Data Integration for Industrial Big Data Applications

Vermue, Laurent

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

PhD Thesis

Data Integration for Industrial Big Data Applications

Laurent Vermue

Kongens Lyngby 2021

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

In modern applications, there are found several operational data storage systems and large amounts of heterogeneous data that is being collected, both in business and science contexts. At the same time, the data generation is in general error prone, meaning that the data entry process always will produce dirty data to some extent, either caused by human or system failure. Data integration comprises the task of cleaning dirty data and reconciling the different data sources into one homogeneous data set, which is a crucial step on the way to developing big data applications, such as machine learning models that rely on a vast amount of data. However, the pace of data creation has by far exceeded the capability of current data integration approaches, as these often rely on domain experts. As a consequence, a large fraction of valuable data is not utilized for analysis and thus leaves unused potential in every business field, which needs to be addressed. Regarding this, this thesis seeks to develop data integration methods for big data applications by employing machine learning algorithms to ease the data integration problem in domain specific contexts, which is covered by a threefold contribution.

First, this thesis has provided research contributions that investigated algorithms in the area of data fusion and relational machine learning with a focus on their suitability to solve data integration challenges. A case study of an enzyme producer in the process industry that involved multiple data sources evidently showed that before the actual task of developing a machine learning solution, multiple data sources require careful data integration steps, which are inherently hard. Many of the data integration challenges are complex, because they require relational knowledge of the data at hand that often goes beyond the data source or is not even contained in the data itself, which often leads to the necessity of involving domain experts, who possess the required knowl-

edge. To accommodate this requirement of understanding relational knowledge this thesis has covered three research contributions in the field of relational machine learning. This includes the *Bayesian Cut*, which is a specialized model for community detection in graphs, as well as two contributions that investigate knowledge graph embedding models in greater detail. In particular, a knowledge graph embedding model framework was developed that enables composability and reproducibility and was accompanied by a large-scale benchmarking study, which made the contribution of individual components transparent, such that the necessary design decisions are more comprehensible and accessible. Furthermore, a new measure was proposed that alleviates the conceptually flawed way of measuring the performance of knowledge graph embedding models, which is vital when relying on knowledge graph embedding models in big data applications.

Second, as knowledge graph embedding models are suitable knowledge integrators that can be used to predict new knowledge based on relations between objects, this thesis proposed a data integration framework purely based on machine learning that allows the combination of other machine learning approaches with knowledge graph embedding models as an artificial domain expert to solve the above-mentioned data integration challenges. In the proof-of-concept experiments that were covered in this thesis, this approach has been shown to be a very suitable approach that has promising merits. These merits include that it can scale vastly while it can accommodate various existing data profiling approaches and various types of information that are handled through the knowledge graph embedding model in a relational fashion.

Third, all research contributions as well as the experiments are based on research software that was created as a part of this thesis, which is openly available, accompanied by journal publications and thoroughly documented, and thus fosters future research in the covered research areas and beyond as well as applications that build on it, such as the data integration framework proposed in this thesis.

Summary (Danish)

I moderne applikationer findes der adskillige operationelle datalagringssystemer og der indsamles store mængder heterogene data, både i forretnings- og videnskabelige sammenhænge. Samtidig er datagenerering generelt udsat for fejl, hvilket betyder, at dataindsamlingsprocessen altid i nogen grad vil producere fejlbehæftede data, enten forårsaget af menneskelige- eller systemfejl. Dataintegrering omfatter opgaverne, at rense fejlbehæftet data og forene de forskellige datakilder i et homogent datasæt, hvilket er et afgørende skridt på vejen til at udvikle big data-applikationer, såsom maskinlæringsmodeller, der er afhængige af en enorm mængde af data. Hastigheden af datagenereringen har imidlertid langt overskredet mulighederne for de nuværende dataintegreringsmetoder, da disse ofte behøver domæneekspertviden. Som en konsekvens af dette analyseres en stor del af den værdifulde data ikke, og efterlader dermed et uudnyttet potentiale på alle forretningsområder, som bør adresseres. På denne baggrund søger denne afhandling at udvikle dataintegreringsmetoder til big data-applikationer ved at anvende maskinlæringsalgoritmer til at lette dataintegreringsproblemet i domænespecifikke kontekster, uddybet i de 3 følgende dele.

For det første har denne afhandling givet et forskningsbidrag, der undersøgte algoritmer inden for datafusion og relationel maskinlæring med fokus på deres egnethed til at løse udfordringer med dataintegrering. Et casestudie af en enzymproducent i procesindustrien, der involverede flere datakilder, viste tydeligt, at før selve opgaven med at udvikle en maskinlæringsløsning kunne løses, krævede flere datakilder omhyggelige og komplicerede dataintegreringstrin. Mange af dataintegreringsudfordringerne er komplekse, fordi de kræver relationskendskab til det foreliggende data, der ofte går ud over datakilden eller ikke altid er indeholdt i selve data, hvilket ofte fører til nødvendigheden af at involvere

domæneeksperter. For at imødekomme kravet om forståelse af relationel viden har dette speciale givet tre bidrag inden for relationel maskinlæring. Dette inkluderer Bayesian Cut, som er en specialiseret model for samfundsdetektering i grafer samt to bidrag, der undersøger knowledge graph embedding modeller mere detaljeret. Især blev der udviklet teoretiske rammer for knowledge graph embedding modeller, der muliggør sammensætning og reproducerbarhed. Dette blev ledsaget af en storstilet benchmarking-undersøgelse, som gør individuelle komponenters bidrag mere synlige, så de nødvendige designbeslutninger er mere forståelige og tilgængelige. Desuden blev der foreslået en ny metode, der forbedrer den konceptuelt mangelfulde måde at måle knowledge graph embedding modellernes ydeevne på, hvilket er afgørende, når man skal stole på knowledge graph embedding modeller i big data-applikationer.

For det andet, siden modeller for knowledge graph embedding er egnede vidensintegratorer, der kan bruges til at forudsige ny viden baseret på relationel viden om forholdet mellem objekter, foreslår denne afhandling en dataintegreringsramme udelukkende baseret på maskinlæring, der tillader kombination af andre maskinlæringstilgange med knowledge graph embedding modeller som en kunstig domæneekspert for at løse de ovennævnte dataintegreringsudfordringer. I de proof-of-concept-eksperimenter, der blev udført i denne afhandling, har denne fremgangsmåde vist sig at være en meget passende tilgang, der har lovende fordele. Disse fordele inkluderer, at den er skalerbar, samtidig med at den kan rumme forskellige eksisterende dataprofileringstilgange og forskellige former for information, der håndteres gennem knowledge graph embedding modeller på en relationel måde.

For det tredje er alle forskningsbidrag såvel som eksperimenterne baseret på forskningssoftware, der blev udarbejdet som en del af denne afhandling. Softwaren er åbent tilgængelig, ledsaget af tidsskriftspublikationer og grundigt dokumenteret, og bidrager til at fremme fremtidig forskning både inden for de omfattede, men også andre, forskningsområder og applikationer, der bygger videre på det, som f.eks. dataintegreringsrammen foreslået i denne afhandling.

Preface

This thesis is presented in fulfillment of the requirements for acquiring a Ph.D. in Engineering, and was prepared at the Section for Statistics and Data Analysis and the Section for Cognitive Systems of DTU Compute, under the supervision of Professor Bjarne Kjær Ersbøll and Professor Lars Kai Hansen.
The thesis covers aspects of how to deal with data integration challenges in modern applications that create vast amounts of data as well as underlying research challenges that can help to solve these challenges. The thesis includes 3 journal papers, 1 conference paper and 1 conference poster that are published and 1 paper that is currently under review at a journal.

