RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
BONN
INSTITUT FÜR INFORMATIK III

Re-evaluating Learning: On the Power of Knowledge Graph Embedding Models

Bachelor Thesis

Reviewer 1: Prof. Dr. Jens Lehmann
Reviewer 2: Prof. Dr. Andreas Behrend

Shayan Shahpasand

November 2021

UNIVERSITÄT BONN

Declaration of Authorship

I hereby declare that all the work described within this Bachelor thesis is the original work of the author. Any published (or unpublished) ideas or techniques from the work of others are fully acknowledged in accordance with the standard referencing practices.

Shayan Shahpasand

Signed: ____

Date: ____

To my companion, Niloufar

Acknowledgements

*First of all, I would like to express my gratitude to both dear **Dr. Sahar Vahdati** and dear **Mr. Mojtaba Nayyeri** for their outstanding support, guidance and professional cooperation. I would also thank them for trusting and giving me the opportunity to start such extensive project. I have made a pleasing experience working with you.*

***I would also like to thank Prof. Dr. Jens Lehman** and **Prof. Dr. Andreas Behrend** for kindly accepting to be my examiners.*

*I would also thank my lovely wife, **Niloufar** and **family** for being there for me and support me through all steps of my life. I highly appreciate your being in my life.*

I would also like to thank the following people for their support:

***Matei Climescu** for his detailed proof reading of this thesis.*

***Michael Galkin** for giving me my first insights of working with TU-HPC systems.*

***The Centre for Information Services and High Performance Computing** [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] TU Dresden and their **HPC Support Team** for providing their facilities for high throughput calculations, answering my questions through the past 6 month and their kind cooperation.*

***Mirza Mohtashim Alam** and **Gökçe Çil** for providing some of their scripts and let me to implement upon them.*

Contents

Chapter 1

Introduction

Artificial Intelligence (AI) has reached hyped attention in recent years. This has both technical and resource-related reasons. On the technical aspect, the computational power is increasing rapidly while the cost of computation decreases continuously. Besides, the phenomena of Big Data, which grows rapidly, contains hidden but valuable information, which motivates tech companies, and research groups to find methods to drain as much knowledge as possible from these data. On the resource aspect, AI-based approaches help societies to accelerate processes in a way which is not comparable with the capacity of human operators. Accuracy is another advantage of using AI. For example, it would take months(if not years) for a store manager in a medium-size supermarket to notice that replacing whipped cream next to strawberries raises their number of sales, while a computer algorithm would notice this as soon as the number of sales for these two items crosses its average value.

Despite such abilities and many more advances in all areas of science facilitated by AI, there are still challenges when deploying AI in our daily life. For example, unlike humans, it is hard for computers to understand the context of information. This is why there has been much work done in representing data in a machine-understandable format.

The legacy approach of relational data as representing data in tabular format has been the only popular approach even for the AI-based tasks due to their power of representation and querying. Finding specific entries upon their relation between a small number of tables is also efficient. Yet, the complexity and thus, the efficiency of such queries grow exponentially as the number of involved tables in the queries grows. Especially with the large scale data explosion, and the need for learning approaches for huge quantities of data made relational representation impossible to be followed by AI-based techniques.

To overcome such issue, **Knowledge Graphs(KG)** technology have emerged. A Knowledge Graph is a representation of information in a graph form, where entities(subjects and objects) are the nodes of the graph, and the relation which connects these entities are the edges of the graph. Knowledge Graph embedding represents KGs in low dimensional space, and Knowledge Graph Embedding Models try to infer information out of KGEs.

KGE Models are used in various applications such as question answering, recommendation systems, security, and link prediction. An example of link prediction is Facebook's friend suggestion system. While Amazon uses KGE to suggest new products that match a user's preference, KGE models are applied in the finance sector to recognize fraudulent transactions.

The new era of KGE models started in 2011 by a proposed model named RESCAL [9] and followed by many models which appeared shortly afterward while pushing their predecessor models' accuracy and performance. Section 4.3 goes deeper into KG and KGE's world, describing them in detail with examples and compares different KGE models and their approach with each other.

These KGE models are used in real-world application for different tasks for example **Question Answering**, **Recommendation Systems** and **Link Prediction**. As an example of a recommendation system, Amazon's offering system can be mentioned. It offers multiple products as a package which are often bought together. i.e., Basketball and basket hoop

KGE models can be differentiated from each other in their score functions. Some, such as TransE, use translation based score function while others like DistMult use complex multiplication based score function. The Power of KGE models is different in learning various patterns. Some are more accurate to recognize and learn symmetric relations between the entities, and other models are better at recognizing patterns like asymmetric, implication, or inversion.

Besides score functions, there is another essential component in training KGEs, which plays a significant role in the models' efficiency and performance. This component is the loss function. Loss functions are used to optimize the entities and relation embeddings in low dimensional spaces and can be used in combination with different KGE models.

While KGE models and their score functions are essential in training KGEs, being able to run experiments to train KGEs in a wide range of evaluation configurations is still a significant obstacle for many students and research groups since it requires a massive amount of computational power and secure infrastructure.

This thesis tackles the technical experimenting issues by introducing and implementing a semi-automated system to run experiments with wide ranges of configurations and then apply the same implementation to compare five state of the art models' accuracy by executing an extensive configuration.

1.1 Research Questions

To give our research a clear path, we defined two research questions which needed to be answered during this research.

Research Question 1: *How can leveraging high performance computing facilitate a comprehensive analysis of KGE Models?*

This research question aims to understand if and in which aspects utilizing high-performance computation can affect the analysis of KGE models.

Research Question 2: *How do different metrics in hyperparameter settings affect the efficiency of KGEs models?*

In this research question, we aim at a more detailed comparison between different model's performance based on different evaluation metrics, for example, the hidden dimensions of the embeddings or utilized loss functions.

1.2 Thesis Structure

The thesis is organized as follows: The preparation, as well as the installation of the implementation on different operation systems, are described in the chapter 2. Chapter 3 zooms back one level and explains the software architecture of the implementation as a system and its life cycle. In chapter 4 the configuration of KGExperimentify used for the wide range of experiments in this thesis are discussed in detail, and the results' evaluation of these experiments are presented in Chapter 5. In the last chapter, the concept of an on-demand service is introduced as future work, which will ease and accelerate the cooperation among the community while saving an enormous amount of resources and energy.

Chapter 2

Conceptual and Technical Background

2.1 Implementation

For this thesis, the KGE-Pattern [10] repository is used, which is built on top of the RotatE Knowledge Graph Embedding framework [8]. All modifications and specific branches are based on this code.

2.2 Code Optimizations and Modifications

Besides the primary implementation of this thesis, some slight changes have been made in some files, for example, `run.py` to optimize readability and logging of the results. Some parts of the code were modernized to catch up with best practices, for example, using formatted strings in outputs.

In addition to above-mentioned optimizations, two bash script files and three python file are added. These files are `command_genarator.sh`, `hpc_job_file.sh`, `default_logs_transporter.py`, `pattern_logs_transporter.py` and `logs_to_excel.py` which are explained in chapter 3.

2.3 Command Line and Command Line Arguments

Executing an experiment with the mentioned implementation requires multiple arguments that specify the parameters used during the experiments, i.e., learning rate and batch size. In this section, these parameters and their order are explained.

Listing 1 indicates an example of an execution code, and table 2.1 describes the command line arguments with their order of appearance.

Listing 1 Example of an experiment execution

```
\$ python3 run.py --do_grid --cuda --do_test --data_path data/FB15k
↪  --model RotatE -d 200 --negative_sample_size 10 --batch_size
↪  1024 --gamma 50 --adversarial_temperature 1
↪  --negative_adversarial_sampling -lr 0.1 --max_steps 400000
↪  -save models/RotatE/margin_ranking/FB15k --loss margin_ranking
```

The aforementioned command should be executed in the `codes` folder under the repository's main directory. In case of executing the command from another directory, the relative path to `run.py` file must be adjusted.

Option	**Description**
`--do_grid`	Grid flag. start a grid run
`--cuda`	GPU flag. Skip if running on CPU
`--do_test`	Evaluation on test set
`--data_path`	Relative path to data set which is being trained and tested
`--model`	The model with which the dataset is being trained with. Example: RotatE. All models which this repository is capable of.
`--d`	Hidden dimension as integer
`--negative_sample_size`	Negative sample size as integer
`--batch_size`	Batch size as integer
`--gamma`	Gamma as integer
`--adversarial_temperature`	Adversarial temperature as integer
`--negative_adversarial_sampling`	Negative adversarial sampling as integer
`--lr`	Learning rate as integer
`--max_steps`	Max steps as integer
`--save`	Relative save path for logs, checkpoint and embedding vectors
`--loss`	Loss function for example `margin_ranking_loss`
`--init_checkpoint`	Path to the stored checkpoint and config file

TABLE 2.1: `run.py` executing arguments

2.4 Technical Requirements to Start an Experiment

In order to run an experiment, there are some requirements which need to be fulfilled. In this section, these requirements are grouped and explained.

Python: This project can be executed both with python2 and python3. It is recommended to use python3 due to its compatibility with new plugins and further development.

Python Packages: Table 2.2 shows all required python packages for this project and their used version for the experiments done during this thesis.

Package Name	Version
Pandas	1.0.5
Scikit-learn	0.22.01
Numpy	1.18.1
Pytorch	1.6.0

Table 2.2: required python packages and used versions in this thesis

2.5 Experiment Environments

For this thesis, three different environments were used. In this section, these environments are introduced, and the preparation of each environment for executing experiments are explained. The first section describes common steps among all three environments. Special modifications for each environment might be necessary and will be discussed in the specified section.

2.5.1 Common Steps

There are some fundamental steps which are shared among different environments regardless of their structure. In this section, these steps are explained. All parts are explained for Macintosh and Linux systems.

2.5.1.1 Git

Git is a version control system which helps the developer to keep track of its work easily. Since there are many changes, adjustments, and optimization needed in such a project, such as this bachelor thesis, git is used to organizing changes, experiments, and results. That is the reason it is necessary to install git as the first step.

Mac OS An easy way to install git on a Macintosh machine is through homebrew. To install homebrew, the following two commands in listing 2 need to be executed in a terminal window:

Linux In order to install git on a Debian based distribution of Linux, the command in listing 3 need to be executed in a terminal window.

Listing 2 Installing Git on macOS

```
\$ curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
↪  master/install.sh

\$ brew install git
```

Listing 3 Installing Git on Linux

```
\$ sudo apt install git-all
```

RSA Key The process of running multiple experiments requires tense communication with the remote system, where the experiments are executed. Therefore adding an RSA key to the remote system is recommended to save time.

2.5.1.2 Cloning The Project

It is always suggested to have a general "Workspace" directory and clone your projects within this general folder. The directory tree of such a "Workspace" directory should look like this:

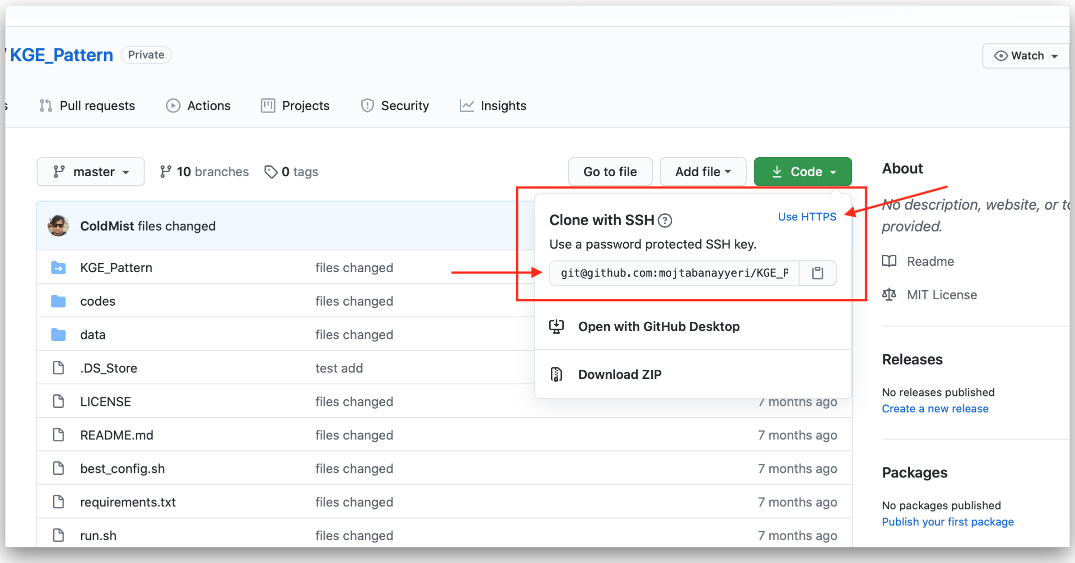
```
|workspace
|        |KGE_Pattern
|        |project2
|        |project1
|        |...
```

To clone the code, the proper link, SSH or HTTPS, should be chosen from the marked place in the figure 2.1. This link can be used in the command listed under listing 4 in the correct directory to clone the project.

Listing 4 Example of cloning the project from git with SSH and HTTPS link

```
\$ git clone git@github.com:mojtabanayyeri/KGE_Pattern.git
\$ git clone https://github.com/mojtabanayyeri/KGE_Pattern.git
```

There is an easier way to clone the repository when using PyCharm on a local machine, which is explained in section Cloning the code in PyCharm.



No description, website, or t
provided.

No releases published

No packages published

Use a password protected SSH key.

```
git@github.com:mojtabanayyeri/KGE_P
```

ColdMist files changed		
KGE_Pattern	files changed	
codes	files changed	
data	files changed	
.DS_Store	test add	
LICENSE	files changed	7 months ago
README.md	files changed	7 months ago
best_config.sh	files changed	7 months ago
requirements.txt	files changed	7 months ago
run.sh	files changed	7 months ago

FIGURE 2.1: Repository's cloning options.

2.5.1.3 Python 3 Aliased

It is highly recommended to work with python3 since there are major optimizations in this version compared to the older versions. To avoid typing "python3" every time a python program needs to be started with python3, an aliased can be set such that every time a program is started with "python" it actually runs with python3. Please note if there are still old python programs being executed on the machine which are executable only using python2, this step is not recommended. To add an alias to python3 in macOS or a Debian based Linux, the command line, `\$ alias python=python3` must be executed in a terminal window.

2.5.1.4 Python Virtual Environments

In order to run an experiment with the `run.py` some python modules need to be installed. Python virtual environment is used to distinguish installed packages for different projects. To create a new virtual environment named env01 using python3, the following code needs to be executed in a terminal window.
`\$ python3 -m venv envs/env01`

After creating the env01 environment, it can be activated using the following command.

```
\$ source envs/env01/bin/activate
```

Now that the virtual environment is activated, all required packages can be installed using the following command.

```
\$ pip install torch torchvision numpy sklearn
```

It is essential to know that for the successful execution of `run.py`, the aforementioned virtual environment must be activated regardless of the operating system. An easy way to install required packages via PyCharm is explained in the section 2.5.2.3

2.5.2 Local Machine for Debug Purposes

Being able to run and debug the code on the local machine is crucially essential in order to allow a better understanding of the implementation. This paragraph is explaining how to prepare and run the code on a local Macintosh machine. Running the code on other Unix based machines is also similar to these methods.

