Comparison of Symbolic & Sub-Symbolic Approaches for Knowledge Graph Completion

Master Thesis

presented by
Lars Joormann
Matriculation Number 1721931

submitted to the
Data and Web Science Group
Prof. Dr. Stuckenschmidt
University of Mannheim

August 2022

Abstract

Link prediction is the task of predicting the existence of a relation between two entities in a knowledge graph. To solve this task various research has been done to improve and create new models. Most of these models are from the sub-symbolic approach while not as much research has been done to create symbolic models. Recent published models like AnyBURL have shown that the symbolic models can compete with the more popular sub-symbolic models.

In this master thesis the two approaches are compared in regard to their ability to correctly predict facts and focus on the characteristics of these facts.

To be more precise, the prediction results of AnyBURL and three different sub-symbolic models were analysed to determine whether one of the approaches performs better if we only regard a subset of the test data which shares a common characteristic i.e. whether one of the approaches has an advantage if a triple has a certain characteristic.

During this comparison it was found that the relation class and relation frequency of a triple has a slight influence on which approach performs better while the entity frequency did not show to have any influence. Moreover, it was found that the sub-symbolic approach performs better in predicting a triple if the trainings data already contained a similar triple while the symbolic approach achieved better results for triples without a similar triple. Lastly, clusters of similar entities were found where one of the approaches performed significantly better than the other.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Research Questions . . . 2
- 1.3 Thesis Structure . . . 2

2 Knowledge Graphs **3**

3 Knowledge Graph Completion **6**
- 3.1 Symbolic Approaches . . . 7
  - 3.1.1 AnyBURL . . . 7
- 3.2 Sub-Symbolic Approaches . . . 11
  - 3.2.1 RESCAL . . . 12
  - 3.2.2 DistMult . . . 14
  - 3.2.3 ComplEx . . . 15
  - 3.2.4 ConvE . . . 15
- 3.3 Comparison of the Approaches . . . 17
- 3.4 Model Evaluation . . . 17
- 3.5 Datasets . . . 19

4 Experimental Setting **21**
- 4.1 Datasets . . . 21
- 4.2 Models . . . 22
- 4.3 Data Aggregation . . . 23

5 Comparison **25**
- 5.1 Overall Performance . . . 25
- 5.2 Better Predicted by & Thresholds . . . 29
- 5.3 Comparison based on Characteristics . . . 31
  - 5.3.1 Relation Class . . . 31
  - 5.3.2 Relation Frequency in Trainings Data . . . 37

5.3.3 Entity Frequency in Trainings Data 41
5.3.4 Existence of Similar Triples in the Trainings Data 45
5.3.5 Existence of Similar Entities in the Dataset 48

6 Test Sets for Vulnerabilities 55

7 Research Limitations and Further Research 61

8 Conclusion 63

A Further Comparison Figures & Tables 69
A.1 Metrics of all trained Models 69
A.2 Comparison of Rankings for all Models and Datasets 71
A.2.1 CoDEx-M 71
A.2.2 FB15k-237 73
A.2.3 YAGO3-10 75
A.3 Comparison in regard to the Relation Classes 76
A.4 Relation Frequency in Trainings Data 78
A.5 Comparison in regard to the Relation Frequency 79
A.6 Entity Frequency in Trainings Data 80
A.7 Comparison in regard to the Entity Frequency 81
A.7.1 AnyBURL and ConvE on CoDEx-M 81
A.7.2 AnyBURL and RESCAL on CoDEx-M 82
A.7.3 AnyBURL and ComplEx on FB15k-237 84
A.7.4 AnyBURL and ConvE on FB15k-237 85
A.7.5 AnyBURL and RESCAL on FB15k-237 87
A.7.6 AnyBURL and ComplEx on YAGO3-10 88
A.8 Comparison in regard to the Existence of Similar Triples in the Trainings Data 90
A.8.1 Binary 90
A.8.2 Stepwise 92
A.9 Comparison in regard to the existence of Similar Entities 95
A.9.1 AnyBURL and ComplEx on CoDEx-M 95
A.9.2 AnyBURL and ConvE on CoDEx-M 98
A.9.3 AnyBURL and RESCAL on CoDEx-M 99
A.9.4 AnyBURL and ComplEx on FB15k-237 100
A.9.5 AnyBURL and ConvE on FB15k-237 101
A.9.6 AnyBURL and RESCAL on FB15k-237 102
A.9.7 AnyBURL and ComplEx on YAGO3-10 103

List of Figures

2.1 Example of a Knowledge Graph 3
2.2 Example of a Google Side Window 4
2.3 Applications of Knowledge Graphs 5

3.1 Example of a generalization lattice of an acyclic path (Ambiguous Prediction $\dagger$, Shorter Bottom Rule $*$, Useless Atom $\diamond$) 9
3.2 Example of an Embedding for a Knowledge Graph 11
3.3 RESCAL vs. DistMult 14
3.4 Example of a Convolution Filter 16
3.5 Steps of the ConvE Model 16

5.1 Comparison of Ranks predicted by AnyBURL and ComplEx for CoDEx-M 26
5.2 Comparison of Ranks predicted by AnyBURL and ComplEx by prediction direction on CoDEx-M 27
5.3 Comparison of difference-ψ between AnyBURL and ComplEx on CoDEx-M 28
5.4 Comparison of difference-ψ between AnyBURL and ComplEx on WN18RR 29
5.5 Comparison of different thresholds for the $better\_predicted\_by$ variable 30
5.6 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the relation classes 32
5.7 Comparison of AnyBURL and ConvE (left) / RESCAL (right) on CoDEx-M in regard to the relation classes 34
5.8 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the relation classes 36
5.9 Relation frequency in trainings data for CoDEx-M 37

5.10 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the relation frequency 38
5.11 Comparison of AnyBURL and ConvE (left) RESCAL (right) on CoDEx-M in regard to the relation frequency 39
5.12 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the relation frequency 40
5.13 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the relation frequency 41
5.14 Entity frequency in trainings data for CoDEx-M 42
5.15 Comparison of AnyBURL and ComplEx on CodEx-M in regard to the question entity frequency 43
5.16 Comparison of AnyBURL and ComplEx on CodEx-M in regard to the answer entity frequency 44
5.17 Comparison of AnyBURL and ComplEx on CodEx-M in regard to the combined entity and relation frequency 45
5.18 Comparison of AnyBURL and ComplEx on CodEx-M in regard to the existence of similar triples in the trainings data 46
5.19 Comparison of AnyBURL and ConvE (left) / RESCAL (right) on CodEx-M in regard to the existence of similar triples in the trainings data 46
5.20 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar triples in the trainings data 47
5.21 Comparison of AnyBURL and ComplEx on CodEx-M in regard to the amount of similar triples in the trainings data 48
5.22 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on the cosine similarity 49
5.23 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on the cosine similarity 50
5.24 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 51
5.25 Comparison of AnyBURL and ConvE (left)/ RESCAL (right) on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 52
5.26 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 53
5.27 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 54

A.1 Comparison of Ranks predicted by AnyBURL and ConvE for CoDEx-M . . . 71
A.2 Comparison of Ranks predicted by AnyBURL and RESCAL for CoDEx-M . . . 72
A.3 Comparison of Ranks predicted by AnyBURL and ComplEx for FB15k-237 . . . 73
A.4 Comparison of Ranks predicted by AnyBURL and ConvE for FB15k-237 . . . 74
A.5 Comparison of Ranks predicted by AnyBURL and RESCAL for FB15k-237 . . . 74
A.6 Comparison of Ranks predicted by AnyBURL and ComplEx for YAGO3-10 . . . 75
A.7 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the relation classes . . . 76
A.8 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation classes . . . 76
A.9 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the relation classes . . . 77
A.10 Relation frequency in trainings data for FB15k-237 . . . 78
A.11 Relation frequency in trainings data for YAGO3-10 . . . 78
A.12 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation frequency . . . 79
A.13 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation frequency . . . 79
A.14 Entity frequency in trainings data for FB15k-237 . . . 80
A.15 Entity frequency in trainings data for YAGO3-10 . . . 80
A.16 Comparison of AnyBURL and ConvE on CodEx-M in regard to the question entity frequency . . . 81
A.17 Comparison of AnyBURL and ConvE on CodEx-M in regard to the answer entity frequency . . . 81
A.18 Comparison of AnyBURL and ConvE on CodEx-M in regard to the combined entity and relation frequency . . . 82
A.19 Comparison of AnyBURL and RESCAL on CodEx-M in regard to the question entity frequency . . . 82
A.20 Comparison of AnyBURL and RESCAL on CodEx-M in regard to the answer entity frequency . . . 83
A.21 Comparison of AnyBURL and RESCAL on CodEx-M in regard to the combined entity and relation frequency . . . 83
A.22 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the question entity frequency . . . 84

A.23 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the answer entity frequency 84
A.24 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the combined entity and relation frequency 85
A.25 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the question entity frequency 85
A.26 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the answer entity frequency 86
A.27 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the combined entity and relation frequency 86
A.28 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the question entity frequency 87
A.29 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the answer entity frequency 87
A.30 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the combined entity and relation frequency 88
A.31 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the question entity frequency 88
A.32 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the answer entity frequency 89
A.33 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the combined entity and relation frequency 89
A.34 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar triples in the trainings data 90
A.35 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar triples in the trainings data 91
A.36 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar triples in the trainings data 91
A.37 Comparison of AnyBURL and ConvE on CodEx-M in regard to the amount of similar triples in the trainings data 92
A.38 Comparison of AnyBURL and RESCAL on CodEx-M in regard to the amount of similar triples in the trainings data 92
A.39 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the amount of similar triples in the trainings data 93
A.40 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the amount of similar triples in the trainings data 93
A.41 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the amount of similar triples in the trainings data 94
A.42 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the amount of similar triples in the trainings data 94

A.43 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=10) 95
A.44 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=25) 96
A.45 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100) 96
A.46 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=10) 97
A.47 Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=25) 97
A.48 Comparison of AnyBURL and ConvE on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100) 98
A.49 Comparison of AnyBURL and ConvE on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100) 99
A.50 Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100) 100
A.51 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100) 101
A.52 Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 101
A.53 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100) 102
A.54 Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100) 102
A.55 Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar head entities based on K-Means Clustering (k=100) 103

List of Tables

3.1 Dataset statistics 20

4.1 Conducted experiments per dataset 23
4.2 Example of LibKGE ranks for ComplEx on WN18RR 23
4.3 Example of LibKGE ranks for ComplEx on CoDEx-M expanded by the ranks and confidence values of AnyBURL 24

5.1 Hits@1, Hits@10 and MRR of models trained for CoDEx-M and FB15k-237 25
5.2 Hits@1, Hits@10 and MRR of models trained for WN18RR and YAGO3-10 26
5.3 Percentage of better predicted triples per category by AnyBURL and ComplEX on CoDEx-M 33
5.4 Percentage of better predicted triples per category by AnyBURL and ComplEX on CoDEx-M with fixed relation class assignments 33
5.5 Percentage of better predicted triples per category by AnyBURL and ConvE on CoDEx-M with fixed relation class assignments . . 34
5.6 Percentage of better predicted triples per category by AnyBURL and RESCAL on CoDEx-M with fixed relation class assignments . 34
5.7 Percentage of better predicted triples per category by AnyBURL and ConvE on FB15k-237 with fixed relation class assignments . . 35
5.8 Percentage of better predicted triples per category by AnyBURL and ComplEx on FB15k-237 with fixed relation class assignments 35
5.9 Percentage of better predicted triples per category by AnyBURL and RESCAL on FB15k-237 with fixed relation class assignments 35
5.10 Percentage of better predicted triples per category by AnyBURL and ComplEx on YAGO3-10 with fixed relation class assignments 36

6.1 Hits@1, Hits@10 and MRR on relation class test sets 57
6.2 Hits@1, Hits@10 and MRR on relation frequency test sets 58

6.3 Hits@1, Hits@10 and MRR on similar triples test sets 59
6.4 Hits@1, Hits@10 and MRR on entity cluster test sets 60

A.1 Hits@1, Hits@10 and MRR of all models trained for CoDex-M and FB15k-237 69
A.2 Hits@1, Hits@10 and MRR of all models trained for WN18RR and YAGO3-10 70

Chapter 1

Introduction

Link Prediction is the task of predicting the existence of a relation between two entities in a knowledge graph. To solve the task a link prediction model exploits the knowledge from the existing links in a knowledge graph to infer missing ones.

Current research is mainly concerned with sub-symbolic approaches to solve the problem. Here latent representations of the entities and relations of a knowledge graph are learned and applied to an equation in order to calculate predictions about missing links. Popular models from this approach include TransE [3], ComplEx [28] and ConvE [7].

As an alternative the symbolic approach has attracted much less attention. [29] Models from this approach learn knowledge in a symbolic language from an existing knowledge graph. Often in form of rules. The learned knowledge then can be applied to generate predictions for the task at hand. Models from this approach include AMIE [8], AnyBURL [17] and RLvLR [9]. Here especially AnyBURL delivers result competitive to the sub-symbolic approaches.

1.1 Motivation

In this master thesis I want to compare the results of symbolic and sub-symbolic approaches on commonly used datasets e.g. Codex [23], FB15k-237 [27] and WNRR [7]. Usually the performance of models are compared by calculating an aggregated metric over the results for the test dataset e.g. mean reciprocal rank or hits@k. I plan to do the comparison in a more granular approach. The goal here is to determine whether subsets of the datasets exists which share a common characteristic where one approach significantly outperforms the other.

If I am able to identify subsets which satisfy this criteria this found knowledge could help to deepen our understanding of the strengths and weaknesses of symbolic and sub-symbolic approaches and could open the opportunity to create hybrid models which combine the advantages of both approaches.

1.2 Research Questions

The main research question of this work is: Are there patterns in subsets of a dataset where one approach (symbolic or sub-symbolic) outperforms the other? Subsets here are created from the test data and divided by characteristics of the included triples. I plan on investigating following characteristics:

(i) Relation Class

(ii) Relation Frequency in the Trainings Data

(iii) Entity Frequency in the Trainings Data

(iv) Existence of Similar Triples in the Trainings Data

(v) Existence of Similar Entities in the Dataset

Furthermore, if such subsets exist another question I plan to answer is, if these sets can be used to test models for vulnerabilities in specific areas e.g. to test whether a model has problems handling less frequent entities.

1.3 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 and 3 outline the theoretical background, first chapter 2 discusses what knowledge graphs are in general and for what they are usually utilized. Afterwards, chapter 3 further defines what link prediction is, how symbolic and sub-symbolic approaches differ and presents the models and datasets used in the following analysis. Next, in chapter 4 the decisions around the experiments executed to obtain data, used for the analysis in chapter 5, are discussed. In chapter 5 the data is then analysed based on the characteristics presented in section 1.2. With the results of the comparison testsets, to answer the second research question, are created and evaluated in chapter 6. Chapter 7 discusses the research limitations and points out where to expand my research in the future and finally chapter 8 concludes this thesis.

Chapter 2

Knowledge Graphs

Knowledge graphs are graph-structured knowledge bases. They store information in the form of relationships between entities. A single information in the graph is referred to as fact. It consist of two entities and a relation between those two. This can be expressed as a triple: $(head, relation, tail)$. Knowledge Graphs are referred to as graphs since their entities can be interpreted as nodes and the relations as labelled and directed edges in a graph. The label here indicates which kind of relation the entities share and the direction indicates which entity is the head entity and which the tail entity i.e. an edge points from the head to the tail. An example for a knowledge graph can be seen in figure 2.1. [18]

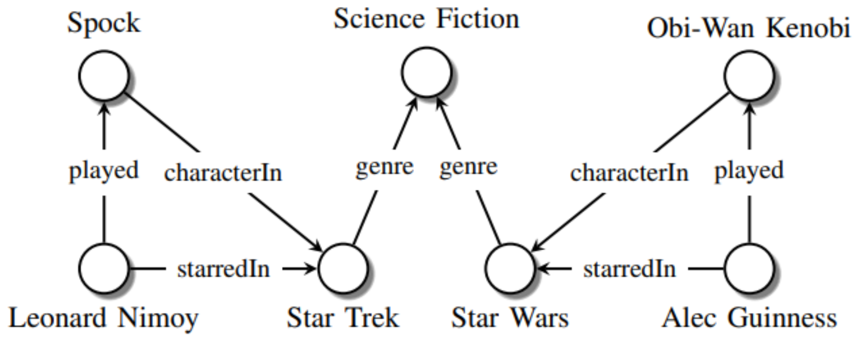
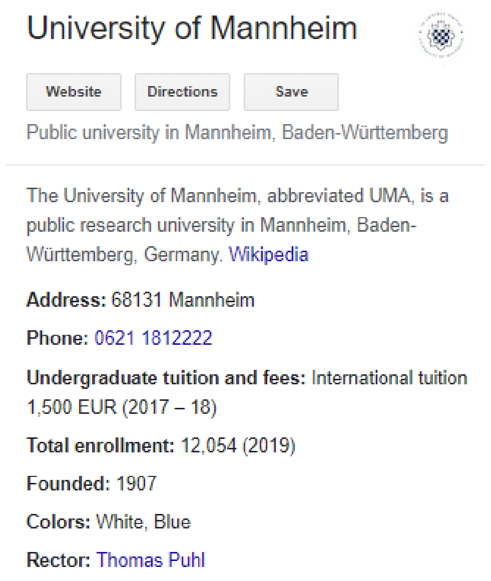


Figure 2.1: Example of a Knowledge Graph

The term 'knowledge graph' was first popularized by Google in 2012 [25]. In their blog they spoke about how they use their knowledge graph to enrich search engine results. The most noticeable part of how they use knowledge graphs are the side windows when searching for an entity with their search engine. An example of such can be seen in figure 2.2. Here we can see what kind of knowledge Google has about the University of Mannheim. For example it seems to be that *(University of Mannheim, founded_in, 1907)* is one of the facts in their Knowledge Graph.


University of Mannheim

Public university in Mannheim, Baden-Württemberg

The University of Mannheim, abbreviated UMA, is a public research university in Mannheim, Baden-Württemberg, Germany. Wikipedia

Address: 68131 Mannheim

Phone: 0621 1812222

Undergraduate tuition and fees: International tuition 1,500 EUR (2017 – 18)

Total enrollment: 12,054 (2019)

Founded: 1907

Colors: White, Blue

Rector: Thomas Puhl

Figure 2.2: Example of a Google Side Window

In the recent years knowledge graphs have become more and more popular and have found their way into further applications than search engines. An overview of applications can be seen in figure 2.3. Question Answering systems use knowledge graphs to enhance their results. Examples here include social chatbots and digital assistance like Siri. Recommender Systems leverage knowledge graphs as side information to improve and diversify their recommendations. Moreover, knowledge graphs are also used in information retrieval, domain-specific applications and more. [32]

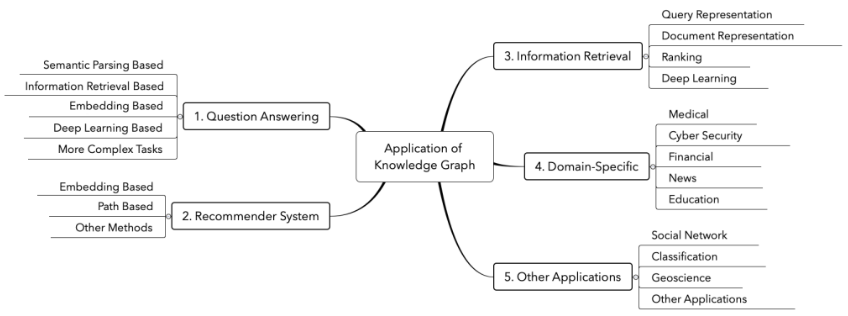


Figure 2.3: Applications of Knowledge Graphs

According to Paulheim [22] a knowledge graph is defined by the following four characteristics:

1. ”mainly describes real world entities and their interrelations, organized in a graph”
2. ”defines possible classes and relations of entities in a schema”
3. ”allows for potentially interrelating arbitrary entities with each other”
4. ”covers various topical domains”

The first characteristic defines that knowledge graphs consist of two kind of instances, entities and relations. An entity can be almost everything from an individual person to any kind of object. These entities are then linked through different kind of relations, which forms our graph.

The schema of the graph plays only a minor role. In most cases the instance-level statements (entities and triples) far outnumber the schema-level statements (entity classes and relations).

With the third characteristic Paulheim opens up the possibility that there are arbitrary relations between entities which are not included in the knowledge graph. The chapter 3 will discuss this further.

Lastly, another characteristic of knowledge graphs is that they do not focus on a single domain but interlink multiple topical domains.

Chapter 3

Knowledge Graph Completion

As discussed in section 2 knowledge graphs are not complete. They contain noisy and incomplete data. It is practically impossible to cover every possible entity and relation existing in the real-world or even in their domain. There might be missing entities and relations or a Knowledge Graph can include two entities/ relations representing the same real-world entity. Knowledge graph completion tries to tackle these and other problems. It can be seen as a way of data cleaning and expansion for knowledge graphs. In the following three examples are given for knowledge graph completion tasks:

Entity Resolution is according to Talburt ”the process of determining whether two [entities] are referring to the same object or to different objects”. [26]

Entity Prediction is the task of integrating new entities into the knowledge graphs. These entities are are discovered from other external sources and the knowledge graph includes no information about them. The goal is to find all possible relations this new entity has to the entities already existing in the graph. [1, p. 1]

Link Prediction is quiet similar to entity prediction. Instead of finding links for a new entity the goal here is to find all missing relations between already existing entities. [10, p. 125] Link prediction can be approached in two different ways: entity classification and triple classification. ”Entity classification tries to predict the type or the class of an entity [...]” [13] For a triple with a missing tail $(h, r, ?)$ the goal would be to list all entities which fit into the tail along with their confidence. Triple classification on the other hand is a binary tasks. Here the input is a compete triple (h, r, t) and the goal is to predict whether this triple is true or not. [13]

In the following we are going to focus on the task of link prediction. There are various models tackling the task. They can be categorized into one of the following two categories: symbolic and sub-symbolic approaches. Symbolic approaches try to reason about the problem with logic, this makes them comprehensible for humans. Sub-symbolic approaches on the other hand, solve the problem through less explainable mathematical equations. [13]

3.1 Symbolic Approaches

Symbolic approaches, or also sometimes referred to as ”Good Old Fashioned Artificial Intelligence”, focus on learning hypothesises in a symbolic (logical) language. [12] The strength of models from this approach lies in their explainability. Since the learned hypothesis are expressed in a logical language, either as axioms or rules, they are interpretable for a human and we can therefore reason about their predictions. In the following we will focus on rule-based models because they are more present for the link prediction task. [11] Rules are basically if-then clauses. Each rule has one or more conditions which have to be true and if they hold true the rule ”fires” and its conclusion can be derived. They are often written down as Horn clauses, an example for a Horn clause can be seen in its disjunctive and implicative form in equation 3.1 and 3.2 respectively. [5]

$$A \wedge \neg B_1 \vee \neg B_2 \vee ... \vee \neg B_n \tag{3.1}$$

$$A \leftarrow B_1, B_2, ..., B_n \tag{3.2}$$

Both equations express the same clause which states that if all conditions B, also called body atoms, are true so is the conclusion A, which is also referred to as the head.

3.1.1 AnyBURL

AnyBURL [17], short for Anytime Bottom-Up Rule Learning, is a link prediction algorithm that generates a set of rules from an existing knowledge graph and leverages the generated rules to make predictions. The core idea behind AnyBURL is that ”[...] sampled paths from a knowledge graph (random walks) are examples of very specific rules, which can be transformed into more general rules.” [21]

Rule Generalization

Rules are learned bottom-up, meaning we start with a sampled random path in the knowledge graph which then gets generalized into a rule. The sampled random path is called the ground path or grounding. In [17] generalized rules are classified into three types AC_1, AC_2 and C.

C rules are generalizations of cyclic ground paths:

$$h(Y, X) \leftarrow b_1(X, A_2), ..., b_n(A_n, Y). \tag{3.3}$$

AC_2 rules are generalizations of acyclic ground paths:

$$h(c_0, X) \leftarrow b_1(X, A_1), ..., b_n(A_n, A_{n+1}). \tag{3.4}$$

AC_1 rules can be generalized from both cyclic (with $c_0 = c_{n+1}$) and acyclic ones ($c_0 \neq c_{n+1}$):

$$h(c_0, X) \leftarrow b_1(X, A_1), ..., b_n(A_n, c_{n+1}). \tag{3.5}$$

X and Y are for variables appearing in the head of the Horn clause, A_i is a variable appearing only in the body and c_i is a constant. Furthermore, $h(e_1, e_2)$ and $b_i(e_1, e_2)$ are another notation to express a triple (e_1, h, e_2) or (e_1, b_i, e_2).

To get from a randomly sampled path to a rule a generalization lattice is build. The root of this lattice is the randomly sampled path and from there it is step-wise modified by either of the two following operations:

1. all occurrences of a constant are replaced by a new variable
2. an atom of the body is dropped

After applying one of these a operations, the resulting rule is put into one of the following three categories:

1. Ambiguous Prediction
2. Shorter Bottom Rule
3. Useless Atom

Ambiguous predictions have a variable in the head which does not appear in the body of the rule. Therefore the rule does not make a concrete prediction and is not useful for the task of link prediction. Shorter bottom rules are rules which might be useful but can also be deprived from the generalization lattice of a shorter path. To avoid duplicates, rules are only created from the shortest path. The last category, useless atom, is the only one which gets generalized further. Rules fitting into this category have either an atom with only constants or an atom with a constant and an unbound variable. A unbound variable is a variable not appearing in any other body atoms or the head.

If a rule does not fit into either of these categories it is a fully generalized rule and can be added to the final set of rules. In figure 3.1 an example of a generalization lattice for an acyclic path can be seen. [16]

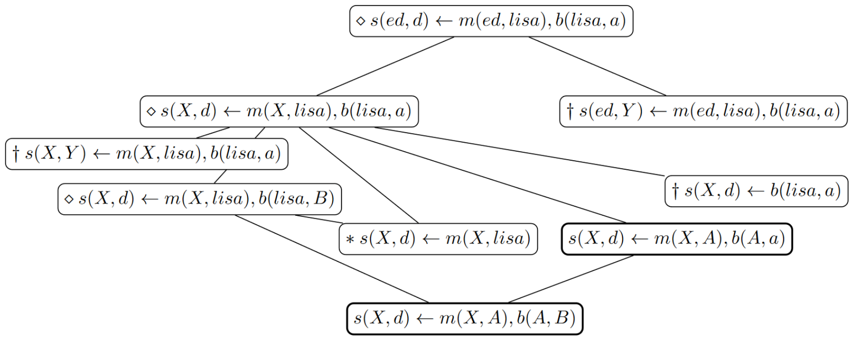

$\diamond\ s(ed,d) \leftarrow m(ed,lisa), b(lisa,a)$
$\diamond\ s(X,d) \leftarrow m(X,lisa), b(lisa,a)$
$\dagger\ s(ed,Y) \leftarrow m(ed,lisa), b(lisa,a)$
$\dagger\ s(X,Y) \leftarrow m(X,lisa), b(lisa,a)$
$\dagger\ s(X,d) \leftarrow b(lisa,a)$
$\diamond\ s(X,d) \leftarrow m(X,lisa), b(lisa,B)$
$*\ s(X,d) \leftarrow m(X,lisa)$
$s(X,d) \leftarrow m(X,A), b(A,a)$
$s(X,d) \leftarrow m(X,A), b(A,B)$

Figure 3.1: Example of a generalization lattice of an acyclic path (Ambiguous Prediction $\dagger$, Shorter Bottom Rule $*$, Useless Atom $\diamond$)

The lattice starts with the path $s(ed,d) \leftarrow m(ed,lisa), b(lisa,a)$ through applying the the generalization operations to final rules are created, the AC_1 rule $s(X,d) \leftarrow m(X,A), b(A,a)$ and the AC_2 rule $s(X,d) \leftarrow m(X,A), b(A,B)$.