Lyngby, July 20-2021

Laurent Vermue

Scientific contributions

Submitted manuscripts

A Mehdi Ali, Max Berrendorf*, Charles Tapley Hoyt*, **Laurent Vermue***, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework. *Under review at IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. *Equal contribution

Peer-reviewed journal publications

B 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 Embeddings. *Journal of Machine Learning Research*, 22(82):1–6, 2021. *Equal contribution

C Petr Taborsky, **Laurent Vermue**, Maciej Korzepa, and Morten Morup. The Bayesian Cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020

D Andreas Baum and **Laurent Vermue**. Multiblock PLS: Block dependent prediction modeling for Python. *Journal of Open Source Software*, 4(34): 1190, 2019

Peer-reviewed conference publications

E Max Berrendorf, Evgeniy Faerman, **Laurent Vermue**, and Volker Tresp. Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 371–374, 2020. doi: 10.1109/WIIAT50758.2020.00053

Peer-reviewed conference posters

F Andreas Baum, Rasmus Devantier, **Laurent Vermue**, Rayisa Moiseyenko, and Thomas Martini Jørgensen. Unraveling fermentation data - a Novozymes case study. In *Recent Advances in Fermentation Technology (RAFT 2017)*, 2017

The above contributions are part of this thesis and are included in the corresponding appendices A to F.

Acknowledgements

First of all, I would like to thank my supervisors Professor Bjarne Kjær Ersbøll and Professor Lars Kai Hansen for supporting me in my research ideas and giving me the freedom to explore broadly while providing me with exactly the impulses that I needed to progress.

I would also like to thank my fellow colleagues at the Section for Cognitive Systems and Section for Statistics and Data Analysis at DTU Compute. Especially, I would like to thank Jeppe Nørregaard and Philip Havemann Jørgensen for making me feel home at DTU and in Denmark in general even though I just had moved to Denmark three days prior to starting my PhD. Also, I would like to thank Maciej Korzepa, Petr Taborsky and Professor Morten Mørup for collaborating on the Bayesian Cut paper. This intense period has really pushed my understanding of machine learning and allowed me to develop into the right direction of my research. Thank you Associate Professor Andreas Baum for introducing me to chemometrics. The collaboration with you on several projects and many technical discussions have helped me to mature my research ideas and broaden my perspectives on data science. In this regard, I would also like to thank Jesper-Bryde Jacobsen from BIOPRO, who has helped fund this PhD project and made many of those projects possible, as well as Innovation Fund Denmark for also funding this PhD project and the Otto Mønsted Foundation for supporting my external research stay.

I'm also especially thankful to Prof. Dr. Volker Tresp for hosting my stay as guest PhD at the Ludwig-Maximilians-Universität in Munich and incorporating me into his research team at Siemens. Your philosophical view on machine learning as a whole rather than an isolated research topic has helped me to

vastly improve my understanding of many machine learning concepts. From this research team I would like to thank especially Max Berrendorf and Sahand Sharifzadeh. I immensely enjoyed working with you and learned so many things from you. You not only made this research stay enjoyable, but also very productive. It was also here that we established the new PyKEEN team together with Mehdi Ali (University of Bonn) and Charles Tapley Hoyt (now Harvard Medical School), whom I also would like to thank very much. Working in this international and talented team, which is still ongoing, has taught me countless new aspects of computer science that together with the many ideas and discussions we have had gave me a special feeling of purpose and significance about my research work.

Last but not least, I would like to thank my family for supporting me throughout my PhD. My sincere gratitude goes out to my significant other. Anna, thank you for supporting me throughout my PhD and giving me the space to perform this work, especially in the last part handling more than one year of home office due to the COVID-19 pandemic together with our son Carsten, whom I would like to thank for patiently peer-reviewing parts of this thesis and my research from my lap. I cannot express in words how much this means to me and would have never been here without your help.

Acronyms

API Application Programming Interface. 4, 18

BCEL Binary Cross Entropy Loss. 33, 35

CVAE Convolutional Variational AutoEncoder. 55, 58, 59

CWA Closed World Assumption. 31, 32

HPO Hyper Parameter Optimization. 36, 38

KG Knowledge Graph. 12, 21, 25–29, 31–33, 36, 39–43, 53–57, 60, 62, 64

KGEM Knowledge Graph Embedding Model. 12, 13, 26–43, 45, 55–65, 68

LCWA Local Closed World Assumption. 32–36, 60

MBPLS Multi-Block Partial Least Squares. 11, 15, 17–19, 45, 63

ML Machine Learning. 2, 5–8, 13, 49, 54, 56, 63, 64, 67, 68

MR Mean Rank. 40, 41

MRL Margin Ranking Loss. 33, 35

NLP Natural Language Processing. 26, 27

NN Neural Network. 6, 27, 54

NSSAL Self-adversarial negative sampling loss. 34, 35

Open IE Open Information Extraction. 26, 27

OWA Open World Assumption. 31, 33, 54, 56

PCA Principal Component Analysis. 15, 55, 60, 61

PLS Partial Least Squares. 14–16, 18

RDF Resource Description Framework. 26, 31

RML Relational Machine Learning. 12, 20, 21, 45, 63, 67, 68

SBM Stochastic Block Model. 12, 21–24, 42

sLCWA stochastic Local Closed World Assumption. 32–36

SOTA state-of-the-art. 6, 15, 29, 35

SPL Softplus Loss. 33, 35, 60, 62

SRL Statistical Relational Learning. 12, 21

Contents

Chapter 1

Introduction

1.1 Background and motivation

In recent years the term *big data* has evolved into an ubiquitous and global phenomenon [47] that by some might be referred to as 'hype' [136]. Looking at data in general, a study estimated that from 2005 to 2020 the digital universe will grow by a factor of 300, from 130,000 million gigabytes to 40,000,000 million gigabytes [49]. Considering these numbers, the term *big data* certainly is appropriate. In fact, in 2015 the famous Gartner hype cycle for emerging technologies completely eradicated big data from their hype cycle, stating that "big data [...] has become prevalent in our lives" [136].

While it is expected that at least 25% of this data contains valuable information, if analyzed, as of 2012 only 3% of the data is tagged and only 0.5% is analyzed [49]. Considering that the prevalence of big data virtually stretches across all fields of business and science, it is standing to reason that this leads to a considerable extent of lost information. The two main reasons for only analyzing a fraction of the data available are as follows. First, the problem of *dirty data*, i.e. data that contains errors, missing values, mixed formats etc. [68, 30], and second, *data heterogeneity*, i.e. data that is describing the same real world object by using different names and formats amongst different databases [32, 21, 82, 28], whose detection and correction often is the most time consuming task in data

analysis or leads to inaccurate and unreliable analysis results, if not performed. In addition, this has posed a problem for developing successful Machine Learning (ML) applications [50], since ML problems can be highly sensitive to dirty and heterogeneous data [30].

In modern applications there are found several operational database systems and large amounts of heterogeneous data that is being collected, both in business and science contexts [113], while "each data warehouse and source can be unique in its own way" [88]. Furthermore, "data entry and acquisition is inherently prone to errors, both simple and complex" [86], meaning that the data entry process always will produce dirty data to some extent, either caused by human or system failure, and the problem of already existing dirty data always remains. Correspondingly, the literature states that up to 40% of the data is erroneous or inaccurate [46, 88].

The above mentioned problems show that heterogeneous and dirty data is an inherent problem in every business field, which needs to be addressed. In fact, the case study included in this thesis [10], which is described in more detail in section 2.1.1.1, shows that basically all of the above named problems occur in typical industrial settings. The ever growing use of sensors in factories is one of the main contributors to this. Often the data created and stored is only used for regulatory documentation purposes while new applications such as process prediction by ML models first appear after years of storing this data.

The negligible amount of data currently being utilized considering the value left unused as outlined above shows that humans are overwhelmed with the vast amount of data and the tools currently available do not provide the help needed to the extent necessary. Therefore, the interest and research activity to obtain methods and tools to clean dirty data and homogenize heterogeneous data is surging, both amongst academia and the industry [21, 94, 30, 3, 88, 27, 26]. Last but not least, driven by the recent development of desired transparency and traceability of computations and reinforced by the new General Data Protection Regulation of the EU [53], there is an interest in obtaining transparent and reproducible results. Likewise, it can be concluded that business and science share this interest, since data integration presents the foundation of the subsequent analysis and usage of the new data created.