2.5.2.1 IDE

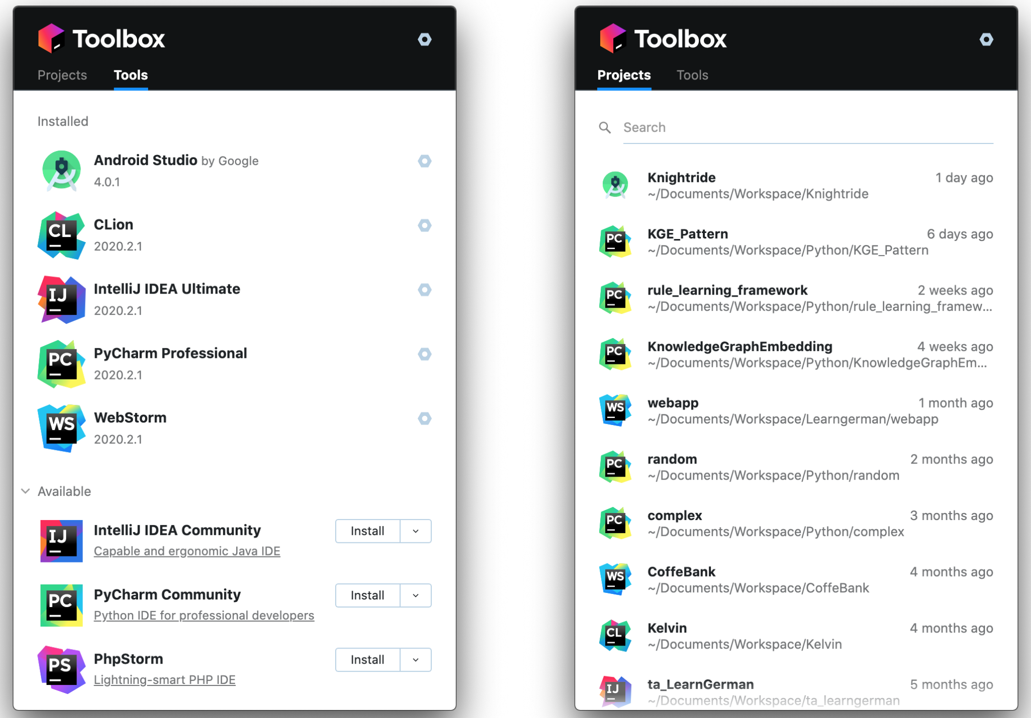
For this project, PyCharm is used for developing and debugging the code. It is suggested to install Jetbrain Toolbox (Figure 2.2) [11] and use it to install PyCharm. JetBrains Toolbox helps to install a specific version of IntelliJ software and makes it easy to switch between IDE versions. Besides, it also helps to manage and efficiently organize projects.

2.5.2.2 Cloning The Code in PyCharm

The easiest way to check out the repository is to click `Get from version control` under `VCS` option in the toolbar. Then add paste the repository link in the `URL` field and click `Clone`.

2.5.2.3 Installing Required Python Packages

Since the file `reqirements.txt` is added to the project, after checking out the project, users will be automatically be asked if they want to install the required



Installed

Android Studio by Google	4.0.1
CLion	2020.2.1
IntelliJ IDEA Ultimate	2020.2.1
PyCharm Professional	2020.2.1
WebStorm	2020.2.1

Available

IntelliJ IDEA Community	Capable and ergonomic Java IDE	Install
PyCharm Community	Python IDE for professional developers	Install
PhpStorm	Lightning-smart PHP IDE	Install

Knightride	~/Documents/Workspace/Knightride	1 day ago
KGE_Pattern	~/Documents/Workspace/Python/KGE_Pattern	6 days ago
rule_learning_framework	~/Documents/Workspace/Python/rule_learning_framew...	2 weeks ago
KnowledgeGraphEmbedding	~/Documents/Workspace/Python/KnowledgeGraphEm...	4 weeks ago
webapp	~/Documents/Workspace/Learngerman/webapp	1 month ago
random	~/Documents/Workspace/Python/random	2 months ago
complex	~/Documents/Workspace/Python/complex	3 months ago
CoffeBank	~/Documents/Workspace/CoffeBank	4 months ago
Kelvin	~/Documents/Workspace/Kelvin	4 months ago
ta_LearnGerman	~/Documents/Workspace/ta_learngerman	5 months ago

FIGURE 2.2: Jetbrain toolbox

packages.
For the manual installation it is needed to open the `preferences...` from `file` menu and then navigate to `Project: KGE_Pattern`. Now under `project inspector` project inspector can be selected.

In the case of this thesis, both Python3 and Conda interpreters are available. It is a personal choice which one to use. For using Conda, it should be installed in advance. This section describes how to install packages with the help of Conda. After activation of Conda package manager, specific Python packages can be installed using the `+` button on the bottom left corner of the window. There, package names can be searched in the search field, as shown in the figure 2.3. Then the exact version of the package can be selected to be installed.

2.5.2.4 Setting Up The Run Configuration

It is always possible to execute a run command in the terminal but being able to debug the code and using breakpoints is much more helpful to understand and optimize code. For this purpose, `Edit Configurations...` under `Run` menu in

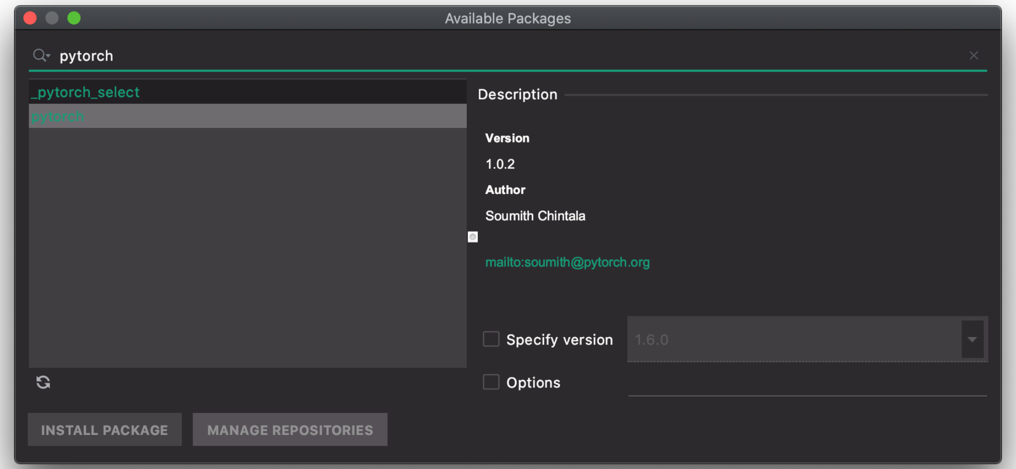


FIGURE 2.3: Installing Pytorch using conda in PyCharm.

the toolbar needs to be clicked. Then new Python configuration must be added by clicking the `+` button and choosing Python.

An optional name in `Name` field can be added. Now `run.py` file must be added in `script path` either by choosing it in the file system or pasting its absolute path into the field. Under parameters, any of in this section mentioned arguments could be used. Please note that the `-- cuda` flag should not be passed in order to run/debug the code on CPU. An example of such parameters for run/debug is demonstrated in listing 5.

Listing 5 Example of a command line to run an evaluation on a local machine.

```
--do_grid --do_test --data_path data/FB15k --model RotatE -d 50
↪  --negative_sample_size 10 --batch_size 512 --gamma 1
↪  --adversarial_temperature 1 --negative_adversarial_sampling -lr
↪  0.1 --max_steps 400000 -save
↪  ../models/RotatE/rotate/FB15k/dim-50/gamma-1/learning-rate-0.1
↪  /batch-size-512/negative-sample-size-10/ -de --loss rotate
↪  --init_checkpoint
↪  ../models/RotatE/rotate/FB15k/dim-50/gamma-1/learning-rate-0.1
↪  /batch-size-512/negative-sample-size-10/
```

By pressing the `debug` button, the code will start to run with the given parameters and stop on breakpoints.

2.5.3 SDA Servers

SDA(Smart Data Analytics) Group offers two GPU servers(sda-server04 sda-server05), which are used for this thesis. Both of these servers use Linux and have python3 installed. These servers are accessible via SSH connection.

2.5.3.1 Connection to SDA Servers via SSH

The user needs to connect to IAI (`login.iai.uni-bonn.de`) server first, either via VPN or SSH, to be able to connect to SDA servers. During this thesis, IAI was always used as a proxy jump station to establish the connection to SDA servers. A computer science (Informatik) student account is needed to be able to connect to the IAI servers. Another account is needed to connect to SDA servers. This account is created by the SDA server admins. The following command is used to connect to the IAI servers:

```
\$ ssh username@login.iai.uni-bonn.de
```

Where `username` is to be replaced with the computer science student account. After connecting to IAI servers the user is able to connect to the desired SDA server with one of the following commands:

```
\$ ssh username@sda-srv04.iai.uni-bonn.de
\$ ssh username@sda-srv05.iai.uni-bonn.de
```

`Username` is to be replaced with SDA account.

2.5.3.2 Time Saving Mechanism

Connecting to IAI and then SDA servers, regardless of using VPN or SSH, is a time-consuming process, and it is a process which needs to be repeated multiple times in a day while working on the project. In order to save time during this action it is suggested to add a Proxy Jump to the SSH config on the local machine. To do this, the config file under `.ssh` directory must be updated. A new config file can be created if there are none existing. Listing 6 indicates a config file which uses a proxy jump to SDA servers.

Listing 6 Example of a proxy jump configuration using `login.iai.uni-bonn.de` as bastion

Part1:

```
Host bastion
    HostName login.iai.uni-bonn.de
    user USERNAME
    port 22
    IdentityFile ~/.ssh/id_rsa
    IdentitiesOnly yes
    UseKeychain yes
    AddKeysToAgent yes
```

Part2:

```
Host server05
    HostName sda-srv05.iai.uni-bonn.de
    user USERNAME
    port 22
    IdentityFile ~/.ssh/id_rsa
    IdentitiesOnly yes
    ProxyJump bastion
    UseKeychain yes
    AddKeysToAgent yes
```

Part1 will create a bastion with IAI server and the **username**. Setting IdentityFile helps to use the private RSA key to connect to this server. Note that RSA public Key has to be added in the login account in IAI Servers, and **USERNAME** must be replaced with the computer science account.

Part2 will use the defined bastion to connect to SDA servers every time a connection to SDA servers is established. In this example, server05 is the name of the defined server in the config file. This means, after configuring the config file, it is enough to use the command `ssh server05` to connect to the SDA server sda-server05 instead of connecting to IAI server first and then to SDA servers manually. Please note that the local machine's public RSA key needs to be added to both SDA and IAI servers.

2.5.3.3 Preparing SDA Servers to Run Experiments

Since a lot of different experiments might run simultaneously and small adjustment might be necessary for each one of those experiments, it is helpful to use a

Workspace directory where multiple versions of the program are cloned in different files in order to first, have a good overview of ongoing evaluations and second, to be able to more simply organize and modify the experiments.

2.5.3.4 Running Experiments on SDA Servers

After cloning the project in the user's directory on SDA Servers, experiments can be started. Here it is important that the specific python environment with installed required packages is activated. Section 2.5.1.4 explains how to create and activate such environments.

2.5.4 TU-HPC [1] Servers

In this project, over than 86000 experiments were executed. Since this number of experiments requires a very powerful computation power and our requirements were much higher than the capacity of SDA GPU Servers, we moved the evaluations to the HPC system of the University of Dresden.

2.5.4.1 HPC Systems

HPC (high-performance computation) system is an offer from the center for information services and high-performance computing (ZIH) of the university Dresden. This system contains multiple efficient hardware combinations for high-performance computation and a batch system that organizes and reserves these hardware combinations for specific tasks. This system provides an optimal environment for research in big data analytics and machine learning.

2.5.4.2 Batch System and Batch System Used in TU-HPC

It is essential to understand that Working with HPC is different from a local machine and SDA servers in many aspects. One of these major differences is the batch system. This section explains Batch System and discusses the differences between such systems with other Linux-based systems that are more usual to use in computer science.

Unlike other standard Linux based systems, a user cannot interact with the hardware controlled by a batch system. Instead, the user creates a job file - like punch cards in the early ages of computation- and submit this file to the batch system. This file contains all the necessary information for the batch system to queue and prioritize the job based on its resource requirements. These job files then get submitted to compute nodes, where the actual execution occurs. The batch system used in HIZ HPC is SLURM [12]. Normally a job file contains hardware requirements like the number of CPUs and GPUs, required memory for each task, and a wall-time that specifies the amount of time the job might reach. After reaching the wall-time, the job will be automatically canceled by the batch system. Section 2.5.4.3 describes job files and their submission to compute nodes with an example.

Another major difference between a normal development environment and the HPC system is log files. Since users cannot interact with the actual hardware which executes their program, they cannot see if something went wrong or see the program logs and outputs. This avoids a real-time interaction between a developer and the running program. Although logs and program outputs cannot be seen, they are still stored. These can be found in the output file in the same directory within the workspace where the job file is submitted. Workspaces are explained in section 2.5.4.3.

In general, working with HPC has some positive and negative aspects. Having a good overview of all, possibly thousands of jobs is one of the positive aspects. In addition, having dedicated resources for every single job ensures extra safety. Not being able to interact with running programs and also providing a close estimation of resource usages before starting each experiment are two disadvantages of the HPC system. Especially that each experiment takes different amounts of time to finish. One point which can be taken as both positive and negative point is that maintenance of the whole system is centralized which means issues like electricity cuts or internet disconnections are rare but if they happen, every single job is affected.

2.5.4.3 Preparing The HPC to Execute Projects

Before running any experiment, some preparation must be done on HPC servers, for example, allocating a workspace, cloning the code, and creating a job submission file. In this section, these steps are explained in detail.

Workspace	Duration	Description
SSD	30 Days	High-IO file system.
Scratch	100 Days	Spinning disks with high streaming bandwidth.
Warm_archive	1 Year	High capacity file system based on spinning disks.

Table 2.3: List of available Workspaces on TU-HPC

Logging in: The first step is to log in to HPC system. The easiest way to log in to the server is via SSH. For this, a TUD account is needed which is requested from a supervisor and is created by the HPC support team. Since an SSH connection is only possible within the TUD network, a VPN connection (https://tu-dresden.de/zih/dienste/service-katalog/arbeitsumgebung/zugang_datennetz/vpn) is required, when connecting from outside of the university's local network. The same username and password for SSH connection is used for VPN. After connecting via VPN to the TUD network, a connection via SSH can be established to the HPC server with the following command: `\$ ssh username@taurus.hrsk.tu-dresden.de`

Where `username` is to be replace with a provided one from the HPC support team.

Workspaces On an HPC system, compute nodes do not have access to `users` directories. That is the reason it is necessary to allocate a special directory between the user's directory and the compute nodes which is accessible by compute nodes. There are different types of workspace-directories (called workspace from now on). In Table 2.3 a list of all available workspaces on TU-HPC and their details is demonstrated.

For this thesis, both Scratch and SSD workspaces are used. There have not been significant performance differences observed.