In [16] it is argued that those two rules are always the result if an acyclic path is generalized and therefore AnyBURL does not need to build the entire generalization for every sampled path but can directly create these rules. The same goes for cyclic path. A generalization will always lead to a C and two AC_1 rules.

Rule Learning

Algorithm 1 shows the AnyBURL algorithm for learning rules. AnyBURL learns rules by sampling random paths of the length n and generalizes these to rules as discussed in the previous section. The function $score(r,s)$ computes the confidence of rule efficiently by using a sampling strategy, where the parameter s specifies the sample size. The confidence can then be used to check whether the quality criteria Q is satisfied. In case it is satisfied, the rule is added to the final set of rules and otherwise discarded. The algorithm tries to find as many rules as possible within the time frame ts. Afterwards it checks if the current amount of rules for paths of the length n satisfy a certain saturation sat. If it does n is increased and if not another timeslot of the length ts is spend searching further rules of the same length. [17]

Algorithm 1 Anytime Botton-up Rule Learning

```
    AnyBURL($\mathbb{G}, s, sat, Q, ts$)
 1: $n = 2$
 2: $R = \emptyset$
 3: loop
 4:     $R_s = \emptyset$
 5:     $start = currentTime()$
 6:     repeat
 7:         $p = samplePath(\mathbb{G}, n)$
 8:         $R_p = generateRules(p)$
 9:         for $r \in R_p$ do
10:             $score(r, s)$
11:             if $Q(r)$ then
12:                 $R_s = R_s \cup \{r\}$
13:             end if
14:         end for
15:     until $currentTime() > start + ts$
16:     $R'_s = R_s \cap R$
17:     if $|R'_s|/|R_s| > sat$ then
18:         $n = n + 1$
19:     end if
20:     $R = R_s \cup R$
21: end loop
22: return $R$
```

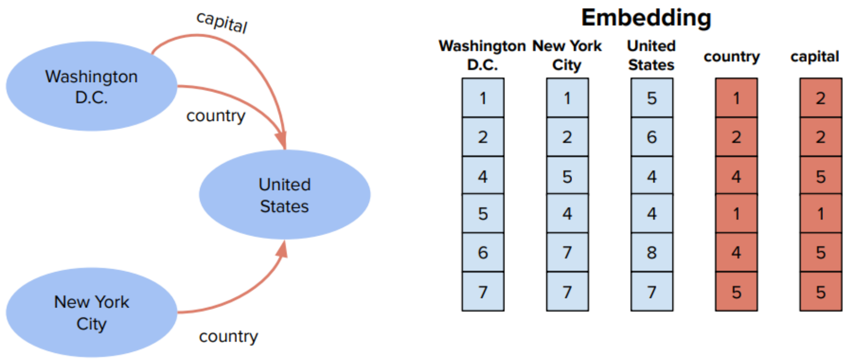
Rule Application

If the algorithm from the previous section is applied to a knowledge graph it returns a set of rules R. These rules can then be used to create a ranking of candidates to fill in an incomplete triple $(h, r, ?)$ or $(?, r, t)$. In [17] two methods are proposed to calculate this ranking of candidates. The first method is refereed to as Noisy-Or and calculates the confidence of a candidate by multiplying the confidences of all rules applicable to the candidate. The second method orders the candidates based on their rule with the highest confidence. In case two candidates got the same confidence, they are ordered by their second best rule and so on, until all candidates are ordered. The first method assumes that all rules are independent, while the second method ranks candidates, for which two independent rules fire, too low.

In the paper it is discussed that the independence assumption is often wrong and therefore it makes more sense to apply the second method. They also provide numbers which support this assumption.

3.2 Sub-Symbolic Approaches

Sub-symbolic approaches are based on statistics. They try to learn numerical models from the existing facts in the knowledge graph. These models can then be used to predict the existence of a triple. [18] The most prominent models for knowledge graph completion of this approach are embedding-based models. These models learn a vector representation for each entity and relation. This is also often referred to as an embedding. In this representation it is then assumed that similar entities and similar relations will have similar vectors. An example for these embeddings can be seen in figure 3.2. On the left side we see a knowledge graph with three entities and two relations. Every entity and relation are represented on the right side by an embedding. Our two entities *Washington D.C* and *New York City* are both cities and therefore we can assume that their semantic meaning are quiet similar. The embeddings of these two entities are also quiet similar which demonstrates that our previous assumption is correct. [2] Our embeddings are also called latent features because they can not be directly observed in the data. Instead our model has to infer these features from the data. [18]



Washington D.C.	New York City	United States	country	capital
1	1	5	1	2
2	2	6	2	2
4	5	4	4	5
5	4	4	1	1
6	7	8	4	5
7	7	7	5	5

Figure 3.2: Example of an Embedding for a Knowledge Graph

An embedding-based model is defined by three characteristics [2]:

1. representations of entities and relationships
2. the scoring function
3. the loss function

As stated earlier our entities and relations are represented through vectors, their embeddings. Some models vary from this a bit and use complex numbers instead of real ones [28] or use matrices to represent relationships [19].

The score function $f(h, r, t)$ calculates the distance between the embeddings of two entities relative to their relation. If the triple holds true, its score should be in an optimal case equal to 1. Lastly the loss function defines the objective which is going to be minimized during the training of our model where the embeddings for our entities and relations are learned. After learning the entity and relation representations through optimizing the loss function for the given score function, the model can be used to create a ranking of all candidates for a triple missing either its head $(?, r, t)$ or its tail $(h, r, ?)$. The score is then calculated for all candidates $c \in C$ such that (c, r, t) or (h, r, c) do not exists in the given knowledge graph. All candidates are then sorted by their score which then results in the ranking.

3.2.1 RESCAL

One such an embedding-based model is RESCAL [19][20]. The model explains triples via pairwise interactions of latent features. It calculates the score of a triple as

$$s(h, r, t) = e_h^T M_r e_t \tag{3.6}$$

where $e_h, e_t \in \mathbb{R}^D$ and $M_r \in \mathbb{R}^{D*D}$. In this formula the interactions between two entity vectors are captured using only multiplicative terms, which makes this a bilinear model.

Furthermore, we can see that entities are encoded into a D-dimensional vector representation and have the same representation regardless of whether they occur as head or tail entity in a triple. Moreover, they also share the same representation for entities independent of the relation in a triple. Meaning that for every relation the vector of an entity stays the same. In [18] the authors of the RESCAL model argue that this allows their model to ”[...] propagate information between triples [...]” and ”[...] capture global dependencies in the data.”. The relation in equation 3.6 is represented via a matrix M_r. Important to note about this matrix is its asymmetry. This allows the model to capture asymmetric relations e.g. while the model should predict the triple $(Darth\_Vader,\ father\_of,\ Luke\_Skywalker)$ as *True*, the triple $(Luke\_Skywalker, father\_of, Darth\_Vader)$ should be predicted as *False*.

Training through RESCAL-ALS

The model is trained through finding estimates for the matrices E and M_r which can be achieved by solving following regularized minimization problem

$$\min_{E,M_r} f(E, M_r) + g(E, M_r) \tag{3.7}$$

where

$$f(E, M_r) = \frac{1}{2}(\sum_{h,r,t} \|X_{hrt} - s(h,r,t)\|_F^2) \tag{3.8}$$

and

$$g(E, M_r) = \frac{1}{2}\lambda(\|E\|_F^2 + \sum_k \|M_r\|_F^2). \tag{3.9}$$

The term g is included as regularization term to prevent the model from overfitting. The regularized minimization problem in equation 3.7 is solved with an for RESCAL adjusted version of the alternating least-squares approach called RESCAL-ALS. This approach consists of a sequence of very efficient, closed-form updates as described in [20]. This training method assumes the closed-world assumption.

Training through Pairwise Loss

In [18] an alternative training method under the open-world assumption is presented which is more similar to the way other embedding-based models are trained. They refer to it as pairwise loss. Here two sets of triples are used to train the model, positives and negatives denoted as D^+ and D^- respectively. The positive triples are every known existing triple in our knowledge graph and the negative triples are created by corrupting existing triples. For the triples in the negative set it is not guaranteed that they are negative since a randomly corrupted triple could actually be an unknown true fact, but they are seen as *assumed-to-be-negative*.

Using stochastic gradient descent or any of its variations following objective function is then minimized

$$\min_{E,M_r} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \mathcal{L}(s(x^+), s(x^-)) + g(E, M_r) \tag{3.10}$$

using the same regularization as in equation 3.9 and where $\mathcal{L}$ is a margin-based ranking loss function such as

$$\mathcal{L}(s, s') = max(1 + s' - s, 0). \tag{3.11}$$

The usage of the margin-based loss has the advantage that negative triples do not have to be necessarily negative, they just have to be ”more negative” than the positive ones.

3.2.2 DistMult

DistMult [30] is another embedding-based model. It also uses the same basic bilinear scoring function as RESCAL in equation 3.6. The difference here is that DistMult restricts the relation matrix M_r to be a diagonal matrix. This reduces the parameters for the learnable parameters for a single relation from $D * D$ to just D. Therefore DistMult can be seen as a simplified version of RESCAL with the score function

$$s(h, r, t) = e_h^T diag(r) e_t \tag{3.12}$$

where e_h, e_t and $r \in \mathbb{R}^D$. The reduction of the relation dimensionality makes the model more computational efficient, especially on large knowledge bases, but also less expressive than RESCAL. Chen [6] provides a good graphic to compare the two models which can be seen in figure 3.3.

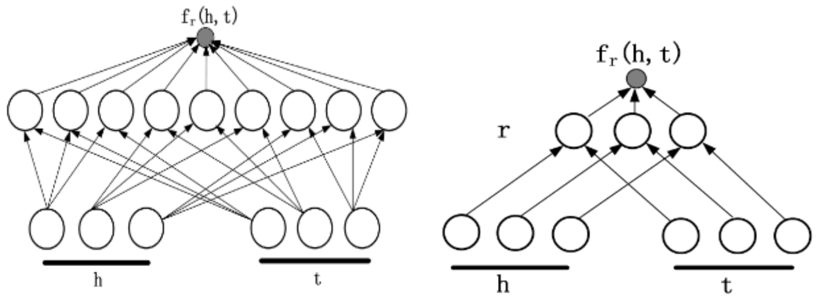


Figure 3.3: RESCAL vs. DistMult

Here our dimension size is $D = 3$. As we can see in RESCAL this leads to us having 3^2 latent features for the relation which have to be learned during the training while DistMult requires only 3.

3.2.3 ComplEx

While DistMult is more effective in regard to time and space complexity, it can not express antisymmetric relations. RESCAL on the other hand can express this but the size of the relation matrix leads to an explosion in the number of parameters. To combine the advantages of both methods Trouillon proposes a model called ComplEx [28]. ComplEx uses a similar score function as DistMult but instead of calculating the dot product between the embeddings as real numbers, it uses complex vectors to represent the entities and relations. The dot product of complex vectors is referred to as the Hermitian dot product. Other than the dot product of real numbers it is not symmetrical. This enables the model to capture asymmetric relations while retaining the efficiency benefits of DistMult. The score function of ComplEx is

$$s(h, r, t) = Re(< w_r, e_h, \bar{e}_t >) \tag{3.13}$$

where e_h, e_t and $w_r \in \mathbb{C}$. To use this score function in our loss function, as defined in equation 3.11, the resulting scores need to be purely real. Therefore after calculating the Hermitian dot product only the real part is kept and the imaginary part simply discarded.

3.2.4 ConvE

ConvE [7] presents another model trying to solve the task of link prediction. This model differentiates itself from other models by using 2D-convolutions over their embeddings. They argue that the use of a 2D convolution creates additional points of interaction between the embeddings and therefore increase the expressiveness of their model.

Convolution modifies an input by a filter. The filter has to have the same amount of dimensions as the input. Therefore if we have an 1-dimensional input we have to either use 1D-convolution or transform our input into a higher dimensional representation. Figure 3.4 depicts an example of how a convolution filter is applied in a two dimensional space. This example has a filter of the size 3×3. A feature map is calculated by multiplying the input with the filter. Here the first element of the input is multiplied with the first element of the filter, then the second with the second and so on. To calculate the next feature map the filter would be shifted one to the right and the same process would be repeated until the filter reaches the end of the input. Filters are also often referred to as kernel and can have different sizes. [31]

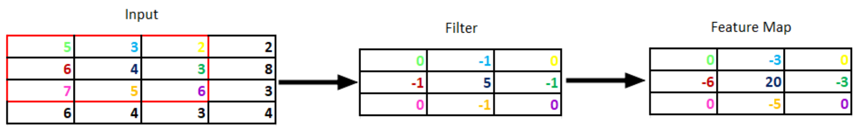


Figure 3.4: Example of a Convolution Filter

In the following I will explain how ConvE integrates the convolution into their model along figure 3.5. In the beginning the embedding of the head entity $e_h \in \mathbb{R}^k$ and of the relation $r_r \in \mathbb{R}^k$ are reshaped into 2D $\bar{e_s}, \bar{r_r} \in \mathbb{R}^{k_w \times k_h}$, where $k = k_w k_h$. The two reshaped embeddings are then concatenated and used as an input for a 2D convolution with filters ω of the size $m \times n$. The resulting feature maps are then transformed into a vector $vec \in \mathbb{R}^{c \times m \times n}$ where c is the number of features maps generated from the convolution. In the next step the vector is projected into a k-dimensional space using linear transformation parametrized by the matrix $W \in \mathbb{R}^{cmn} \times k$. Finally the inner product between the projected vector and the embedding of the tail entity $e_t \in \mathbb{R}^k$ results in the final prediction. [7]

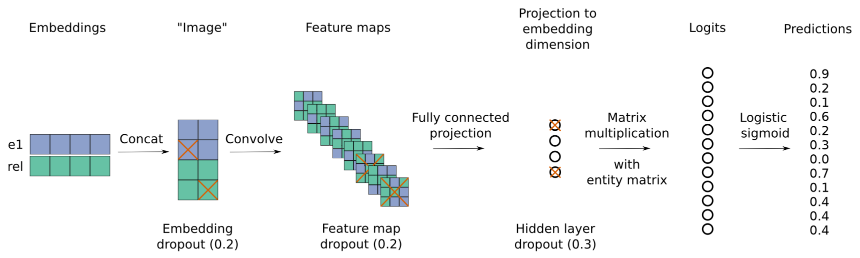

Embeddings
"Image"
Feature maps
Projection to embedding dimension
Logits
Predictions
e1
rel
Concat
Convolve
Fully connected projection
Matrix multiplication with entity matrix
Logistic sigmoid
Embedding dropout (0.2)
Feature map dropout (0.2)
Hidden layer dropout (0.3)
0.9
0.2
0.1
0.6
0.2
0.3
0.0
0.7
0.1
0.4
0.4
0.4

Figure 3.5: Steps of the ConvE Model

Formally the above can be defined as the score function of the model:

$$s(h, r, t) = f(vec(f([\bar{e_h}; \bar{r_r} * \omega))W)e_t \tag{3.14}$$

Furthermore, the authors of the model suggest using a different loss function. Instead of the margin-based loss RESCAL, DistMult and ComplEx are using, ConvE uses binary cross entropy loss:

$$\mathcal{L}(s, s') = s' * log(s) + (1 - s') * log(1 - s)) \tag{3.15}$$

3.3 Comparison of the Approaches

In the previous sections symbolic and sub-symbolic models have been presented. In the following we have a deeper look into the differences in terms of advantages/ disadvantages of these two approaches.

The main advantage for the symbolic approach as previously mentioned is the explainability. Especially in a rule-based system results and often intermediate steps can be directly explained and it can be shown which part of the data lead the model to its results. In real world applications having a result which can be explained is often a wanted feature and sometimes it might also become mandatory through laws such as the European Union's General Data Protection Regulation. A sub-symoblic model does not offer the possibility of an explanation for its results.

A further advantage of rule-based models is the possibility to modularly adjust the rules. Rules can be added or removed at any given time. A sub-symbolic model on the other hand has to be completely retrained once the trainings data changes and its impractical to make small adjustments to trained parameters. This modularity rule-based systems offer also allows for a knowledge transfer between different models. Rules from one model could be copied into the other.

Sub-symbolic approaches on the other side are often easier to scale up and therefore can be better applied to larger knowledge bases. The models are often computational more efficient but have a large amount of hyperparameters which have to be optimized. Moreover, embedding-based models are robuster to noisy or missing data, compared to rule-based models, since rule-based models derive rules from specific facts where embedding-based models only use the facts as one under many to optimize its parameters. This on the other hand makes the embedding-based models more dependent on the distribution of the trainings data. If the trainings data only contains a few examples for a entity or relation the parameters are likely to not be optimized for this entity or relation in favour of being better in predicting an entity or relation more frequently represented in the data. [13]

3.4 Model Evaluation

As discussed in section 3.1 and 3.2 both methods are used to generate a ranking of candidates for an incomplete triple $(h, r, ?)$. As the rank of such an triple one understands the position at which the correct answer is in this ranking. This is also often refereed to as the raw rank. The rank can be filtered by other known true triples which is then called the filtered rank i.e. when evaluating the predictions for a triple (h, r, t) there might be other triples in the knowledge graph (h, r, t') for $t \neq t'$, since (h, r, t') exists in the knowledge graph and therefore is also a true triple it is desirable to give it a high score and a low rank. This could affect the rank of the triple (h, r, t) in a negative way. To avoid this other true triples are not counted for the filtered rank. Therefore the filtered rank is always less or equal to the raw rank. [3]

In most cases when evaluating a link prediction model a test dataset is given which includes a set of prior unknown triples for which the head and tail are predicted. The resulting ranks can then be used to calculate one or more of the following metrics in order to compare the general performance of different models: Hits@k (eq. 3.16), Mean Rank (eq. 3.17) and Mean Reciprocal Rank (eq. 3.18).

$$Hits@k = \frac{|q \in Q|rank(q) \leq k|}{|Q|} \tag{3.16}$$

$$MR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_i \tag{3.17}$$

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} 1/rank_i \tag{3.18}$$

with Q being the set of test triples.

While this allows to compare the overall performance of two models it is also interesting to compare two models in a case by case analysis, meaning with respect to specific completion task e.g. $(ed, s, ?)$. The easiest approach here would be to take the difference between the rank the two models return for the true answer. In [15] it is argued that this approach is flawed. For example if our first model predicts the correct answer on position #12 while the second predicts it on position #77, it looks like the first model solved the task significantly better. But if we take a look at the confidence values it might be that the confidence values given by the first model are very close between rank #10 and #80. To integrate the confidence values into the comparison they propose following formula:

$$\psi(c, \mathcal{E}, r(e, ?)) = \frac{conf(c, r(e, ?))}{conf(\mathcal{A}[\mathcal{E}[c]^{\#}], r(e, ?))} \tag{3.19}$$

where $conf(c, r(e, ?))$ describes the confidence the first model assigned to the task for candidate c, $\mathcal{E}[c]^{\#}$ is the position the second model ranks the candidate c, $\mathcal{A}[i]$ is the entity the first model has at the rank i and therefore $conf(\mathcal{A}[\mathcal{E}[c]^{\#}], r(e, ?))$ is the confidence the first model assigns to the entity at the rank where the candidate c appears in the ranking of the second model. A score of around 1 would show that the models performed quiet similar, scores above 1 mean that the first model predicted the triple better and scores below 1 would mean that the second model predicted the triple better.

3.5 Datasets

To test and benchmark knowledge graph completion models various datasets have been proposed. Such datasets always contain a set of trainings, validation and test triples. While the training and validation part is used to train/ prepare the models the final evaluation get executed on the test set.

In the following I will present the datasets relevant for this thesis.

Freebase is a huge knowledge base of general facts. The data was collaboratively collected by its community.

From this data source Bordes [3] created two datasets for link prediction evaluation, FB15k and FB1M. While FB1M was not often used after its publication FB15k became quiet prominent and was frequently used.

Later it was found that FB15k has the problem of containing many test triples that invert trainings triples or were almost a duplicate of a trainings triple. To avoid this data leakage Toutanova [27] extracted a new dataset FB15k-237 from FB15k avoiding these problems. FB15k-237 still contains a few flaws in [23] it is stated that the dataset has skewed relations e.g. relations which always point towards a single entity or a relation referring to the gender of a person pointing in 78.41% of all cases to the entity male. Moreover the dataset is also said to contain fixed-set relations, relation types that connect entities to a fixed set of values. Nevertheless FB15k-237 is still widely used and valued as a dataset for evaluating link prediction models.

Wordnet is a lexical database. It expresses semantic relations between words.

Here Bordes [3] also extracted a dataset called WN18 similar to the way he extracted FB15k which was published in the same paper.

In his publication about ConvE, Dettmers [7] argued that WN18 also contains flaws and therefore created the dataset WN18RR as a correction of WN18. Here he does not go into detail what these flaws are. Other authors later stated that the problem was the same as FB15k had: inverse relation test leakage. [24]

YAGO is a knowledge base that combines the information from Wikipedia in multiple languages. Here WordNet was used to create a connection between the different languages.

The knowledge base was extended by Mahdisoltani in 2015 [14]. This new version is referred to as YAGO3.

Since YAGO3 is huge dataset a subset was also proposed called YAGO3-10. This dataset only contains entities which have a minimum of 10 relations each. This drastically lowers the overall size of this dataset.

CoDEx [23] is a dataset extracted from Wikidata and Wikipedia. The dataset was created with the intent of creating a dataset for link prediction evaluation. The authors state that they carefully created the dataset to be a hard set without flaws other comparable sets such as FB15k-237 contain. The dataset comes in three variations, differing in size: S, M and L.

Table 3.1 compares the amount of entities, relations and facts each of the previously mentioned datasets contain.

	# of Entities	**# of Relations**	**# of Facts**
FB1M	1 000 000	25 000	17 000 000
FB15k	14 951	1 345	592 213
FB15k-237	14 541	237	310 116
WN18	40 943	18	151 442
WN18RR	40 943	11	93 003
YAGO3	4 400 000	77	24 000 000
YAGO3-10	123 182	37	1 089 040
CoDEx-S	2 034	42	36 000
CoDEx-M	17 050	51	206 000
CoDEx-L	77 952	69	612 000

Table 3.1: Dataset statistics

Chapter 4

Experimental Setting

Before being able to compare the prediction results of both approaches, we need to run experiments with various models to obtain their rankings over different datasets. These rankings will then be compared in the next chapter.

4.1 Datasets

As datasets I first selected CoDEx-M and WN18RR. CoDEx-M was selected because it is a dataset explicitly created for link prediction which leads to it having an appropriate difficulty for the task. Moreover with its 51 relations and 206k triples it seems to have decent size for having enough data to analyse but not taking to much computational power to conduct my experiments. This also what lead me to pick CoDEx-M over the other CoDEx variants (CoDEx-S and CoDEx-L).

WN18RR was picked because of its popularity. Furthermore it includes a smaller set of triples than CoDEx-M, only containing 93k and eleven relations. It could be that the smaller size leads to more comprehensible results during the comparison than the larger CoDEx-M. While conducting the analysis it turned out that having the smaller data size lead to it not being very useful for my case therefore I focused more on the other datasets and as a consequence of that only two models were trained for this dataset, as can be seen in table 4.1. Furthermore, for this reason WN18RR will not be focused on in upcoming analysis.

After ruling out WN18RR I still wanted to have a dataset to compare the results I obtained on CoDEx-M. Therefore I decided to include FB15k-237 which similar sized to CoDEx-M but has some flaws which is the reason I initially picked CoDEx-M over FB15k-237. But despite it flaws, FB15k-237 is still often used as a benchmark and most researchers come to the conclusion that it is nevertheless a useful tool.

Furthermore, I also wanted to run the comparison on one larger dataset and therefore also included YAGO3-10. Because of its size it is quiet expensive computational-wise to run experiments on it and I therefore only trained two models on it.

4.2 Models

As for the models I selected AnyBURL as the only model to represent the symbolic approach. Out of the currently published symbolic models, it is the best performing one. To apply this model to the previously mentioned datasets I used the code published on its website[1]. The model has a set of hyperparameters to optimize the learning of rules but since not much research has been done yet finding the optimal values, I used the default values for these.

The sub-symbolic approach gets represented by ComplEx and RESCAL. Both models are often used as benchmarks for other models and produce competitive results. These models are to some extend similar as they can both be categorized as multiplicative models, due to the nature of their score function. To include a model which differs more from these two, I later decided to verify my results with ConvE.

To run the experiments for the sub-symbolic models LibKGE was used. LibKGE is a library which produces fast and reproducible results for knowledge graph embeddings. [4] The library also comes with a set of pretrained models and the corresponding configuration file, to reproduce the experiment. I utilized these configuration files to train my models.

The embbedings for these models are initialized with random values before they get optimized during the training. This results in slightly different models every time one gets trained, even when the same configuration is being used. To avoid this random factor influencing my comparison I run every experiment for the sub-symbolic models five times and aggregated the results. This is discussed further in the next section.

Table 4.1 shows which model was applied to which dataset. As can be seen ConvE and RESCAL were not applied to all datasets, the reason for that was already discussed in section 4.1. All configurations and experiment data is reviewable on Github[2], as well as the results and code for the comparison of the next chapters.

[1]https://web.informatik.uni-mannheim.de/AnyBURL/ [last accessed: 15.07.2022]

[2]https://github.com/LJ171/Thesis

	CoDEx-M	**FB15k-237**	**WN18RR**	**YAGO3-10**
AnyBURL	x	x	x	x
ComplEx	x	x	x	x
ConvE	x	x		
RESCAL	x	x		

Table 4.1: Conducted experiments per dataset

4.3 Data Aggregation

In this step the data from the symbolic experiments with AnyBURL and the data from the sub-symbolic experiments with LibKGE gets combined.

