 Mar;4:210-220. doi: 10.1200/CCI.19.00117. PMID: 32142370; PMCID: PMC7101249.

[27] Sameh K Mohamed, Aayah Nounu, Vít Nováček, Biological applications of knowledge graph embedding models, *Briefings in Bioinformatics*, Volume 22, Issue 2, March 2021, Pages 1679–1693, https://doi.org/10.1093/bib/bbaa012

[28] Mehdi Ali, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, Hajira Jabeen, BioKEEN: a library for learning and evaluating biological knowledge graph embeddings, *Bioinformatics*, Volume 35, Issue 18, 15 September 2019, Pages 3538–3540, https://doi.org/10.1093/bioinformatics/btz117

[29] PyTorch-BigGraph Documentation. https://torchbiggraph.readthedocs.io/en/latest/index.html. Accessed July 2021.

[30] Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, Tudorache T, Musen MA. BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Res. 2011 Jul;39(Web Server issue):W541-5. Epub 2011 Jun 14.

[31] S. T. Sherry, M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, K. Sirotkin, dbSNP: the NCBI database of genetic variation, *Nucleic Acids Research*, Volume 29, Issue 1, 1 January 2001, Pages 308–311, https://doi.org/10.1093/nar/29.1.308

[32] Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, Assempour N, Iynkkaran I, Liu Y, Maciejewski A, Gale N, Wilson A, Chin L, Cummings R, Le D, Pon A, Knox C, Wilson M. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res. 2017 Nov 8. doi: 10.1093/nar/gkx1037.
PubMed: 29126136

[33] Online Mendelian Inheritance in Man, OMIM®. McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD), 10 July 2021. World Wide Web URL: https://omim.org/

[34] M. Whirl-Carrillo, E.M. McDonagh, J. M. Hebert, L. Gong, K. Sangkuhl, C.F. Thorn, R.B. Altman and T.E. Klein. "Pharmacogenomics Knowledge for Personalized Medicine" *Clinical Pharmacology & Therapeutics* (2012) 92(4): 414-417.

[35] Huntley RP, Sawford T, Mutowo-Meullenet P, Shypitsyna A, Bonilla C, Martin MJ, O'Donovan C. The GOA database: Gene Ontology annotation updates for 2015.
Nucleic Acids Res. 2015 Jan; 43:D1057-63

[36] Bryony Braschi, Paul Denny, Kristian Gray, Tamsin Jones, Ruth Seal, Susan Tweedie, Bethan Yates, Elspeth Bruford, Genenames.org: the HGNC and VGNC resources in 2019, *Nucleic Acids Research*, Volume 47, Issue D1, 08 January 2019, Pages D786–D792, https://doi.org/10.1093/nar/gky930

[37] Bio2RDF version of The Life Science Resource Registry.
https://download.bio2rdf.org/files/release/3/lsr/lsr.html. Accessed July 2021.

[38] Sioutos N, de Coronado S, Haber MW, Hartel FW, Shaiu WL, Wright LW. NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information. J Biomed Inform. 2007 Feb;40(1):30-43. doi: 10.1016/j.jbi.2006.02.013. Epub 2006 Mar 15. PMID: 16697710.

[39] Theodore Dalamagas, Nikos Bikakis, George Papastefanatos, Yannis Stavrakas, and Artemis G. Hatzigeorgiou. 2012. Publishing life science data as linked open data: the case study of miRBase. In *Proceedings of the First International Workshop on Open Data* (*WOD '12*).

Association for Computing Machinery, New York, NY, USA, 70–77. DOI:https://doi.org/10.1145/2422604.2422615

[40] *Navarro-Gallinad, A., The Semantic Combining for Exploration of Environmental and Disease Data Dashboard for Clinician Researchers, In Ivanova, V., Lambrix, P., Pesquita, C., Wiensn, V. (Editors). Proceedings of the Fifth International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 19th International Semantic Web Conference (ISWC 2020), 73 - 85*

[41] Zehra, D., Jha, A., Khan, Y., Hasnain, A., d'Aquin, M., & Sahay, R. (2019). A Cancer Genomics Data Space within the Linked Open Data (LOD) Cloud. *SeWeBMeDa@ISWC*.