Initializing Workspaces A Scratch workspace can be allocated with the following command in the user directory:

```
\$ ws_allocate -F scratch -r 10 -m name.lastname@tu-dresden.de Name 100
```

Table 2.4 explains each flag or option in workspace allocation's command line.

Workspace extension It is possible to extend the lifetime of any Workspace two times. Scratch workspaces are an exception and can be extended up to 10

Flag/Option	Description	Allowed Values
`-F`	File system.	Scratch, SSD, Warm_archive.
`-r`	How many days before workspaces' lifetime should a notification be sent.	Any integer between 1 and the file system's maximum lifetime.
`-m`	Email address to which the notification will be send.	Any valid email address.
Name	Desired name for the workspace.	Valid strings.
100	Duration of the workspace in days.	Any integer between 1 and the file system's maximum lifetime.

TABLE 2.4: List of flags and options in workspace allocation's command line

times. After a workspace's lifetime reaches its end, its data will be archived for some time and then removed automatically from the system.

A workspace's duration can be extended with the following command:

```
\$ ws_extend -F scratch NameOfTheDirectory01 100
```

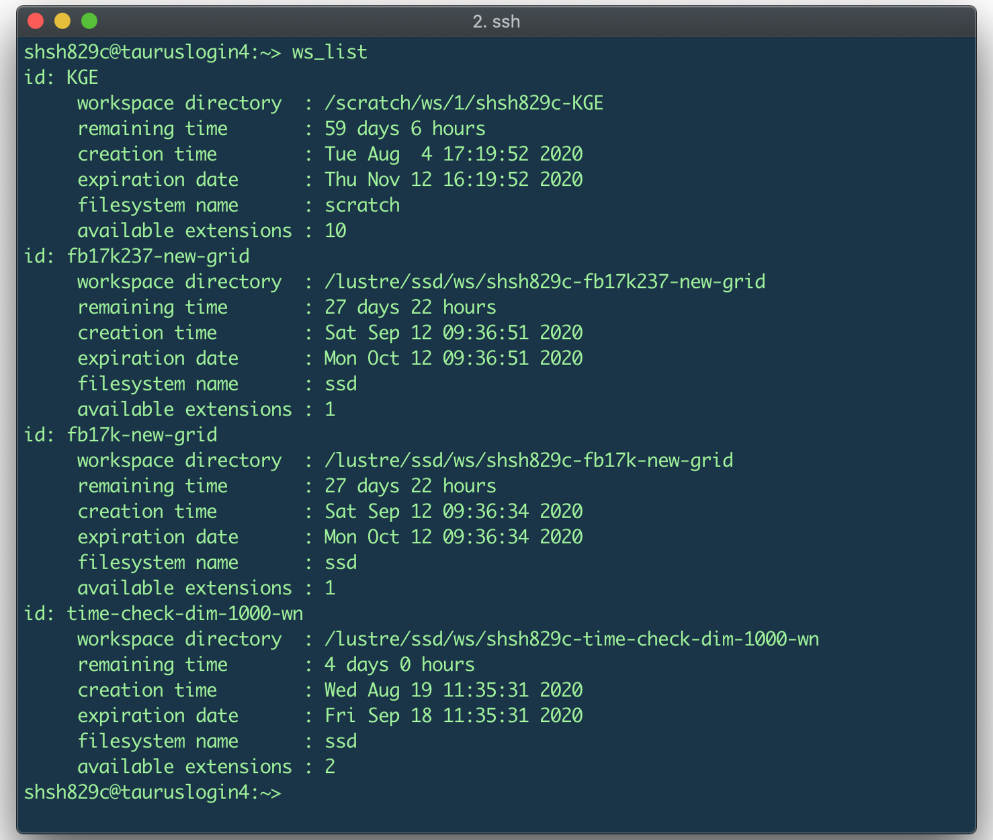
All working workspaces which are created by an account can be listed with the command `\$ ws_list` as shown in the figure 2.4.

Cloning code For cloning the repository into the workspace, It is necessary to navigate to the workspace's directory and follow the steps explained in section 2.5.1.2.

Submitting a job After cloning the code, jobs can be submitted to the compute nodes. To do so, a job file is needed. A job file contains all necessary information for SLURM to allocate dedicated resources to the job. This information and its valid values are explained on a job file example in this section.

An example of a job submission file is demonstrated in Listing 7. Such files can be submitted to compute nodes with `sbatch jobfile\_name` command. Table 2.5 gives an overview of job options which can be used in a job file.

Array Jobs Array jobs provide the possibility to run similar jobs as a group of jobs. This is helpful for job monitoring and organizing experiments. Array job



```
shsh829c@tauruslogin4:~> ws_list
id: KGE
     workspace directory  : /scratch/ws/1/shsh829c-KGE
     remaining time       : 59 days 6 hours
     creation time        : Tue Aug  4 17:19:52 2020
     expiration date      : Thu Nov 12 16:19:52 2020
     filesystem name      : scratch
     available extensions : 10
id: fb17k237-new-grid
     workspace directory  : /lustre/ssd/ws/shsh829c-fb17k237-new-grid
     remaining time       : 27 days 22 hours
     creation time        : Sat Sep 12 09:36:51 2020
     expiration date      : Mon Oct 12 09:36:51 2020
     filesystem name      : ssd
     available extensions : 1
id: fb17k-new-grid
     workspace directory  : /lustre/ssd/ws/shsh829c-fb17k-new-grid
     remaining time       : 27 days 22 hours
     creation time        : Sat Sep 12 09:36:34 2020
     expiration date      : Mon Oct 12 09:36:34 2020
     filesystem name      : ssd
     available extensions : 1
id: time-check-dim-1000-wn
     workspace directory  : /lustre/ssd/ws/shsh829c-time-check-dim-1000-wn
     remaining time       : 4 days 0 hours
     creation time        : Wed Aug 19 11:35:31 2020
     expiration date      : Fri Sep 18 11:35:31 2020
     filesystem name      : ssd
     available extensions : 2
shsh829c@tauruslogin4:~>
```

FIGURE 2.4: List of available working spaces

files are similar to normal job files with small differences. There are different ways to create array job files. In this thesis, a simple method is used, which helps run experiments with a lot of different combinations of hyper parameters. The only option which needs to be added to a provided job file in the previous section is `-array=` with the number of array jobs which will be executed by the submitted job file. E.g. `-array=1-250` means there will be 250 jobs executed within this array job.

In addition to the above-mentioned option, the following line should replace the absolute or relative path to the python file:

```
srun \$(head -n \$SLURM_ARRAY_TASK_ID commands.txt | tail -n 1)
```

This line tells the SLURM that there is a `commands.txt` file which contains all running commands. `command.txt` contains commands such as the command in Listing 8 which are separated by a new line.

Listing 7 A example of a HPC job file.

```
#!/bin/bash

#Submit this script with: sbatch thefilename
#SBATCH --time=25:00:00   # walltime
#SBATCH --nodes=1   # number of nodes
#SBATCH --gres=gpu:1
#SBATCH --ntasks=1      # limit to one node
#SBATCH --cpus-per-task=4  # number of processor cores (i.e.
↪  threads)
#SBATCH --mem-per-cpu=3875M   # memory per CPU core
#SBATCH -J "wn18-test-sym-run"   # job name
#SBATCH -o wn18-test-sym-run%j.out
#SBATCH --mail-user=name@mailbox.tu-dresden.de   # email address
#SBATCH
↪  --mail-type=BEGIN,END,FAIL,REQUEUE,TIME_LIMIT,TIME_LIMIT_90
#SBATCH --reservation=p_ml_nimi_137
#SBATCH -A p_ml_nimi
#SBATCH --array=1-162

#source /home/username/venv/env1/bin/activate

srun $(head -n $SLURM_ARRAY_TASK_ID commands.txt | tail -n 1)

exit 0
```

Listing 8 Example of a command line in `ands.txt`

```
python3 ../codes/run.py --do_grid --cuda --do_test --data_path
↪  ../data/FB15k --model RotatE -d 50 --negative_sample_size 10
↪  --batch_size 512 --gamma 1 --adversarial_temperature 1
↪  --negative_adversarial_sampling -lr 0.01 --max_steps 400000
↪  -save
↪  ../models/RotatE/rotate/FB15k/dim-50/gamma-1/learning-rate-0.01
↪  /batch-size-512/negative-sample-size-10/ -de --loss rotate
```

An automatic way to create such commands is implemented for this thesis and is explained in Chapter 3.

Job Monitoring TUD-HPC system provides a helpful job monitoring system [13] where a user can observe submitted, running, and finished jobs along with other useful information like time and energy-consuming, memory usage, and other information in the form of diagrams and also detailed view for further observation.

Option	Description	Allowed Values
`--time`	Walltime. Job will be canceled if it reaches walltime.	Any value between 00:00:00 up to 7 days.
`--nodes`	Number of utilized nodes.	Depending on the partition it can start from one up to 1328 nodes.
`--ntask`	Number of tasks this job will start by executing the command file.	Integer. Default value 1. Limited to one node.
`--gres=gpu:`	Number of utilized GPUs for the job.	Depending on the partition can vary between 1 and 4.
`--cpus-per-task`	Number of utilized cores for each task.	It should match the number of threads used by the program.
`--mem-per-cpu`	Amount of allocated memory for each CPU in MB.	Integer + M.
`-j`	Job name.	String.
`-o`	The job's output file name where logs are stored.	Recommended to use %j in the name to have the unique job ID to avoid overwriting.
`--mail-user`	Email address where job notification will be sent.	Valid email address.
`--mail-type`	Events which trigger an email notification.	BEGIN, END, FAIL, REQUEUE, TIME_LIMIT, TIME_LIMIT_90.
`-A`	Project name.	Case sensitive project name.

Table 2.5: List of flags and options in a job file

Figure 2.5 represents some diagrams which were used in this project created with the job monitoring tool.

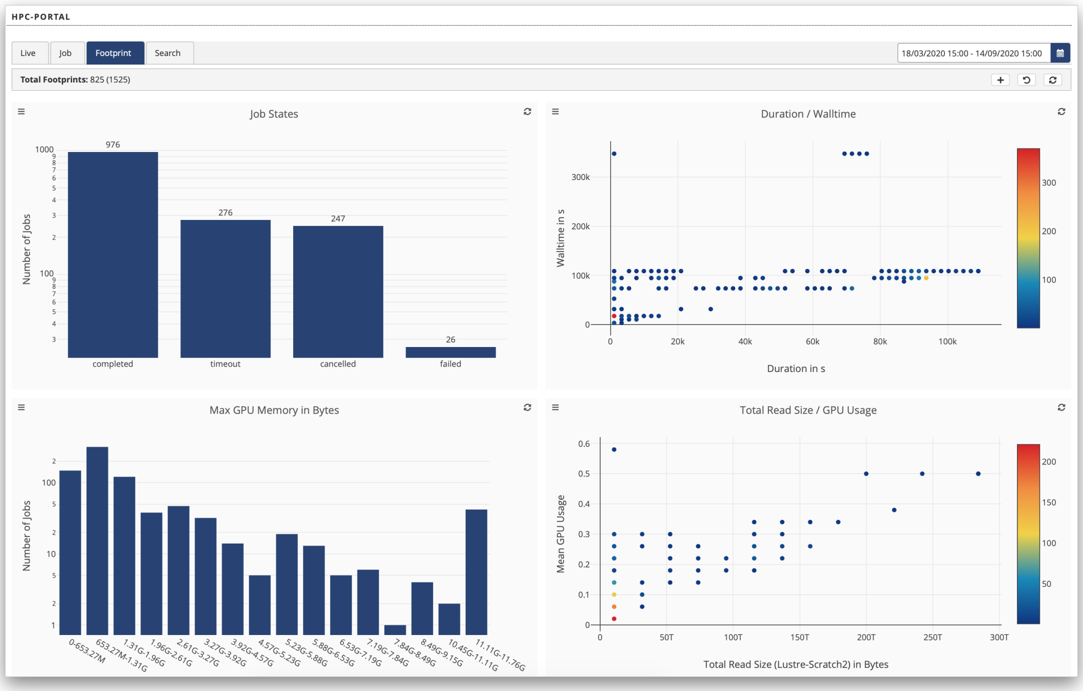


FIGURE 2.5: Some job monitoring diagrams

Chapter 3

3. KGExperimentify System Design

To overcome a wide range of evaluations settings, i.e. different hyperparameter and multiple datasets, a semi-automatic system is designed which is capable of running the evaluations directly on multiple GPU or prepare them to run on HPC compute nodes. This system which is called **KGExperimentify**. In addition to the aforementioned features, it contains the implementation of different state-of-the-art KGE models as well as loss functions.

This chapter describes the architecture and workflow of KGExperimentify, goes deeper into details, and explains each step of its workflow and the reason behind them. To understand the evaluation's process more precisely, this process is divided into three steps pre-execution, run, and post-execution. Figure 6.1 illustrate these steps in their orders.

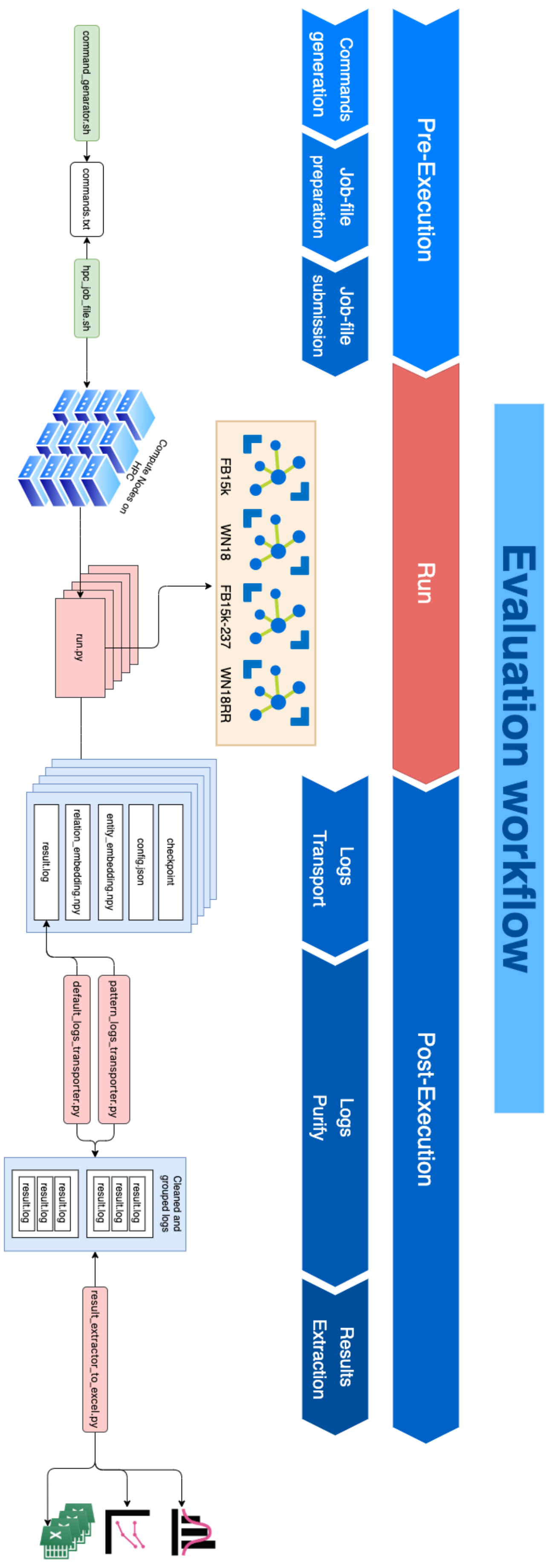


FIGURE 3.1: Evaluation workflow.