LibKGE produces during the prediction of the test dataset a YAML file which contains an entry per triple of the test dataset which contains the triple (h, r, t), the rank (raw & filtered) and the prediction direction, whether the head or the tail of the triple was predicted. Since the sub-symbolic experiments are repeated five times I combine their results and aggregate the resulting five ranks per triple into an average rank. An example for this can be seen in table 4.2. Furthermore, we can already see within these three examples that the rank for the individual models can differ significantly. As argued in the previous section this is the reason why the sub-symbolic experiments were repeated five times.

	h	**r**	**t**	**predicted _head**
0	Q73437	P1412	Q1860	False
1	Q73437	P1412	Q1860	True
2	Q95076	P106	Q10798782	False

	rank _filtered _kge_0	**rank _filtered _kge_1**	**rank _filtered _kge_2**	**rank _filtered _kge_3**	**rank _filtered _kge_4**	**rank _filtered _kge**
0	1	1	1	1	1	1
1	1111	1649	1187	108	1755	1162
2	1236	1701	190	357	519	801

Table 4.2: Example of LibKGE ranks for ComplEx on WN18RR

The output file from AnyBURL contains three lines per triples. First the triple

(h, r, t), second the ranking for the prediction of the head and and lastly the ranking for the prediction of the tail. The ranking is the filtered ranking and contains for each rank the additional information of the confidence AnyBURL assigned to each rank. Here I extract the two ranks for each triple and the corresponding confidence. Since I want to be able to calculate the difference-ψ from equation 3.19, I extract the confidence AnyBURL assigns to the rank given by the LibKGE model as well. If we expand table 4.2 by this information we end up with table 4.3.

	h	**r**	**t**	**predicted _head**
0	Q73437	P1412	Q1860	False
1	Q73437	P1412	Q1860	True
2	Q95076	P106	Q10798782	False

	rank _filtered _kge_0	**rank _filtered _kge_1**	**rank _filtered _kge_2**	**rank _filtered _kge_3**	**rank _filtered _kge_4**	**rank _filtered _kge**
0	1	1	1	1	1	1
1	1111	1649	1187	108	1755	1162
2	1236	1701	190	357	519	801

	rank _filtered _anyburl	**rank _difference**	**anyburl _confidence _anyburl**	**anyburl _confidence _kge**	**difference _psi**
0	1	0	0.382083	0.382083	1
1	6960	-5798	0.282316	0.536634	0.526087
2	614.0	187	0.470588	0.368421	1.277311

Table 4.3: Example of LibKGE ranks for ComplEx on CoDEx-M expanded by the ranks and confidence values of AnyBURL

In this table we can see that difference-ψ and the difference between the ComplEx and AnyBURL rank indicate the same information (which model is better) but while the rank difference let it seem like ComplEx predicted the second triple significantly better than AnyBURL, including the confidence lessens this effect.

Chapter 5

Comparison

Before the two approaches can be compared in regard to the characteristics defined in section 1.2, we need to compare their models in regard to their overall performance. The performance of the models will have a direct influence on the following comparison.

5.1 Overall Performance

Tables 5.1 and 5.2 show the hits@1, hits@10 and mean reciprocal rank for each model and dataset combination. As we can see the models achieve all similar results, especially between the sub-symbolic models there is only a small deviation. AnyBURL performs slightly worse than the other models on CoDEx-M, FB15k-237 and YAGO3-10 but performs better on WN18RR. Interesting to note here is that AnyBURL nevertheless produces the best hits@1 for CoDEx-M and YAGO3-10. The tables here only include one of the trained models for each sub-symbolic model. The results for the other models are almost identical as can be seen in appendix A.1.

	CoDEx-M			**FB15k-237**		
Model	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**
AnyBURL	0.313	0.412	0.285	0.238	0.501	0.322
ComplEx	0.261	0.475	0.334	0.253	0.534	0.346
ConvE	0.231	0.457	0.309	0.246	0.522	0.337
RESCAL	0.241	0.453	0.314	0.264	0.539	0.356

Table 5.1: Hits@1, Hits@10 and MRR of models trained for CoDEx-M and FB15k-237

	WN18RR			**YAGO3-10**		
Model	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**
AnyBURL	0.495	0.553	0.479	0.559	0.629	0.525
ComplEx	0.438	0.544	0.474	0.464	0.672	0.540
ConvE	-	-	-	-	-	-
RESCAL	-	-	-	-	-	-

Table 5.2: Hits@1, Hits@10 and MRR of models trained for WN18RR and YAGO3-10

This slight difference in performance can be seen as well when comparing all ranks the models produced. Figure 5.1 shows the filtered ranks of AnyBURL compared to the filtered ranks of ComplEx in a boxplot. While the median for both models is close to the optimum of one, ComplEx third quartile and maximum are lower than AnyBURLs. Furthermore AnyBURL shows a few more extreme outliers.

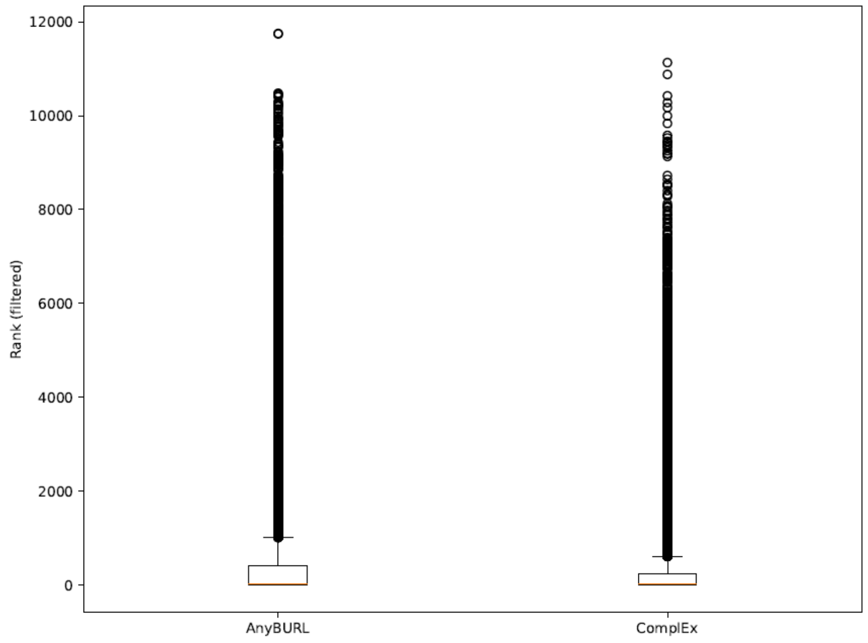


Figure 5.1: Comparison of Ranks predicted by AnyBURL and ComplEx for CoDEx-M

If we split up the above boxplot via the prediction direction i.e. whether the head was predicted $(?, r, t)$ or the tail was predicted $(h, r, ?)$, as can be seen in figure 5.2 we can make the same observations as for the figure above: in both prediction directions ComplEx slightly outperforms AnyBURL. Furthermore, we can see that both models are better at predicting the tails while struggling more with the prediction of the heads.

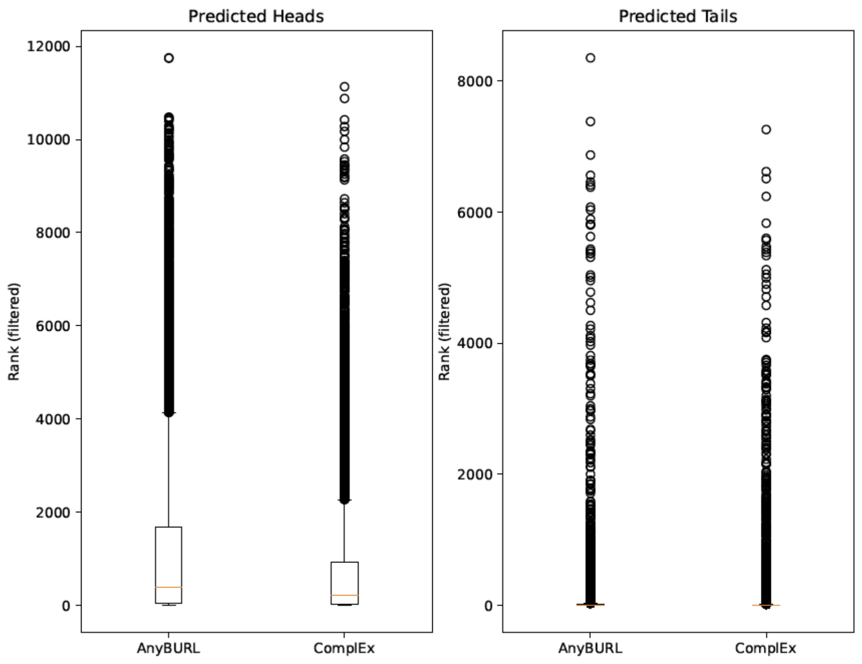


Figure 5.2: Comparison of Ranks predicted by AnyBURL and ComplEx by prediction direction on CoDEx-M

As was to expect the distribution of difference-ψ follows the same pattern as can bee seen in figure 5.3.

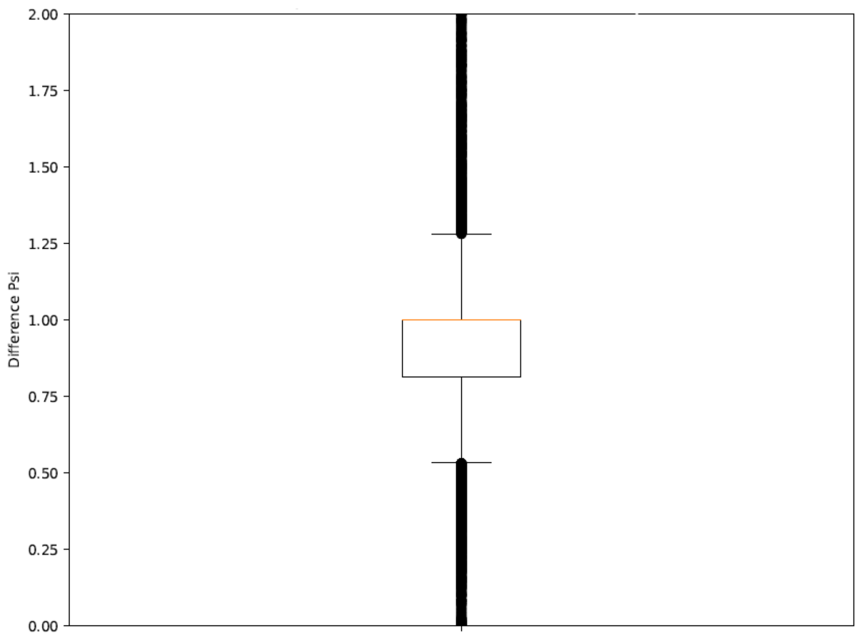


Figure 5.3: Comparison of difference-ψ between AnyBURL and ComplEx on CoDEx-M

If we plot the same graphs for the other model-dataset combinations, we end up with similar results. The corresponding figures can be found in appendix A.2.

The only outlier here are the results for WN18RR. A triple only counts as better predicted by one of the models if difference-ψ is above or below one. The more it deviates from one the stronger the rank difference between the compared models. If we take a look at figure 5.4 we can see that for this model-dataset combination only outliers deviate significantly from one. For my comparison I am mostly interested in the cases where one model predicts a triple better than the other. Given that there are only a few cases for WN18RR, this is one of the main reasons why this dataset proved not very useful for my comparison. To elaborate on this point a bit more the figure does not show that both models predict WN18RR always perfect but the figure shows that both models predict the true answer always on a similar rank and therefore shows us that there are triples in the test dataset which seem to be easier to predict than others for all models.

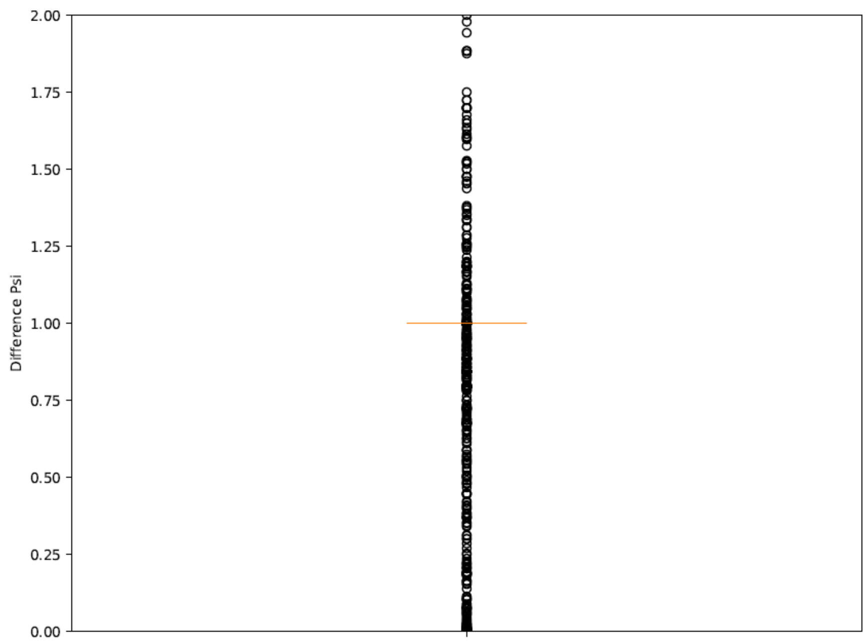


Figure 5.4: Comparison of difference-ψ between AnyBURL and ComplEx on WN18RR

To sum up the results of the comparison of the overall performance comparison, we saw that for all datasets except for WN18RR the sub-symbolic approach performed slightly better than the symbolic approach. The consequence of this is that our data is also slightly skewed in favour of the sub-symbolic models. For that reason when we compare the models based on the difference-ψ value we need to take that into consideration.

5.2 Better Predicted by & Thresholds

I decided to expand my dataset by a further variable called better $\_predicted\_by\_anyburl$ which transforms the difference-ψ values into a ternary variable. While the size of the deviation from one of difference-ψ encodes some information, this variable discards this part of the information in favour of creating a simpler variable which is easier to understand, interpret and work with. $better\_predicted\_by\_anyburl$ can obtain three values:

- -1 if ranking produced by the embedding-based model is significantly better,

- 0 if both models performed equally or
- 1 if AnyBURLs ranking is significantly better.

An open question here is how to define significantly better. To find an answer I created the variable based on different thresholds and compared the results. The threshold here is the amount which difference-ψ has to deviate at least from one to not count as an equally good performance any more. The distribution of $better\_predicted\_by\_anyburl$ for different thresholds can be seen in figure 5.5. Here we can notice that if the threshold goes slightly above 0 the amount of triples equally good predicted by both models increases a lot. If we increase the threshold further from 5 to 10 or 25 the amount of equally good predicted triples increases while the amount of better predicted triples decreases by a similar amount for both models. Above 50 almost all triples are labelled as equally good predicted while the amount of better predicted triples per model seem to equalize. Furthermore if the threshold is set to 100 there are no triples better predicted by ComplEx. Here we see a small short coming of the difference-ψ formula. Since two confidence values get divided the result can be at a minimum 0 while the maximum value of difference-ψ goes against ∞.

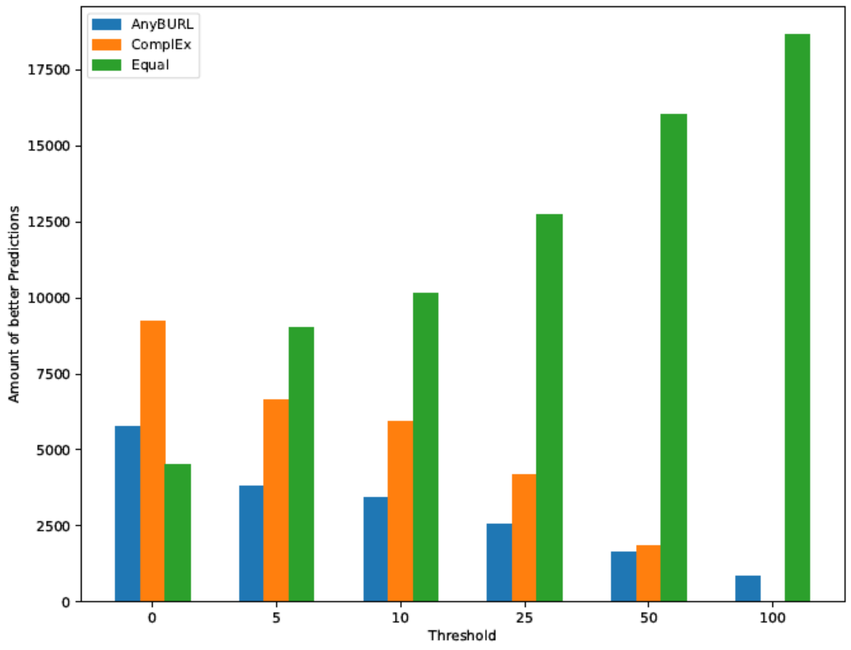


Figure 5.5: Comparison of different thresholds for the $better\_predicted\_by$ variable

In the end I decided to use 20 as the final threshold because a value lower than that leaves in my opinion to many cases where the ranks produced by both models were quiet close as better predicted by one of the models. A value above 20 on the other hand puts to many cases into to the equal bin and since for my comparison the cases where a triple is better predicted by one model are more interesting, keeping as many cases out of the equal bin seemed rational.

Nevertheless I also generated the comparisons with higher and lower thresholds and noticed that as long as the threshold does not get set to an extreme value (close to 0 or above 50) the results do not change significantly.

5.3 Comparison based on Characteristics

In the following the models will be compared in regard to the characteristics defined in the section 1.2.

5.3.1 Relation Class

The first characteristic I want to investigate is the relation class of a triple. Bordes [3] defines four relation classes ”1-To-1”, ”1-To-Many”, ”Many-To-1” and ”Many-To-Many” or as I will refer them to *1-1*, *1-N*, *M-1* and *M-N*. These relation classes categorize the relations according to the cardinalities of their head and tail entities. For example a relation gets put in to the *1-1* category if a head can appear with at most one tail and the head does also not appear with any other tail.

Before we can determine whether a certain approach performs better than the other in regard to a specific relation class it first needs to be determined if certain relations get better predicted by one of the approaches. For that I averaged the ”$better\_predicted\_by$” variable per relation. The closer the average is to 1 the more likely it is that AnyBURL is better in predicting this relation and the closer the value is to -1 the more likely it is that the sub-symbolic approach is better in predicting this relation. A value of 0 or close to 0 would indicate that both models perform equally good in regard to this relation.

If we now plot the averaged $better\_predicted\_by$ values per relation for AnyBURL and ComplEx on CoDEx-M and overlay the relation class category we get figure 5.6. One the first things one notices when looking at the figure is that all *1-1* relations have a value above 0, meaning that (almost) all triples containing a relation of that category get better predicted by AnyBURL in comparison to ComplEx. Furthermore, we can see that the *M-1* relations seem evenly distributed and that most relations belonging to the *M-N* category get better predicted by ComplEx. Another interesting takeaway from this figure is that for some reason CoDEx-M does not contain any relations belonging to the *1-N* relation class.

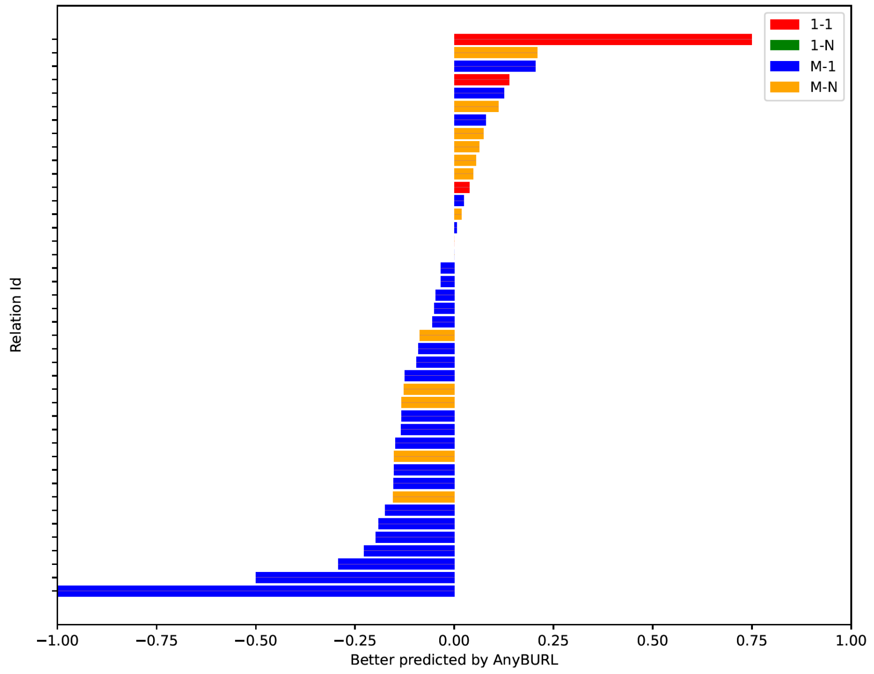


Figure 5.6: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the relation classes

Since not all relations are equally often represented in the dataset, averaging the $better\_predicted\_by$ value of each triple by their relation distorts the data. To give a better perception of the real data table 5.3 shows the percentage of better predicted triples for each model per relation class. What the previous graph missed to show is that 88.81% of all triples containing a *1-1* relation are equally good predicted by both models while only 10.49% are better predicted by AnyBURL.

Nevertheless that still indicates that AnyBURL performs better on *1-1* relations but in a less significant manner.

Moreover, the table shows us that ComplEx predicts roughly 10% more triples of the *M-1* and *M-N* category better than AnyBURL. Here we have to take into account that AnyBURL performed overall slightly worse than ComplEx. Therefore, in my opinion the data indicates that AnyBURL is better in predicting triples with *1-1* relations while only hinting that the same goes for ComplEx and *M-1* and *M-N* relations.

	AnyBURL	**ComplEx**	**Equal**
1-1	10.49%	0.7%	88.81%
1-N	0%	0%	0%
M-1	11.44%	21.66%	66.9%
M-N	16.01%	25.82%	58.18%

Table 5.3: Percentage of better predicted triples per category by AnyBURL and ComplEX on CoDEx-M

In table 5.3 all data points containing a relation of the *1-N* class are in that category. During the evaluation on the test dataset every triple gets used twice. Once of predicting the head and once for predicting the tail. When predicting the head of an *1-N* relation its essentially like predicting the tail of an *M-1* relation. Therefore I assigned all cases of the *1-N* category where the head was predicted to the *M-1* category. In case CoDEx-M had contained any *M-1* relations I would have done the same swap there as well. The result of this can be found in table 5.4. This corrections does not lead to any new findings the same interpretation of the numbers for the *1-N* and *M-N* relation classes now goes for the *M-1* class as well.

	AnyBURL	**ComplEx**	**Equal**
1-1	10.49%	0.7%	88.81%
1-N	16.01%	31.14%	52.85%
M-1	7.24%	12.91%	79.85%
M-N	16.01%	25.82%	58.18%

Table 5.4: Percentage of better predicted triples per category by AnyBURL and ComplEX on CoDEx-M with fixed relation class assignments

Comparing ConvE and RESCAL to AnyBURL in regard to the relation classes results in a similar outcome. This can be seen in figure 5.7 and tables 5.5 and 5.6. This shows us that my previous observations are not only valid for one model sub-symbolic model but for multiple and therefore maybe for the whole approach.

A slight take back here is the strong variations in the tables. While all tables display comparable results they do so in varying magnitudes. While AnyBURL only predicts 10% of the *1-1* relations better than ComplEx (table 5.4), it predicts 29% better than ConvE (table 5.5) and 81% better than RESCAL (table 5.6). By further analysing the data I could not determine an explanation for this strong variation.

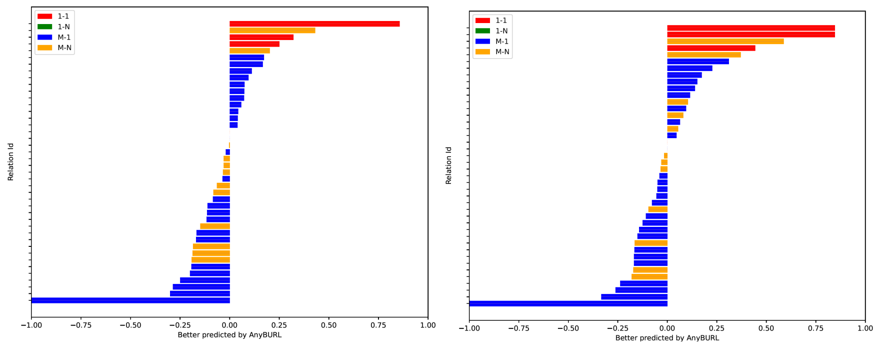


Figure 5.7: Comparison of AnyBURL and ConvE (left) / RESCAL (right) on CoDEx-M in regard to the relation classes

	AnyBURL	**RESCAL**	**Equal**
1-1	29.29%	0%	70.71%
1-N	22.66%	26.19%	51.15%
M-1	9.24%	13.21%	77.55%
M-N	15.29%	27.94%	56.77%

Table 5.5: Percentage of better predicted triples per category by AnyBURL and ConvE on CoDEx-M with fixed relation class assignments

	AnyBURL	**RESCAL**	**Equal**
1-1	81.1%	0%	18.9%
1-N	18.93%	27.65%	53.43%
M-1	13.55%	11.42%	75.04%
M-N	15.86%	26.24%	57.9%

Table 5.6: Percentage of better predicted triples per category by AnyBURL and RESCAL on CoDEx-M with fixed relation class assignments

If creating the same graph (as in figure 5.6 and 5.7) for the experiments on FB15k-237, the graph is incomprehensible due to the amount of relations in the dataset. This can be seen in appendix A.3. Therefore in the following I will only refer to the tables for FB15k-237.

First of for ConvE we can make the same observations as before on CodEx-M. As can be seen in table 5.7, while ConvE is better than AnyBURL in most relation classes, AnyBURL outperforms ConvE in regard to the *1-1* relation class.

	AnyBURL	ConvE	Equal
1-1	21.05%	6.58%	72.37%
1-N	16.2%	20.68%	63.12%
M-1	7.21%	15.89%	76.9%
M-N	11.37%	25.29%	63.35%

Table 5.7: Percentage of better predicted triples per category by AnyBURL and ConvE on FB15k-237 with fixed relation class assignments

When looking at the data for ComplEx (table 5.8) and RESCAL (table 5.9), we can still see that the embedding-based models outperform AnyBURL in the relation classes *1-N*, *M-1* and *M-N* while they perform similar in the *1-1* category. If we take into account here again that the sub-symbolic models generally outperformed AnyBURL, this data still slightly indicates that AnyBURL might be better than the sub-symbolic models in predicting relations of the *1-1* class.

	AnyBURL	ComplEx	Equal
1-1	5.66%	5.66%	88.68%
1-N	12.45%	22.3%	65.25%
M-1	7.04%	15.1%	77.86%
M-N	9.57%	24.99%	65.44%

Table 5.8: Percentage of better predicted triples per category by AnyBURL and ComplEx on FB15k-237 with fixed relation class assignments

	AnyBURL	RESCAL	Equal
1-1	8.54%	7.54%	83.92%
1-N	14.75%	22.57%	62.68%
M-1	7.01%	16.03%	76.96%
M-N	10.17%	25.71%	64.13%

Table 5.9: Percentage of better predicted triples per category by AnyBURL and RESCAL on FB15k-237 with fixed relation class assignments

In figure 5.8 we can see that the experiments on YAGO3-10 do not produce the same results as the experiments on CoDEx-M and FB15k-237 did. Here no model seems to be better than the other in regard to any relation class. If we refer to table 5.10 we notice that both models seem to perform equal in all relation classes except for the *M-1* class where AnyBURL is better than ComplEx.