[42] Detwiler LT, Suciu D, Brinkley JF. Regular paths in SparQL: querying the NCI Thesaurus. AMIA Annu Symp Proc. 2008 Nov 6;2008:161-5. PMID: 18999137; PMCID: PMC2656016.

[43] Celebi R, Uyar H, Yasar E, Gumus O, Dikenelli O, Dumontier M. Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings. BMC Bioinformatics. 2019 Dec 18;20(1):726. doi: 10.1186/s12859-019-3284-5. PMID: 31852427; PMCID: PMC6921491.

[44] PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., & Lehmann, J. (2020). Journal of Machine Learning Research, 22(82), 1–6.

[45] Q. Wang, Z. Mao, B. Wang and L. Guo, "Knowledge Graph Embedding: A Survey of Approaches and Applications," in IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724-2743, 1 Dec. 2017, doi: 10.1109/TKDE.2017.2754499.

[46] Ben Ellefi, Mohamed & Bellahsene, Zohra & Breslin, John & Demidova, Elena & Dietze, Stefan & Szymanski, Julian & Todorov, Konstantin. (2017). RDF Dataset Profiling - a Survey of Features, Methods, Vocabularies and Applications. Semantic Web. 9. 10.3233/SW-180294.

[47] Sadeghi, Afshin & Collarana, Diego & Graux, Damien & Lehmann, Jens. (2021). Embedding Knowledge Graphs Attentive to Positional and Centrality Qualities.

[48] SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/. Accessed July 2021.

[49] PyKEEN Repository. https://github.com/pykeen/pykeen. Accessed August 2021.

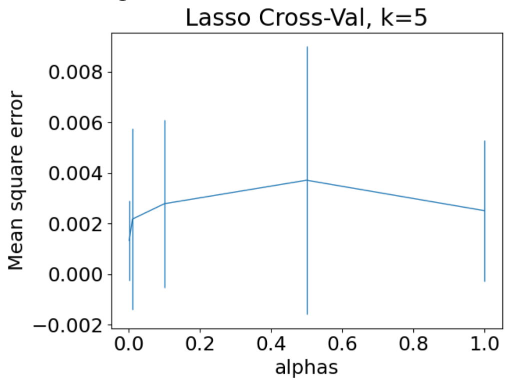
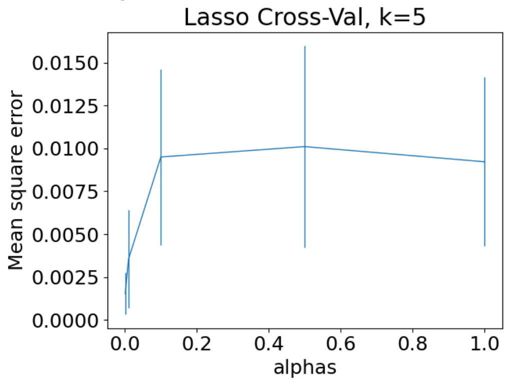
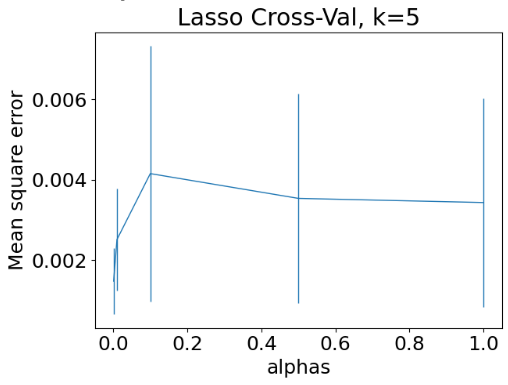
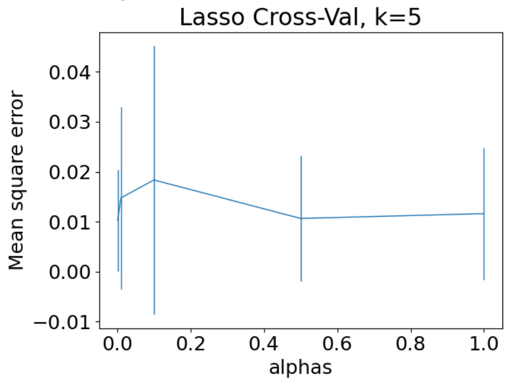
[50] Web Ontology Language (OWL). https://www.w3.org/OWL/. Accessed August 2021.