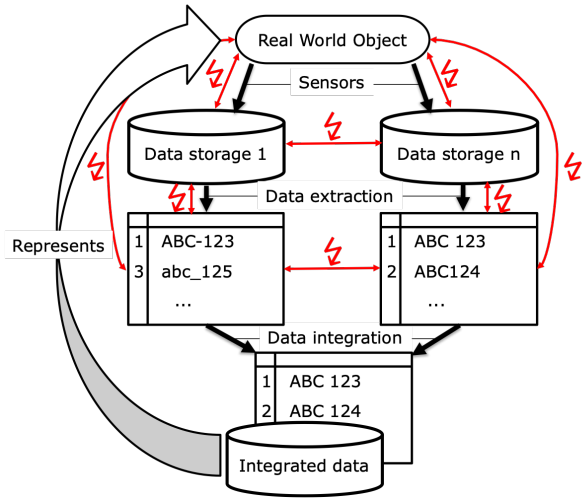


Figure 1.1: Data Integration Context

1.2 Problem definition and research questions

In general, data integration can be described as "...the problem of combining data residing at different sources, and providing the user with a unified view of these data." [76]. From an application perspective, data integration contains the tasks of cleaning dirty data and reconciling the different data sources into one homogeneous dataset [21, 76].

To better understand where the data integration challenges originate the context and environment of data integration have to be understood. In Figure 1.1 the data integration context is shown. In general, the data creation starts with sensors observing real world objects, which are considered the ground truth for the purpose of this thesis and are an abstract representation for objects of interest, e.g., a production process. Whereas in most settings sensors are electrical devices, humans can be considered as proxy sensors that subsequently use electrical devices to enter data, which presumably increases the error rate even further [86]. Accordingly, this process often introduces errors and the data stored now contradicts in parts the real world object. Furthermore, due to the heterogeneity of sensor systems, e.g., humans and automatic sensory systems, data about the same object is stored in different databases and types of storage systems. Because of the nature of these technical systems and the errors introduced by the different sensors, it can often be observed that the data stored not only contradicts the real world object, but that the data stored in

different storage systems contradict each other. Looking at the illustration, it could be argued that the data integration process could already follow as next step directly connecting to the different data storage systems, however, even the direct extraction from storage systems requires the configuration by humans, e.g., language settings and time zone settings, as well as the data extraction protocol often handled over Application Programming Interfaces (API) that are also configured by humans. In industry settings it can also be the case that direct access to the storage systems is not possible or restricted due to organizational reasons, as observed in our case study [10], which requires a pure human based manual extraction procedure, e.g., writing SQL-queries or importing production data over spreadsheet application plugins. During this process of data extraction from different data storage systems the amount of contradiction is generally increased as the different procedures and technical systems involved allow for additional errors. It is only now, once all data has been extracted, the data integration process starts, which should reconcile the data at hand into one homogeneous dataset [21, 76] that optimally truthfully represents the real world object. Elaborating this statement, this does not mean that data integration should be able to create more knowledge about the real world object than it has received, but that it should optimally only make correct statements about the real world object and also know what part of the data is contradicting or seems incorrect and thus has to be dropped or marked as such.

In general, the starting point in data integration is cleansing the dirty data in each source itself. More specifically, this can be stated as the problem of how erroneous data can be detected efficiently. Whereas the terminology of errors in databases has been well defined in the literature [94, 65, 100], the detection of these remains a problem. The literature has proposed a variety of methods and algorithms to detect mostly isolated kinds of data errors based on either rigid rules or statistical measures [86, 3]. However, in general there seems to be a consensus that well conducted data cleansing cannot be performed without domain experts [94]. A recent study tested the performance of available tools and algorithms on real data errors and concluded that their performance is far from 100% [3]. Even if available current algorithms are enriched with additional knowledge, they cannot identify errors that humans can [3]. In addition, data cleansing has focused on detecting and fixing errors, while neglecting the scalability to huge datasets by using expensive computations [64].

After cleansing, the reconciliation of related data is the remaining data integration problem due to heterogeneous data. Heterogeneity can in general be caused by the information itself, i.e. semantic, structural and syntactic, as well as the system, e.g. using different file types or database systems as shown in Figure 1.2. Asides the systematic heterogeneity, which is rather a technical aspect and not a focus of this thesis, the information heterogeneity means that even though two data sources themselves are error-free, they may differ from

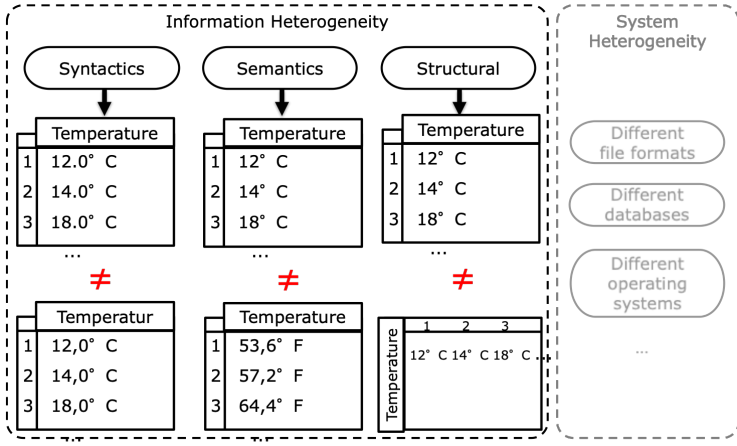


Figure 1.2: Heterogeneity of Data based on [101]

each other semantically, syntactically and structurally. As an example a Danish database stating the constant "frokost" meaning "lunch" in Danish and a Norwegian database stating "frokost" meaning "breakfast" in Norwegian can be taken. Whereas they are both syntactically correct and look similar, their semantics differ.

Since the theory and foundation of data integration itself is well established, this thesis does not cover it and instead refer to the book "The Principles of Data Integration" by Doan *et al.* [7] that covers this topic in great detail. Still, our knowledge at present only allows us to take care of the problem manually, but does not provide approaches to solve existing problems in an automated fashion. This issue is emphasized by the literature through the recurrent emphasis on the necessity of domain experts and domain knowledge [32, 74]. To alleviate this problem the literature from early on called for domain specific ontologies, which describe the data logic of each domain in a formal representation [101, 74], but as of today the lack of ontologies and data understanding is a remaining problem [127, 27].

The consent in the literature seems to be that these problems can be addressed by ontology learning, i.e. constructing the ontology out of existing data also by the means of ML [32, 22, 75, 27], but at the same time learning ontologies has turned out to be substantially more difficult than anticipated and it is often cheaper to build the ontology from scratch rather than learning it from the data [75, 111]. Summarizing these impediments, one can comprehend why challenges outlined in 1.1 remain.

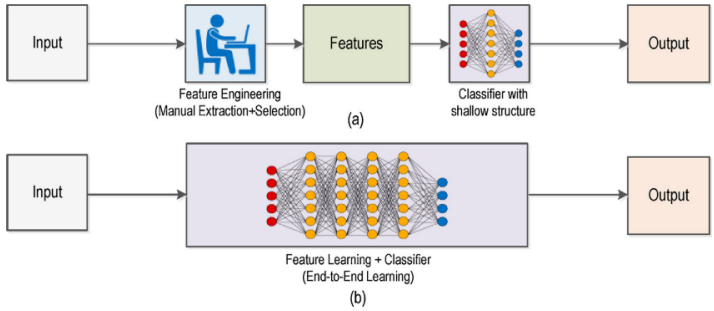


Figure 1.3: Comparison of (a) domain specialist based ML and (b) pure ML. Figure from [128]

Therefore, this thesis takes a different stance and makes an approach to solve the data integration from a pure ML perspective. Even though there are ML approaches to be found in the Ontology Learning community, it can be seen that the focus remains on using the domain knowledge and structure that are already established in this community. Drawing a direct comparison to other computer science domains as shown in Figure 1.3, a good example is the computer vision domain that for a long time has tried to solve the image recognition problem with domain expertise, domain techniques also combined with ML techniques, e.g., using hand-crafted kernels together with Support Vector Machines, whereas the ML community with the rise of Neural Networks (NNs) soon achieved superior results without the use of any domain knowledge and as of today remains the state-of-the-art (SOTA) technique [102, 128].