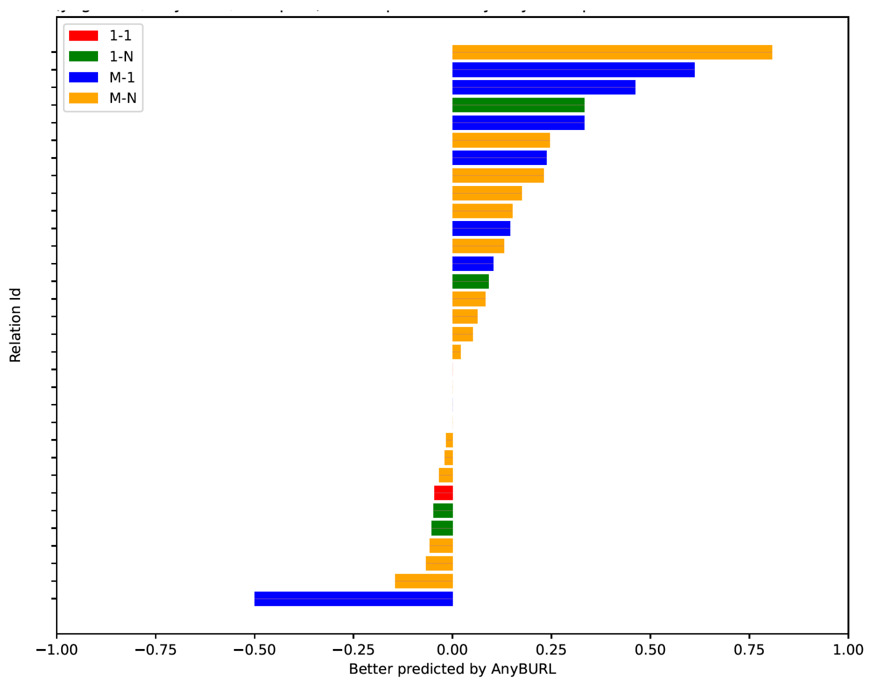


Figure 5.8: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the relation classes

	AnyBURL	**ComplEx**	**Equal**
1-1	3.57%	5.36%	91.07%
1-N	16.5%	15.46%	68.04%
M-1	21.92%	6.8%	71.28%
M-N	7.38%	8.35%	84.28%

Table 5.10: Percentage of better predicted triples per category by AnyBURL and ComplEx on YAGO3-10 with fixed relation class assignments

In this section we have now seen that AnyBURL seems to be better in predicting triples with *1-1* relations while the sub-symbolic models favour relations with a multi-cardinality. On the CoDEx-M dataset the data showed this almost unambiguously and on FB15k-237 the data showed the same result but not as clearly as on CoDEx-M. YAGO3-10 is an outlier here, on this dataset both models performed equally good in regard to the relation classes.

5.3.2 Relation Frequency in Trainings Data

The next characteristic I want to investigate is the relation frequency in the trainings data. To be more specific, in figure 5.9 we can see that in CoDEx-M some relations occur more often than others in the trainings data, therefore I wanted to see whether one approach might be better than the other in predicting triples containing less often occurring relations or containing often occurring relations. In appendix A.4 the same graph can be seen for FB15k-237 and YAGO3-10 which also show that some relations are more frequent than others in these datasets.

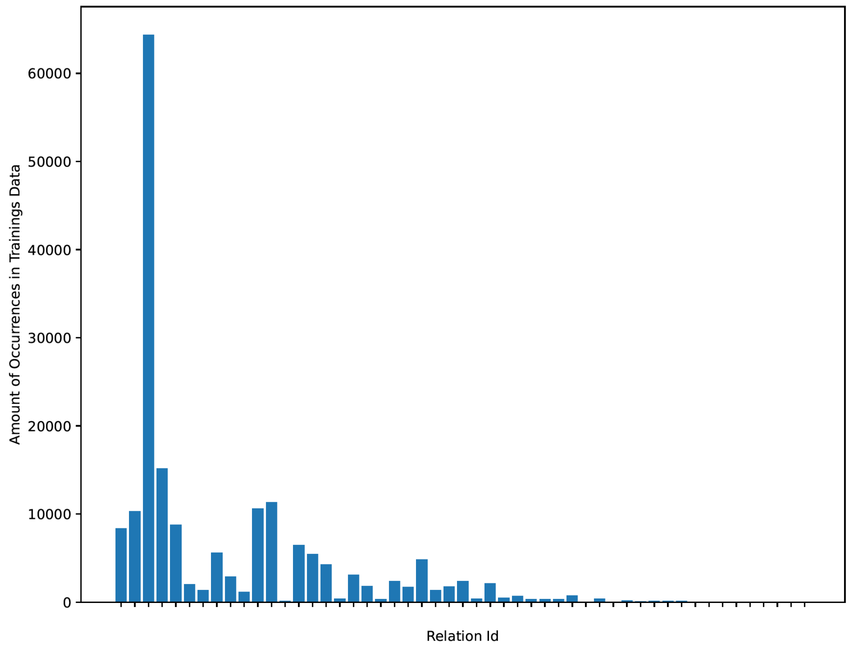


Figure 5.9: Relation frequency in trainings data for CoDEx-M

In figure 5.10 the relation frequency data is normalized between 0 and 1 overlayed on the per relation aggregated $better\_predicted\_by$ data, the same aggregation as in the previous section. Important to note before viewing this graph is that as we can see on the scale on the right the colouring of the bars is not evenly spread out along the range of the normalized frequency data. Instead I set the maximum of the color scale to the second most frequent relation. Because the most frequent relation is extremely dominant, the colors of the other bars would otherwise be almost indistinguishably.

In this graph AnyBURL and ComplEx are compared on CoDEx-M. We notice here that the six most frequent relations are all better predicted by ComplEx. While only a few somewhat frequent relations are better predicted by AnyBURL.

This observation has two limitations. For once there are only a handful of relations which can be counted as frequently occurring and furthermore the frequent relations better predicted by ComplEx all have an average $better\_predicted\_by$ value smaller than -0.25. This shows us that there is no strong correlation and we can only assume a slight connection between the relation frequency and the performance of both approaches.

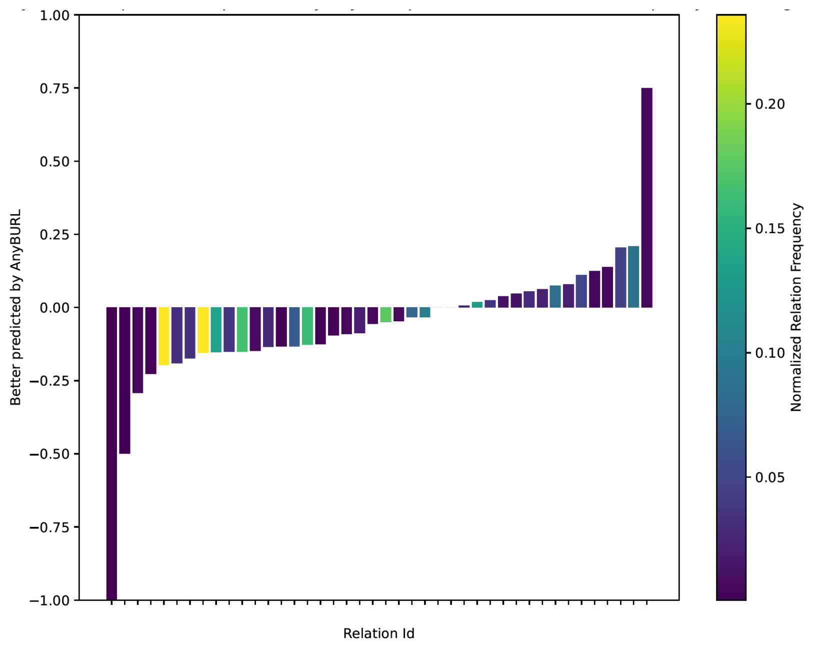


Figure 5.10: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the relation frequency

When plotting the same graph for ConvE and RESCAL, as can be seen in figure 5.11, we notice that here the most frequent relations are all better predicted by the embedding-based models than by AnyBURL. But the same limitations as before also hold.

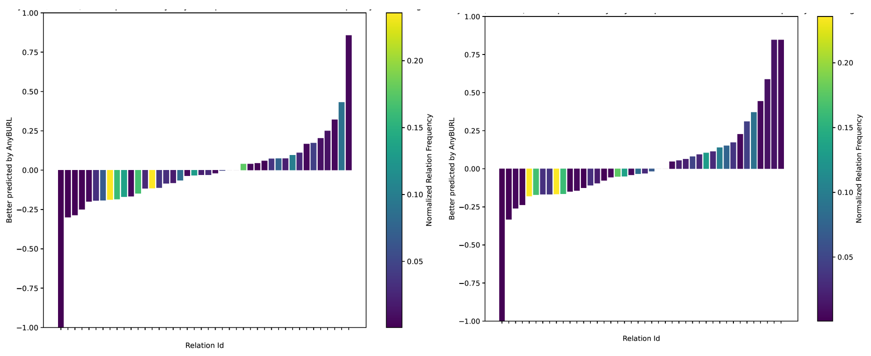


Figure 5.11: Comparison of AnyBURL and ConvE (left) RESCAL (right) on CoDEx-M in regard to the relation frequency

On FB15k-237 we can make similar observations. Figure 5.12 compares Any-BURL and ComplEx. Most frequent relations are in average all better predicted by ComplEx but only with the average being close to -0.25 which again indicates that there is no strong correlation but that there might possibly be a weaker connection between the relation frequency and the performance of the approaches. The same comparison on FB15k-237 with ConvE and RESCAL produces similar results as can be seen in appendix A.5.

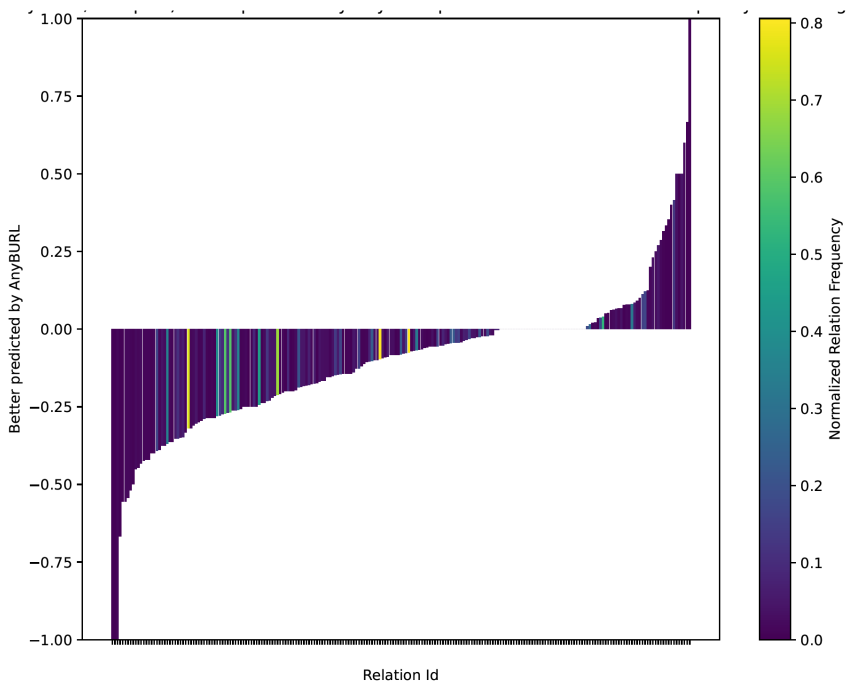


Figure 5.12: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the relation frequency

On YAGO3-10 two relations dominate in occurrence all others. In figure A.11 we see that the second and third relation occur both more than 300000 times in the trainings data while the next frequent relation occurs less than 100000 times and the ones after that significantly less often.

Figure 5.13 shows us that both of these relations are better predicted by ComplEx but the average $better\_predicted\_by$ value is so close to 0 that it can not be argued that ComplEx predicts relations frequently occurring in the trainings data better than AnyBURL on YAGO3-10.

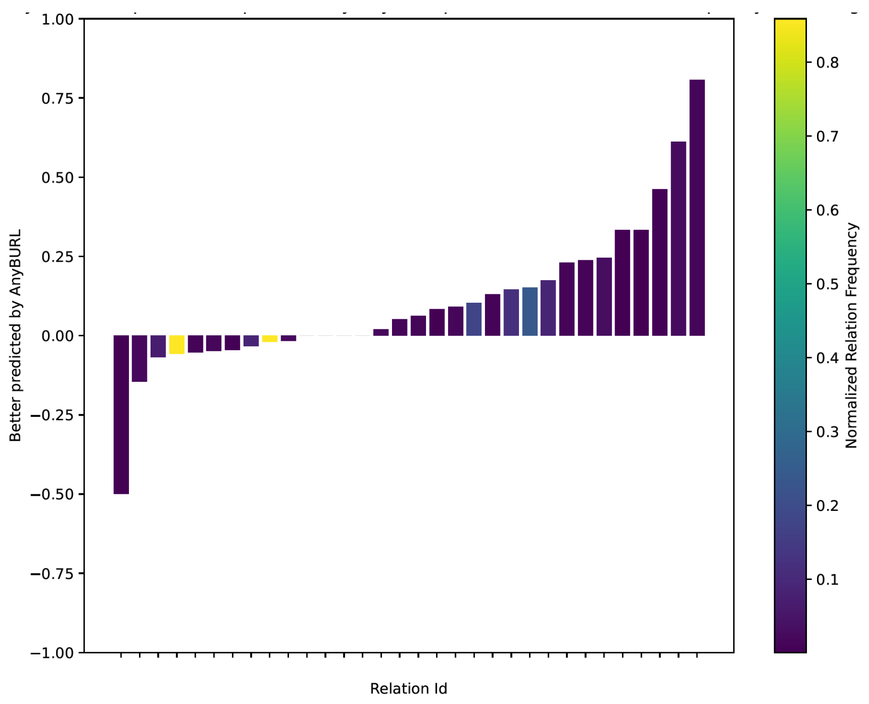


Figure 5.13: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the relation frequency

To sum up the results in regard to the relation frequency, we have now seen that on CoDEx-M and FB15k-237 triples with frequent relations tend towards being better predicted by the embedding-based models compared to AnyBURL. But this tendency is only small and furthermore this observation can not be made on YAGO3-10.

5.3.3 Entity Frequency in Trainings Data

Similar to the relations the entities are also not equally often represented in the datasets. For CoDEx-M we can see the entity distribution of the trainings data in figure 5.14 and for the other dataset in appendix A.6.

Therefore I also wanted to investigate whether the frequency of the entities in the trainings data influences which approach performs better.

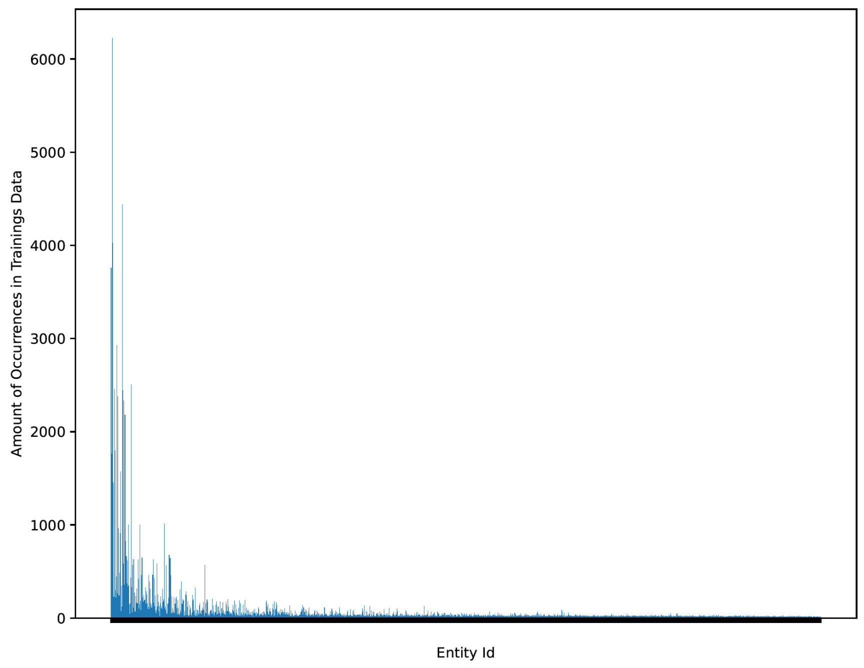


Figure 5.14: Entity frequency in trainings data for CoDEx-M

Different than when making the comparison in regard to relation frequency is that for the entity frequency there are always two values one frequency value for the head and one for the tail. In first attempts at creating this comparison I simply used either of these value and created one graph for each. Since the triples are predicted in both directions, the head/ tail is sometimes a part of the question and sometimes the answer. Therefore I found that it makes more sense to rephrase the question and ask whether the frequency of the question entity or the frequency of the answer entity influences which approach performs better. The question entity here is in case the tail is predicted the head and in case the head is predicted the tail. For the answer entity it is the other way around.

Another problem for creating this comparison is the amount of unique entities the datasets contain. Plotting a bar for each entity and colouring it like it was done for the relation frequency would result in an incomprehensible mess. Therefore I created ten normalized frequency bins in steps of 0.1 and calculated the average $better\_predicted\_by$ value for each bin.

Figure 5.15 plots this average and overlays the question entity frequency. Most of the bins average $better\_predicted\_by$ values are close to 0 with the only outliers being the $(0.7, 0.8]$ and $(0.5, 0.6]$ bins. These two slightly tend towards being better predicted by AnyBURL. But since the bins above these all tend towards ComplEx I would not say that this shows any pattern.

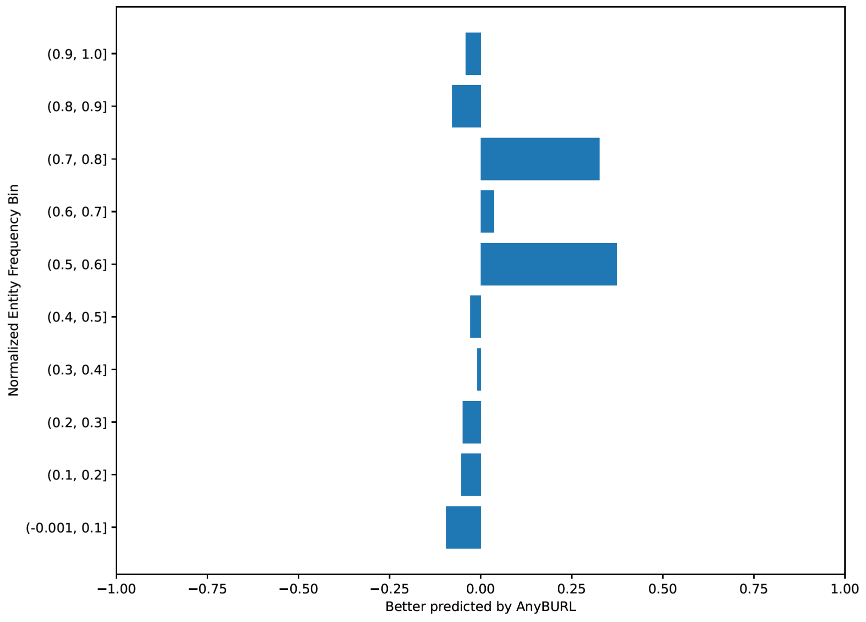


Figure 5.15: Comparison of AnyBURL and ComplEx on CodEx-M in regard to the question entity frequency

When plotting the same figure for the frequency of the answer entity, as can be seen in figure 5.16, we notice that all bins above 0.4 tend towards AnyBURL, while all below tend towards ComplEx. While this shows a slight tendency since the values are quiet low I would argue that this also does not express a clear pattern.

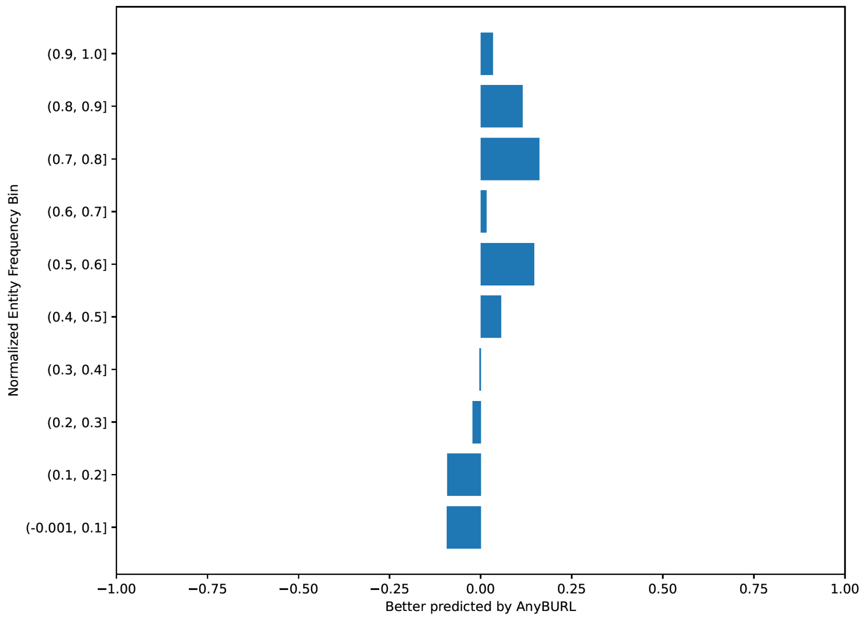


Figure 5.16: Comparison of AnyBURL and ComplEx on CodEx-M in regard to the answer entity frequency

In appendix A.7 the same figures can be found for the other model-dataset combinations. All these also do not show a clear pattern, especially not one overarching different embedding-based models and/ or datasets.

Even though I could not find a correlation between the model performances and the entity frequency, I still also wanted to test if it has an influence when all three parts of the triple are frequent. For that I added the two entity frequencies together and also included the relation frequency. I used the same bins as define before and the result of this can be seen in figure 5.17. While some bins tend towards either of the models there also is no clearly recognizable pattern. As can be seen in appendix A.7 the same holds for all other model-dataset combinations.

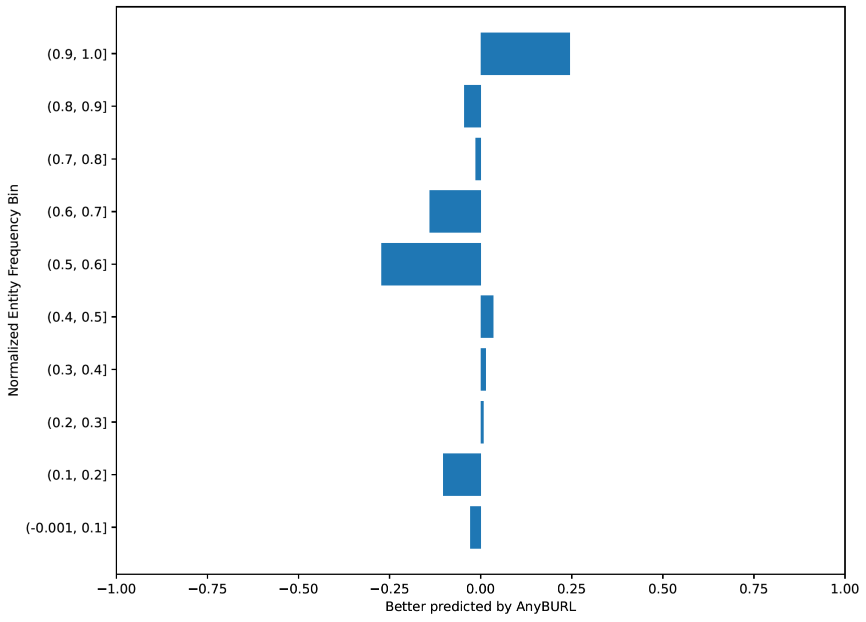


Figure 5.17: Comparison of AnyBURL and ComplEx on CodEx-M in regard to the combined entity and relation frequency

In this section I have now shown that I could not find any evidence that the frequency of the entities in the trainings data influences which approach predicts a triple better.

5.3.4 Existence of Similar Triples in the Trainings Data

Next up I wanted to investigate whether it is easier for one of the approaches to predict a triple if a similar triple already exists in the trainings dataset. For example if we have a test triple (h, r, t), does the trainings data contain a similar triple (h, r, t') when predicting the tail or (h', r, t) when predicting the head with $h \neq h'$ and $t \neq t'$.

To achieve that my code labels every triple from the test set with a binary variable expressing whether at least one similar triple exists or not.

In figure 5.18 expresses this information and compares AnyBURL and ComplEx on CoDEx-M where the $better\_predicted\_by$ value is the average of all triples belonging to the same binary class. As we can see the value averages are low values indicating that both models are almost equally good at predicting triples with or without similar triples in the trainings data. Interesting here is that ComplEx is slightly better in predicting triples when there is a similar triple and AnyBURL is better when there is non. What makes this interesting is that, as we can see in figure 5.19, ConvE and RESCAL create a similar result.

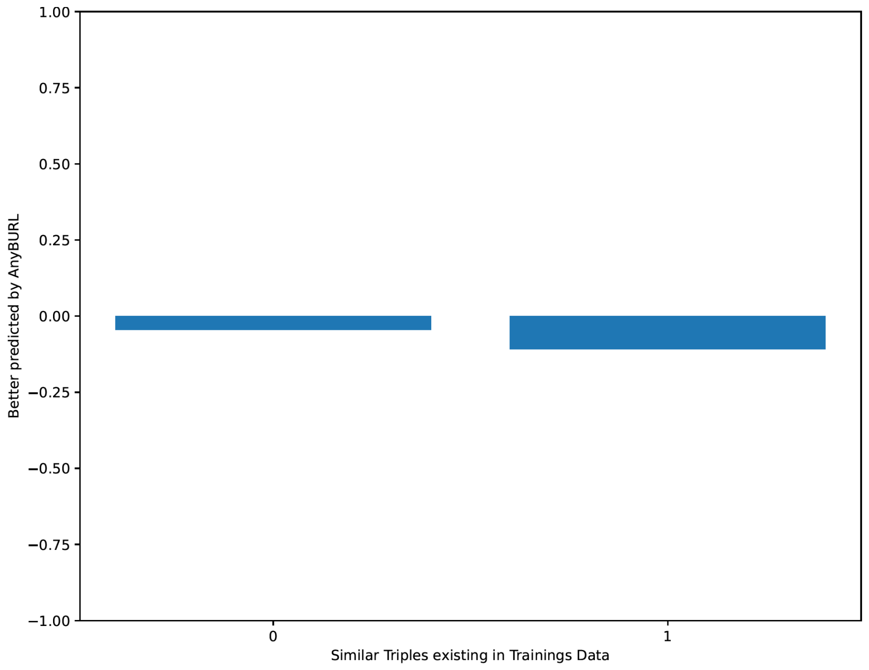


Figure 5.18: Comparison of AnyBURL and ComplEx on CodEx-M in regard to the existence of similar triples in the trainings data

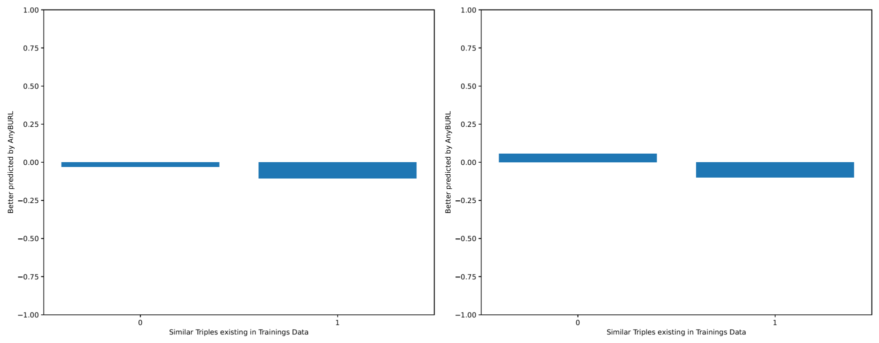


Figure 5.19: Comparison of AnyBURL and ConvE (left) / RESCAL (right) on CodEx-M in regard to the existence of similar triples in the trainings data

Plotting the same data for FB15k-237 and YAGO3-10 leads to comparable results. As can be seen in figure 5.20 comparing ComplEx and AnyBURL on FB15k-237 results in the same observations as before. Appendix A.8 shows that this also goes when comparing AnyBURL to ConvE and RESCAL on FB15k-237 and when comparing AnyBURL and ComplEx on YAGO3-10.