[51] PyTorch-BigGraph GitHub. https://github.com/facebookresearch/PyTorch-BigGraph. Accessed August 2021.

Appendix

I. Cross validation of structure-performance regression models, round 1

Figure I.1 below gives all cross-validation plots created during 5-fold cross-validation. Cross-validation was performance against alpha, the hyperparameter to the regression model that serves as a coefficient to the L_1 penalty term, and evaluated in terms of the mean-square error (MSE) score of the regression model. The effectiveness of all these classifiers was measured by their AUC scores. All values of alpha were cross-selected over the range of 1e-3, 1e-2, 1e-1, 0.5, and 1.

Data used	**BioPortal Configuration**	**General Configuration**
Sink-repeat ratios	Predicting AUC:  Predicting r1: 	Predicting AUC:  Predicting r1: 
Centrality distribution	Predicting AUC:	Predicting AUC:

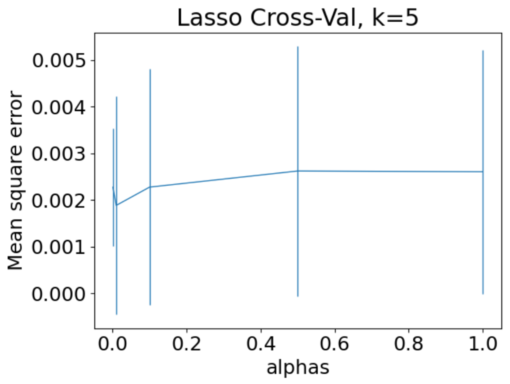
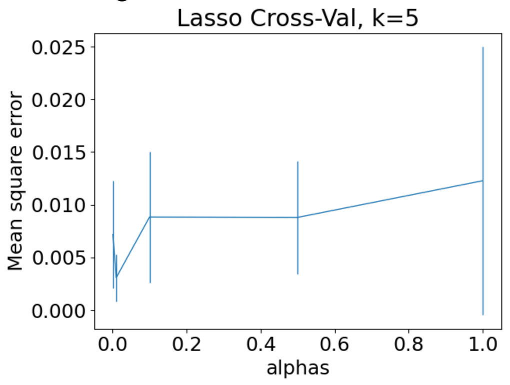
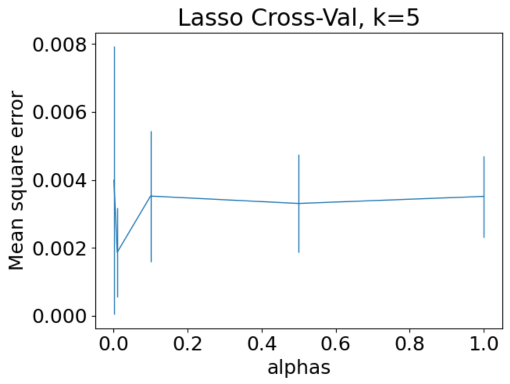
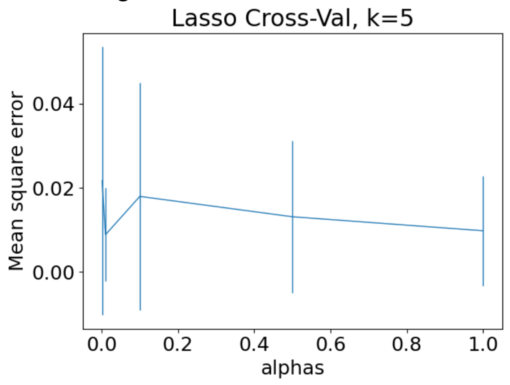
	 Predicting r1: 	 Predicting r1: 

Figure I.1. Cross-validation plots for all models created.

Based on the above plots, values of alpha were selected that led to minimal MSE (means squared error) values were selected, subject to a set of criteria to avoid over- and under-fitting. The selection criteria were as follows: all else being equal, larger values of alpha were preferred since smaller values of alpha can drive the penalty term to zero and lead to overfitting. Similarly, the extreme high-end values of alpha (notably the value of 1) were avoided in the absence of strong evidence in favour of them, since such values often lead to underfitting. Smaller values of alpha were selected when their standard error interval was small, and when their point-estimate MSE values were notably below those corresponding to higher alpha values. The results obtained from this are shown below in Table I.1.