3.1 Pre-Execution

Every necessary step before the start of the evaluation will take place in the pre-execution part, for example, creating the experiment commands and configuring the job file. Here two cases are explained. The generated commands are either executed directly on GPUs or stored into a secure file for execution on HPC servers.

3.1.1 Command Generation

The first step is to create the commands which will later be used on compute nodes of an HPC system to start training and evaluating with a specific hyperparameter. This step is done in `commands_generator.sh`.

`commands_generator.sh` is a shell script which receives separate arrays of possible values for each hyperparameter. Users can specify the required dimensions, batch sizes, learning rates, datasets, models, loss functions, negative sample sizes, temperatures, gammas, and max steps as arrays. Then, depending on its run configuration, this file performs two different actions.

In order to run the experiments directly on GPUs, for example when evaluations are planned to run on GPU servers, the flag `RUN_ON_GPU` must be set to `true`. KGExperimentify then takes the user-defined number of available GPUs in `AVAIL_GPUS` and runs all possible combinations of the aforementioned configuration parallel on all of the GPUs. The number of simultaneously running experiments depends on the free available memory of each GPU. In order to ease the process, a threshold for a free amount of memory is used. This threshold is also configurable by the user in `FREE_MEM_THRESHOLD`. KGExperimentify then checks all available GPUs, and whenever more memory than the defined threshold by the user is available on a GPU, a new experiment is started. If a GPU is already occupied, the next GPU will be checked for available memory. It continues this process until all of the experiments are started.

It is important to notice that this method is efficient only if executed on a dedicated GPU server. Since this script starts many experiments in parallel, there will be much CPU power needed to transport datasets to RAM and then to GPUs. If other processes are running on the server, the server will slow down considerably. By default, the GPU's free memory threshold is set to 2GB. This can vary for different variables, especially datasets and hidden dimensions. From observations, the lowest value of GPU memory for an experiment is about 500 MB, and the highest is value is about 11.5 GB (This is the result of a personal observation of many experiments and not a calculated value.).

If it is intended to run the experiments on an HPC system, the flag `RUN_ON_GPU` option must be set to `false`. In this case, KGExperimentify creates the execution command lines and stores them automatically in a file named `commands.txt` in the same directory. The `commands.txt` file is then used to execute each command in different jobs on HPC.

3.1.2 Job-file Preparation

To run all created commands, a Job-file which executes each command on compute nodes is needed. Since there are multiple commands to run, an array job needs to be configured. The job file in this project is `hpc_job_file.sh`. In section 2.5.4.3 job files are introduced. The most important parameters in an array Job-file are the name, output name, and the number of commands. Besides, the time wall is needed to be set. From experience, it is recommended to set the wall time to 25 hours for WN18 and WN18RR data sets and 40 hours for FB15K and FB15k237. In most cases, depending on the hyperparameters' combination, an evaluation takes between 50% to 70% of the wall time. In some cases, it may reach 80%, and in very rare cases, some evaluations will reach over to 90% of the wall time.

3.1.3 Job-file Submission

After the first two steps, both `commands.txt` and `hpc_job_file.sh` in addition to data sets are needed to start evaluations. A Job-file can be submitted to the compute nodes by `sbatch JOB_FILE_NAME` in the Job-file's directory on users node of HPC systems.

3.2 Post-Execution

From experience, experiments will not always run and finish perfectly. That is the main reason that after running the experiments, the result logs may contain incomplete, corrupt or duplicated files. The post-execution steps are the download and purification of result logs from embedding files and extraction of results and storing them in into human-readable formats of useful log. The post-execution step contains downloading and purification result logs, separation of the result logs from embedding files, and extraction of the results from purified log files and storing them in a human-readable format.

3.2.1 Transferring Trained Models and Results

Depending on the types of workspaces on HPC Computers, they have a limited lifetime and need to be extended after some time to prevent them from being removed. Therefore, all successfully trained models are downloaded into a safe location for future use. Due to the size and number of results, it is recommended to download these files through `taurusexport.hrsk.tu-dresden.de`. TU-Dresden offers this server for the transportation of large files. It is also recommended to use `rsync` command to skip already downloaded files in case of connection interruptions. An example of copying results from the remote location `A` to the local location `B` using a `SSH` connection and `rsync` command is provided below:

```
rsync -ravP -e ssh USERNAMEc@taurusexport.hrsk.tu-dresden.de:A B
```

3.2.2 Purifying The Results

Purifying the result logs is mandatory to have noiseless results. For this step, there are two written scripts named `pattern_logs_transporter.py` and `default_logs_transporter.py`. The first script is responsible for purifying the patterned based results, which in this thesis are **symmetric**, **inverse**, **one_to_many** and **implication**. Patterns, as well as KGE models, can be set by the user to ease the extraction process. This feature comes in handy when the user evaluated only a specific model or pattern and wants to only extract these results. The second script is responsible for the extraction of results on the default test sets.

Both scripts have an option which allows the user to set the dataset, from which the result files need to be organized. These scripts remove every duplicate result log and also eliminate every incomplete result log. In other words, it takes only one result logs from each evaluation, which contains results, and transports this file into a specific log directory. This means the user can run the incomplete evaluations multiple times without taking care of the result files. If no result log contains results, it transports the largest log into the specific log directory to ensure the security of the files' integration.

Each log file's name contains a prefix part, the model's name with which the dataset is trained, and the loss function's name, which is used to train the model. The user can modify all these names. Despite a specific name for each log and the use of a timestamp in the log files' name, there will be cases where logs have the same name, although they contain different results. This is due to running multiple evaluations in parallel where the timestamp is the same. For this reason, a random number in the range of 0 to 999999999 as a prefix is added to every log file after transporting them in new logs directories. Figure 3.2 represents an example of results' directory tree.

3.2.3 Extracting Results into Human-readable Tables

With the increasing importance of data science and artificial intelligence, more and more people are involved in this field than before. Not everyone has the skills to generate their desired information through code manipulation. Although having machine-readable data is the first requirement to work on big data, human-readable and intractable data is always a proper way to experiment and analyze the results. Besides, navigating through thousands of log files pursuing a good result takes much time and highlights the dependency between data scientists and data analytics. Therefore, the important information of each evaluation log is converted to an Excel-table line, which contains all results for a dataset. Using these tables, a non-professional user, like students who are starting to work on this topic, can have a good overview of the goal of these experiments and how close the performances of different models are together.

The python file `logs_extractor.py` is responsible for extracting information from cleaned up log files and converting them to excel entries. It gets a dataset's name and navigates through all log files under the log directory of the dataset and converts them into an excel file that is then stored under the excel directory. In the `output_modes` array variable, users can pass the desired patterns such as default, **symmetric** or **implication**.

```
results/
 ├── wn18/
 ├── wn18rr/
 ├── fb15k-237/
 └── fb15k/
      ├── excel/
      │    ├── results-fb15k-default.xlsx
      │    ├── results-fb15k-symmetric.xlsx
      │    ├── results-fb15k-inverse.xlsx
      │    └── results-fb15k-implication.xlsx
      ├── logs/
      │    ├── default/
      │    │    ├── ComplEx/
      │    │    │    └── 573344test_ComplEx_margin_ranking_2020_10_06__at_03_30_02.log
      │    │    ├── DistMult/
      │    │    ├── QuatE/
      │    │    ├── RotatE/
      │    │    └── TransE/
      │    └── pattern/
      └── models/
           ├── ComplEx/
           ├── DistMult/
           ├── QuatE/
           ├── RotatE/
           └── TransE/
```

Figure 3.2: Results directory structure.

Chapter 4

Evaluation Configuration

There are many different configurations like models, loss functions, and hyperparameters which are used in each experiment and play a big role in the end results of the experiments. This section describes all of those configurations used for the experiments whose results are presented in chapter 5.

4.1 Implementation

Implementation of models and loss functions may have minor effects on the results of evaluations. The implementation of models and loss functions which are used in this thesis are found in this repository [10].

4.2 Datasets

To deliver comparable results and stress the improvement, most research groups work on specific well known Knowledge Base datasets like WordNet(WN)[14] or Freebase(FB)[15]. These types of datasets contain entities representing objects or entities in the world and relationships that connect these entities. This section describes the used datasets in this thesis and goes further to explain the pattern extraction process, from which new test sets are created from the default test sets used in WN and FB.

Dataset	Entities	Relationships	Train triples	Valid triples	Test triples
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466

Table 4.1: The statistics of datasets FB15k and FB15k-237

Dataset	Entities	Relationships	Train triples	Valid triples	Test triples
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134

Table 4.2: The statistics of datasets WN18 and WN18RR.

4.2.1 Knowledge Base Datasets

Freebase Freebase is a large, shared, and open Knowledge Base(KB) containing data from different real world contexts. It was introduced as a shared database for entity-relationship data. Besides harvested data from sources like Wikipedia[16] and MusicBrainz[17], a lot of the structured entries are added by normal users.

There are two subsets of Freebase KB used in this thesis, FB15K[18] and FB15K-237[19]. FB15K was introduced to accelerate the experiments, but due to a significant test leakage through inverse relations, FB15K237 was introduced, in which all triples in the test set are removed, whose inverse triples existed in the training set.

Table 4.1 indicates the exact numbers of entities, relationships and triples in both FB15k and Fb15k-237.

Wordnet WordNet is a lexical database for English where verbs, adjectives and nouns are the entities. These entities are then connected to each other by relationships such as **synonym**, **hypernym**, **part_of etc**. Like Freebase, two subsets of WordNet database are used in this thesis: WN18[18] which contains all 18 found relations in WordNet's synsets and WN18RR[20] which was introduced as a subset of WN18 with only 11 relations, which are not mutually interchangeable.

Table 4.2 indicates the exact numbers of entities, relationships, and triples in both WN18 and WN18RR.

Dataset	FB15k	FB15k-237	WN18	WN18RR
# Original test set triples	59,071	20,466	5,000	3,134
# Symmetric test set triples	3,583	300	1,169	1,169
# Implication test set triples	1,974	387	3,478	2,439
# Inverse test set triples	3,749	277	2,682	2,381
# one-to-many test set triples	1,400	410	161	222

Table 4.3: Size of each pattern-based test set compared to the original test set of each dataset. Each pattern-based test set is created based on the top three relations with the highest occurrences in each pattern in the training sets.

4.2.2 Relational-patterned Datasets

Substructures in knowledge graphs often form shapes of relational patterns for which the legacy of logic-based reasoning comes into a plan for facilitating inference of the learning approaches. Therefore, one crucial aspect of working with learning methods on knowledge graphs is their ability to capture such patterns both in structure and semantic. To gain detailed information about each relation's characteristics in every dataset, all four training sets' triples are analyzed by a script based on this work[21]. Upon each analysis, a figure of merit was created for each dataset to see which relations are more likely to demonstrate a pattern.

We denote E as the set of entities and R as the set of relations for the dataset.

Table 4.3 indicates the size of each pattern-based test set for all four datasets. Only the top three relations with the highest occurrences in the training sets are taken to create pattern-based test sets.

Symmetric A relation is defined as *symmetric* if for a triple (h,r,t), there is a another triple in training set with (t,r,h). An example of such relation is "*is_friend*".

Based on the mentioned definition, all relations in the training set are examined, and the results are extracted into a figure of merit which shows the number of *symmetric* occurrences for each relation. Figure 4.1 indicates the extracted figure of merit from the WN18 training set. A new symmetric-test set is created which only contains triples with the top three relations with the highest *symmetric* occurrences in the statistic. This symmetric-test set is created for all four datasets. The results are shown in Chapter 5.

Table 4.1 indicates the figure of merit for *symmetric* relations in WN18 datasets. Similar statistics for other used datasets in this thesis can be found in Appendices.

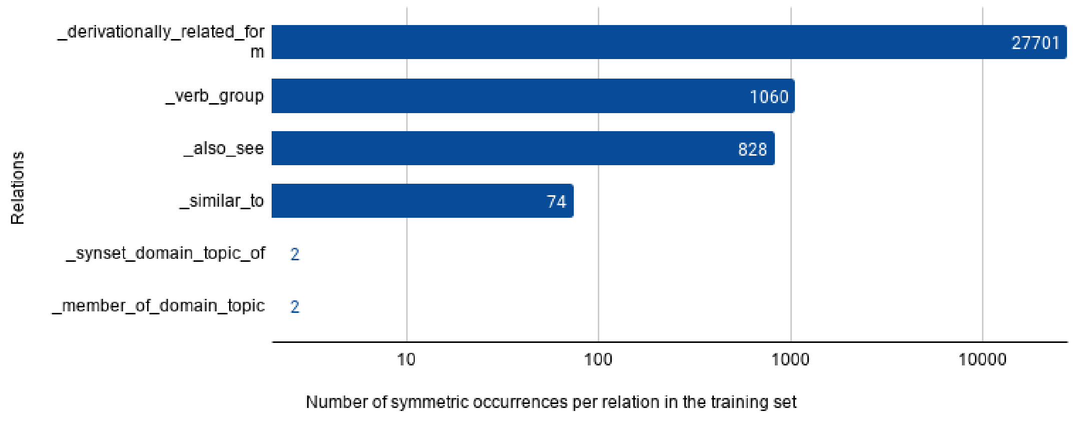


FIGURE 4.1: Symmetric occurrences for each relation in the WN18 dataset.

Implication For a triple (h,r,t) a relation r'≠r does count as *implication* if triple (h,r',t) exists. Based on this definition, for each triple (h,r,t) in the training set and every relation r'∈R is checked to see how many triples (h,r',t) exists in the training set. The total number of such occurrences for each relation is then stored and displayed in Figure 4.2 shows for the WN18 training set. An implication-based test set is then created, which contains the three relations with the highest number of *implication* incidences.

Inverse To create a test set containing only reversible relations, for each triple (h,r,t) of the training set is checked if there is a triple (t,r',h) in the training set where $r \neq r'$. If such a triple exists, r and r' are inverse relations. After creating a statistics of such triples (h,r,t) with relations like r in each training set, we have a set of relations R, for which there is another relations set R', where:

$$\{\forall\, r \in R \rightarrow \exists\, r' \in R' : r \neq r' \; and \; r, r' \text{ are inverse } \}\,. \tag{4.1}$$

Figure 4.3 shows the figure of merit for *inverse* relations in the WN18 training set. Now the default test set is filtered by relations in R to create a test subset, such that there are only triples (h,r,t) in the subset where r ∈ R.