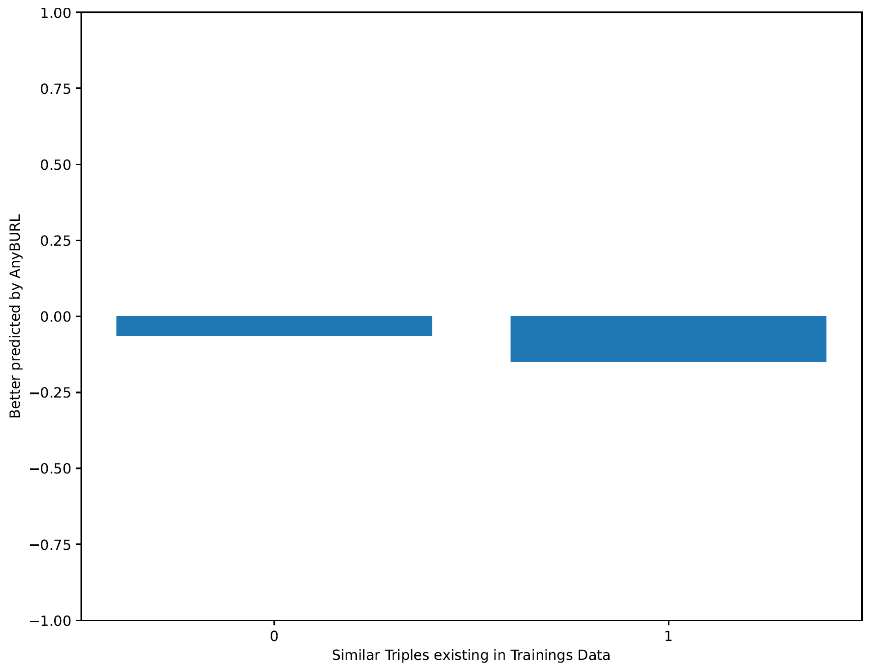


Figure 5.20: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar triples in the trainings data

As argued before having the averaged $better\_predicted\_by$ values this close to 0 shows that this is not a strong correlation but since all dataset-model combinations produce the same result, I still think that this shows that AnyBURL is slightly better in handling cases where no similar triple exists in the trainings data while the embedding-based models seem to be better in handling cases where such a triple exists.

Based on the previous observations I thought that maybe not only the existence of similar triples could have an influence but that the amount of similar triples might do as well.

In figure 5.21 triples are grouped based on the amount of similar triples existing in the trainings data. I would have expected for the $better\_predicted\_by$ value to continuously decrease. While there is a decrease initially for an higher amount of similar triples his pattern does not continue throughout figure with the value starting to increase again after more than 80 similar triples.

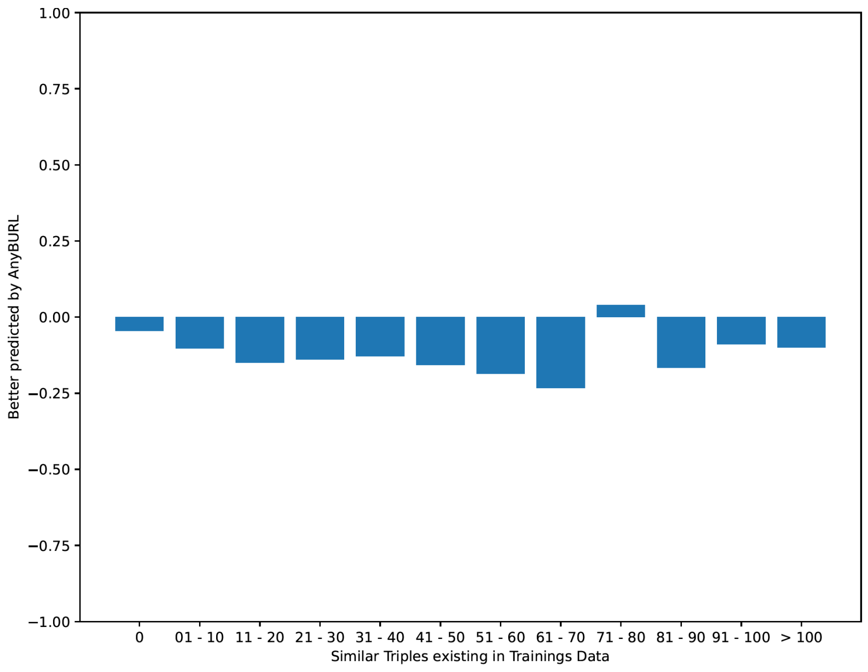


Figure 5.21: Comparison of AnyBURL and ComplEx on CodEx-M in regard to the amount of similar triples in the trainings data

If we now also consider the data from the other model-dataset combinations, as can be seen in appendix A.8, we notice that the average $better\_predicted\_by$ values seem to be more less random without a clear pattern, indicating that the amount seems to have no more influence than the general existence of similar triples.

This part has now shown us that the general existence of similar triples in the trainings data lightly benefits embeddings-based models compared to AnyBURL but the amount of similar triples does not play a larger role.

5.3.5 Existence of Similar Entities in the Dataset

The last characteristic I wanted to investigate is if the existence of similar entities in the dataset has an influence on which approach works better for a triple i.e. whether one approach is better in predicting a triple if for the triple (h, r, t) an entity exists which is similar to either h or t.

Based on the Similarity Score

To measure the similarity I tried two approaches. In my first one I calculated a cosine similarity matrix between all entity embeddings of the embedding-based model and summed up each row. That resulted in a similarity score for each entity which is higher for entities which have a lot of similar entities and lower for the ones that do not.

Figure 5.22 shows the per head entity averaged $better\_predicted\_by$ value with the entity similarity scores for AnyBURL and ComplEx on CoDEx-M. The entities with high similarity scores and the ones with low similarity scores are evenly distributed throughout the figure. Indicating that no model is better than the other in predicting entities with a lot or a few similar entities in the dataset.

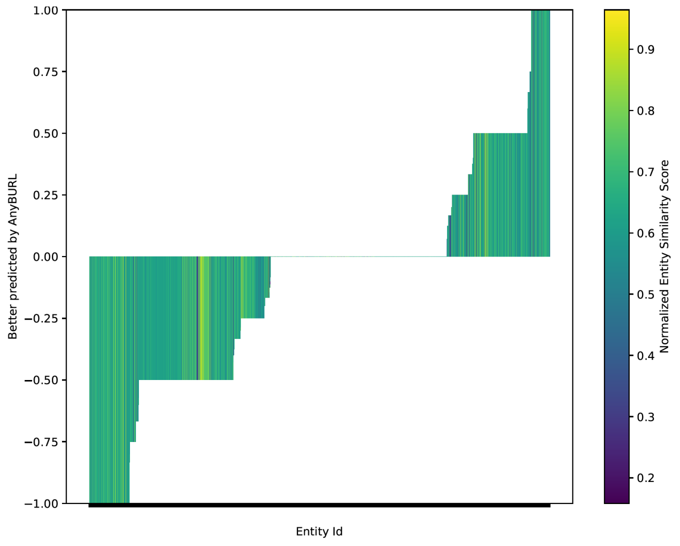


Figure 5.22: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on the cosine similarity

The same data displayed for the per tail entity averaged $better\_predicted\_by$ values, as can be seen in figure 5.23, has the high and low similarity scores stronger grouped together. The entities with low similarity scores appear mostly in the middle of the figure, indicating that both models predict them equally well, and the ones with a high similarity score have an higher/ lower averaged $better\_predicted\_by$ value showing that they are better predicted by one of the models. Since entities with high similarity appear on both sides we can not conclude that either model is better in predicting triples containing entities for which a lot of similar entities are included in the dataset.

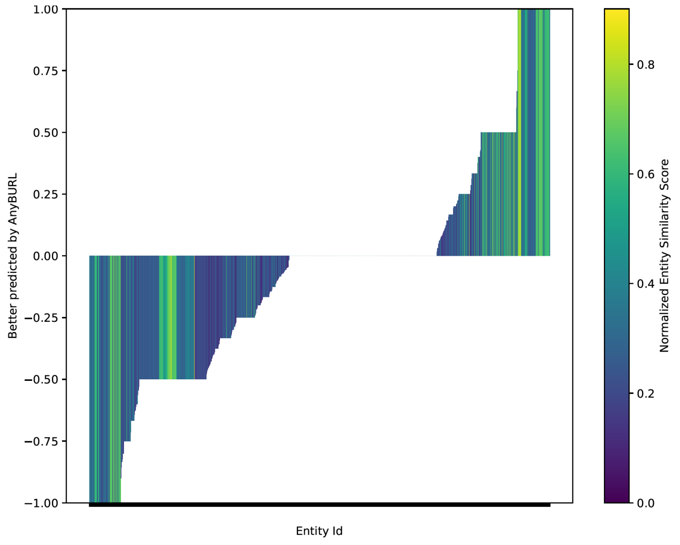


Figure 5.23: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on the cosine similarity

But it is still interesting that entities with high similarity scores are better predicted by one of the models. Maybe the models are learning specific groups better than the other model. To further investigate this I tried to cluster the entities in the next section.

Based on K-Means Clustering

To cluster the entities I applied K-means clustering onto the entity embeddings. I tried different cluster sizes and in the end a cluster size of $k = 100$ proved to create the comprehensible results.

Important to note here is that while I tried all embedding-based models to create clusters, the clusters in the upcoming figures are always the ones based on the ComplEx embeddings e.g. in figure **??** AnyBURL and ConvE are compared but the clusters are based on ComplEx. I made this decision to be better able to compare the clusters across different embedding-based models. Furthermore, we have the option of assigning a triple to a cluster either based on its head or tail entity. In the previous section we have seen that the similarity score works best if it is assigned based on the tail entity and the same goes for the clustering here. In appendix A.9 we can also see the clustering based on the heads but the results based on the tails are more useful and therefore in the following I will only address these.

In figure 5.24 we can see the averaged $better\_predicted\_by$ value per cluster for AnyBURL and ComplEx on CoDEx-M. There are clusters which are clearly better predicted by one of the models. While the largest clusters are tending towards a $better\_predicted\_by$ value of 0 the clusters tending towards either of the models still contain a few triples. It is an interesting result that some entities, which are in the eyes of the ComplEx embeddings similar, are clearly better predicted by one of the models.

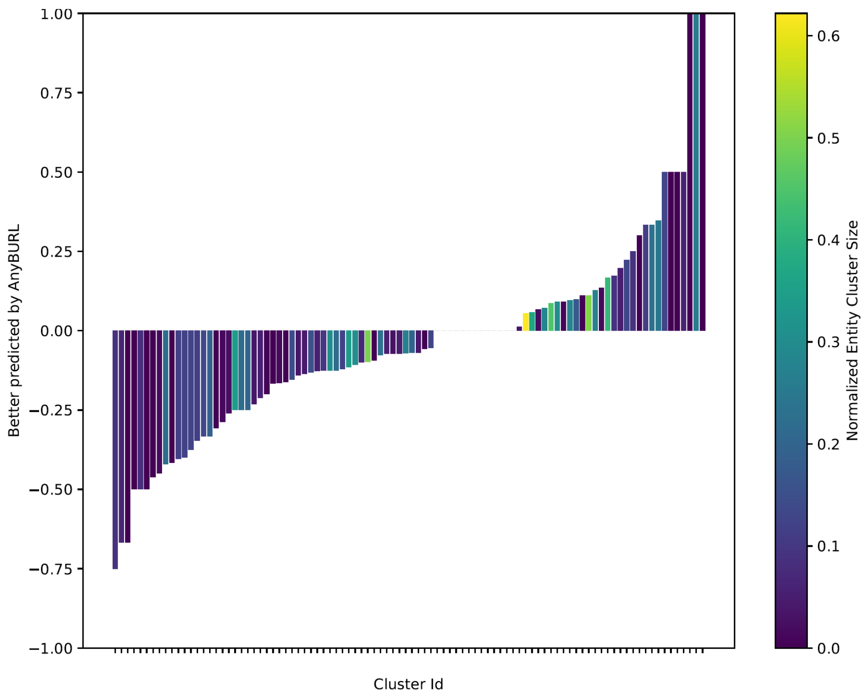


Figure 5.24: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

In figure 5.25 we can see the same comparison for AnyBURL and ConvE/ RESCAL on CoDEx-M. Here there are also clearly a few clusters better predicted by one of the models. What can not be seen in these figures is that the clusters tending towards either AnyBURL or one of the embedding-based models are the same clusters across all three comparisons. Showing that these clusters are a group of tails which are easier to predict for one of the approaches.

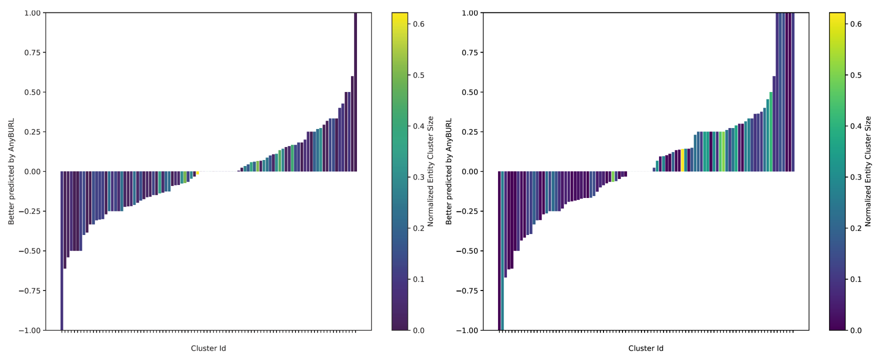


Figure 5.25: Comparison of AnyBURL and ConvE (left)/ RESCAL (right) on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

Doing the same analysis on FB15k-237 shows less expressive results than on CoDEx-M. While here some clusters tend more towards on of the models all averaged $better\_predicted\_by$ values are quiet low. This indicates that neither of the models predict a cluster significantly better than the other. This can be seen in figure 5.26 for AnyBURL and ComplEx and for the other models the same figure can be found in appendix A.9.

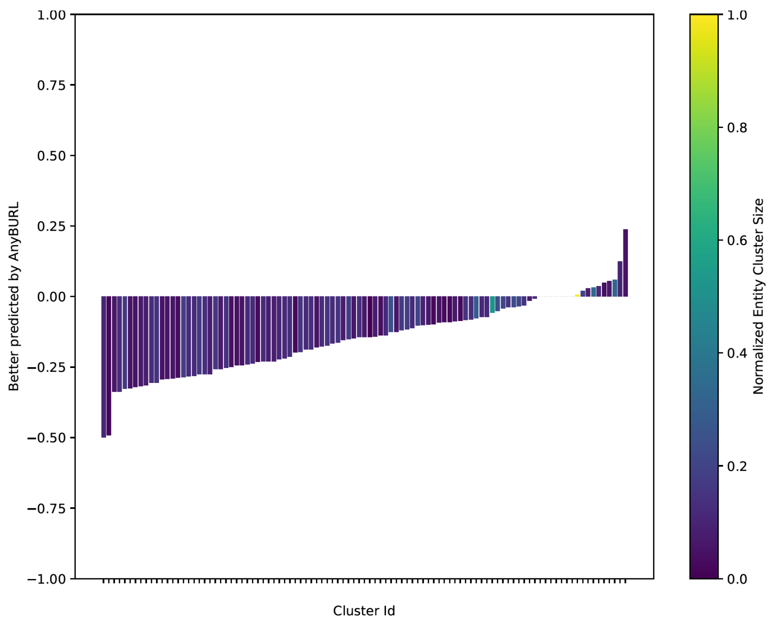


Figure 5.26: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

On YAGO3-10 on the other hand clusters were found which can be clearly assigned to one of the approaches. In figure 5.27 we can see that different from the result from CoDEx-M that for YAGO3-10 there larger clusters clearly better predicted by one of the approaches. For example the largest cluster has an average $better\_predicted\_by$ of above 0.95 and is therefore clearly better predicted by AnyBURL.

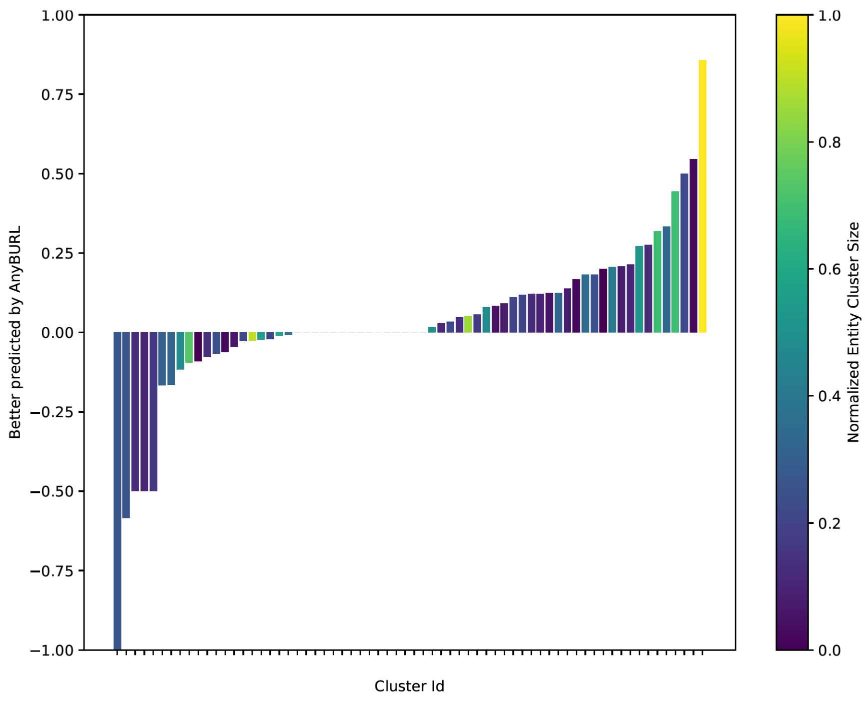


Figure 5.27: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

Chapter 6

Test Sets for Vulnerabilities

In this chapter the second research question I will try answer the second research question. Based on the result from chapter 5 I decided to create eight testsets as subsets of the original test set. In the following I will first present the test sets, explain what kind of triples they contain and afterwards I will show the MRR the different models achieved on these sets.

1-1 relations test set. This test set only contains triples with relations belonging to the *1-1* class. In the last chapter was saw that AnyBURL performed better on these kind of relations and i therefore expect it to achieve an higher MRR.

Mult-cardinal relations test set. Here all other relations appear. In that we have seen the embedding-based models predicted more triples from the other classes better and I therefore expect these models to achieve an higher MRR here.

25% least frequent relations test set. This test set includes all triples containing one of the 25% least frequently occurring relations. Here I expect all models to perform quiet similar.

10% most frequent relations test set. In this test set only triples containing one of the 10% most frequent relations appear in this set. I decided to only use 10% instead of 25% as before since the least frequent set would otherwise have been significantly smaller. Based on the result from the previous chapter I expect the embedding-based models to perform slightly better.

No similar triples in the trainings data test set. This test set only contains triples for which no similar triple, as defined in section 5.3.4, exists. Our previous data has shown that these triples are sometimes better predicted by AnyBURL therefore I expect it to achieve a slightly better MRR.

At least one similar triple in the trainings data test set. Here the triples occur for which the trainings data contains at least one similar triple. The embedding-based models should be better on this set.

AnyBURL cluster test set. This test set was created from the top-10 clusters better predicted by AnyBURL predicted better than ComplEx, therefore AnyBURL should also achieve a higher MRR.

ComplEx cluster test set. The last test set is the opposite of the previous test set, here only triples occur which were in the top-10 clusters better predicted by ComplEx. Here I expect the embedding-based models to perform better.

In table 6.1 the hits@1, hits@10 and MRR is shown for the two test sets about the relation classes. For CoDEx-M everything is as expected. AnyBURL outperforms all embedding-based models on the *1-1* relations test set with RESCAL being significantly worse than any other model and on the multi-cardinal relations test set AnyBURL performs slightly worse than the embedding-based models. This completely mirrors the observations we made in chapter 5 about the relation classes on CoDEx-M.

On FB15k-237 on the other hand the best performing model for the *1-1* relations test set was ComplEx - AnyBURL still outperformed the other embedding-based models. Previously it was stated that the relation class effect is not as strong on FB15k-237 as on CoDEx-M, this showed especially with ComplEx as the comparison model. For the multi-cardinal relations test set the results where as expected with AnyBURL achieving the worst scores.

YAGO3-10 was already an outlier in regard to the relation classes in the last chapter and here again we can see that the results differ from the previously mentioned datasets. ComplEx outperforms AnyBURL here in both sets.

I also included WN18RR here to see if my results are transferable. Both AnyBURL and ComplEx achieved quiet similar scores and therefore the results for the relation class test sets are not transferable.

	1-1 relations			Multi-cardinal relations		
Model	h@1 (filtered)	h@10 (filtered)	MRR (filtered	h@1 (filtered)	h@10 (filtered)	MRR (filtered
CoDEx-M						
AnyBURL	0.731	0.753	0.729	0.309	0.409	0.281
ComplEx	0.651	0.720	0.674	0.341	0.474	0.332
ConvE	0.532	0.715	0.604	0.231	0.454	0.308
RESCAL	0.172	0.500	0.282	0.242	0.453	0.315
FB15k-237						
AnyBURL	0.479	0.539	0.462	0.308	0.443	0.283
ComplEx	0.430	0.604	0.485	0.252	0.533	0.345
ConvE	0.245	0.505	0.340	0.248	0.521	0.339
RESCAL	0.401	0.521	0.445	0.264	0.539	0.355
YAGO3-10						
AnyBURL	0.783	0.833	0.770	0.558	0.627	0.523
ComplEx	0.800	0.850	0.823	0.466	0.673	0.540
WN18RR						
AnyBURL	0.976	0.976	0.976	0.488	0.547	0.473
ComplEx	0.964	0.976	0.970	0.429	0.539	0.467

Table 6.1: Hits@1, Hits@10 and MRR on relation class test sets

For the relation frequency test sets AnyBURL was on CoDEx-M and FB15k-237 in both categories worse than the embedding-based models as can be seen in table 6.2. For CoDEx-M the gap here is larger on the most frequent relations test set, this fits the observations from the previous chapter where the frequent relations had a slight tendency towards the embedding-based models while the infrequent ones where distributed more or less equally between both approaches. The results from FB15k-237 showed an equal gap in performance for both test sets.

On YAGO3-10 we receive similar results as on CoDEx-M.

When trying out these test subsets on WN18RR we notice that AnyBURL is significantly better on the least frequent relations test set while performing similar to ComplEx on the most frequent relations test set.

	25% least frequent relations			25% most frequent relations		
Model	h@1 (filtered)	h@10 (filtered)	MRR (filtered	h@1 (filtered)	h@10 (filtered)	MRR (filtered
CoDEx-M						
AnyBURL	0.167	0.233	0.145	0.318	0.412	0.288
ComplEx	0.133	0.400	0.237	0.272	0.476	0.343
ConvE	0.133	0.333	0.184	0.256	0.460	0.329
RESCAL	0.100	0.267	0.167	0.273	0.468	0.341
FB15k-237						
AnyBURL	0.475	0.601	0.423	0.226	0.371	0.206
ComplEx	0.420	0.678	0.506	0.182	0.502	0.287
ConvE	0.409	0.669	0.497	0.179	0.493	0.281
RESCAL	0.423	0.655	0.501	0.195	0.514	0.299
YAGO3-10						
AnyBURL	0.225	0.325	0.235	0.642	0.754	0.623
ComplEx	0.200	0.375	0.261	0.605	0.833	0.688
WN18RR						
AnyBURL	0.387	0.519	0.389	0.170	0.249	0.152
ComplEx	0.198	0.283	0.231	0.111	0.246	0.156

Table 6.2: Hits@1, Hits@10 and MRR on relation frequency test sets

Table 6.3 shows us the results for the similar triples test sets. When there is no triple AnyBURL performs similar to ConvE and RESCAL on CoDEx-M and FB15k-237 while still getting outperformed by ComplEx. For the test set including only triples with similar triples in the trainings data the embedding-based models perform best. Interesting to note here is that in all cases AnyBURL achieves a significant higher hits@1 than the other models. YAGO3-10 achieves comparable results, on the test set without similar triples AnyBURL performs slightly better than ComplEx and slightly worse on the other test set.

Another interesting observation made when creating the test subsets is that WN18RR does not include any triples in its test set which are similar to any triples in the trainings data. Therefore WN18RR could not be evaluated with this test subset. Furthermore, this might explain why WN18RR is the only dataset for which AnyBURL reports better overall results than ComplEx. Embedding-based models seem to be good in predicting triples for which similar triples exist in the trainings data but this advantage does not apply here since such triples do not exist.

	No similar triples in the trainings data			At least one similar triple in the trainings data		
Model	h@1 (filtered)	h@10 (filtered)	MRR (filtered	h@1 (filtered)	h@10 (filtered)	MRR (filtered
CoDEx-M						
AnyBURL	0.350	0.441	0.320	0.310	0.410	0.282
ComplEx	0.321	0.512	0.387	0.259	0.475	0.333
ConvE	0.274	0.442	0.334	0.243	0.455	0.309
RESCAL	0.276	0.453	0.337	0.243	0.455	0.317
FB15k-237						
AnyBURL	0.379	0.462	0.355	0.310	0.445	0.285
ComplEx	0.330	0.508	0.393	0.253	0.537	0.347
ConvE	0.323	0.485	0.380	0.250	0.525	0.341
RESCAL	0.336	0.509	0.397	0.266	0.544	0.358
YAGO3-10						
AnyBURL	0.320	0.361	0.303	0.561	0.630	0.526
ComplEx	0.260	0.356	0.296	0.469	0.677	0.544
WN18RR						
AnyBURL	—	—	—	—	—	—
ComplEx	—	—	—	—	—	—

Table 6.3: Hits@1, Hits@10 and MRR on similar triples test sets

Lastly for the entity cluster test sets AnyBURL performed always best on the AnyBURL cluster subset and the embedding-based models performed best on the ComplEx cluster subset. Since these datasets were created based on which model predicted which entity best this is no surprise. But it still proves that there are entities for which AnyBURL is the better model and there are entities where the embedding-based models were most successful. Sadly I could not find any pattern between those entities.