Data used	**BioPortal Configuration**	**General Configuration**

Sink-repeat ratios	`Predicting AUC:`
	`1e-3`
	`Predicting r1:`
	`1e-3`
		`Predicting AUC:`
		`1e-3`
		`Predicting r1:`
		`1e-3`

Centrality distribution	`Predicting AUC: 1e-2 Predicting r1: 1e-2`	`Predicting AUC: 1e-2 Predicting r1: 1e-2`

Table I.1. The selected values of alpha for each regression model.

II. Cross validation of structure-performance regression models, round 2

Figure II.1 below gives all cross-validation plots created during 5-fold cross-validation. Cross-validation was performance against alpha, the hyperparameter to the regression model that serves as a coefficient to the L_1 penalty term, and evaluated in terms of the mean-square error (MSE) score of the regression model. The effectiveness of all these classifiers was measured by their AUC scores. All values of alpha were cross-selected over the range of 1e-3, 1e-2, 1e-1, 0.5, and 1.

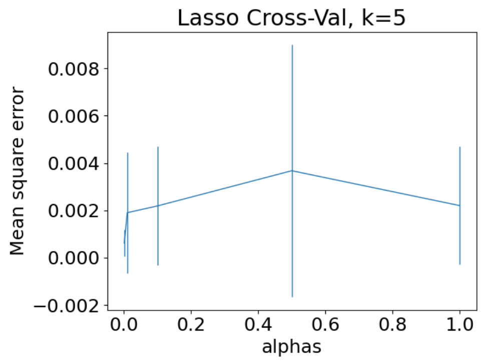
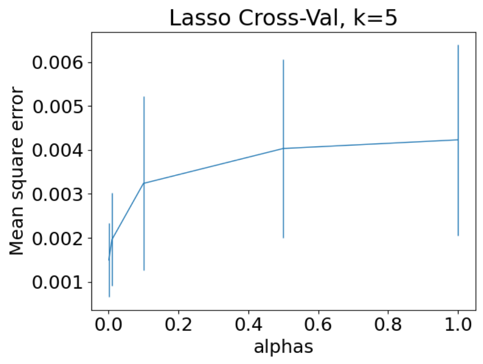
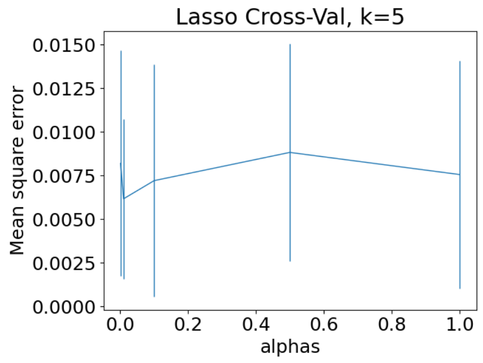
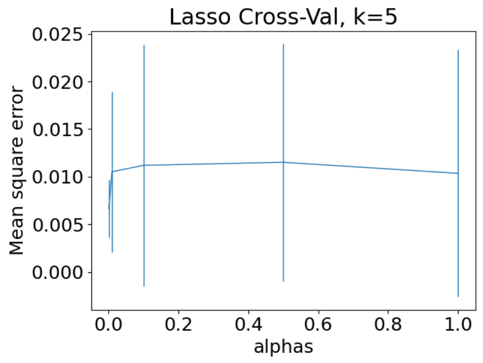
Regression Task	**BioPortal Configuration**	**General Configuration**
Predicting AUC		
Predicting r1		

Figure II.1. Cross-validation plots for all models created.