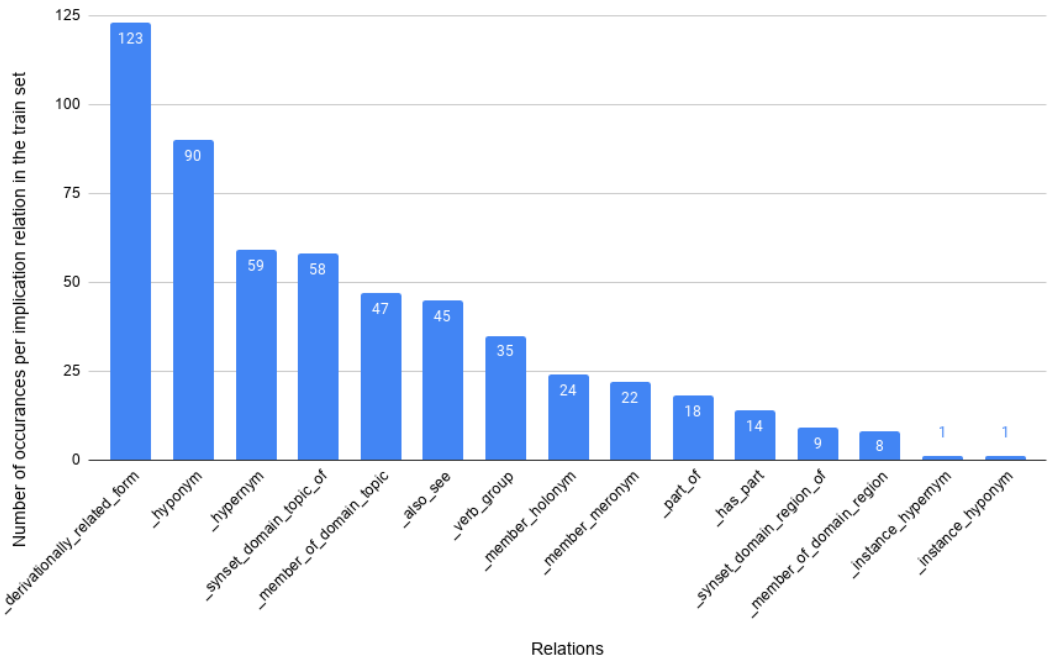


FIGURE 4.2: Number of implication occurrences for each relation in the WN18 dataset.

1-N *One_to_many* relation is defined as: If for an entity h ∈ E and r ∈ R there is subset of E' ∈ E with: ∀ t' ∈ E' there is a triple (h,r,t') in the training set, we call the r a *one_to_many* relation.

r is a *one_to_many* relation if for:

$$\text{For } h \in E \text{ } and \text{ } r \in R, \exists\, E' \subseteq E : \forall\, t' \in E' \Rightarrow \exists(h, r, t) \in \text{training set} \tag{4.2}$$

To identify the candidate relations in the training set, for each unique relation r ∈ R is checked how many *one_to_many* triple in the training set exist. For each relation, the minimum, maximum, and average of *one_to_many* occurrences are noted and saved into a figure of merit to compare how likely it is for a relation to be a *one_to_many* relation.

A *one_to_many* test set is created for each dataset based on the top three relations with the highest average *one_to_many* occurrences.

Figure 4.4 indicates the average one-to-many occurrences for each relation in the WN18 training set. Overview tables for WN18RR, FB15k, and FB15k-237 in all four test sets are attached in the appendix section.

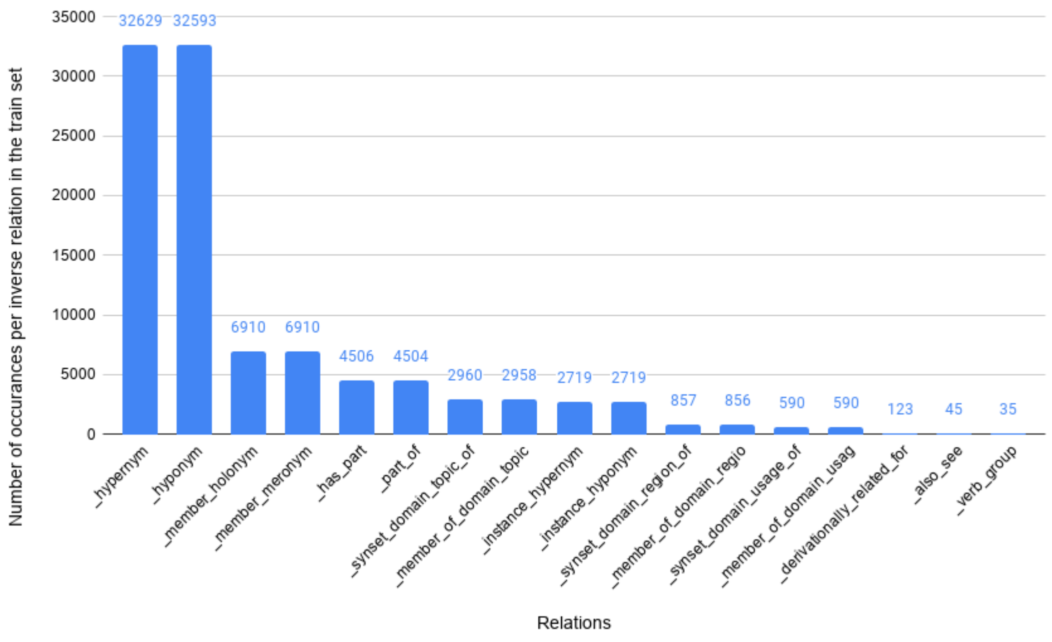


FIGURE 4.3: Number of inverse occurrences for each relation in the WN18 dataset

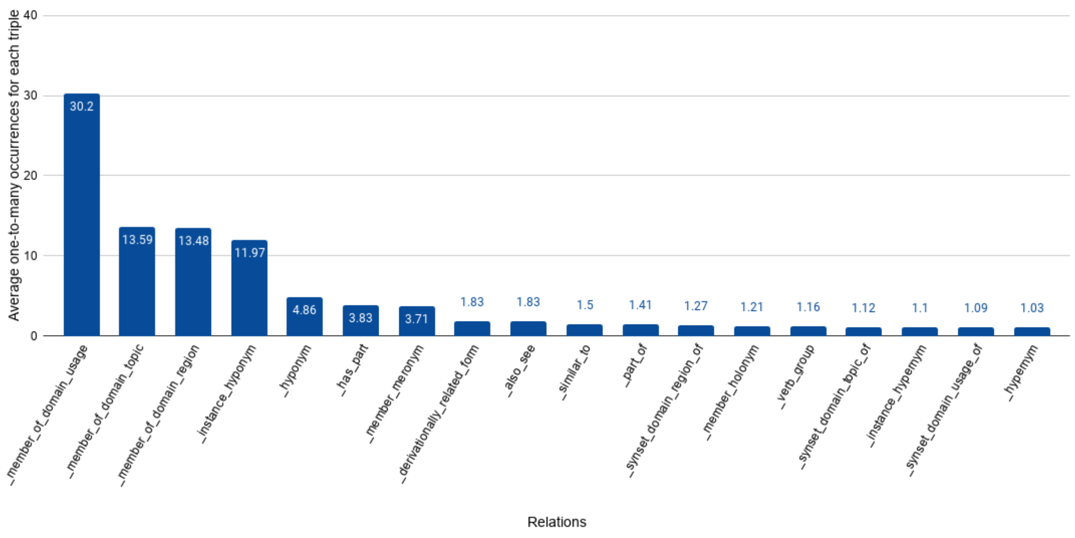


FIGURE 4.4: Average *one-to-many* occurrences for each relation in the WN18 dataset.

4.3 Models

Knowledge Graphs (KG), as the name suggests are graphs, in which the nodes of the graph are from a specific set, which is called *entities* and edges are from another set, which is called *relations*. The *entities*(nodes) are connected to each other through the *relations*(edges). These Knowledge Graphs can be built in different domains and contexts. An example of a knowledge graph with the entities set E = {Mary, John, Alex, Berlin, Germany} and relation set R = {"capital_of", "mother_of", "married_to"} is demonstrated in figure 4.5.

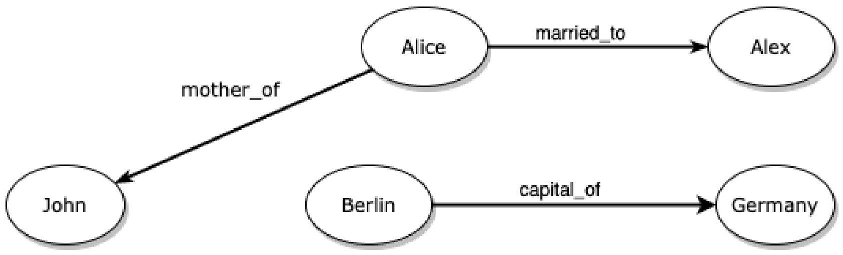


FIGURE 4.5: An example of Knowledge Graph.

Triples are one important units of KG which are the form (h,r,t) where h,t $\in$ E and r $\in$ R. Examples of such triples in aforementioned KG are: {(Mary, "married_to", Alex), (Berlin, "capital_of", Germany), (Mary, "mother_of", "John")}

Now that the concept of KG is clear, **Knowledge Graph Embedding(KGE)** can be introduced. KGE is the representation of a KG in a low dimensional space. A representation of the aforementioned KG in a twodimensional space is shown in Figure 4.6.

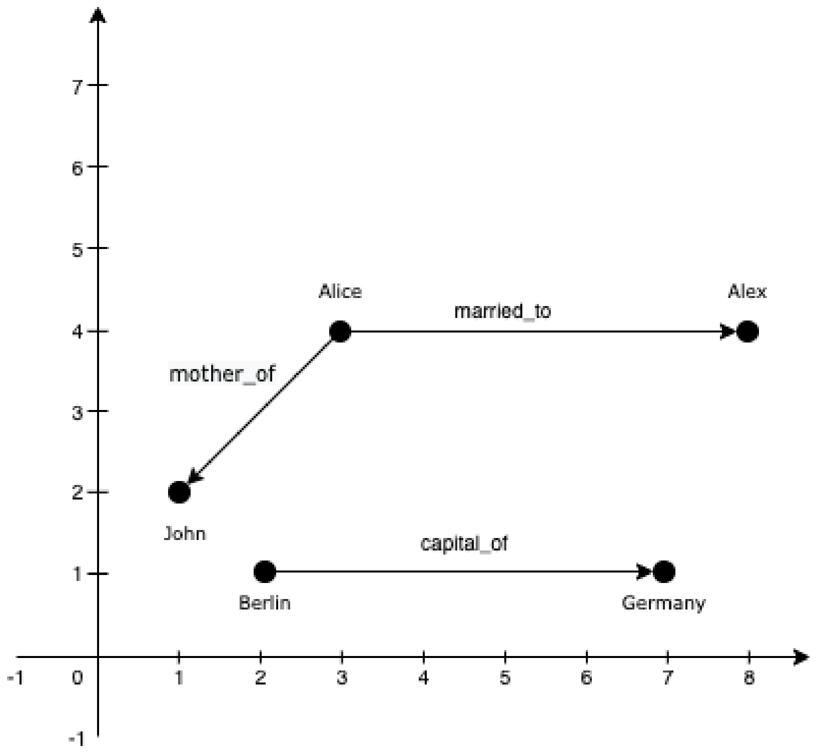


FIGURE 4.6: The representation of KG in a two-dimensional space

Gaining knowledge from such KGEs is one of the KGE models' duties. These models assign vectors in a low dimensional space to entities and relations. Then they guess *triples* of *(entity, relation, entity)* and calculate their score using a model-specific score function. These scores are then used in a loss-function, which will be minimized. This means correct guesses or guesses with a high likelihood of being correct, get a higher score, and incorrect guesses get a lower score. This process is repeated so much until most of the guessed triples are true. Then the model is trained and can evaluate new datasets.

There are many useful KGE-models introduced from 2011. In this thesis, five state-of-the-art models are used for the evaluation.

For the rest of this section the following rules apply:

- h, t $\in$ E = { All entities in a dataset }.
- r $\in$ R = { All relations in a dataset }.
- $<$ A,B $>$ denotes the dot product between two vector A and B which are in $\mathbb{R}^k$ and $\mathbb{C}^k$ for ComplEx and RotatE.

- $\parallel$ A $\circ$ B $\parallel$ denotes the element-wise product between A and B, where both A and B have the same dimension.
- $\|A\|$ denotes the absolute value of A.

4.3.1 ComplEx [2]

ComplEx is a Knowledge Graph Embedding model which represents entities and relations in a low dimensional complex instead of real space. It then uses the dot product for a triple (h,r,t), as shown in the equation 4.3, to find its score. In a dot product between two vectors in a complex space(Hermitian product), a Hermitian transpose (conjugate transpose) of one of the vectors is used. Unlike the dot product between two vectors in a real space, a dot product in a complex space is asymmetric. This means, depending on the appearance order of the entities in the score function, the results are different. ComplEx benefits this property to gain information about antisymmetric relations.

Complex is able to detect and learn the following patterns: symmetry, anti-symmetry, asymmetry, inversion.

$$\mathrm{Re}\left(< r, h, \bar{t} >\right) \tag{4.3}$$

4.3.2 DistMult [3]

Like ComplEx, DistMult uses the dot product between heads, relations, and tails as its score function. The difference between DistMult and ComplEx comes from the characteristic of the dot product in real and complex spaces. The dot product in real space is symmetric, which means replacing h and t in <h,t> does not affect the result. Based on this property, DistMult can handle symmetric relations.

DistMult is able to detect symmetric patterns.

Score-function:

$$< r, h, t > \tag{4.4}$$

4.3.3 QuatE [4]

QuatE is a KGE model which represents entities and relations in quaternion vectors. Quaternion are 4 dimensional extension of complex numbers and quaternion inner product which is used in QuatE's score function 4.5 is a rotation in this space. The advantage of Quaternion rotations is the greater number of available dimensions for rotations.

QuatE is able to detect and learn the following patterns: Symmetry, anti-symmetry, inversion

$$\phi(h,r,t) = Q'_h \cdot Q_t = \langle a'_h, a_t \rangle + \langle b'_h, b_t \rangle + \langle c'_h, c_t \rangle + \langle d'_h, d_t \rangle \tag{4.5}$$

where $Q_x = a + b\,\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where a,b,c and d $\in \mathbb{R}$ and i,j and k are imaginary units.

4.3.4 RotatE [5]

RotatE is one of the state of the art KGE models, that like ComplEx, represents entities and relations in a complex space. The goal of RotatE, as the authors state, is to aim for a more powerful model in regards to patterns. Since RotatE uses r as a rotation of the head entity h towards the tail entity t as its score function 4.6 using an element-wise product between h and r, it is able to distinguish between symmetry, inversion, and composition.

RotatE is able to detect and learn the following patterns: inversion, composition, symmetric, antisymmetric.

$$\|h \circ r - t\| \text{ where h,r,t} \in \mathbb{C}^{\text{k}}. \tag{4.6}$$

4.3.5 TransE [6]

Like many other KGE models, TransE assign vectors in low dimensional space to the entities and relations. It then uses relation vectors as translation vectors to translate head entities to tail entities. The goal is to bring h+r-t to zero while keeping h and t as different as possible. This model can infer inversion and composition as translation between vectors does this. Equation 4.7 shows the score function used in TransE.

TransE is able to detect and learn the following patterns: Anti-symmetry, Inversion, Composition.

$$-\|h + r - t\| \tag{4.7}$$

4.4 Loss Functions

Although each model's score function has a big impact on the its accuracy, loss functions play a big role in embedding models' performance as well. This is shown in Chapter 5 by training multiple KGE Models with a fixed hyperparameter configuration and three different loss functions. This section gives a detailed overview of the three used loss functions in this thesis.