	AnyBURL cluster			ComplEx cluster		
Model	h@1 (filtered)	h@10 (filtered)	MRR (filtered	h@1 (filtered)	h@10 (filtered)	MRR (filtered
CoDEx-M						
AnyBURL	0.184	0.276	0.182	0.146	0.232	0.128
ComplEx	0.066	0.289	0.119	0.199	0.459	0.284
ConvE	0.066	0.211	0.109	0.180	0.456	0.269
RESCAL	0.079	0.184	0.119	0.191	0.453	0.278
FB15k-237						
AnyBURL	0.419	0.562	0.379	0.105	0.240	0.107
ComplEx	0.348	0.582	0.423	0.113	0.443	0.218
ConvE	0.308	0.567	0.395	0.112	0.427	0.213
RESCAL	0.348	0.568	0.425	0.140	0.461	0.242
YAGO3-10						
AnyBURL	0.246	0.290	0.217	0.294	0.361	0.271
ComplEx	0.149	0.273	0.193	0.296	0.483	0.366
WN18RR						
AnyBURL	0.474	0.551	0.457	0.285	0.449	0.296
ComplEx	0.357	0.491	0.403	0.304	0.627	0.415

Table 6.4: Hits@1, Hits@10 and MRR on entity cluster test sets

Chapter 7

Research Limitations and Further Research

The previous shown research has a few limitations which I want to list here and address how the research could be improved in the future.

The first research limitation I want to mention is that only one model was used to represent the symbolic approach. While for the sub-symbolic approach three different models were tested to show that the results can be generalized, the same was not done for the symbolic approach, here only AnyBURL was used. A next step here would be to replicate the experiments with other symbolic models like AMIE [8] or RLvLR [9].

In chapter 5 the analysis was done on the $better\_predicted\_by$ variable. This variable only classifies a triple as either being better predicted by one of the two compared models or as being equally good predicted. This leads to two limitations. For once it does not include the gap between the ranks of both models. We have seen in many cases that AnyBURL achieves good hits@1 often outperforming the embedding-based models in this metric but in hits@10 the embedding-based models already achieve higher scores. This leads me to believe that AnyBURL might be partly better in predicting the solution but not as good as the embedding-based models when it comes to proposing alternative solutions which might be true in case the triple at the first rank is not. To analyse this it could be interesting to extend the comparison by including the rank differences between the two models.

The second limitation coming from the $better\_predicted\_by$ variable is that since it is created from the difference-ψ value it is a relative measure. Meaning it compares only the difference and does not regard the general ranking. For example it might be that there is the same difference for a triple where one model predicted rank $\#100$ and the other rank $\#1000$ and for a triple where one predicted rank $\#1$

and the other rank $\#25$. Here it would be interesting to separate the cases where both models where completely wrong like in the first example and the cases where at least one model was correct or at least close to being correct.

Another point which could be expanded in the future is to analyse why there are clusters of entities which can be clearly better predicted by one of the approaches.

Chapter 8

Conclusion

The main research question of this thesis was if it is possible to find characteristics of triples for which subsets of the dataset exist, where either the symbolic or the sub-symbolic approach outperforms the other. The second research question if the findings produced by the first question could be utilized to create test subsets to analyse model for specific vulnerabilities.

To answer this question five characteristics where analysed.

First the relation classes where investigated. Here it was shown that for CoDEx-M and FB15k-237 the symbolic approach worked better on $1-1$ relations and the sub-symbolic approach on relations with a multi-cardinality. For YAGO3-10 this observation could not be made, here all models performed equally. When creating the test sets the result for CoDEx were validated while for FB15k-237 the data for ConvE and RESCAL also support the findings but ComplEx outperformed AnyBURL on all relation classes and therefore generally saying that symbolic approaches are better on $1-1$ relations is wrong but the data still showed that the relation classes have a small impact on how the approaches compare. For other two datasets a pattern between the relation classes and the model performances could not be found.

The second characteristic analysed was the frequency of the relations in the trainings data. The observation here made was that the sub-symbolic approach is better in predicting frequent relations while the infrequent ones were equally predicted by both approaches. Through the creation of the test set it was shown that this observation holds on CoDEx-M, YAGO3-10 and especially on WN18RR. On FB15k-237 the metrics did not seem to be influenced by the relation frequency.

A similar analysis was done for the frequency of the entities in the trainings data. Here was saw no influence.

After that was analysed whether the existence of similar triples in the trainings data has an influence. While the amount of similar triples did not seem to have a significant influence the general existence did. The creation of the test subsets support this hypothesis. WN18RR here is exempted while since it does not contain any triples in its test set for which similar triples exist in the trainings data.

Lastly the influence of the existence of similar entities in the dataset was investigated. While no approach generally performed better on triples containing entities with similar entities in the dataset, clusters of entities could be found where one approach performed significantly better than the other. Interesting here was that the clusters for the three sub-symbolic models were all quiet similar and that even though the test subset was only created based on the cluster from ComplEx, ConvE and RESCAL achieved similar scores. Furthermore, to no surprise each approach worked best on the subset created from its models.

To sum up five characteristics were analysed. For the relation class and frequency a slight influence was found in regard to the performance comparison of the two approaches which differs in significance from dataset to dataset. The entity frequency did not show to have any influence and the existence of similar triples in the dataset was shown to have a clear influence. The existence of similar entities also showed no influence but clusters of entities were found where one approach performed better than the other. In these clusters I was not able to identify a further pattern.

Bibliography

[1] Matthias Baumgartner, Daniele Dell'Aglio, and Abraham Bernstein. Entity Prediction in Knowledge Graphs with Joint Embeddings. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 22–31. Association for Computational Linguistics, June 2021.

[2] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge Graph Embeddings and Explainable AI. *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, 47:49, April 2020.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. *Advances in Neural Information Processing Systems*, 26, 2013.

[4] Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. LibKGE - A knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 165–174. Association for Computational Linguistics, 2020.

[5] Ashok K. Chandra and David Harel. Horn clause queries and generalizations. *The Journal of Logic Programming*, 2(1):1–15, April 1985.

[6] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge Graph Completion: A Review. *IEEE Access*, 8:192435–192456, 2020.

[7] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, pages 1811–1818. AAAI, July 2018.

[8] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. ACM Press, 2013.

[9] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. Scalable Rule Learning via Learning Representation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2149–2155, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization.

[10] Jennifer Golbeck. *Analyzing the Social Web*, volume 1. Newnes, February 2013.

[11] Ioannis Hatzilygeroudis and Jim Prentzas. Neuro-Symbolic Approaches for Knowledge Representation in Expert Systems. *International Journal of Hybrid Intelligent Systems*, 1(3-4):111–126, January 2005.

[12] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*, 12(2):1–257, September 2021.

[13] Eleni Ilkou and Maria Koutraki. Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies? In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, November 2020.

[14] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Proceedings of the 7th biennial conference on innovative data systems research*, 2015.

[15] Christian Meilicke, Patrick Betz, and Heiner Stuckenschmidt. Why a Naive Way to Combine Symbolic and Latent Knowledge Base Completion Works Surprisingly Well. In *Proceedings of the 3rd Conference on Automated Knowledge Base Construction*, 2021.

[16] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion, April 2020.

[17] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3137–3143. International Joint Conferences on Artificial Intelligence Organization, August 2019.

[18] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[19] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the The 28th International Conference on Machine Learning*, 2011.

[20] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280, April 2012.

[21] Simon Ott, Christian Meilicke, and Matthias Samwald. SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models. *arXiv preprint arXiv:2109.08002*, September 2021.

[22] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, December 2016.

[23] Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark, October 2020.

[24] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end Structure-Aware Convolutional Networks for Knowledge Base Completion, November 2018. arXiv:1811.04441 [cs].

[25] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[26] John Talburt. *Entity Resolution and Information Quality*. Elsevier, January 2011.

[27] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, July 2015.

[28] Theo Trouillon, Johannes Welbl, and Sebastian Riedel. Complex Embeddings for Simple Link Prediction. In *Proceedings of the International conference on machine learning*, pages 2071–2080, 2016.

[29] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, December 2017.

[30] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the 3rd International Conference on Learning Representations*, August 2015.

[31] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*, 29(32):4790–4797, November 1990.

[32] Xiaohan Zou. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series*, 1487(1):012016, March 2020.

Appendix A

Further Comparison Figures & Tables

A.1 Metrics of all trained Models

	CoDEx-M			**FB15k-237**		
Model	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**
AnyBURL	0.313	0.412	0.285	0.238	0.501	0.322
ComplEx (1)	0.261	0.475	0.334	0.253	0.534	0.346
ComplEx (2)	0.262	0.476	0.336	0.251	0.532	0.345
ComplEx (3)	0.260	0.479	0.335	0.253	0.534	0.345
ComplEx (4)	0.262	0.474	0.335	0.253	0.534	0.346
ComplEx (5)	0.258	0.472	0.331	0.254	0.537	0.348
ConvE (1)	0.231	0.457	0.309	0.246	0.522	0.337
ConvE (2)	0.234	0.460	0.312	0.246	0.520	0.338
ConvE (3)	0.235	0.456	0.311	0.248	0.522	0.338
ConvE (4)	0.233	0.456	0.310	0.246	0.522	0.338
ConvE (5)	0.233	0.456	0.310	0.248	0.521	0.338
RESCAL (1)	0.241	0.453	0.314	0.264	0.539	0.356
RESCAL (2)	0.248	0.456	0.319	0.262	0.532	0.351
RESCAL (3)	0.240	0.454	0.313	0.264	0.539	0.356
RESCAL (4)	0.245	0.455	0.318	0.266	0.540	0.358
RESCAL (5)	0.246	0.458	0.319	0.266	0.540	0.358

Table A.1: Hits@1, Hits@10 and MRR of all models trained for CoDex-M and FB15k-237

	WN18RR			**YAGO3-10**		
Model	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**	**h@1 (filtered)**	**h@10 (filtered)**	**MRR (filtered)**
AnyBURL	0.495	0.553	0.479	0.559	0.629	0.525
ComplEx (1)	0.438	0.544	0.474	0.464	0.672	0.540
ComplEx (2)	0.440	0.546	0.476	0.467	0.674	0.541
ComplEx (3)	0.439	0.542	0.474	0.473	0.676	0.545
ComplEx (4)	0.436	0.542	0.472	0.455	0.670	0.533
ComplEx (5)	0.436	0.545	0.473	0.459	0.671	0.534
ConvE (1)	-	-	-	-	-	-
ConvE (2)	-	-	-	-	-	-
ConvE (3)	-	-	-	-	-	-
ConvE (4)	-	-	-	-	-	-
ConvE (5)	-	-	-	-	-	-
RESCAL (1)	-	-	-	-	-	-
RESCAL (2)	-	-	-	-	-	-
RESCAL (3)	-	-	-	-	-	-
RESCAL (4)	-	-	-	-	-	-
RESCAL (5)	-	-	-	-	-	-

Table A.2: Hits@1, Hits@10 and MRR of all models trained for WN18RR and YAGO3-10

A.2 Comparison of Rankings for all Models and Datasets

A.2.1 CoDEx-M

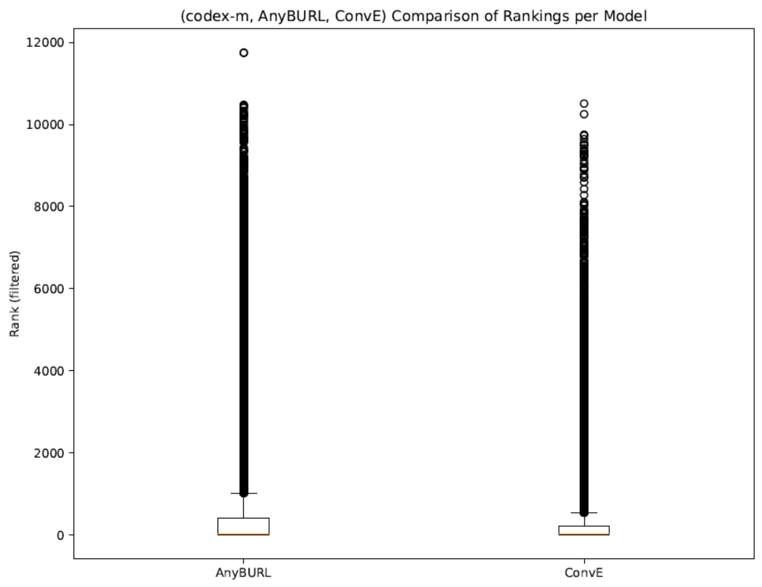


Figure A.1: Comparison of Ranks predicted by AnyBURL and ConvE for CoDEx-M

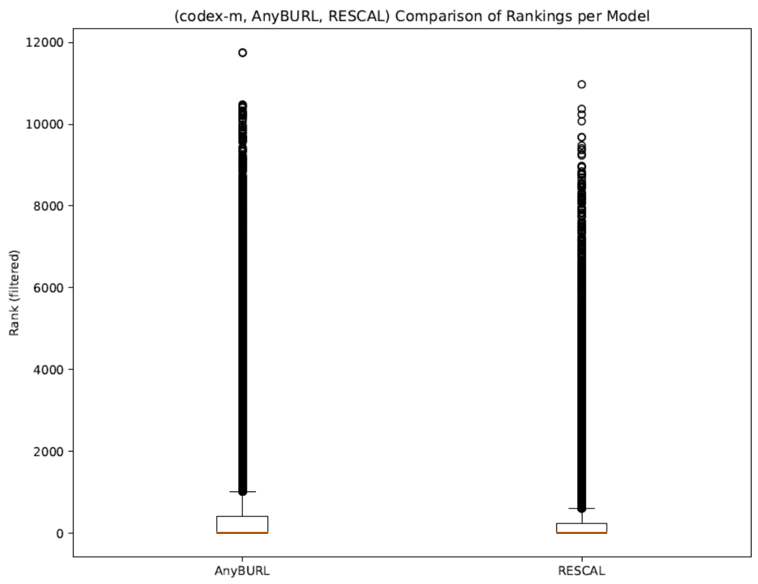


Figure A.2: Comparison of Ranks predicted by AnyBURL and RESCAL for CoDEx-M

A.2.2 FB15k-237

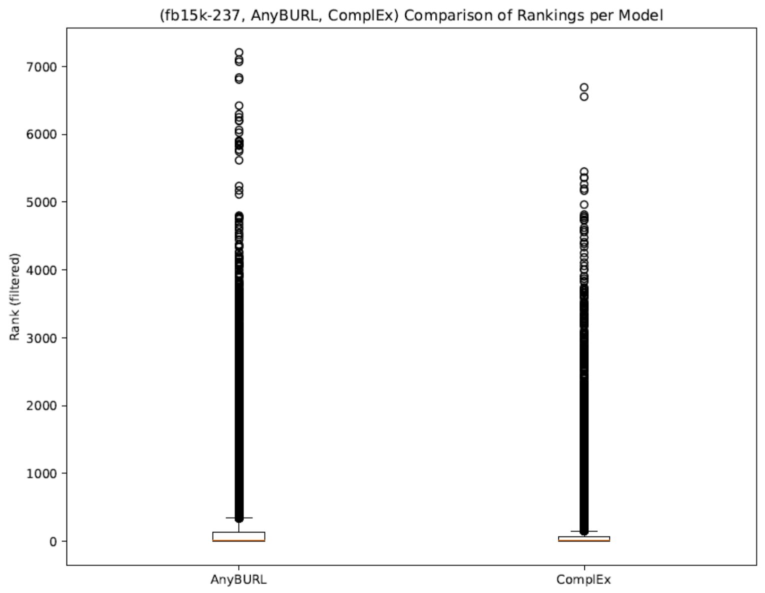


Figure A.3: Comparison of Ranks predicted by AnyBURL and ComplEx for FB15k-237

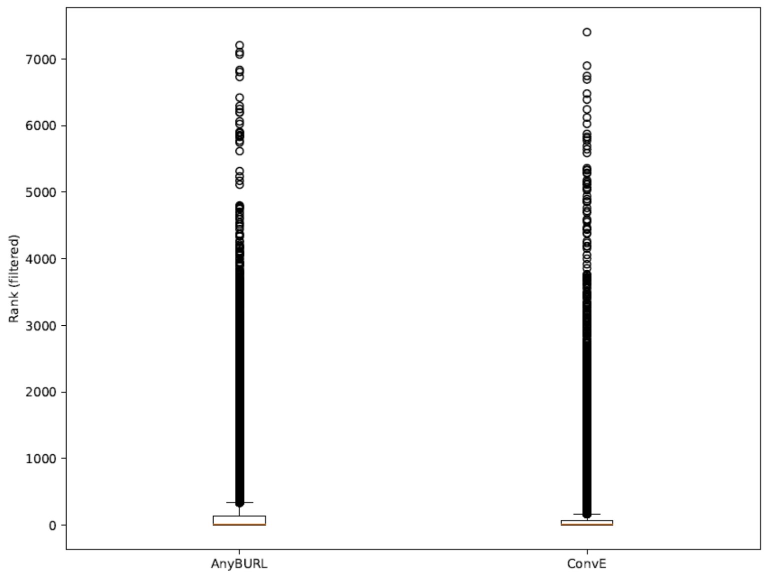


Figure A.4: Comparison of Ranks predicted by AnyBURL and ConvE for FB15k-237

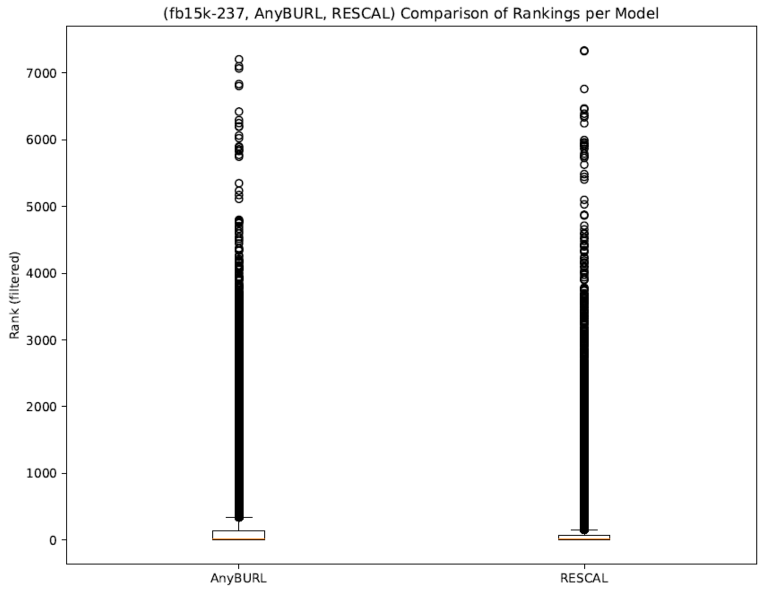


Figure A.5: Comparison of Ranks predicted by AnyBURL and RESCAL for FB15k-237

A.2.3 YAGO3-10

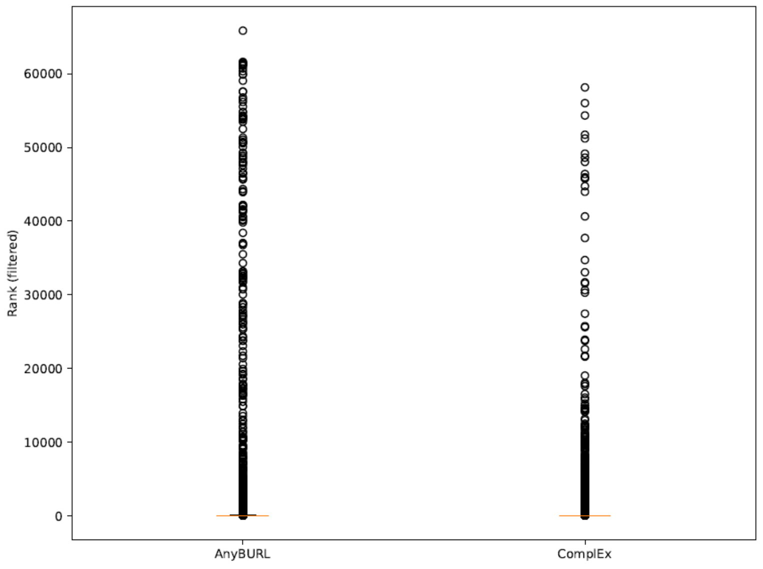


Figure A.6: Comparison of Ranks predicted by AnyBURL and ComplEx for YAGO3-10

A.3 Comparison in regard to the Relation Classes

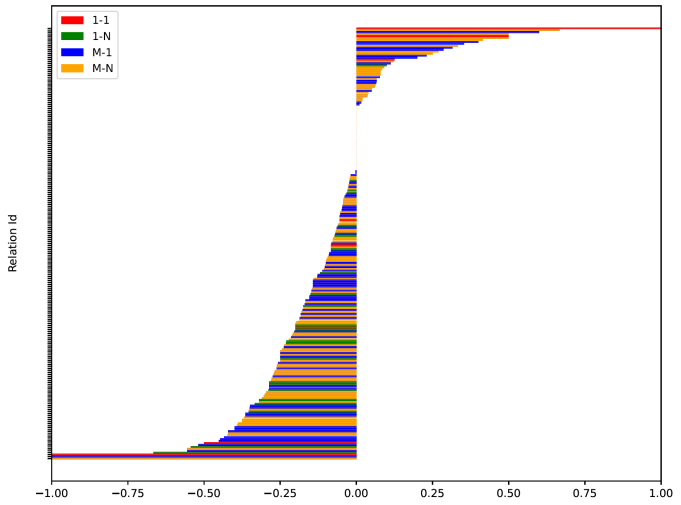


Figure A.7: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the relation classes

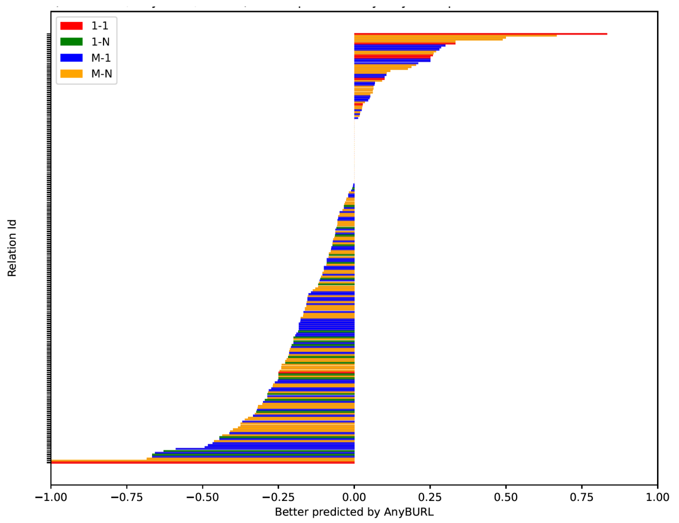


Figure A.8: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation classes

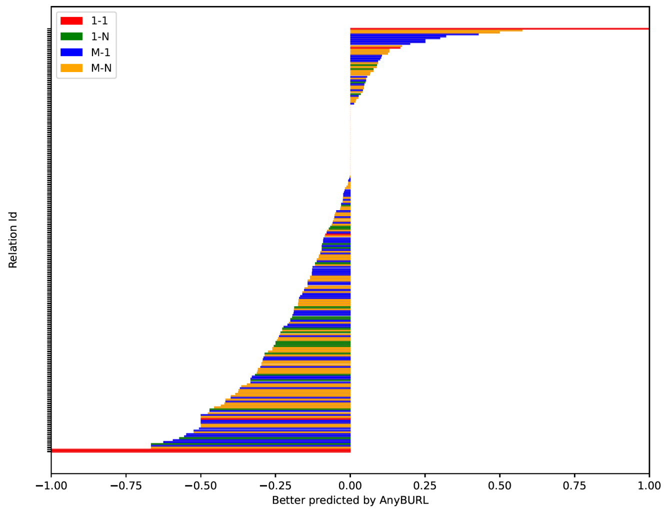


Figure A.9: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the relation classes

A.4 Relation Frequency in Trainings Data

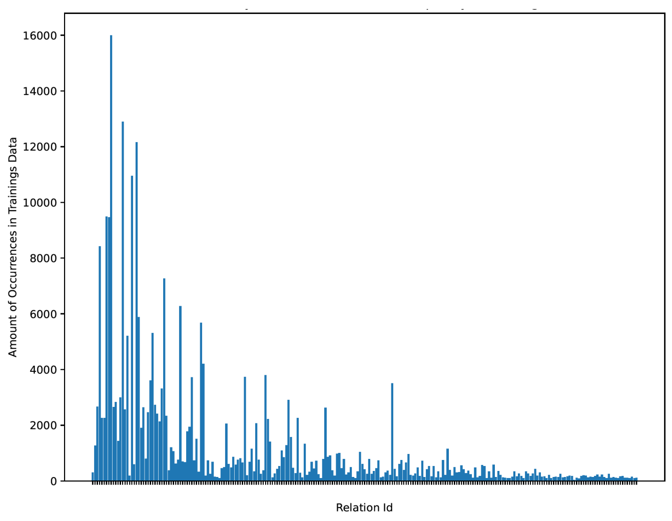


Figure A.10: Relation frequency in trainings data for FB15k-237

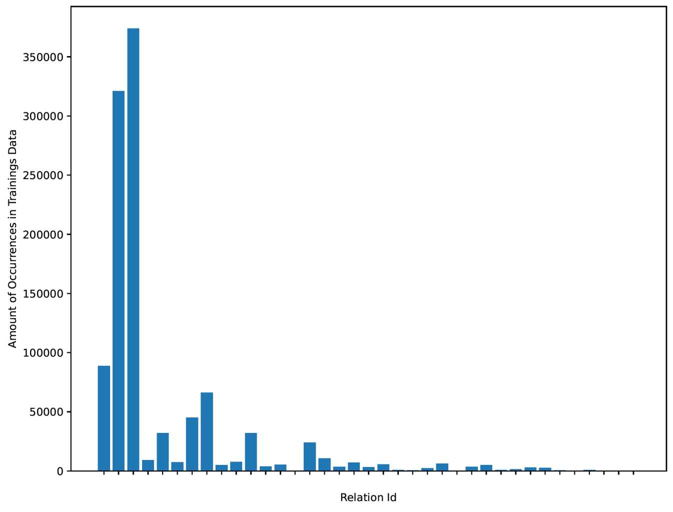


Figure A.11: Relation frequency in trainings data for YAGO3-10

A.5 Comparison in regard to the Relation Frequency

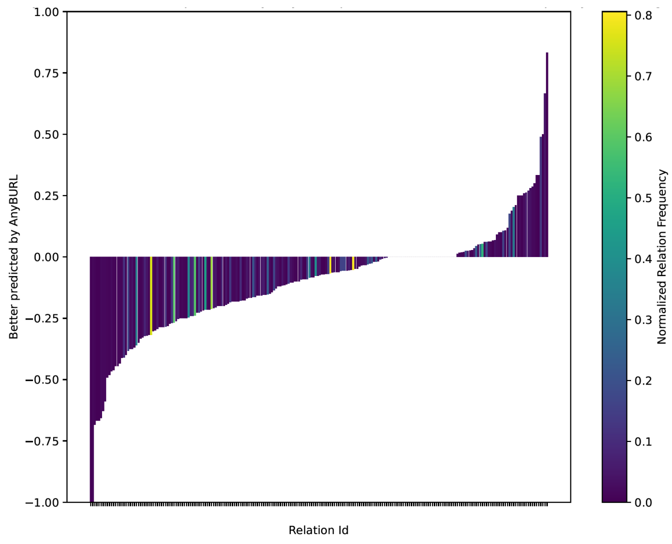


Figure A.12: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation frequency

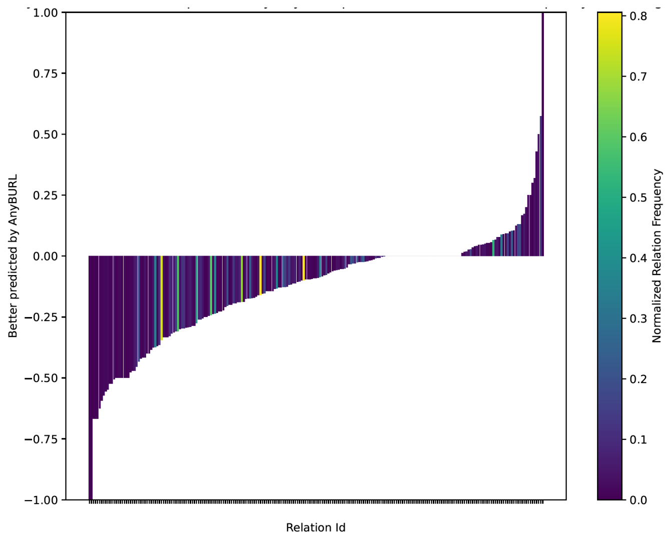


Figure A.13: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the relation frequency