Here it could be seen that a pure ML approach is better suited for a domain that handles vast amounts of data with a high complexity, i.e. big data. Based on this, the context shown in Figure 1.1, and in order to cope with the challenges outlined in Section 1.1, this thesis seeks to conceptualize data integration in a more holistic approach that tries to solve these challenges in a principled manner by using ML in order to help practitioners and researches integrate their data at hand without the help of domain experts. This leads to the following to research questions:

1. **Which ML methods are suitable for data integration?**

   Similar to the computer vision domain that has shifted towards the ML domain, there are specific algorithms for specific tasks within that domain itself [102]. Another aspect, especially with regards to big data applications, is the issue of scalability and explicit interaction of users, i.e. while

some approaches might deliver good results their poor scalability or complexity can make them less suitable for big data applications. Therefore, this thesis investigates ML based approaches that either allow to handle multiple data sources directly or that support the reconciliation of data sources in big data applications through knowledge integration, while focusing on the scalability.

2. **What is a suitable generic framework of data integration under the pure ML approach?**

   Overarching the first part, it can be assumed that a general solution will consist of multiple methods and algorithms. Since the order of which the computations are performed has a huge influence on the overall performance of such a solution [3], it can be concluded that the overall framework has a computational and sequential optimum. Therefore, the goal is to develop a general data integration framework that allows for an incorporation of different algorithms to cope with data integration challenges as a whole in order to provide an efficient way of performing data integration that is suitable for big data applications.

1.3 Outline and guideline of the thesis

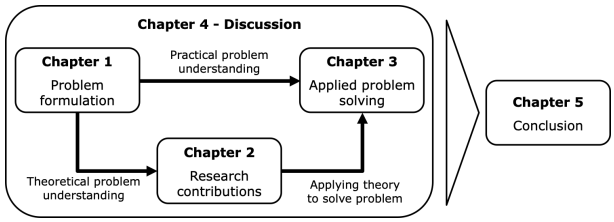


Figure 1.4: Thesis outline

As a guideline and to clearly differentiate the intent of this thesis, it should be noted that the intent is not to conduct a detailed examination of what data integration is. Also, the intent is not to establish a form of ontology learning even though some versed readers might be tempted to draw that conclusion. For readers interested in ontology learning, this thesis refers to Lehmann *et al.* [75]. To put it straight, this thesis intends to cover how the data integration challenges that arise with the age of big data can be solved by the means of ML, which the thesis covers with 5 chapters as seen in Figure 1.4 and that are shortly described in the following:

1. **Introduction**

   In chapter 1, the problem is formulated and the research questions are raised. From here the reader has to options,

   - continue to chapter 2 that will help the reader to understand the mathematical challenges of ML algorithms with regards to data integration.
   - go straight to chapter 3 for more a more practical view and application of data integration as intended by this thesis.

2. **Research contributions**

   In chapter 2, we address the first research question described in section 1.2. This chapter handles the publications that are part of this thesis and will investigate data integration challenges from a more theoretical perspective in order to gain a theoretic understanding to provide the right principled tools to solve this problem.

3. **Practical application**

   In chapter 3, the second research question described in section 1.2 is covered. Therefore, this chapter starts by constructing a theory on how data integration challenges should be considered in general to provide a straight forward approach to solve these challenges. Subsequently, the practical problem understanding of data integration challenges is handled from the applied perspective and the knowledge drawn from chapter 2 will be applied to exemplary cases.

4. **Discussion**

   Chapter 4 will put the first three chapters into perspective and cover remaining challenges with regards to the research questions.

5. **Conclusion**

   Chapter 5 will conclude this thesis by summarizing the main contributions and outcomes of this thesis.

Chapter 2

Research contributions

In this chapter the publications of this thesis, which are attached in appendices A to F, are summarized and put in context to the goal of this thesis. These publications represent the research topics that were covered during the PhD program and explore different research directions that allow coping with data integration challenges and build the underlying foundation. The two directions examined are described in the following:

1. **Data Fusion**

   handled in section 2.1. Here the focus is on joining multiple data sources in order to boost the expressiveness of algorithms, which is covered in the research contribution shown in appendix F that uses the Multi-Block Partial Least Squares (MBPLS) algorithm in an industry use-case to explore fermentation data with multiple data sources. Since the only available implementation of the MBPLS algorithm showed severe scalability issues, another research contribution was made with appendix E. In this publication all available MBPLS algorithm versions were investigated and re-implemented in a unified framework available as open-source Python package for researchers and practitioners.

2. **Relational Machine Learning**

   In Relational Machine Learning (RML) the goal is to learn the underlying relations of relational data that represents objects and the relationships between them [38]. A sub-field of this is called Statistical Relational Learning (SRL), the goal is to model data with complex and relational structures in such a way that their underlying probabilistic nature is represented [67]. As relational data often is represented in graphs representing the relationships between objects, we look at two methods that can be used to learn these relations.

   (a) **Stochastic Block Models**

   handled in section 2.1 One of such methods is the Stochastic Block Model (SBM) that allows detecting the underlying statistical nature of data represented in graphs, also known as community structures [62]. A valuable feature of SBM is that it creates a generative model that mimics the data. Therefore, adaptions have been proposed to enhance the expressiveness of this models by converting it to a Bayesian model, i.e. parametrizing the model parameters using Bayesian statistics [110, 57]. As all these models suffer from the major shortcoming that they do not necessarily represent a reasonable community structure, a new model named "The Bayesian Cut" was introduced in the research contribution presented in appendix D. This model has been shown to overcome the shortcomings of the other models while preserving the desired properties of full Bayesian models.

   (b) **Knowledge Graph Embedding Models**

   handled in section 2.2.2. In the recent years Knowledge Graphs (KGs) have become an increasingly popular way to store complex structural data that describes the relations between real-world objects as well as abstract concepts in both academia and the industry [60]. To exploit the knowledge stored in such KGs, Knowledge Graph Embedding Models (KGEMs) can be used [98, 129]. In the same fashion as in SRL KGEMs learn the relational structure of the data and thus can be used to predict new facts [98], which is why they also are widely applied in many tasks, such as relation extraction and question answering amongst many others [129]. Even though works such as [98] and [129] have provided great contributions to establish the theoretic framework of KGEMs in the literature [6] could observe that actual publications proposing KGEM often do not follow this framework and thus, the understanding of the impact due to architectural design choices in the field remained unclear. Therefore, the research contribution of this thesis shown in appendix A proposes a unified framework and conducts a large-scale evaluation study of

existing KGEMs to unify existing research and provide a structured approach to conduct future research for both academia and industry. Since this effort required a robust and modular software framework, another research contribution is provided by this thesis with appendix B in which an open-source Python package is developed that takes a principled approach to KGEMs based on the established theoretical framework by [6]. Last but not least, in [13] it was observed that ranking measures, which are used to evaluate the performance of KGEMs, are fundamentally flawed and can lead to deceiving results that hide the actual performance of these models. Therefore, the research contribution of this thesis shown in appendix C analyzes this issue and proposes a new metric that effectively overcomes this issue.