For the rest of this section the following rules apply:

- $sf(h, r, t)$ is the score function for a positive triple (h,r,t).
- $sf(h, r, t)'$ is the score function for a negative triple (h',r,t').
- γ denotes the margin between positive and negative triples' scores.
- σ denotes the sigmoid function and $\log \sigma$ is the log-sigmoid function.
- α denotes the temperature of sampling.

4.4.1 Margin Ranking

Margin Ranking Loss tries to separate positive and negative triples by a margin of γ through the equation below:

$$Loss = max\,[\,0\,,\ (sf(h, r, t) - sf(h, r, t)' + \gamma\,]\,. \tag{4.8}$$

As shown in [7] Margin Ranking Loss cannot guarantee that the overall difference between the highest score of positive samples and the lowest score of negative samples is big enough such that all positive and negative samples are separated from each other by the margin γ. This issue was solved in adaptive_margin loss with the introduction of a new slack variable.

4.4.2 Adaptive Margin Ranking [7]

Adaptive Margin Ranking Loss's approach is to use a γ and a slack variable ξ to separate the score of positive and negative triples as follows:

$$Loss = [\mathrm{Relu}\,(sf(h,r,t) - \gamma + \xi)\ +\ \mathrm{Relu}\,(sf(h,r,t) + \gamma + \xi)] \tag{4.9}$$

In this approach, γ specifies the center of the space between the highest score of positive samples and lowest score of negative samples. The slack variable ξ is used to stretch or compress the margin between positive and negative triples' scores.

4.4.3 Negative Self Adversarial Sampling[8]

Negative sampling loss with self adversarial was introduced with RotatE KGE Models, that is the reason why it is used under the name "rotate" in this thesis.

Rotate loss function is a modified version of the original negative sampling loss [22] which is adapted for optimizing distance-based models. The other difference between these two loss functions is that they use different negative triples. RotatE's loss function uses an approach called self-adversarial, which uses negative triples for the current positive samples and uses the equation 4.10 as the weight of negative sample for each positive sample in the actual loss function(Figure 4.11).

$$p((h_j, r, t_j)' \mid \{(h_i, r_i, t_i)\}) = \frac{\exp\ \alpha\ sf(h,r,t)}{\sum_i \exp\ \alpha\ sf(h_i, r, t_i)}. \tag{4.10}$$

$$Loss = -\log\sigma(\gamma - sf(h,r,t)) - \sum_{i=1}^{n} p(h_i, r, t_i)' \log\sigma\left(sf(h_i, r, t_i)' - \gamma\right). \tag{4.11}$$

Values	**Variable**
Dataset	{FB15k, FB15k-237, WN18, WN18RR}
Model	{TransE, RotatE, ComplEx, QuatE, DistMult}
Loss Function	{adversarial , margin_ranking, adaptive_margin}
Hidden dimension	{10, 100, 1000}
Learning Rate	{0.1, 0.01, 0.05}
Gamma	{1, 10, 20, 30, 40, 50}
Negative Sampling Size	{10, 100, 1000}

Table 4.4: Overview of used configuration variables

4.5 Experiment Configurations' Statistics

Besides five KGE Models and three loss functions, there is a wide range of hyperparameter used for the experiments in this thesis. These hyperparameters have a direct effect on the accuracy and efficiency of models. Table 4.4 indicates all used configuration variables for the experiments during this thesis. 9720 experiments were generated as combinations between models, loss functions, and other hyperparameters. It is essential to notice that these 9720 experiments are repeated for each patterned test set(*symmetric, inverse, implication and one_to_many*) as well, which results in 48600 experiments. However, the actual number of executed parameters are noticeably higher due to technical issues such as job cancellations due to HPC maintenance. Figure 4.7 shows the number of executed experiments grouped by their average amount of running time.

There have been 86799 experiments executed in total. As a comparison, it would have taken a powerful PC over 13.8 years to run these experiments successively. Executing these experiments on Google's HPCs[23] with the lower yearly commitment price would cost 34,442$.

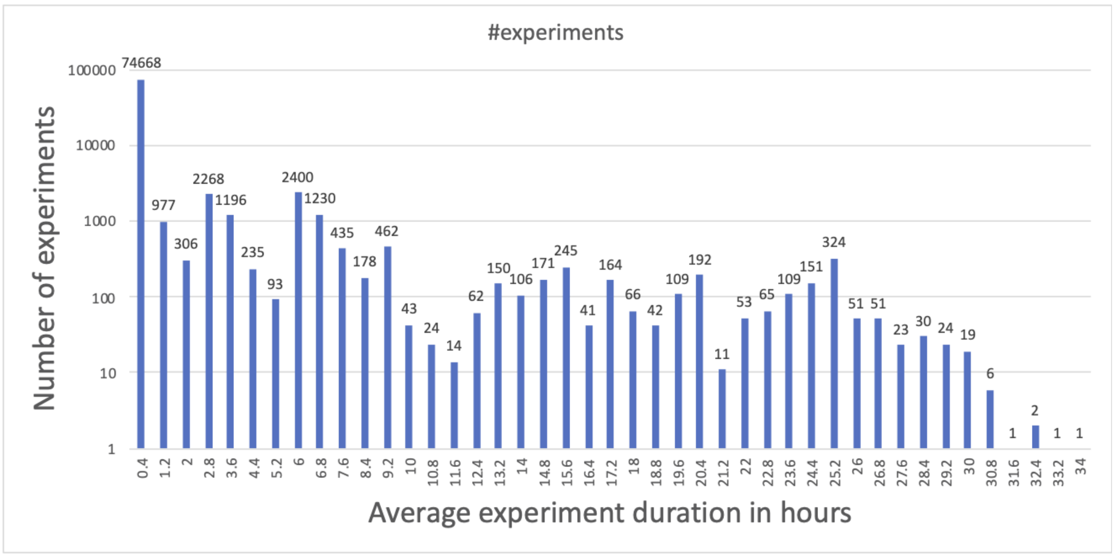


FIGURE 4.7: Number of executed experiments ordered by their average duration time.

Chapter 5

Evaluation and Results

A total of 86799 evaluation was done during this work. The results of these evaluations are grouped depending on the test sets which are evaluated and are presented in different subsections.

It is important to mention that the presented results in this section are the best overall results among all used evaluation configuration(hyperparameters). Some results with a specific configuration, for example, comparison on fixed dimension or models, are also presented.

All the results are presented with three digits after the decimal separator. The highest results in a row and columns are highlighted. In some cases, if multiple models' results are exactly the same, all those results are highlighted.

5.1 Evaluation Results on Original Test Sets

As the tables 5.1 and 5.2 indicate, in almost all four datasets and every metric, RotatE delivers the best overall results. QuatE shows the best HITS@1 in WN18RR, while the best Mean Reciprocal Rank is shared between both QautE and RotatE. For FB15k and FB15k-237, TransE performs much better on Mean Rank but again for all other metrics, RotatE wins the race in some cases with a minimal margin and in others with a obvious one.

Model	WN18					WN18RR				
	MR	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**	**MR**	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
ComplEx	779.429	0.945	0.942	0.947	0.950	9326.271	0.385	0.369	0.387	0.424
DistMult	1058.586	0.834	0.751	0.916	0.936	6525.354	0.434	0.404	0.444	0.499
QuatE	711.111	0.946	0.943	0.948	0.952	6297.141	**0.467**	**0.429**	0.473	0.521
RotatE	**291.488**	**0.948**	**0.943**	**0.951**	**0.956**	**4185.274**	**0.467**	0.427	**0.482**	**0.544**
TransE	366.388	0.823	0.718	0.930	0.951	6497.995	0.182	0.035	0.305	0.415

Table 5.1: Results of WN datasets evaluations.

Model	FB15k					FB15k-237				
	MR	MRR	Hit@1	Hit@3	Hit@10	MR	MRR	Hit@1	Hit@3	Hit@10
ComplEx	144.734	0.729	0.673	0.765	0.823	882.274	0.202	0.135	0.221	0.338
DistMult	67.416	0.762	0.691	0.817	0.874	214.618	0.315	0.225	0.346	0.497
QuatE	188.035	0.743	0.692	0.776	0.832	324.829	0.285	0.189	0.318	0.481
RotatE	57.567	**0.770**	**0.702**	**0.820**	**0.875**	187.072	**0.334**	**0.237**	**0.370**	**0.531**
TransE	**43.813**	0.736	0.647	0.806	0.870	**167.377**	0.320	0.221	0.356	0.519

Table 5.2: Results of FB datasets evaluations.

5.2 Evaluation Results on Pattern Based Test Sets

As described in section 4.2.2, there are four different test sets extracted from the original test set of each dataset. A complete evaluation is executed on each one of these pattern-based test sets. The result of these evaluations is presented in this section.

5.2.1 Symmetry

Every analyzed model in this thesis can learn *symmetric* pattern and thus have good performance evaluation symmetric-based test set. Figure 5.1 indicates a good overview of Models Mean Ranks in different Datasets.

While RotatE occupies the best results in four out of five metrics for both WN18 and WN18RR, QuatE shows slightly better results in HITS@1 in both datasets.

The competition in FB15k and FB15k-237 is, on the other hand, between ComplEx and DistMult. In FB15k, ComplEx delivers a very good Mean Rank and wins the race for HITS@3 and HITS@10 with a small margin. However, DistMult has with 91.1% the best HITS@1 results, which much more than ComplEx's accuracy. On the Other side, ComplEx has the best HITS@1 with 11% compared to DistMults' 7.6% accuracy. Starting from HITS@3, DistMults takes over with its better results and delivers a result with a 50.25% advantage over ComplEx.

Model	WN18-symmetric					WN18RR-symmetric				
	MR	MRR	Hit@1	Hit@3	Hit@10	MR	MRR	Hit@1	Hit@3	Hit@10
ComplEx	875.303	0.938	0.935	0.941	0.943	1039.160	0.934	0.930	0.937	0.941
DistMult	849.181	0.940	0.936	0.941	0.949	864.700	0.938	0.934	0.940	0.946
QuatE	1042.303	0.941	**0.937**	0.945	0.949	834.571	0.938	**0.935**	0.940	0.946
RotatE	**218.771**	**0.942**	0.934	**0.945**	**0.955**	**370.222**	**0.939**	0.934	**0.941**	**0.950**
TransE	869.334	0.093	0.007	0.120	0.263	912.633	0.133	0.007	0.203	0.364

Table 5.3: Results of WNs-symmetric test evaluations.

Model	FB15k-symmetric					FB15k-237-symmetric				
	MR	MRR	Hit@1	Hit@3	Hit@10	MR	MRR	Hit@1	Hit@3	Hit@10
ComplEx	**9.756**	0.869	0.787	**0.944**	**0.971**	681.407	0.208	**0.110**	0.245	0.396
DistMult	12.833	**0.930**	**0.911**	0.942	0.961	**213.561**	**0.236**	0.076	**0.295**	**0.595**
QuatE	79.467	0.922	0.896	0.941	0.963	415.089	0.212	0.087	0.25	0.483
RotatE	30.514	0.806	0.732	0.860	0.934	467.637	0.151	0.051	0.196	0.350
TransE	186.014	0.171	0.062	0.216	0.371	275.297	0.101	0.029	0.092	0.264

Table 5.4: Results of FBs-symmetric test evaluations.

Tables 5.3 and 5.4 indicate the best results of each model in all four datasets.

The lack of efficiency in TransE for the symmetric test set is studied in section 5.3.

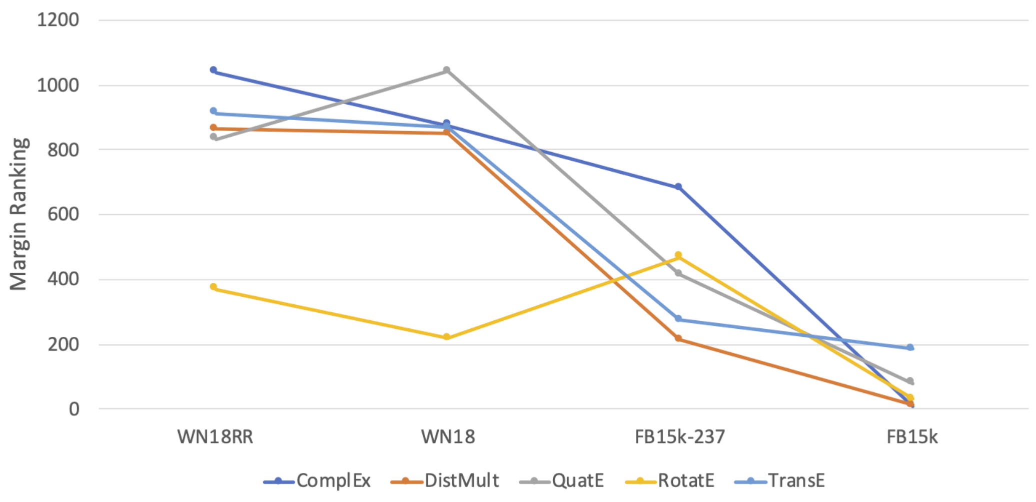


Figure 5.1: Comparison of KGE models' Mean Rank in different symmetric test sets.

5.2.2 Inverse

All five used models in this project are able to learn inverse relations. Some such as RotatE perform better while others, such as ComplEx, deliver lower performance on different Datasets. Tables 5.5 and 5.6 indicate the best results of each model in all four datasets.

Model	WN18-inverse					WN18RR-inverse				
	MR	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**	**MR**	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
ComplEx	638.104	0.952	0.951	0.953	0.955	9172.631	0.462	0.451	0.463	0.489
DistMult	972.936	0.792	0.666	0.917	0.943	7047.995	0.497	0.475	0.505	0.549
QuatE	572.307	0.955	0.952	0.956	0.959	**6264.070**	**0.519**	**0.496**	**0.531**	**0.565**
RotatE	**328.830**	**0.957**	**0.954**	**0.958**	**0.962**	6298.047	0.514	0.494	0.522	0.557
TransE	350.101	0.948	0.943	0.949	0.956	6544.647	0.194	0.015	0.350	0.465

Table 5.5: Results of WNs-inverse test evaluations.

Model	FB15k-inverse					FB15k-237-inverse				
	MR	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**	**MR**	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
ComplEx	70.461	0.882	0.853	0.903	0.924	806.696	0.225	**0.124**	0.276	0.424
DistMult	32.674	0.883	0.847	0.910	0.941	246.5	**0.262**	0.086	**0.341**	**0.673**
QuatE	53.396	0.900	0.885	0.906	0.929	464.427	0.237	0.102	0.294	0.530
RotatE	**11.980**	**0.913**	**0.894**	**0.923**	**0.946**	232.485	0.191	0.079	0.245	0.429
TransE	12.167	0.906	0.885	0.917	0.943	**205.848**	0.223	0.037	0.317	0.608

Table 5.6: Results of FBs-inverse test evaluations.