A.6 Entity Frequency in Trainings Data

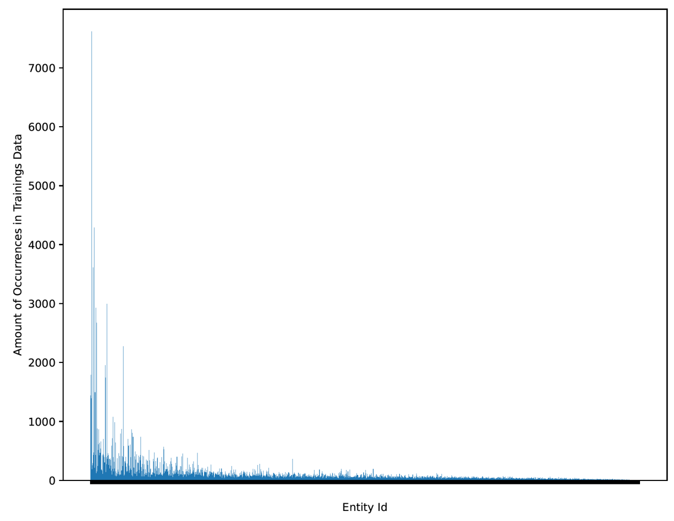


Figure A.14: Entity frequency in trainings data for FB15k-237

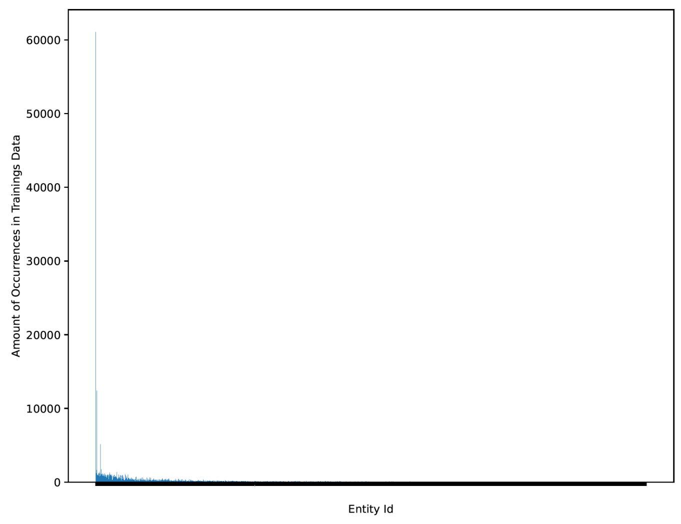


Figure A.15: Entity frequency in trainings data for YAGO3-10

A.7 Comparison in regard to the Entity Frequency

A.7.1 AnyBURL and ConvE on CoDEx-M

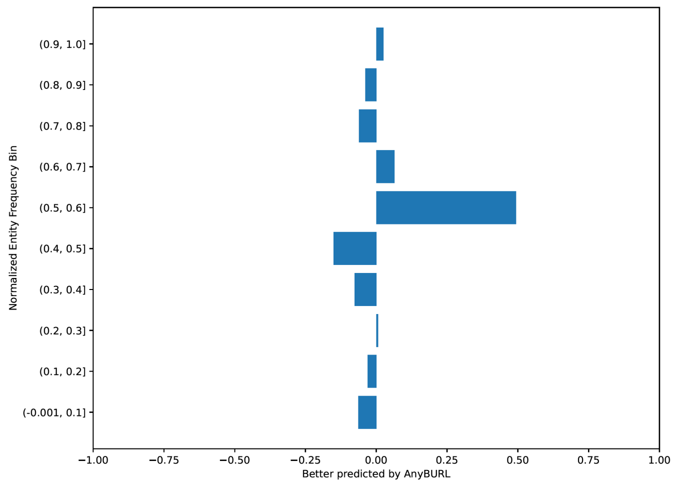


Figure A.16: Comparison of AnyBURL and ConvE on CodEx-M in regard to the question entity frequency

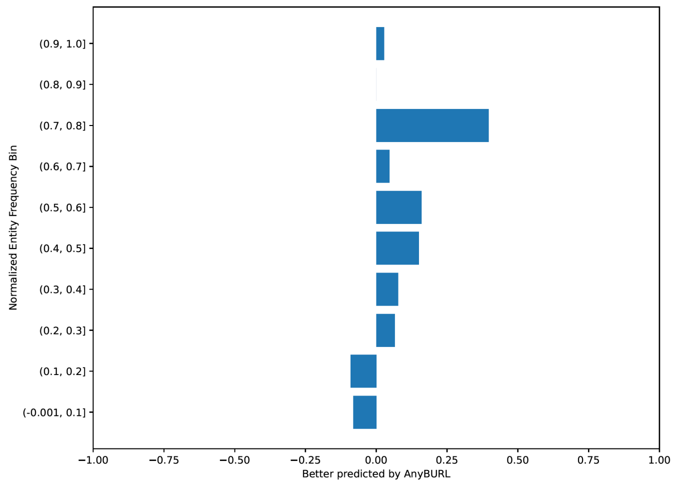


Figure A.17: Comparison of AnyBURL and ConvE on CodEx-M in regard to the answer entity frequency

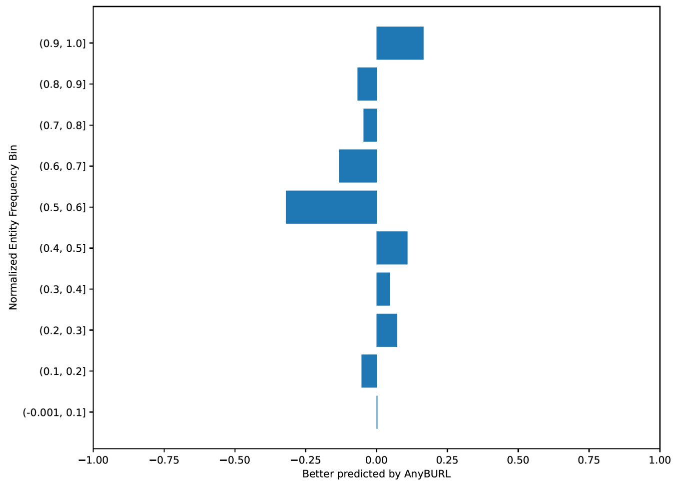


Figure A.18: Comparison of AnyBURL and ConvE on CodEx-M in regard to the combined entity and relation frequency

A.7.2 AnyBURL and RESCAL on CoDEx-M

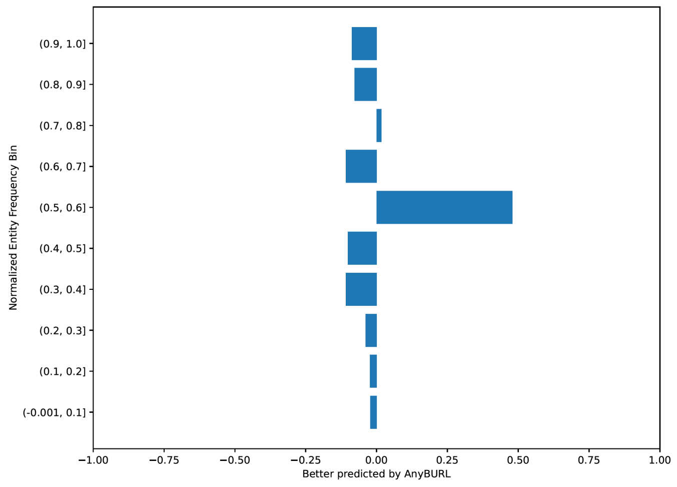


Figure A.19: Comparison of AnyBURL and RESCAL on CodEx-M in regard to the question entity frequency

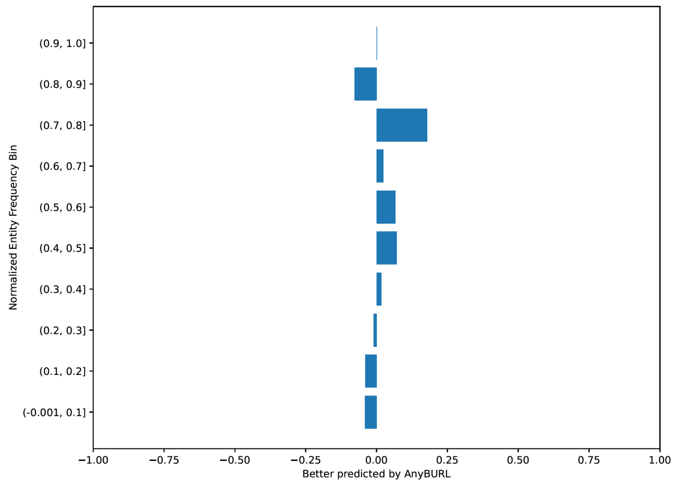


Figure A.20: Comparison of AnyBURL and RESCAL on CodEx-M in regard to the answer entity frequency

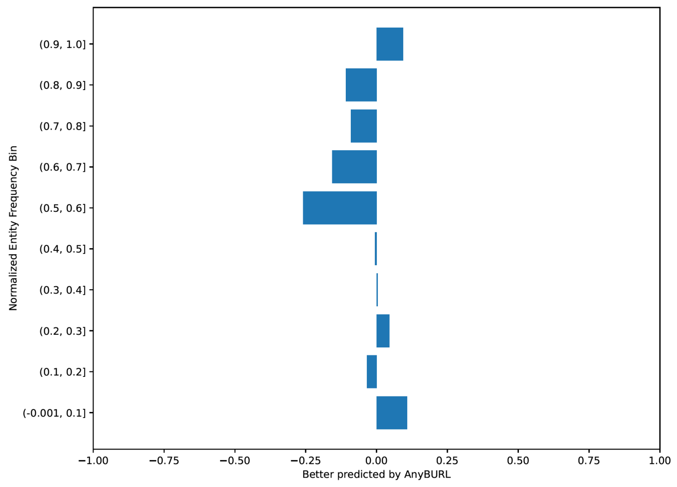


Figure A.21: Comparison of AnyBURL and RESCAL on CodEx-M in regard to the combined entity and relation frequency

A.7.3 AnyBURL and ComplEx on FB15k-237

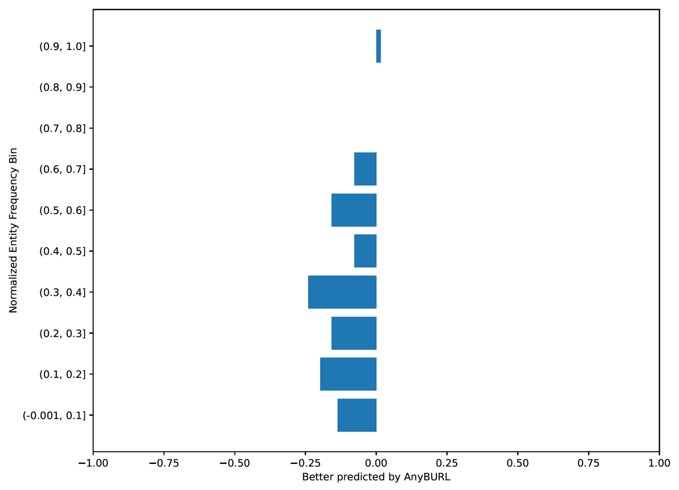


Figure A.22: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the question entity frequency

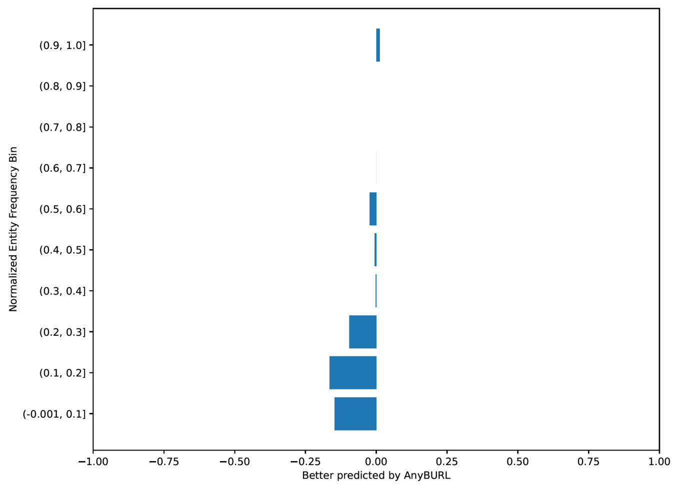


Figure A.23: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the answer entity frequency

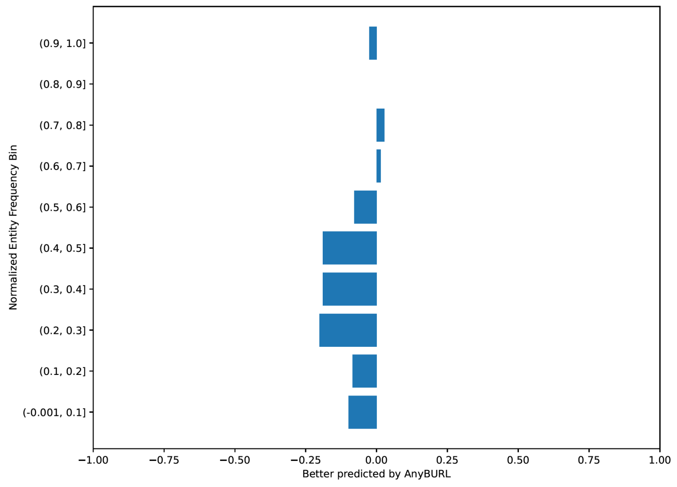


Figure A.24: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the combined entity and relation frequency

A.7.4 AnyBURL and ConvE on FB15k-237

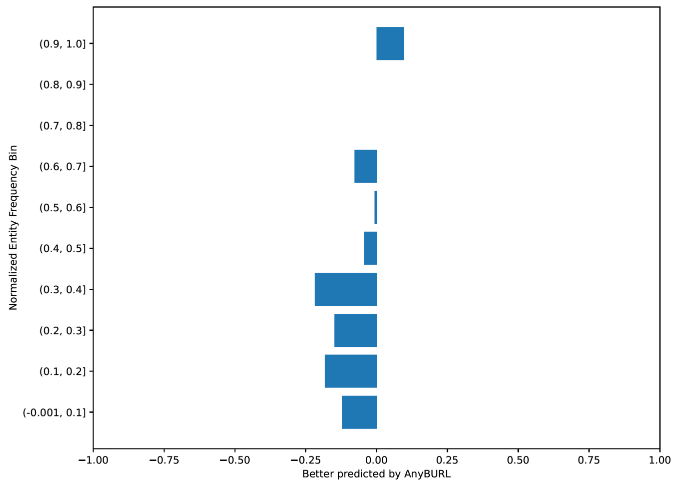


Figure A.25: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the question entity frequency

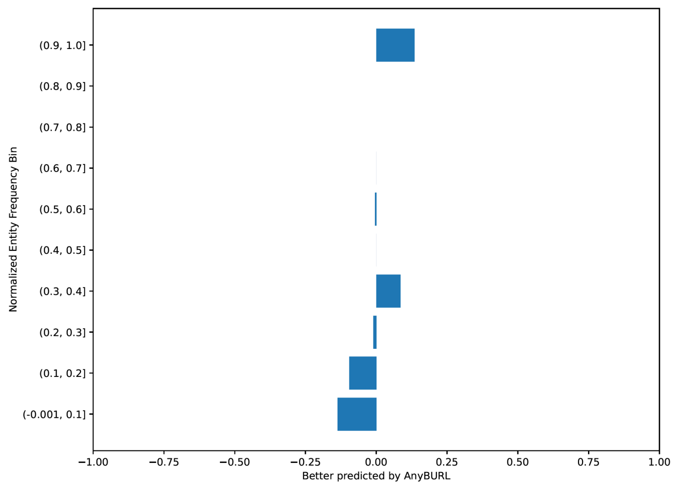


Figure A.26: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the answer entity frequency

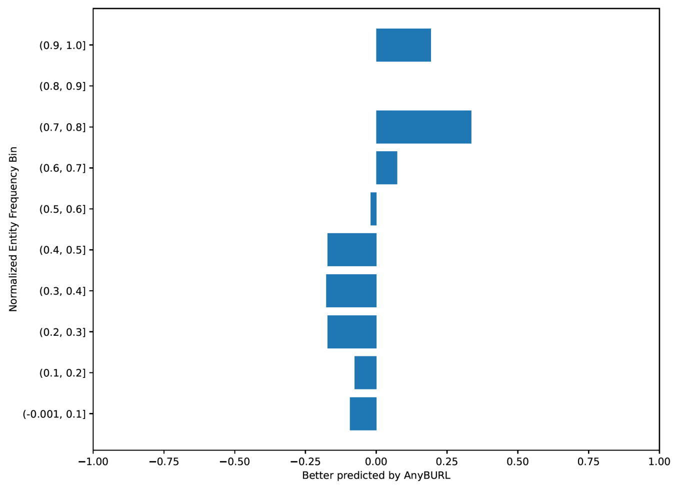


Figure A.27: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the combined entity and relation frequency

A.7.5 AnyBURL and RESCAL on FB15k-237

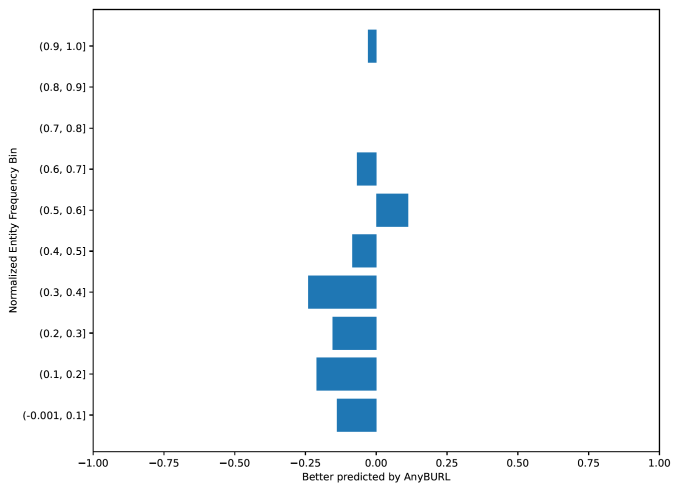


Figure A.28: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the question entity frequency

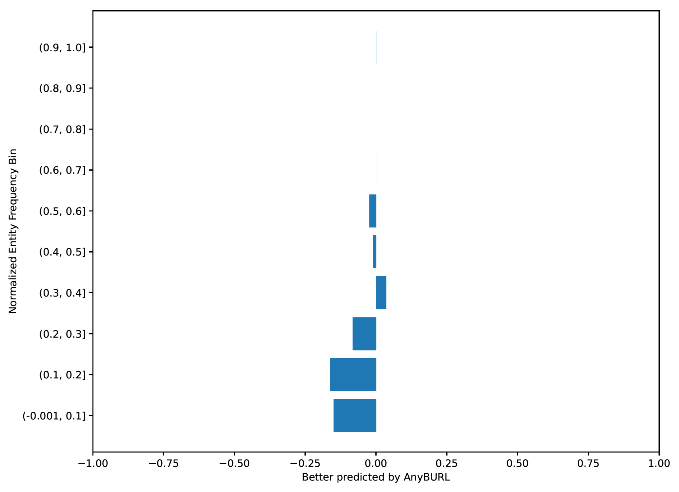


Figure A.29: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the answer entity frequency

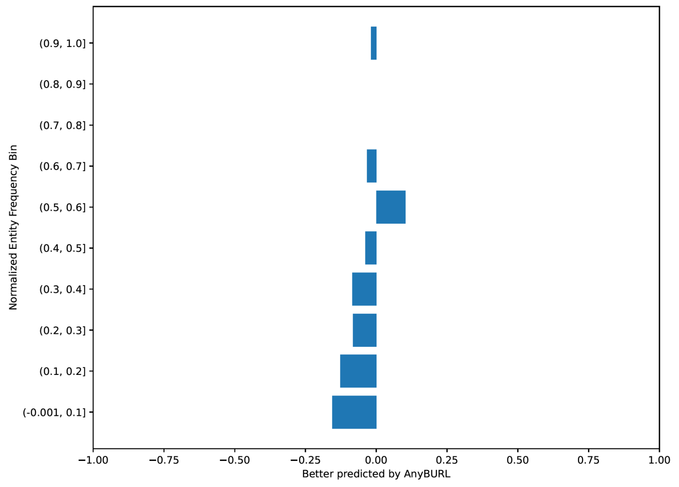


Figure A.30: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the combined entity and relation frequency

A.7.6 AnyBURL and ComplEx on YAGO3-10

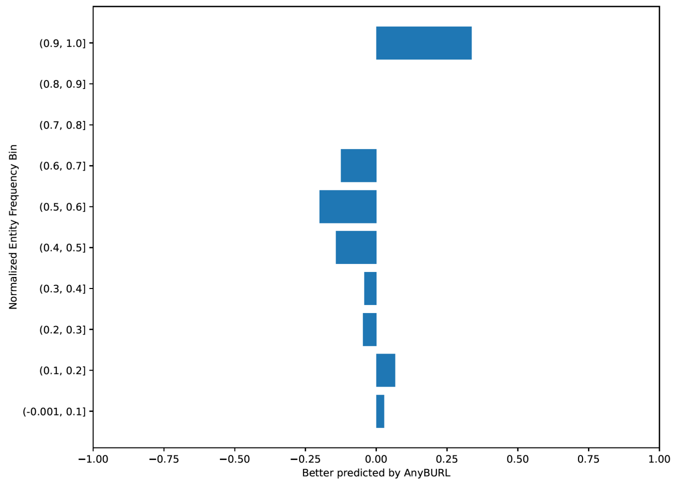


Figure A.31: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the question entity frequency

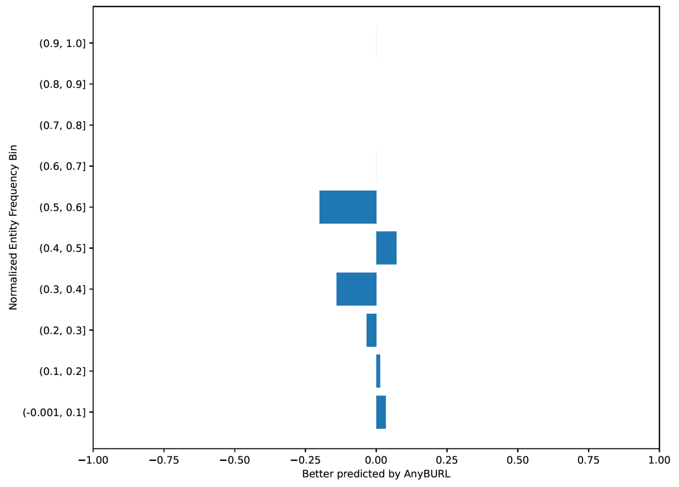


Figure A.32: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the answer entity frequency

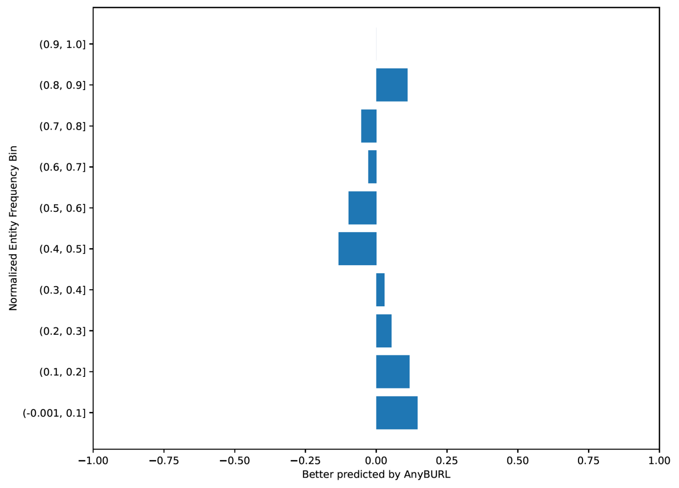


Figure A.33: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the combined entity and relation frequency

A.8 Comparison in regard to the Existence of Similar Triples in the Trainings Data

A.8.1 Binary

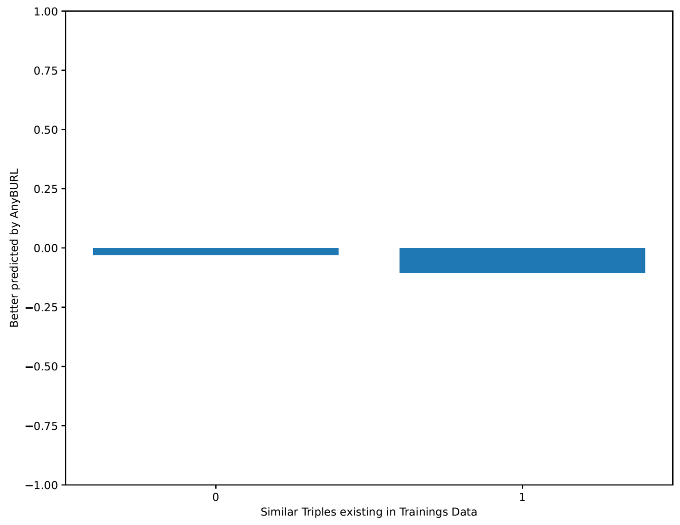


Figure A.34: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar triples in the trainings data