2.1 Data Fusion

In combination with data integration one might hear the term "data fusion", or one of its synonyms such as "data merging", "data consolidation", "entity resolution", or "finding representations/survivors" [15], which might cause confusion. In general, data fusion is joining a variety of data from multiple sensors and sources into a single data set to improve the accuracy of algorithms and analyses [55, 54]. This is different from data integration, which is the general task of combining data from multiple sources, including the cleaning of dirty data and reconciliation of entries, to provide a unified view to the user [21, 76]. Accordingly, it can be derived that contrary to data integration, data fusion does not concern techniques to produce more consistent, accurate or correct data than the individual data sources provide themselves. In short, data fusion only addresses the question how to merge multiple sources into one in a principled manner, and therefore should be considered a sub-procedure of data integration, which is supported by [15].

Nonetheless, the logic of how to merge data is very relevant and has a significant impact on the overall data integration pipeline. Thereby, data fusion also covers the necessary considerations as to why data sources are merged in specific ways as shown e.g. by the Joint Directors of Laboratories (JDL) process model for data fusion that guides how data should be merged and which techniques should be used based on process and application considerations [24]. However, for ML and the goal of this thesis it is less relevant to know how things should be done based on a framework, as it is to understand how things actually can be done and what impact that has on the integrity and semantics of the data at hand.

Therefore, this thesis will follow the "Three Processing Architectures" according to [55], that define the three ways of joining data as follows

- **direct association** of all sensor data, i.e. direct aggregation of the data through averaging or more advanced estimation techniques such as Kalman filtering. This is often done to reduce the noise of multiple independent sensors that measure the same physical phenomena. As an example we can take two different cameras that take a picture of the same real-world object that subsequently are joined into one file containing the RGB information as if it was one camera.
- **feature extraction** of sensors and subsequent association. Different from the first case we can envision a LIDAR sensor instead of the second camera to add depth perception to the image. Now the RGB file information from the camera in the first example is augmented with the LIDAR depth data.
- **processing** of the data for each sensor and subsequent joining of the inferred data, which is also called information fusion, since it is a higher semantic level [24]. As an example we can look at a facial recognition algorithm. Whereas in the first two ways the data is combined and then given to the inference algorithm that predicts the person, this approach would predict the person based on the RGB data and LIDAR data separately first and then join those predictions into one data set.

At this point it should be noted that the literature generally attributes more tasks to data fusion than the merging logic of multiple sources, such as conflict resolution or resolving errors, which might be the sole reason causing confusion of these two terms. The reason for this might be that data fusion has had a long history in military research and has grown its own field of research that naturally has evolved in directions that try to cope with the challenges of merging multiple data sources, as shown in [24] and [54]. However, this thesis argues that these tasks should be strictly separated from the data fusion principle in order to allow a principled logic distinction in the view of data integration, which will be covered in more detail in section 3.2. For the inclined reader, this thesis refers to [55] and [24] for a detailed examination of the data fusion field as well as [15], which considers data fusion in the context of data integration.

2.1.1 Multi-Block Partial Least Squares

In 1984 [135] proposed the Partial Least Squares (PLS) algorithm. Contrary to regression algorithms that try to find the direct relation between the independent space represented by the matrix $\mathbf{X}$ and dependent space represented by the

matrix $\mathbf{Y}$, PLS projects each matrix $\mathbf{X}$ and $\mathbf{Y}$ into a latent space and maximizes the covariance between the two latent spaces. This results in two desirable properties, namely that it is inherently good at handling collinearity as well as the fact that it can handle datasets that have more variables than samples. In addition to the resulting prediction model it also offers declarative insights about the latent space in the same fashion as Principal Component Analysis (PCA) does. Regardless of its age, it can still be found in recent SOTA applications, such as [81], in fields that deal with high collinearity combined with high dimensionality and small sample sizes and thus rely on the properties provided by the PLS algorithm.

As shown in section 2.1, the object with data fusion is to join multiple data sources into one with the purpose of improving the prediction of algorithms. Analogue to having one independent matrix $\mathbf{X}$, having multiple data sources can be seen as having multiple matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, while still being interested in the same dependent matrix $\mathbf{Y}$. In order to obtain one common matrix $\mathbf{X}_{total}$ from these multiple matrices one would now have to choose one of the three processing architectures presented in section 2.1. However, this requires sincere consideration as the decision can have a severe impact on the resulting data quality. As an example we can assume that the **direct association** processing approach was chosen, since the data consists of two sensors measuring the same physical phenomenon. Assuming that one of the two sensors is malfunctioning and thus providing incorrect values, this information would be lost and the resulting averaged values would lead to a useless dataset.

Fortunately, the MBPLS algorithm was introduced by [132] that allows us to avoid the delicate decision of choosing the data fusion processing architecture, since it can handle multiple matrices without further ado. The underlying algorithm of MBPLS is the same as in PLS, but in MBPLS each matrix is projected in its own latent space. This adds another desirable property to PLS, which is that it can express the importance of each matrix for the resulting prediction model as a high level indicator. In the just illustrated case of one broken and one functioning sensor, the matrix representing the broken sensor would be expected to have a $\sim 0\%$ importance while the other matrix would obtain $\sim 100\%$ importance. Concluding, the MBPLS algorithm not only allows data fusion without having to make data fusion decisions, but at the same time it can provide important insights into how separate data sources contributed to the final prediction model. These properties make it very suitable for exploratory analyses in industrial settings that usually contain multiple data sources spread across the organization for the same real-world object while the researcher confronted with the data not necessarily be able to make the right data fusion decisions, which is bringing us to the first research contribution of this thesis.

2.1.1.1 Contribution: Unraveling fermentation data – a Novozymes case study

The case study presented in appendix F had the goal to investigate the fermentation process of an industrial enzyme producer in order to understand which factors contribute to an improved yield of the process at hand. Industrial fermentation processes often consist of multiple batch production steps that are equipped with a whole array of sensors and controlled variables that define and steer the production process, which was also the case in this case study. Therefore, we were presented with the following data sources:

- Main fermentation sensor values over time for each batch (independent variables), declared as $\mathbf{X}_1$. These are values that are a result of the ongoing process over time, e.g. the pH value.
- Main fermentation input parameters over time for each batch (controlled variables), declared as $\mathbf{X}_2$. These values are e.g. materials that are added to the process over time, but also values that are controlled by the equipment as e.g. the airflow.
- Seed fermentation[1] sensor values over time for each batch (independent variables), declared as $\mathbf{X}_3$. Analogue to the main fermentation values.
- Seed fermentation input parameters over time for each batch (controlled variables), declared as $\mathbf{X}_4$ Analogue to the main fermentation values.
- Yield data (dependent variable), declared as $\mathbf{Y}$. The resulting yield of each batch, which should be explained by the model.

As to be seen from the data description, the interest of this study was to explain the final yield represented by the data block[2] $\mathbf{Y}$ through the data blocks $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ & $\mathbf{X}_4$. Overall the data consisted of roughly 200 batches/samples, which each consisted of time-series with more than 1500 time points for about 200 different variables in the data sources $\mathbf{X}_1$ and $\mathbf{X}_2$ as well as 800 time points for about 50 different variables in $\mathbf{X}_3$ and $\mathbf{X}_4$. Doing a quick calculation with these numbers we can see that the dataset consists of 68 million data points. An amount that exceeds the human capability to go through and comprehend in detail and which this thesis considers to be big data. Another challenge was, and will remain, that the persons that conduct the data analysis and work with

[1] Seed fermentation is a smaller size fermentation step that precedes the main fermentation.

[2] Since the data in industrial settings often consists of time-series for each sensor per sample the original data has to be seen as a three-way tensor, which is considered a data block. As PLS only handles two-way tensors, these data blocks are unfolded to a two-way tensor, which is why the data blocks are represented using matrix notations.

the data extraction are usually not the process specialists, and thus, often do not carry the knowledge about the meaning nor correctness of the data at hand, which is a main barrier for data integration as outlined in the introduction. This challenge is exacerbated by the fact that due to the vast amount of sensors and processes involved in the industry, there usually is not a single process expert that is knowledgeable about all involved process parts, which this thesis considers another dimension of big data increasing the complexity and might lead to additional errors. In fact, even though the data in this case study was from the same standardized process, it originated from different plants, in different countries, and the data was stored in two different systems that again consisted of multiple databases. As a result, the initial task of creating a unified and correct view of the data for this study, i.e. data integration, which could be consumed by an algorithm or used by the researches for the analysis, posed tremendous challenges, such as:

- variables/column names were misplaced
- missing columns
- different names for the same sensor
- wrong values stored under some variables
- phantom columns created through fields containing commas in a comma separated file. i.e. the value of one variable is split into two columns

As the analysis of the data integration part is not covered in the research contribution, it will be elaborated on in more detail later in this thesis under section 3.1.