Model	WN18-implication					WN18RR-implication				
	MR	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**	**MR**	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
ComplEx	642.416	0.953	0.951	0.953	0.954	9599.563	0.440	0.429	0.441	0.469
DistMult	776.454	0.842	0.757	0.924	0.944	6785.821	0.484	0.460	0.492	0.540
QuatE	590.660	0.955	0.952	0.956	0.960	6411.477	**0.507**	**0.483**	**0.518**	**0.555**
RotatE	**254.633**	**0.957**	**0.953**	**0.957**	**0.965**	**6373.238**	0.502	0.480	0.509	0.546
TransE	374.143	0.792	0.652	0.938	0.955	6648.998	0.192	0.019	0.343	0.452

Table 5.7: Results of WNs-implication test evaluations.

Model	FB15k-implication					FB15k-237-implication				
	MR	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**	**MR**	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
ComplEx	23.155	0.897	0.847	0.945	0.961	981.015	0.196	**0.109**	0.235	0.361
DistMult	6.754	0.926	0.883	0.967	0.982	850.896	0.203	0.072	**0.258**	0.498
QuatE	12.736	0.924	0.883	0.959	0.983	1000.093	0.189	0.089	0.240	0.399
RotatE	2.696	0.945	0.916	**0.971**	**0.987**	376.487	0.146	0.062	0.165	0.335
TransE	**2.194**	**0.947**	**0.923**	0.967	0.986	**50.034**	**0.213**	0.045	0.255	**0.638**

Table 5.8: Results of FBs-implication test evaluations.

5.2.3 Implication

For implication test sets, as tables 5.7 and 5.8 demonstrate, the evaluation on FB15k shows that models perform similarly. For instance, comparing HITS@10 in FB15k shows that four out of five Models deliver almost the same result.

It is also interesting that despite the lowest result in HITS@1 in FB15k-237 from TransE, it shows the highest accuracy in HITS@10 by a large margin.

Model	WN18-one_to_many					WN18RR-one_to_many				
	MR	MRR	Hit@1	Hit@3	Hit@10	MR	MRR	Hit@1	Hit@3	Hit@10
ComplEx	1780.807	0.925	0.925	0.925	0.925	13185.045	0.069	0.045	0.076	0.123
DistMult	1011.080	0.931	0.919	**0.944**	0.944	8208.191	0.159	0.110	0.173	0.279
QuatE	844.627	0.933	0.928	0.937	0.937	7404.409	0.203	**0.157**	0.216	0.290
RotatE	435.074	**0.943**	**0.940**	**0.944**	**0.953**	**5638.123**	**0.211**	0.155	**0.236**	**0.324**
TransE	**295.285**	0.939	0.931	**0.944**	0.950	7601.959	0.149	0.101	0.166	0.231

Table 5.9: Results of WNs-on_to_many test evaluations.

Table 5.10: Results of FBs-on_to_many test evaluations.

	Symmetric test set with inverse triples in negative samples				Symmetric test set without inverse triples in negative samples			
	TransE							
	FB15k	**FB15k-237**	**WN18**	**WN18RR**	**FB15k**	**FB15k-237**	**WN18**	**WN18RR**
MR	186.01	275.29	869.33	912.63	3.37	80.87	476.99	514.95
MRR	0.17	0.10	0.09	0.13	0.78	0.29	0.84	0.85
Hit@1	0.06	0.02	0.007	0.007	0.68	0.15	0.77	0.78
Hit@3	0.21	0.09	0.12	0.20	0.84	0.35	0.92	0.92
Hit@10	0.37	0.26	0.26	0.36	0.94	0.59	0.93	0.94

Table 5.11: Comparing results in all 4 symmetric test sets for TransE. Left: while evaluating, for a triple (h,r,t), corrupted samples set contains the inverse element (t,r,h). Right: while evaluating, for a triple (h,r,t), corrupted samples set does not contain the inverse element (t,r,h)

5.2.4 One to many

As explained in section 4.2.2 the *one_to_many* test sets are created based on the average, number of *one_to_many* occurrences per relation. This means comparing the results should be done under the consideration of differences between each test set. Tables 5.9 and 5.10 indicate the best results of each model in all four datasets.

5.3 Impact of Inverse Triples in Corrupted Samples on TransE Accuracy

TransE reports inefficient results when corrupted samples for each triple (h,r,t) contain inverse triple (h,r,h). This is clearly detected in the experiments for this thesis. The efficiency of TransE increases considerably when inverse triples are removed from corrupted samples. Table 5.11 shows the result using the same symmetric test set created from each origin test set's with a small difference. On the left side of the table, corrupted samples for a triple (t,h,r) contain the inverse triple (t,r,h), and on the right side, the triple (t,r,h) is not included in the corrupted sample set. Figure 5.2 demonstrates the enormous improvement in TransE's performance after removing the inverse triples from corrupted samples.

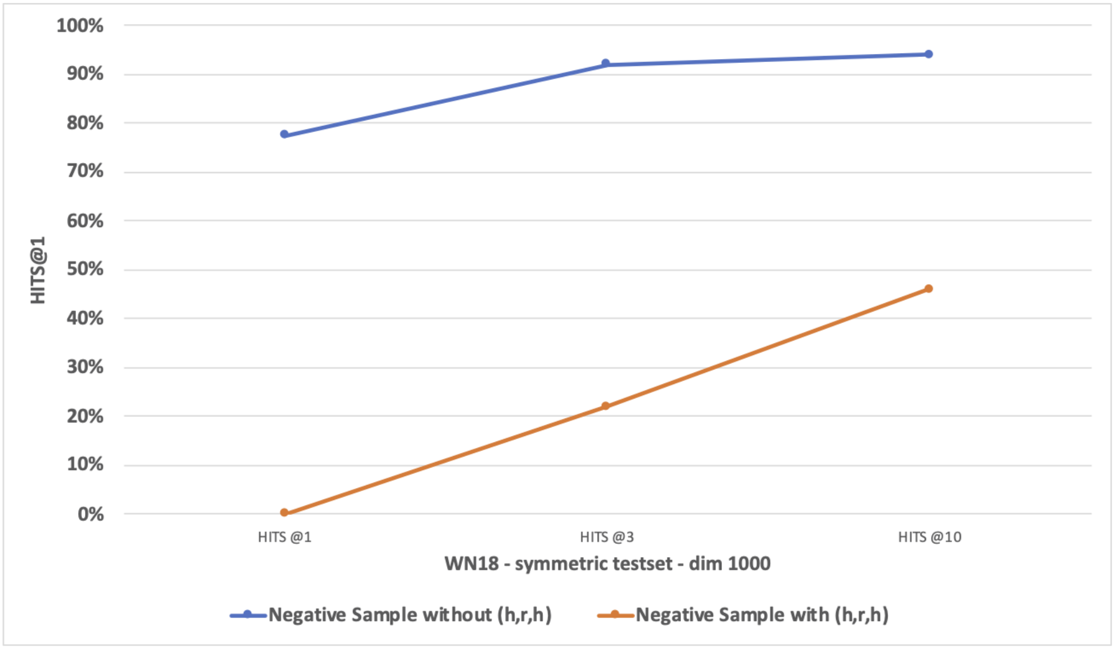


Figure 5.2: TransE's performance optimization by removing inverse triples from negative samples in WN18 symmetric test set - dimension 1000.

5.4 Comparison of DistMult, ComplEx and QuatE

The Score Function of DistMult, Complex and are very similar to each other. The main difference between these models is their space embeddings. While ComplEx and QuatE are subsumes of DistMult and use more complex score function and hence more computational resources, their performance is only different in lower dimensions.

As Figure 5.3 indicates, ComplEx's better performance is obvious in all four datasets compared to its competitors in a 10 dimensional embedding space. Increasing the dimension from 10 to 100 displays a merging of performances between the three models(figure 5.4). This behavior shows that the power of models with similar score functions is much more visible in lower dimensions.

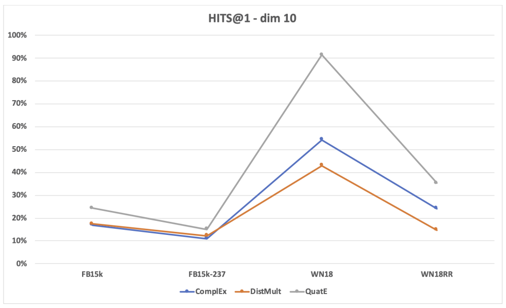


FIGURE 5.3: Dimension 10

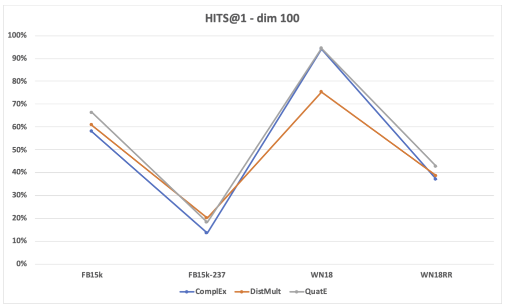


FIGURE 5.4: Dimension 100

FIGURE 5.5: Comparison of HIT@1 results in different datasets for DistMult, ComplEx and QuatE in 10 and 100 dimensional embedding representation.

5.5 Role of Loss Function in Models' Accuracy

Besides score function, loss functions play another significant role in a model's performance. This is shown by analyzing different models with different loss functions in a fixed dimension in this work. As Figure 5.6 demonstrates, margin_ranking has an enormous negative effect on ComplEx's performance in all three different dimensions, while both adaptive margin and rotate losses perform fairly similar high results.

The same behavior is observed for all other Models in both FB15k and WN18. For Some Models, for example, DistMult and RotatE, this negative effect is small, but it is dominant for others.

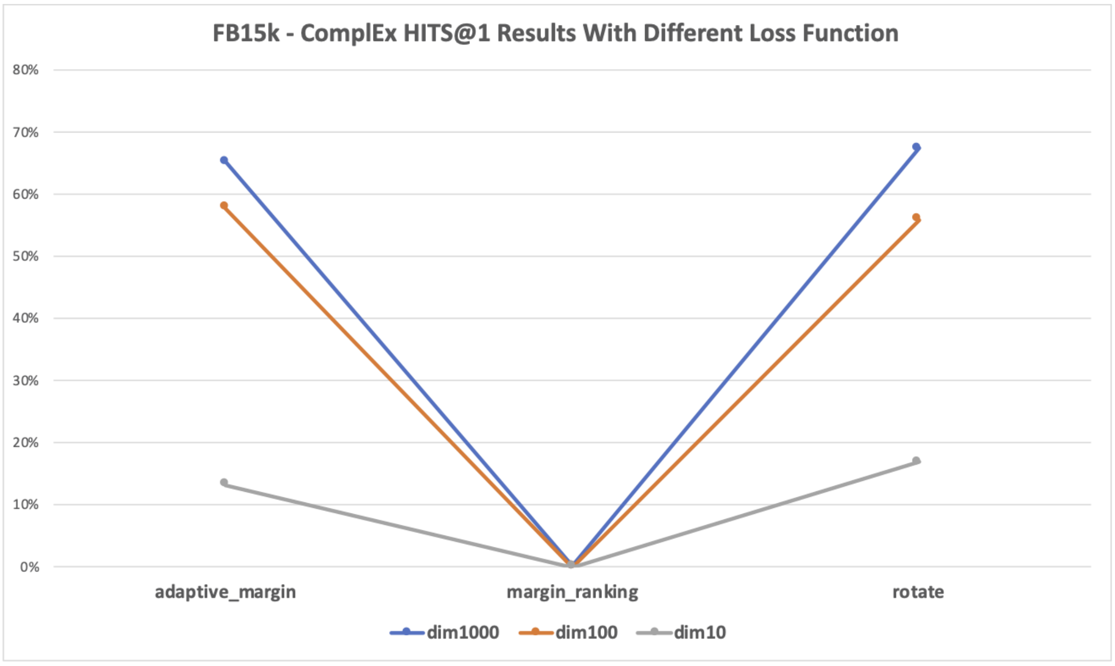


FIGURE 5.6: ComplEx's performance in different dimensions with three loss functions on FB15k.

Chapter 6

6. Conclusion and future work

Training models is by far the most time and resource-consuming step of an evaluation process. Yet reusing trained models to evaluate other datasets or, in this thesis, patterned data sets, saves much time. A new approach to overcome this issue is presented in section 6.2. Related works, the comparison between them, and the suggested future work are discussed in section 6.3. The last section of this chapter concludes our work by mentioning some of the primary knowledge which gained through this thesis.

6.1 Contributions

Here we remind the research questions that was established at the start of this project. A list of contributions is provided for each which shows the research within the time frame of this thesis has been successfully reached its goals and further insights for future work will be discussed in the next section.

Research Question 1: *How can leveraging high performance computing facilitate a comprehensive analysis of KGE Models?*

For this research question, an extensive evaluation have been done as shown in chapter 4. However, this was only possible by the aid of high performance computing clusters. Figure 6.1 shows a summary of the aforementioned evaluation complexity and combination. Such an extensive evaluation made a comprehensive overview of the embedding models possible for us. All of the conclusions and insights have been made by carefully analysing the KGEs.

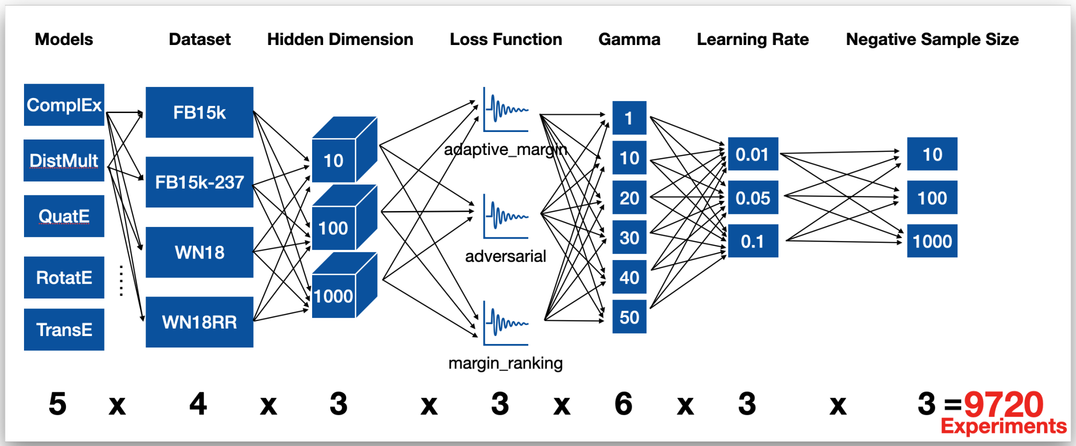


Figure 6.1: Computation of Evaluations Empowered by HPCs.

Research Question 2: *How do different metrics in hyperparameter settings affect the efficiency of KGEs models?*

In reply to this research question, we analysed KGEs from different aspects.

- Loss function-based evaluation
- Relational pattern-based evaluation
- Dimension-based Evaluation

The results of these evaluations are shown in Chapter 5 and the effect of each factor has been extensively discussed. This gives detailed insights on the functionality of the KGE models when it comes to the real world knowledge graphs and in proposing creation and refinement approaches for KGs.

6.2 Future Work

One of the main goals of this thesis is to create a library of trained models with different configurations which facilitates the evaluation process. This concept can be developed as the future work of this thesis to create an open-source service that can help many research groups and create a common ground to share trained models.