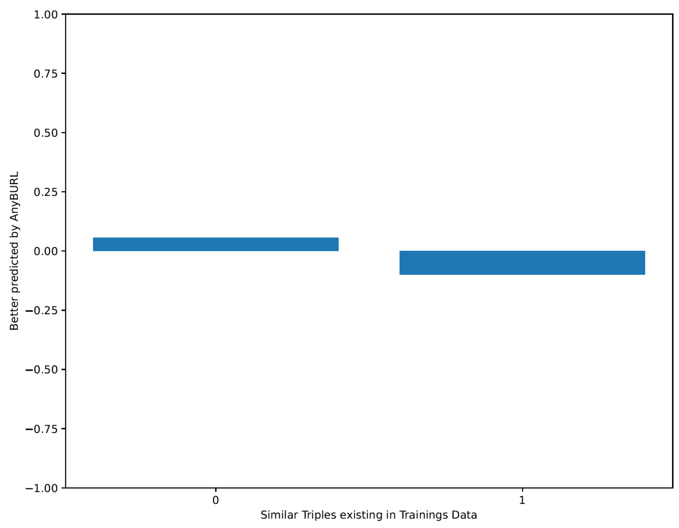


Figure A.35: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar triples in the trainings data

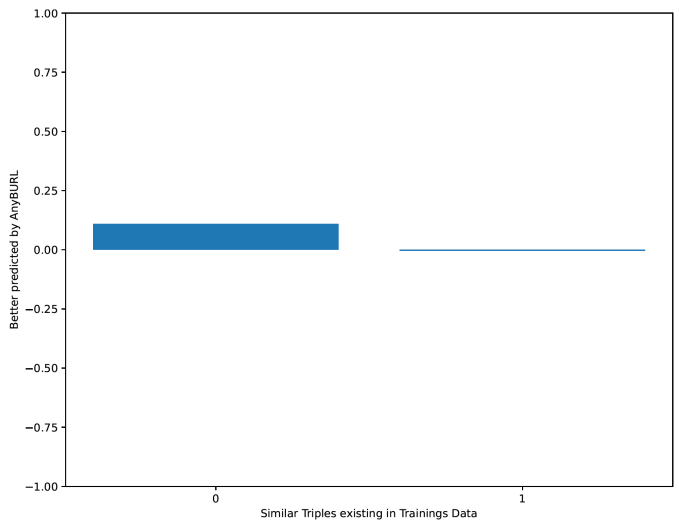


Figure A.36: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar triples in the trainings data

A.8.2 Stepwise

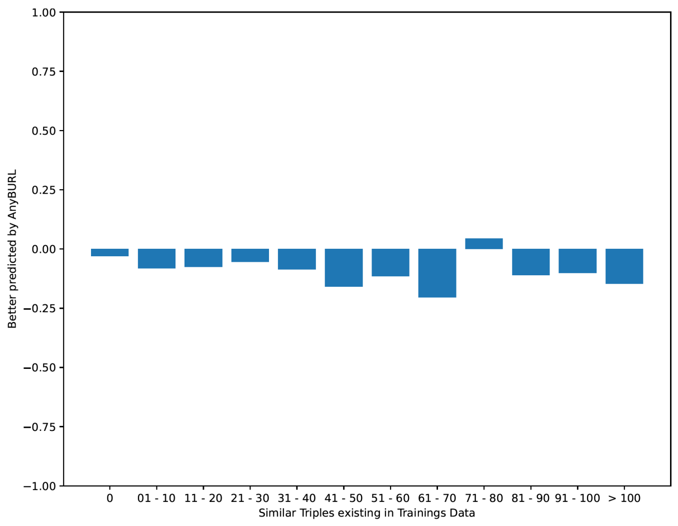


Figure A.37: Comparison of AnyBURL and ConvE on CodEx-M in regard to the amount of similar triples in the trainings data

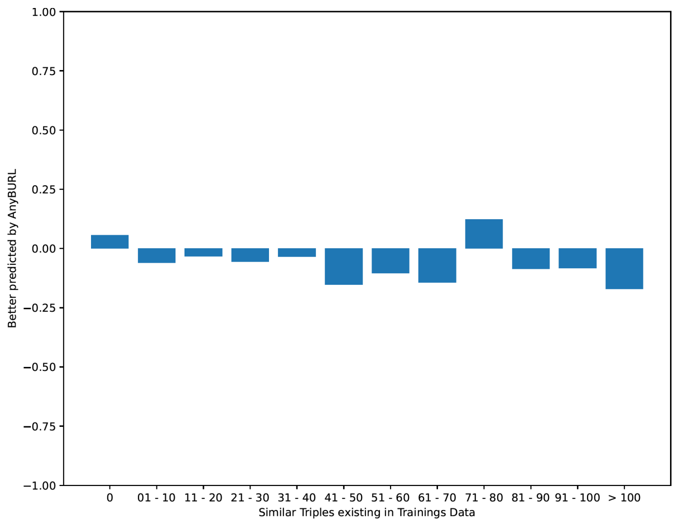


Figure A.38: Comparison of AnyBURL and RESCAL on CodEx-M in regard to the amount of similar triples in the trainings data

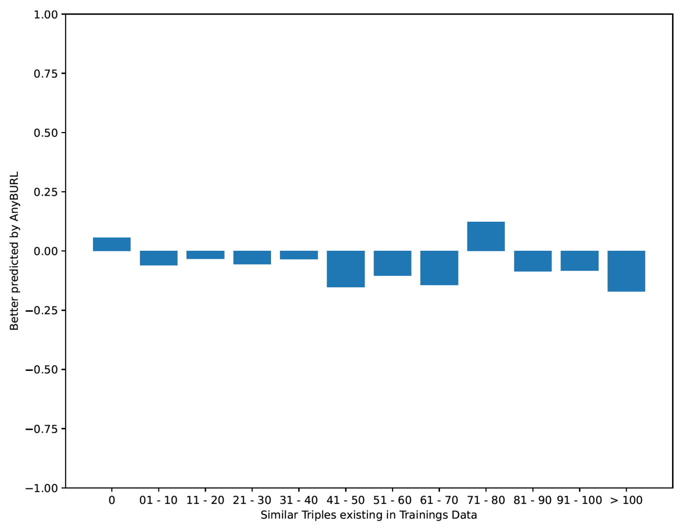


Figure A.39: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the amount of similar triples in the trainings data

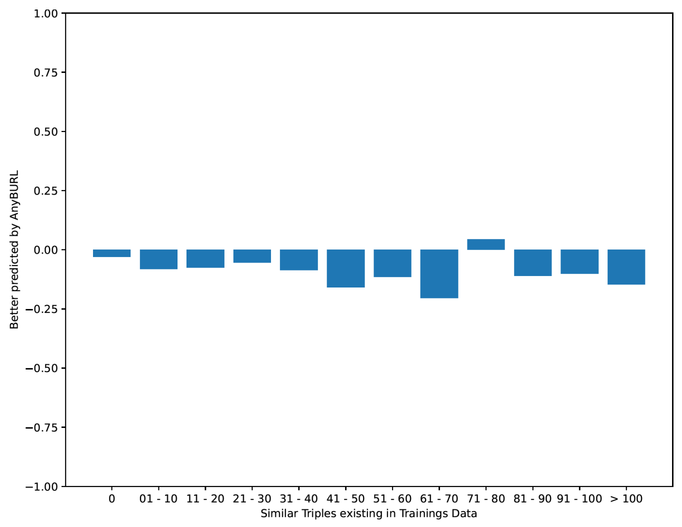


Figure A.40: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the amount of similar triples in the trainings data

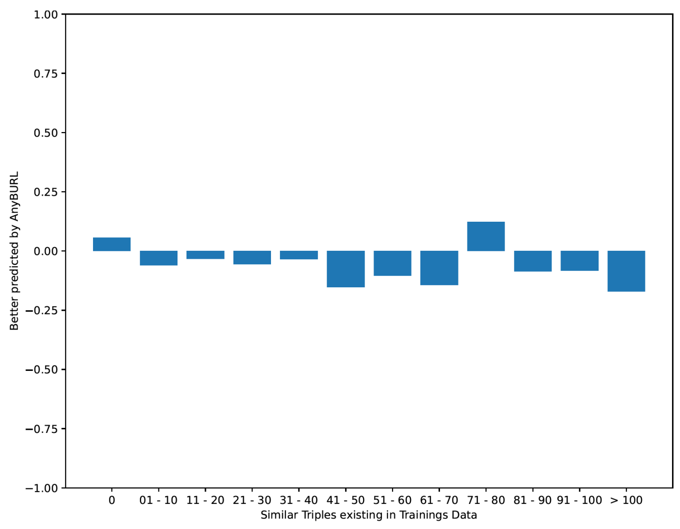


Figure A.41: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the amount of similar triples in the trainings data

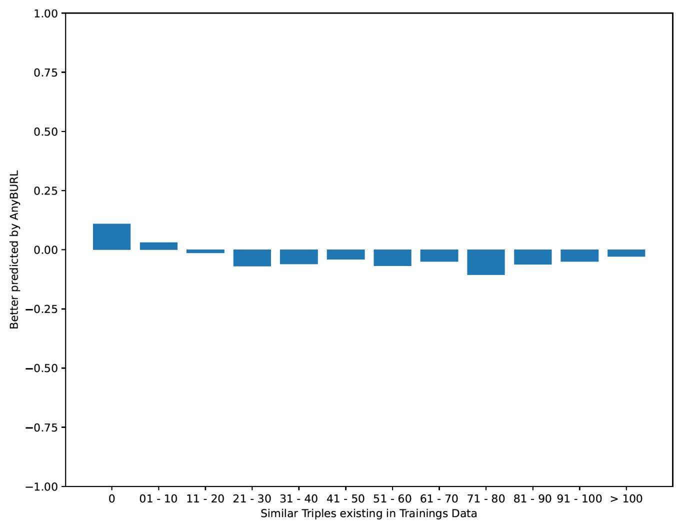


Figure A.42: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the amount of similar triples in the trainings data

A.9 Comparison in regard to the existence of Similar Entities

A.9.1 AnyBURL and ComplEx on CoDEx-M

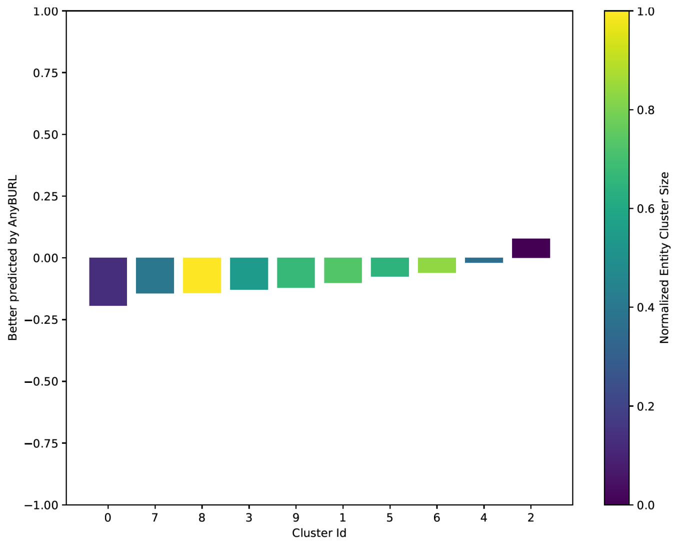


Figure A.43: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=10)

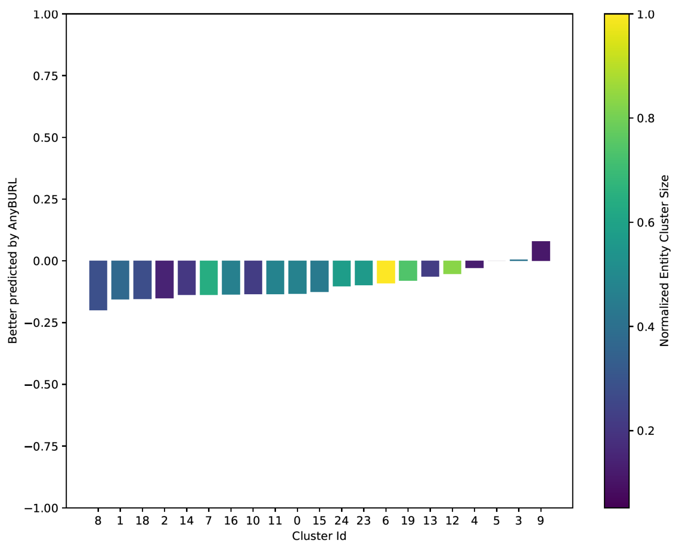


Figure A.44: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=25)

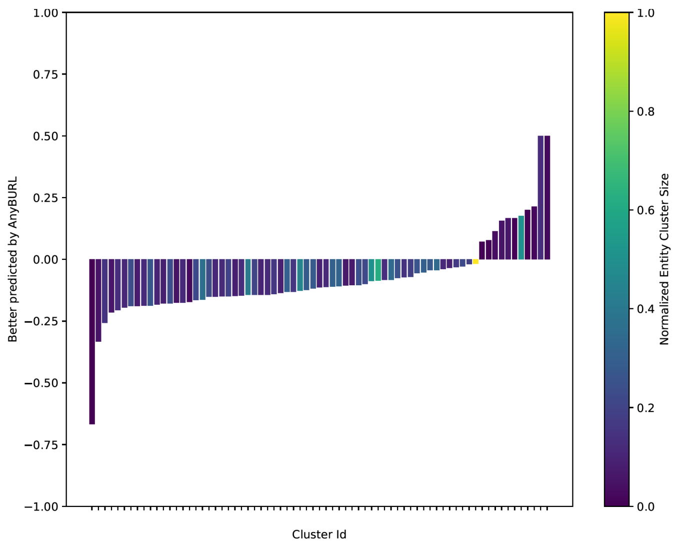


Figure A.45: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100)

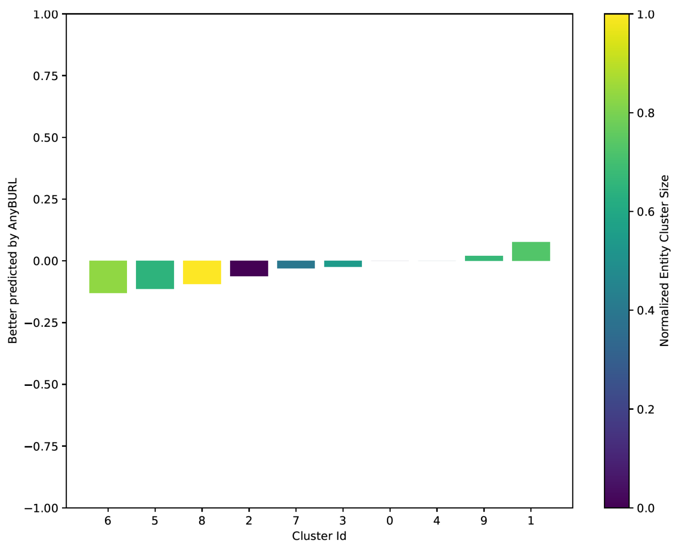


Figure A.46: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=10)

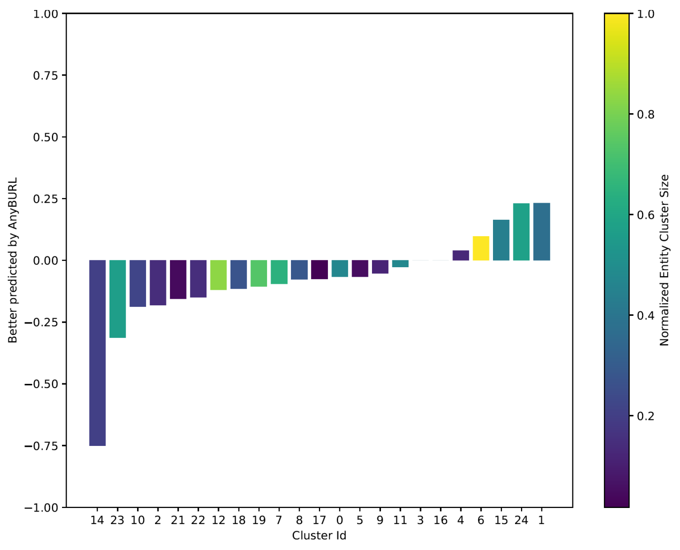


Figure A.47: Comparison of AnyBURL and ComplEx on CoDEx-M in regard to the existence of similar tail entities based on K-Means Clustering (k=25)

A.9.2 AnyBURL and ConvE on CoDEx-M

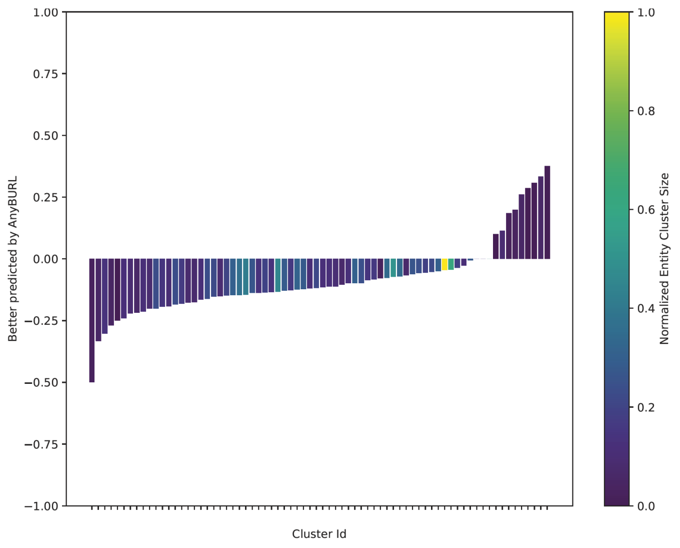


Figure A.48: Comparison of AnyBURL and ConvE on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100)

A.9.3 AnyBURL and RESCAL on CoDEx-M

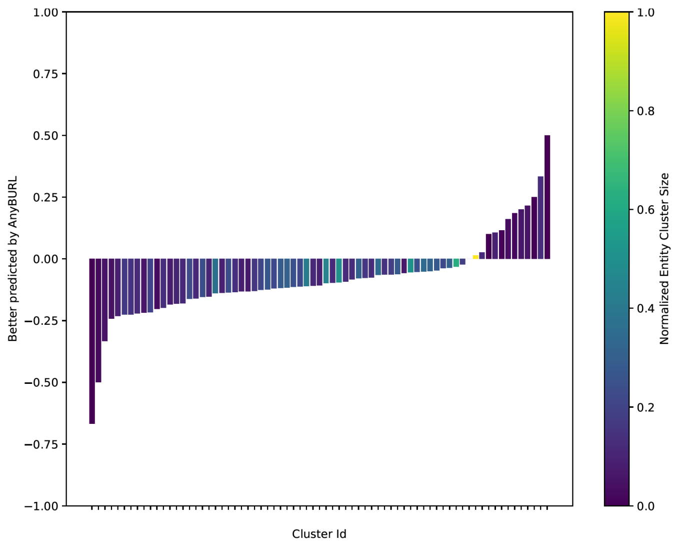


Figure A.49: Comparison of AnyBURL and ConvE on CoDEx-M in regard to the existence of similar head entities based on K-Means Clustering (k=100)

A.9.4 AnyBURL and ComplEx on FB15k-237

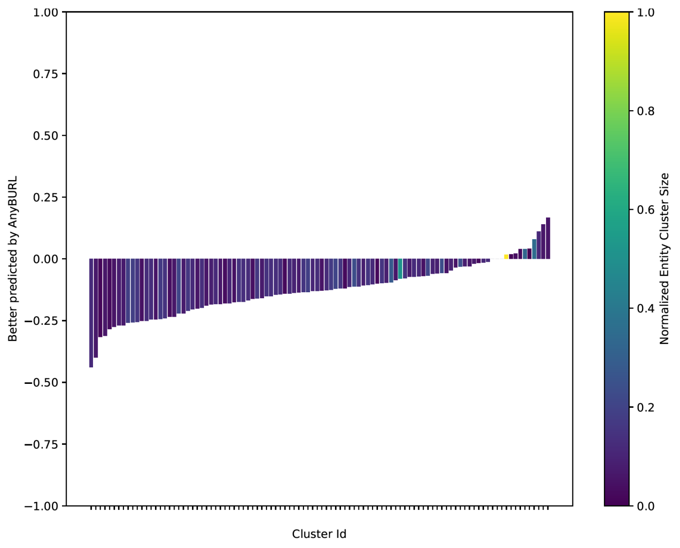


Figure A.50: Comparison of AnyBURL and ComplEx on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100)

A.9.5 AnyBURL and ConvE on FB15k-237

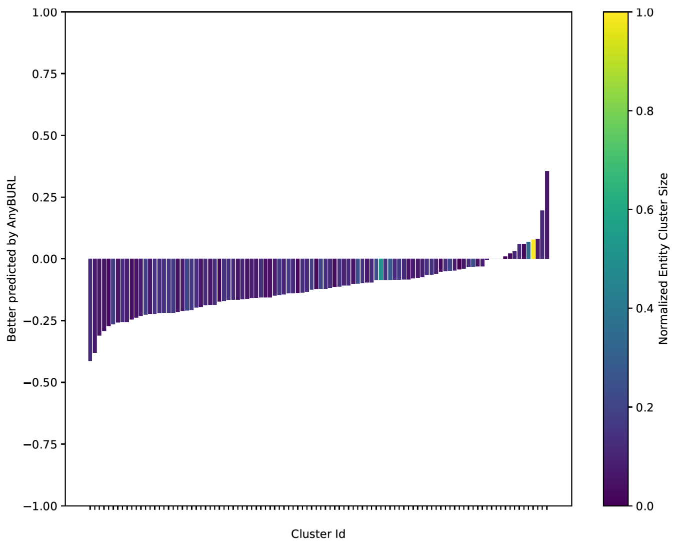


Figure A.51: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100)

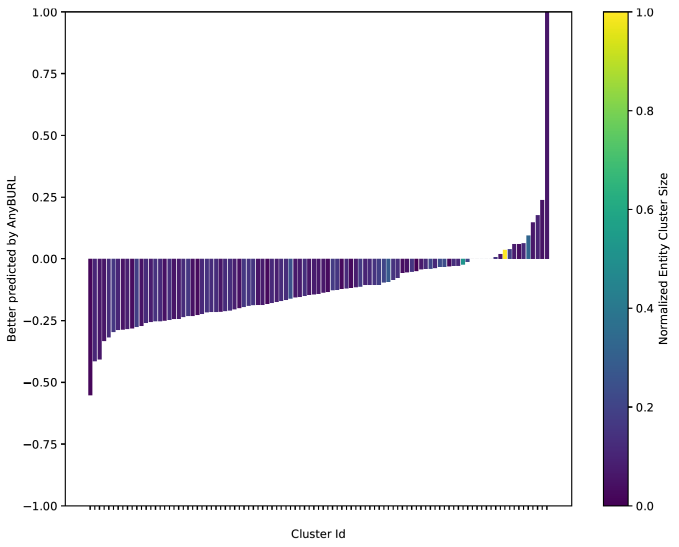


Figure A.52: Comparison of AnyBURL and ConvE on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

A.9.6 AnyBURL and RESCAL on FB15k-237

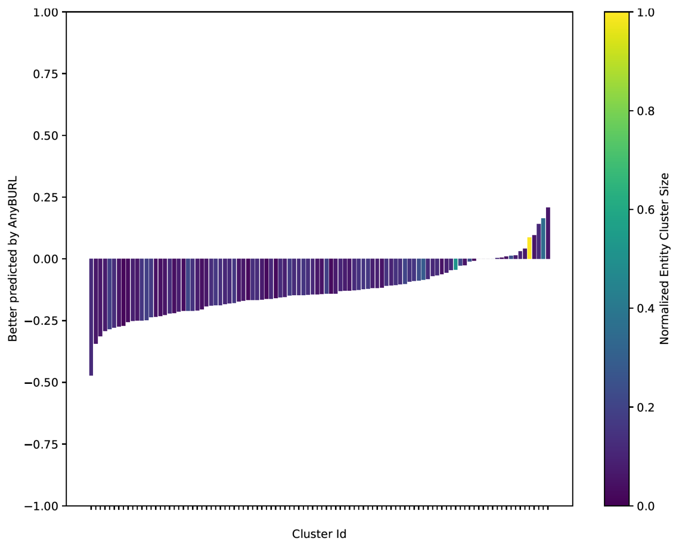


Figure A.53: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar head entities based on K-Means Clustering (k=100)

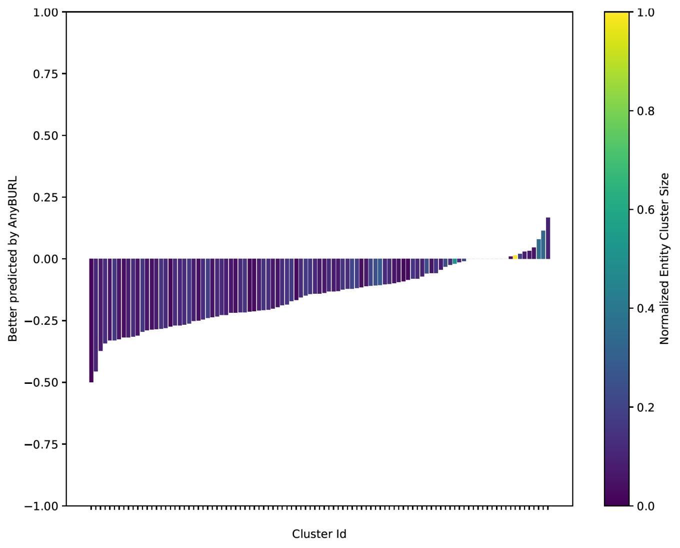


Figure A.54: Comparison of AnyBURL and RESCAL on FB15k-237 in regard to the existence of similar tail entities based on K-Means Clustering (k=100)

A.9.7 AnyBURL and ComplEx on YAGO3-10

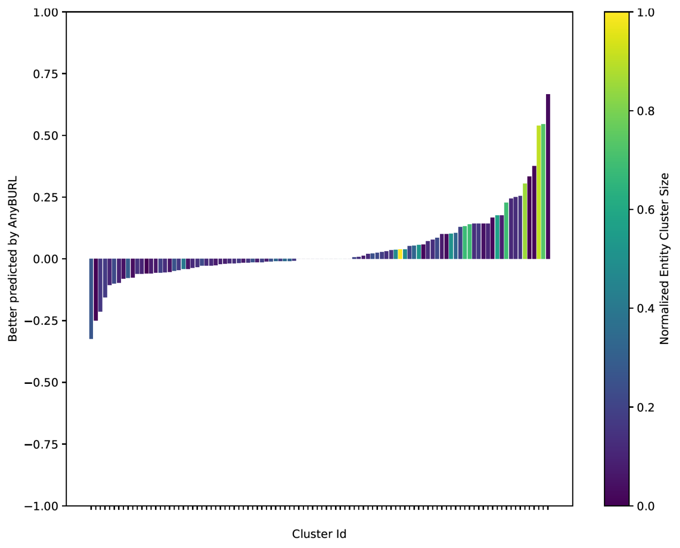


Figure A.55: Comparison of AnyBURL and ComplEx on YAGO3-10 in regard to the existence of similar head entities based on K-Means Clustering (k=100)

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 31.08.2022 Unterschrift