After having solved the data integration challenges, the result was a unified view of the data blocks described above, namely $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$ & $\mathbf{Y}$. Instead of relying on process specialists to cover the remaining data fusion part, the study relied on the MBPLS algorithm and its ability to use multiple data sources. For more details the reader is referred to the research contribution itself in appendix F. In summary, the case study showed that MBPLS was able to use the multiple data sources provided to detect equipment configurations and their underlying contributions that led to higher yields, while at the same time being able to explain the importance of each data source. Having discovered MBPLS as a helpful tool for such data fusion challenges, it was revealed that the available open source implementation of MBPLS had severe runtime issues, which resulted in unfinished runs after more than 24 hours, even when just using 25% of the available dataset. This leads us to the second research contribution of this thesis, which is an open source package for MBPLS algorithms.

2.1.1.2 Contribution: Multiblock PLS: Block dependent prediction modeling for Python

Conducting the case study in section 2.1.1.1 it became evident that the at that time available MBPLS R-package provided by [20], which initially was used for the case study, suffered from convergence problems on larger datasets and therefore had severe scalability issues as also shown in [9]. Additionally, there is a more established machine learning community in the Python programming language than in R and existing Python packages like scikit-learn[103] provide excellent additional functionalities like cross-validation pipelines and evaluation metrics with a standardized API that is known to most users through its widespread usage.

For this reason it was decided to create an open source Python package that optimally does not suffer from scalability issues and should provide a rich, but simple to use, functionality together with a full compatibility with the scikit-learn API. This resulted in research contribution in appendix E. Instead of just implementing the MBPLS[132] algorithm in Python, which is based on the original Nonlinear Iterative Partial Least Squares (NIPALS)[134] algorithm, a unified architecture has been created so the MBPLS algorithm now also can be run based on the UNiversal PArtial Least Squares (UNIPALS)[51] and KERNEL[79, 106, 107] algorithm. Also the method to handle missing data proposed by [87] was added as well as the SIMPLS[37] algorithm, which is the fastest PLS algorithm, but mathematically cannot be adopted to be used with multiple data blocks. To allow a straight forward application of the provided package for practitioners and researchers, the package comes with a full documentation that also shows the application of this package on former MBPLS based research publications, such as [11]. The resulting package has already been used for other research publications, e.g., [81, 23, 80, 43, 59], and provides great runtime improvements compared to the existing R package. As a comparison, the process that did not finish after 24 hours when using the MBPLS package for R finished within few seconds with the Python package provided by this research contribution.

2.1.2 Discussion

As shown in the above described case study handling multiple data sources has become a normality when analyzing industrial processes that often stretch across multiple locations, systems and databases. As to be expected, it was shown that considering all available data sources allows us to obtain better results when performing analyses or training prediction models and thus shows

that data fusion is not a mere academic exercise, but a necessity in big data applications. It was shown that MBPLS can be used to avoid explicit data fusion decisions as to how the data should be merged that otherwise would require a domain expert, while at the same time being scalable to "big data" datasets and having the desirable attribute that it can explain how much each data source contributed to the result.

However, with regards to the goal of this thesis, we have to note that an algorithm such as MBPLS makes several strong implicit assumptions that have to be satisfied. It expects e.g. that the provided time series are equidistant. It also has assumptions about the data quality. The way that MBPLS can detect that two sensors are the same is through projecting those sensor values into the latent space and, in the case of perfect collinearity, reducing them to one variable/sensor. The main problem arises when MBPLS also does this when two variables are substantially different, but coincidentally show high collinearity in the dataset at hand. This is the well known problem of confounding correlation with causality. Given the size of "big data" datasets this risks increases substantially, since the likelihood of confounding correlation with causality grows combinatorial with the amount of included variables. In addition, looking at the example with one broken and one functional sensor measuring the same physical phenomena made in section 2.1.1, we actually would like the method at hand to tell us that the broken sensor is expected to show measurements that resemble the other sensor's values. The reason that MBPLS does not do this is because it focuses on the low-level data and does not incorporate higher-level data, such as the knowledge that this sensor measures the same physical phenomena as the other sensor. It is believed that relating real-world objects to each other and thus transferring prior knowledge from already learned examples allow humans to quickly understand new problems[72], such as this sensor problem, which in the field of data integration usually is solved with the call for domain experts. Accordingly, we can see that data integration goes beyond data fusion and that in order to solve these challenges we need to rely on methods that are capable of incorporating higher-level semantics, such as e.g. the meta data describing the context of the object at hand, which is addressed in the next section.

2.2 Relational Machine Learning

Other than low-level features like sensor readings, there is often an interest in the relations of high-level structures that put these objects into perspective [67]. With regards to data integration the knowledge of how a certain sensor is related to a specific kind of production is substantially more valuable than the specific values of the sensor itself. Therefore, it is argued that the rich logical object-relational structure is crucial for solving the more general and complex problems [67]. This belief goes back to the very beginning of artificial intelligence research as pointed out by the following quote

> **"I am convinced that the crux of the problem of learning is recognizing relationships and being able to use them."**
> Christopher Strachey in a letter to Alan Turing, 1954, [98]

In fact, the hypothesis seems to be correct, as current research in cognitive science shows that a main component of the amazing human learning capability is to understand the relations and similarities between objects and concepts [72].

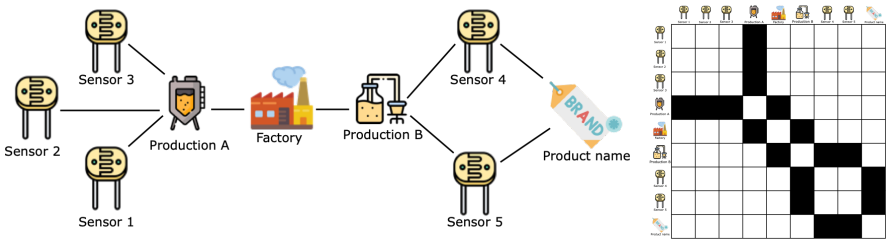
Thus, in RML we want to utilize the information of one object to give knowledge about the other object at hand. Rather than looking at the data of the sensor itself to derive that it measures the pH, we could also look at its attributes, often called *meta data*, to derive important characteristics. When working with relational data the rich and dynamic structure, defined by a varying amount of entities and relations, makes a typical fixed-vector based approach less applicable, if at all [38]. Therefore, this data is often represented as a graph that shows the relations between entities. The goal is to learn similarity based on proximity in the graph rather than looking at all data produced by the entity itself. Referring to the example of two sensors measuring the same physical phenomena from the previous discussion in section 2.1.2, knowing that the sensor is placed on the same tank, is in the same factory, is from the same brand, has the same model name as well as the same settings will from a semantic perspective give us a much better indicator that these sensors actually are measuring the same physical phenomena than looking at the time series values produced by the sensors themselves.

In addition to the uncertainty that arises from the data at hand, relational learning also has to deal with the uncertainty of the number of objects and whether they are related or not [67]. This can be understood as in addition to being unsure whether data points, contained or missing, are correct, there is also an inherent probability to the existence of such data point that is not

equal between all data points. The field focusing on this is SRL, where the goal is to model data with complex and relational structures, such that their underlying probabilistic nature is represented [67]. As covering this field in more detail would go beyond the scope of this thesis, readers looking for a general introduction to the field of SRL and RML are referred to [67] and [38]. Instead we focus on the two main fields that this thesis has contributed to in the next to sections.