Before getting into details of the suggested future work, it is essential to distinguish between a data scientist's and a data analyst's role. Although these two roles are connected in many ways and ease each other's tasks, they can be performed independently from each other. Manipulation and improvement of models and their implementation, which is listed as one of the data scientists' primary tasks, requires high theoretical and technical skills. These skills are not necessarily needed for a data analyst. The suggested future work draws a clear line between these two roles and gives both groups the opportunity to work independently from each other. Separation of these two tasks even accelerates the improvement in Big Data and Machine Learning fields. While data scientists are working on optimizing the models, data analysts can study the results of evaluations and draw attention to the points, which might not be evident at first glance.

Another point which needs to be raised is that training models is costly and requires a lot of recourse which are not available for every student or every research group. This means that research groups with more computation power can benefit from fast feedback by evaluating their newly designed or optimized models, while others cannot keep up.

This thesis suggests a service that would deliver a trained model with the desired configuration and hyperparameter on demand. Although it is clear that with new adjustments, models need to be trained repeatedly, retraining the models with the same configuration should be avoided to reduce resource waste. This service, which would need to be implemented as a Web Application, is suggested as the technical aspect of this thesis's future work. It is a Web Application with which users could upload and download trained models. These pre-trained models would be stored in a database that would deliver them on demand. It would help every student or researcher wishing to evaluate a new or adjusted data set with already trained models and decrease their experiment's time drastically. With time and enough promotion, this system can be a single source of truth for every scientific paper in the field of KGEs by making the comparison between the state of the art models much easier as it would contain a wide range of trained models and evaluated test sets' results.

As a next step, this system could automate training models and evaluation of new data sets such that it would receive datasets and hyperparameter configurations from a user and returns the result of the evaluations. Furthermore, every research group would benefit from the evaluated results as they would be visible to everyone.

Most importantly, it would establish a firm common ground in the community for cooperation on evaluation models as well as on the Web App itself. It would also be possible to add options for training models with a given configuration and model's implementation.

A more in-depth survey on the impact of loss functions on pattern-based test sets could be done as theoretical future work.

6.3 Related works

There has been much effort done in the community to create software solutions for training models. PyKeen[24] is one of the solutions which is designed and developed by the SDA group [25] at the University of Bonn to train and evaluate KGEs, which offers a wide variety of KGE models, loss functions, and datasets. pykg2vec [26] is another python library which covers more than 20 different KGE models. Table 6.13, inspired by this work [27], shows an overview of some state of the art solutions and their supported features. However, the problem which is raised in this thesis is not handled by most of these solutions. Only four out of nine known libraries offer pre-trained models. These pre-trained models are limited in numbers and cover, in most cases, only two different datasets. Suggested future work in section 6.2 tackles this issue more thoroughly while opening an opportunity for cooperation with one or multiple of the aforementioned libraries and accelerating the research progress by reducing the amount of experimenting time drastically.

Software Solution	**Supported Models**									**Pre-trained Models**
	ComplEx	**RotatE**	**TransX**	**RESCAL**	**ConvX**	**DistMult**	**QuatE**	**HolE**	**SimplE**	
PyKEEN	✓	✓	E,R,D,H	✓	E, KB	✓	-	✓	✓	-
Pykg2vec	✓	✓	E,R,D,H,M	✓	E, KB	✓	-	✓	✓	-
OpenKE[28]	✓	-	E,R,D,H	✓	-	✓	-	✓	✓	WikiData, Freebase
AmpliGraph[29]	✓	-	E	-	E, KB	✓	-	✓	-	-
GraphVite[30]	✓	✓	E	-	-	✓	✓	-	✓	WikiData

TABLE 6.1: Over view of state of the art KGE model libraries.

6.4 Conclusion

Despite precise overall results, a more in-depth survey is needed to compare models' performances. As shown in this work, although RotatE delivers the best overall results in all four datasets in almost every metric, other models like QuatE have higher efficiency in lower dimensions.

Besides, a dimension based comparison between the three models ComplEx, DistMult, and QuatE, which use a similar approach in their score function showed, that in lower dimensions, i.e., dim 10 the best performance is delivered by QuatE, then ComplEx, and then DistMult, although the three models perform similarly in dimension 100 and 1000.

In another step of our analysis, it is showed that loss functions have large impacts on the models' performances. As an example, we pointed out the almost-zero accuracy of ComplEx on FB15k while using margin ranking as the loss function. This happened despite the same model's high accuracy while using adaptive margin and rotate loss on the same dataset.

It is also showed that leveraging high-performance computation and having a comprehensive library of pre-trained KGE Models accelerates the process of analyzing KGE Models' performances on modified test sets, i.e., rule-based test sets.

Bibliography

[1] HPC Support team. Knuth: Computers and typesetting. URL `https://tu-dresden.de/zih/hochleistungsrechnen/hpc`.

[2] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[3] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015.

[4] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embedding. *CoRR*, abs/1904.10281, 2019. URL `http://arxiv.org/abs/1904.10281`.

[5] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *CoRR*, abs/1902.10197, 2019. URL `http://arxiv.org/abs/1902.10197`.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 2787–2795. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf`.

[7] Mojtaba Nayyeri, Xiaotian Zhou, Sahar Vahdati, Hamed Shariat Yazdi, and Jens Lehmann. Adaptive margin ranking loss for knowledge graph embeddings via a correntropy objective function. *CoRR*, abs/1907.05336, 2019. URL `http://arxiv.org/abs/1907.05336`.

[8] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkgEQnRqYQ`.

[9] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. 2011.

[10] URL `https://github.com/ShayanSolid/BachelorThesis`.

[11] URL `https://www.jetbrains.com/toolbox-app/`.

[12] URL `https://slurm.schedmd.com/`.

[13] URL `https://selfservice.zih.tu-dresden.de/l/index.php/hpcportal/jobmonitoring//zih/jobs`.

[14] George A. Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[15] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. page 1247–1250, 2008.

[16] URL `https://en.wikipedia.org/wiki/Main_Page`.

[17] URL `https://musicbrainz.org`.

[18] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. pages 2787–2795, 2013. URL `http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf`.

[19] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. pages 1499–1509, September 2015. doi: 10.18653/v1/D15-1174. URL `https://www.aclweb.org/anthology/D15-1174`.

[20] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. 2018.

[21] Mirza Mohtashim Alam. Rulect: A system for extraction,representation and learning ofrelational patterns on knowledge graph embeddings. Master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 12 2020.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[23] URL `https://cloud.google.com/compute/all-pricing#gpus`.

[24] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph emebddings. *arXiv preprint arXiv:2007.14175*, 2020.

[25] URL `https://sda.tech`.

[26] Shih Yuan Yu, Sujit Rokka Chhetri, Arquimedes Canedo, Palash Goyal, and Mohammad Abdullah Al Faruque. Pykg2vec: A python library for knowledge graph embedding. *arXiv preprint arXiv:1906.04239*, 2019.

[27] URL `https://kge-tutorial-ecai2020.github.io/ECAI-20_KGE_tutorial.pdf`.

[28] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.

[29] Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nicholas McCarthy, and Pedro Tabacof. AmpliGraph: a Library for Representation Learning on Knowledge Graphs, March 2019. URL `https://doi.org/10.5281/zenodo.2595043`.

[30] Zhaocheng Zhu, Shizhen Xu, Meng Qu, and Jian Tang. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504. ACM, 2019.

List of Figures

List of Tables

Appendix

	relation	count
1	/award/award_nominee/award_nominations./award/award_nomination/award_nominee	12958
2	/award/award_winner/awards_won./award/award_honor/award_winner	6866
3	/music/performance_role/track_performances./music/track_contribution/role	3068
4	/music/performance_role/regular_performances./music/group_membership/role	2170
5	/location/location/adjoin_s./location/adjoining_relationship/adjoins	1664
6	/base/popstra/celebrity/friendship./base/popstra/friendship/participant	1216
7	/base/popstra/celebrity/dated./base/popstra/dated/participant	1134
8	/government/legislative_session/members./government/government_position_held/legislative_sessions	668
9	/military/military_combatant/military_conflicts./military/military_combatant_group/combatants	620
10	/award/award_nominated_work/award_nominations./award/award_nomination/nominated_for	592

Table 6.2: Symetric- FB15k - top 10

	relation	avg
1	/common/annotation_category/annotations./common/webpage/topic	2962.5
2	/people/marriage_union_type/unions_of_this_type./people/marriage/spouse	719.5
3	/sports/sports_position/players./soccer/football_roster_position/team	452.4
4	/user/tsegaran/random/subject_taxonomy/entry./user/tsegaran/random/taxonomy_entry/subject	410
5	/tv/tv_producer_type/tv_producers_of_this_type./tv/tv_producer_term/producer	220
6	/film/film_job/films_with_this_crew_job./film/film_crew_gig/film	162.16
7	/tv/tv_producer_type/tv_producers_of_this_type./tv/tv_producer_term/program	118
8	/film/personal_film_appearance_type/film_appearances./film/personal_film_appearance/person	116
9	/people/marriage_union_type/unions_of_this_type./people/marriage/location_of_ceremony	103.25
10	/people/profession/people_with_this_profession	98.22

Table 6.3: one-to-many- FB15k - top 10

	relation	count
1	/award/award_nominee/award_nominations./award/award_nomination/award	13834
2	/award/award_category/nominees./award/award_nomination/award_nominee	13805
3	/award/award_nominee/award_nominations./award/award_nomination/nominated_for	13323
4	/award/award_nominated_work/award_nominations./award/award_nomination/award_nominee	13215
5	/sports/sports_position/players./sports/sports_team_roster/team	12608
6	/film/film/starring./film/performance/actor	11517
7	/film/actor/film./film/performance/film	11460
8	/award/award_nominated_work/award_nominations./award/award_nomination/award	10006
9	/award/award_category/nominees./award/award_nomination/nominated_for	9947
10	/people/profession/people_with_this_profession	9496

Table 6.4: inverse- FB15k - top 10

	relation	**count**
1	/sports/sports_team/roster./sports/sports_team_roster/position	9648
2	/award/award_nominee/award_nominations./award/award_nomination/nominated_for	8245
3	/award/award_nominated_work/award_nominations./award/award_nomination/award_nominee	8045
4	/award/award_nominee/award_nominations./award/award_nomination/award_nominee	6723
5	/award/award_winner/awards_won./award/award_honor/award_winner	6612
6	/sports/sports_team/roster./soccer/football_roster_position/position	5707
7	/soccer/football_team/current_roster./sports/sports_team_roster/position	5616
8	/soccer/football_team/current_roster./soccer/football_roster_position/position	5615
9	/award/award_winner/awards_won./award/award_honor/honored_for	5334
10	/award/award_winning_work/awards_won./award/award_honor/award_winner	5282

Table 6.5: impl- FB15k - top 10

	relation	**count**
1	/award/award_nominee/award_nominations./award/award_nomination/award_nominee	6670
2	/award/award_winner/awards_won./award/award_honor/award_winner	6563
3	/award/award_nominee/award_nominations./award/award_nomination/nominated_for	3352
4	/film/actor/film./film/performance/film	2233
5	/soccer/football_team/current_roster./sports/sports_team_roster/position	1833
6	/soccer/football_team/current_roster./soccer/football_roster_position/position	1833
7	/music/performance_role/track_performances./music/track_contribution/role	1524
8	/music/performance_role/regular_performances./music/group_membership/role	1464
9	/people/person/places_lived./people/place_lived/location	1355
10	/award/award_winning_work/awards_won./award/award_honor/award_winner	1302

Table 6.6: impl- FB15k-237 - top 10

	relation	**count**
1	/award/award_nominee/award_nominations./award/award_nomination/award_nominee	6681
2	/award/award_winner/awards_won./award/award_honor/award_winner	6580
3	/sports/sports_position/players./sports/sports_team_roster/team	5519
4	/award/award_nominee/award_nominations./award/award_nomination/nominated_for	4855
5	/award/award_winning_work/awards_won./award/award_honor/award_winner	3986
6	/award/award_nominee/award_nominations./award/award_nomination/award	3890
7	/award/award_category/winners./award/award_honor/award_winner	3883
8	/location/location/contains	2747
9	/award/award_category/nominees./award/award_nomination/nominated_for	2225
10	/award/award_winning_work/awards_won./award/award_honor/award	2218

Table 6.7: inv- FB15k-237 - top 10

	relation	**avg**
1	/people/marriage_union_type/unions_of_this_type./people/marriage/location_of_ceremony	103.25
2	/education/educational_degree/people_with_this_degree./education/education/institution	97.04
3	/organization/role/leaders./organization/leadership/organization	74.4
4	/sports/sports_position/players./sports/sports_team_roster/team	68.98
5	/location/country/second_level_divisions	50.22
6	/user/ktrueman/default_domain/international_organization/member_states	43.75
7	/food/food/nutrients./food/nutrition_fact/nutrient	42.91
8	/award/award_category/nominees./award/award_nomination/nominated_for	41.37
9	/olympics/olympic_sport/athletes./olympics/olympic_athlete_affiliation/country	37.69
10	/government/government_office_category/officeholders./government/government_position_held/jurisdiction_of_office	35.1

Table 6.8: 1-n- FB15k-237 - top 10

	relation	**count**
1	/award/award_nominee/award_nominations./award/award_nomination/award_nominee	12950
2	/award/award_winner/awards_won./award/award_honor/award_winner	6860
3	/music/performance_role/track_performances./music/track_contribution/role	3068
4	/music/performance_role/regular_performances./music/group_membership/role	2170
5	/location/location/adjoin_s./location/adjoining_relationship/adjoins	1660
6	/base/popstra/celebrity/friendship./base/popstra/friendship/participant	1216
7	/base/popstra/celebrity/dated./base/popstra/dated/participant	1134
8	/government/legislative_session/members./government/government_position_held/legislative_sessions	668
9	/military/military_combatant/military_conflicts./military/military_combatant_group/combatants	620
10	/award/award_nominated_work/award_nominations./award/award_nomination/nominated_for	592

Table 6.9: sym- FB15k-237 - top 10

	relation	**count**
1	_derivationally_related_form	61
2	_hypernym	56
3	_synset_domain_topic_of	42
4	_member_meronym	22
5	_verb_group	18
6	_also_see	7
7	_member_of_domain_region	6
8	_has_part	6

Table 6.10: impl- WN18RR - top 10

	relation	**count**
1	_hypernym	76
2	_derivationally_related_form	63
3	_also_see	39
4	_synset_domain_topic_of	36
5	_member_meronym	23
6	_verb_group	18
7	_has_part	10
8	_member_of_domain_region	2
9	_instance_hypernym	1

Table 6.11: inv- WN18RR - top 10

	relation	**avg**
1	_member_of_domain_usage	30.2
2	_member_of_domain_region	13.48
3	_has_part	3.83
4	_member_meronym	3.71
5	_also_see	1.83
6	_derivationally_related_form	1.83
7	_similar_to	1.5
8	_verb_group	1.16
9	_synset_domain_topic_of	1.12
10	_instance_hypernym	1.1
11	_hypernym	1.03

Table 6.12: 1-n- WN18RR - top 10

	relation	**count**
1	_derivationally_related_form	27701
2	_verb_group	1060
3	_also_see	828
4	_similar_to	74
5	_synset_domain_topic_of	2

TABLE 6.13: sym- WN18RR - top 10