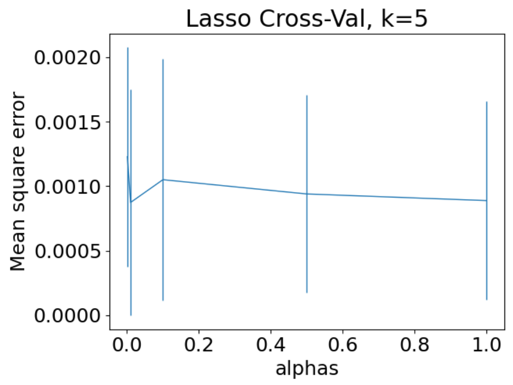
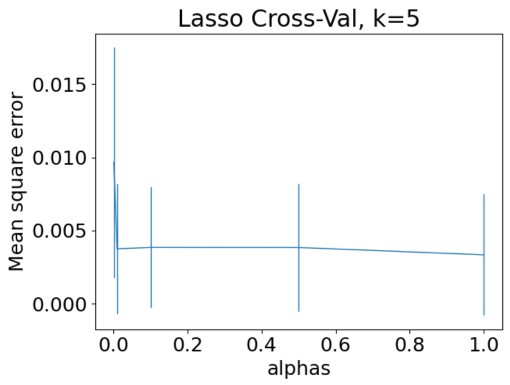
Based on the above plots, the values of alpha were selected. The selection criteria were as follows: all else being equal, larger values of alpha were preferred since smaller values of alpha can drive the penalty term to zero and lead to overfitting. Similarly, the extreme high-end values of alpha (notably the value of 1) were avoided in the absence of strong evidence in favour of them, since such values often lead to underfitting. Smaller values of alpha were selected when their standard error interval was small, and when their point-estimate MSE values were notably below those corresponding to higher alpha values. The results obtained from this are shown below in Table II.1.

Regression Task	**BioPortal Configuration**	**General Configuration**
Predicting AUC	`1e-3`	`1e-3`
Predicting r1	`1e-1`	`1e-3`

Table II.1. The selected values of alpha for each regression model.

III. Cross validation of model score difference regression models, round 1

Figure III.1 below gives all cross-validation plots created during 5-fold cross-validation. Cross-validation was performance against alpha, the hyperparameter to the regression model that serves as a coefficient to the L_1 penalty term, and evaluated in terms of the mean-square error (MSE) score of the regression model. The effectiveness of all these classifiers was measured by their AUC scores. All values of alpha were cross-selected over the range of 1e-3, 1e-2, 1e-1, 0.5, and 1.

Data used	**Predicting AUC**	**Predicting r1**
Sink-repeat ratios		

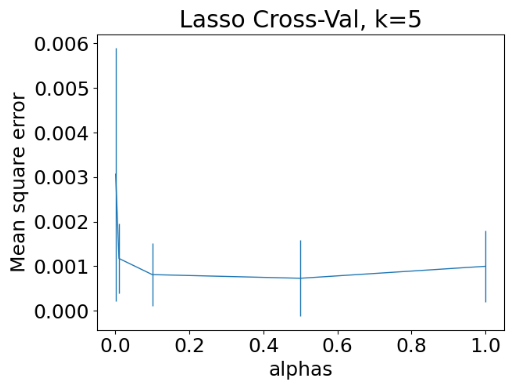
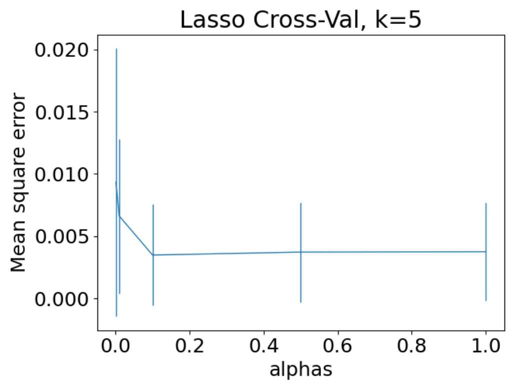
Centrality distribution		

Figure III.1. Cross-validation plots for all models created.

Based on the above plots, the values of alpha were selected. The selection criteria were as follows: all else being equal, larger values of alpha were preferred since smaller values of alpha can drive the penalty term to zero and lead to overfitting. Similarly, the extreme high-end values of alpha (notably the value of 1) were avoided in the absence of strong evidence in favour of them, since such values often lead to underfitting. Smaller values of alpha were selected when their standard error interval was small, and when their point-estimate MSE values were notably below those corresponding to higher alpha values. The results obtained from this are shown below in Table III.1.

Data used	**Predicting AUC**	**Predicting r1**
Sink-repeat ratios	`0.5`	`1e-1`
Centrality distribution	`0.5`	`1e-1`

Table III.1. The selected values of alpha for each regression model.