2.2.1 Stochastic Block Models

First described in [58], SBMs are generative models for random graphs that try to reproduce the graph's underlying structure of connections between all network members through a statistical model, capable of finding similar items [1]. In order to make this a bit more tangible we will define a basic notation[3] and look at a basic SBM. We start with a graph $\boldsymbol{G}$ that has n nodes[4] and N links[5] An example is shown in figure 2.1a with an undirected[6] graph that has $n = 9$ nodes and $N = 9$ links. The connections of the graph can be presented as an adjacency matrix $\boldsymbol{A}$ as shown in 2.1b, where $\boldsymbol{A}_{i,j} = 1$, when the nodes $i, j \in \{1, \ldots, n\}$ are said to have a link, i.e. the node i and j are connected in the graph.[7]



(a) Graph **(b)** Adjacency Matrix

Figure 2.1: (a) An exemplary graph showing objects in a factory represented as nodes and their relations to each other represented as links, and (b), the same graph shown as adjacency matrix.

[3]The literature provides many different notations when describing SBMs. This notation is in line with notation provided in the research contribution appendix D.

[4]Nodes are also commonly called "vertices" in graph theory applications as well as "entities", which is the more common term in KG applications

[5]Links are also commonly called "edges" in graph theory applications and "relations" in KG applications.

[6]In undirected graphs the direction of the relations/links are not taken into account.

[7]It should be noted that in the case of Poisson distribution based SBMs there can be multiple links as $\boldsymbol{A}_{i,j} \in \mathcal{N}$.

In order to describe the underlying structure of the graph, the SBM assumes the graph to be split into C clusters and every node to be assigned to a cluster, which we denote with the vector $\boldsymbol{z}$ so that $z_i = c, c \in C$ can be understood as the node i being assigned to the cluster c. Last but not least, we define $\boldsymbol{\eta}$ as the link density parameter where $\eta_{r,s}$ describes the expected link density between the clusters $r, s \in \{1, \ldots, C\}$. With the basic notation covered we can now define a basic Poisson distribution based SBM as follows:

$$\begin{aligned} P(G|\boldsymbol{\eta}, \boldsymbol{z}) = &\prod_{i<j} \frac{(\eta_{z_i z_j})^{A_{ij}}}{A_{ij}!} \exp(-\eta_{z_i z_j}) \\ &\times \prod_{i} \frac{(\frac{1}{2}\eta_{z_i z_i})^{A_{ii}/2}}{(A_{ii}/2)!} \exp(-\frac{1}{2}\eta_{z_i z_i}) \end{aligned} \tag{2.1}$$

Without analyzing the equation in much detail we can see that the SBM models the likelihood of the graph G represented through the adjacency matrix A given the parameters $\boldsymbol{\eta}$ and $\boldsymbol{z}$. If we now change the parameter $\boldsymbol{z}$ by changing the group assignments of some nodes, the model will have a higher likelihood in cases, where this new assignment fits the underlying structure better, and a lower likelihood if it does not. Thus, we can use this approach to derive which nodes should be grouped together, while at the same time modelling the underlying probabilities of observing links between and inside these groups. This has the advantage that the model, once fitted, can be used to analyze the underlying structure of the clusters and predict missing links. With regards to data integration we can look at the example of sensors and their meta data in factories from earlier, which represented as relational data would allow us to cluster similar sensors in groups to better understand what sensors should be similar and which not.

Overall, there exists a vast amount of different SBM models and approaches that all have their specific applications [73, 48, 52]. Compared to the model described in equation 2.1 there has been one important addition to the model, which is the "degree correction" [62]. The model above assumes that all nodes within one cluster have the same connectivity, i.e. they have the same amount of links, which is called "node degree". This has been shown to be an irrational assumption, which is why degree correction obtains superior results [62]. Another important contribution comes from [57], has defined the model proposed in [62] as a non-parametric Bayesian model that allows the full inference of the model parameters.

One of the main disadvantages of SBMs is that "[...] one has to hope that the model represents a good fit for real data, which does not mean necessarily a realistic model but at least an insightful one" [1]. Aside from their expressiveness SBMs, and also the other models presented so far, often suffer from a tendency to create highly unrealistic results as shown and analyzed in [118]. One of these cases can be illustrated using the previously defined notation. Where η_c describing the inside link density of each cluster $c \in \{1, \ldots, C\}$ and η_{out} describing the link density between clusters, models often have the tendency to create solutions that show the behaviour defined as $\eta_c \leq \eta_{out}$, i.e. nodes are less likely to be related to nodes in the same cluster than to nodes in other clusters. To resolve this issue, [92] introduced a Bayesian SBM that inherently respects this notion of the "community structure", i.e. $\eta_c \geq \eta_{out}$, forcing the model to only create solutions that exhibit higher inside cluster link density than outside density. However, this model does not incorporate the degree correction shown to be useful earlier. This leads us to the third research contribution of this thesis, which is the development of a fully Bayesian degree corrected SBM, termed "The Bayesian Cut".

2.2.1.1 Contribution: The Bayesian Cut

The research contribution covered in appendix D proposes a new fully Bayesian degree corrected SBM with community constraint. The motivation for this was that the degree correction was shown to be useful, while a Bayesian model is more expressive, since it can exhibit the posterior distributions of the used parameters allowing us to derive how certain the model is regarding the solution. At the same time we want to enforce community structure, i.e. avoiding cases where $\eta_c \leq \eta_{out}$. To relate to the example of finding clusters of similar sensors, this would mean that the solution creates one cluster with sensors that are absolutely unrelated to each other, which is diametrical to what we are looking for. Illustrative cases of this behaviour are given in [118]. To elaborate on the title of this publication for the readers less privy to graph theory, clustering is in graph theory often referred to as "graph cutting" derived from the figurative language of cutting the ties, i.e. links, between the nodes of the graph to create graph clusters. Hence, our model being Bayesian, it was termed "the Bayesian Cut".

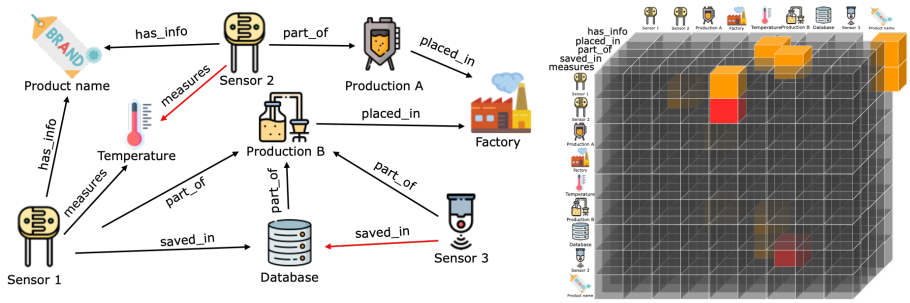
Since the mathematical background of this research contribution is covered in great detail in the publication itself in appendix D, we settle for a shorter version at this point. As a starting point we take the degree corrected SBM proposed by [62] and [57], but in order to avoid cases where $\eta_c \leq \eta_{out}$, we introduce a community density parameter $\boldsymbol{b}$, with the range $[0, 1]$, that controls the link density to be higher inside the cluster than outside the cluster by constraining

the model to solutions where $\eta_c b_c \geq \eta_{out}$ for each cluster c. However, in order to remain a fully Bayesian model the newly constrained η parameter has to be marginalized under the constraint, i.e. solutions that violate the constraint should have 0 probability. Unfortunately, this quickly becomes computationally prohibitive, since we, figuratively speaking, have to integrate over the entire constrained solution space for η to know the probability of a proposed solution. In our publication a new way of approximating this integral with multiple incomplete gamma functions was proposed, which is facilitated with a principled way of controlling the error boundaries of the approximation itself. As a result the computational efficiency of the model is improved by orders of magnitude and at the same time the model significantly outperforms the originally proposed degree corrected SBM on typical social network benchmark datasets, which can be attributed to the community constraint as shown in the research contribution.

2.2.2 Knowledge Graph Embedding Models

KGs have a long history in the logic and artificial intelligence domain and have spread to other domains such as the Semantic Web community in the past years [98]. One of the main drivers for the success of KGs have been internet-based companies that use KGs to gather and unify large amounts of knowledge available on the web. Just to name two examples, we can look at Google that uses their KG to enhance their search engine "Google Search" with knowledge and facts as well as Microsoft that uses their KG, named "Satori", for the same purpose for their "Bing" search engine [104]. Already in 2012 the Google KG contained 18 billion facts, based on 570 million entities and 35,000 relation types [112, 41].

In order to make it more comprehensible how KGs can be utilized, we will establish a notation first. In the KG domain nodes/vertices are usually referred to as entities, while the different relationship types are simply referred to as relations. The actual links/edges are called facts or generally and for the remainder of this thesis triples, since they consist of one relation that marks the connection between two entities. Further characteristics can best be observed by comparing a KG to the undirected graph in figure 2.1a. Opposed to figure 2.1a in figure 2.2a the links have arrows, i.e. the relations are directed, and the relations have identifiers, which mark the different kind of relations between the entities.



(a) KG **(b)** Adjacency Tensor

Figure 2.2: (a) An exemplary KG showing objects in a factory and their relations to each other. The nodes represent entities and links their respective relations. Black colored links represent triples that we know and red links represent triples we would like to learn. (b) The same KG shown as Adjacency Tensor representation, where orange cubes represent triples we know, and red cubes triples we would like to learn.

The exemplary KG in figure 2.2a contains $N_e = 9$ entities, $N_r = 5$ relations and 12 triples. Formally we have a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$ and define the KG $\mathcal{K} \subseteq \mathbb{K} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ as a directed, multi-relational graph that consists of triples $(h, r, t) \in \mathcal{K}$ where $h, t \in \mathcal{E}$ represent a triples' respective head and tail entities and $r \in \mathcal{R}$ represents its relationship.[8] An exemplary triple from figure 2.2a would be *(Sensor 1, part_ of, Production B)*, where *Sensor 1* is the head and *Production B* is the tail entity while *part_ of* is the relationship. Analogue to the adjacency matrix constructed from a graph, as shown in figure 2.1b, we can construct an adjacency tensor $\underline{\mathbf{A}} \in \{0, 1\}^{N_e \times N_e \times N_r}$ from the KG, as shown in figure 2.2b, where $a_{h,t,r} = 1$ (indicated by red and orange cubes) for all triples $(h, r, t) \in \mathcal{K}$ and 0 (indicated as grey cubes) otherwise.

Having established the architecture of the KG the remaining question is how to obtain the necessary triples to construct it. According to [98] we can categorize these approaches as:

- **curated**

  In this approach a closed group of experts creates the KG manually as for example in WordNet [90].

- **collaborative**

  Here an open group of volunteers creates the KG manually. Examples are Freebase [16] and its successor Wikidata [124].

- **automated semi-structured**

  This approach tries to extract data from structured knowledge, e.g. Wikipedia Infoboxes, via handcrafted or learned rules as well as regular expressions as for example in YAGO [114].

- **automated unstructured - without schema**

  In order to utilize the vast amount of knowledge that is contained in free text on the web, this approach uses Open Information Extraction (Open IE) techniques that often comprise Natural Language Processing (NLP) approaches to extract facts from text, such as in REVERB [45]. A disadvantage of the Open IE system is that due to the missing schema it creates ambiguous triples, such as ("DTU", "placed in", "Denmark") and ("Danmarks Tekniske Universitet", "located in", "Kongens Lyngby") where it is not clear whether *"DTU"* and *"Danmarks Tekniske Universitet"* or *"placed in"* and *"located in"* mean the same thing.

[8] It should be noted that we follow the World Wide Web Consortium (W3C) Resource Description Framework (RDF) for representing information on the web. However, other than originally defined by the W3C we do not follow the (subject, predicate, object)/(s,p,o) naming convention [126], but instead use the (head, relation, tail)/(h,r,t) notation that is more common in the KGEM community.

- **automated unstructured - schema based**

  In order to overcome the problems of the Open IE system this approach uses an established schema that is used to train a classifier, which subsequently is used to disambiguate the triples extracted from free text such as in DeepDive/Elementary [99] and Knowledge Vault [41].

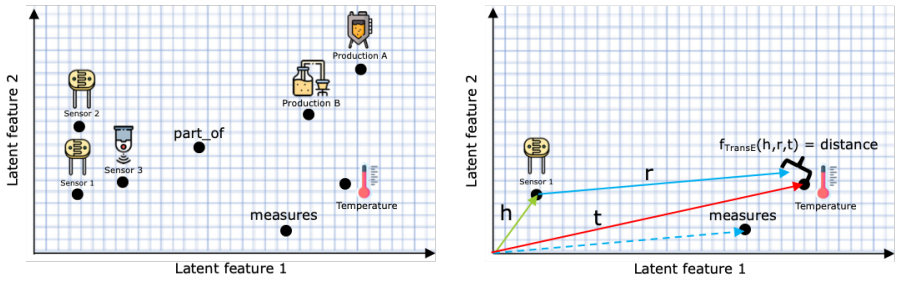
While manual approaches are very accurate, but labour intensive and limited in their scalability and in addition often lead to many missing attributes, even when these are mandatory [98], the more automated approaches based on free text face similar challenges. First, Wikipedia, the main source of knowledge, has left the exponential growth phase and is now growing at a steady rate [115, 133], which can be considered a call for machines to help humans to accelerate the knowledge creation. Second, even more quality controlled sources, such as Wikipedia, have been shown to be noisy, i.e. they contain triples that contradict each other at least to some extent, and thus require additional reasoning in order to deem which triples are correct [131]. In order to overcome this [41] proposed the Google Knowledge Vault that trains a model based on the existing triples to predict whether a newly extracted triple is reasonable, which resulted in superior extraction performance compared to other techniques. With regards to data integration and the goal of this thesis there are many similarities, since, every time we observe a new database with new variables and are confronted with unresolved questions, e.g. is the entity *"Temperatur"* the same as *"Temperature"* in this context, we could revert to a system like this instead of relying on domain experts. This principle is illustrated in figure 2.2 where we have existing triples colored as black links/orange cubes, while we would like to learn the red links/cubes from our "artificial domain expert". In the literature this task is generally called *link prediction*, whereas the task of whether an extracted entity such as *"Temperatur"* is the same as *"Temperature"* is called *entity resolution*, even though both tasks conceptually are very similar [98].

Since the KG itself is merely a way of storing knowledge, [41] used a Multi Layer Perceptron (MLP), i.e. a shallow NN, to create this "artificial domain expert". This model learns to capture the interaction terms between the entities and relations of the KG by building upon vectors representing these entities and relations that are learned during training. This kind of model is in the literature, and also in this thesis, referred to as KGEM. This approach is very similar to NLP approaches where words with similar meanings are located closely together in clusters in the latent space [89]. The vectors defining the location of each entity $h, t \in \mathcal{E}$ and relation $r \in \mathcal{R}$ in the latent space are simply referred to as the embeddings or embedding vectors. This principle is illustrated in figure 2.3a where a subset of the entities and relations from the KG shown in figure 2.2a are placed in a two-dimensional latent space. Here it can be seen that e.g. the sensors are located closely together in the latent space, i.e. they have a similar

meaning and a shared concept of measuring physical phenomena. Together with an interaction model, i.e. the underlying mathematical function of a KGEM that defines how the embeddings are combined, the score of a triple is derived. As an example we can look at the TransE[18] interaction model[9]:

$$f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2 \tag{2.2}$$

It is easy to spot that the interaction model tries to enforce $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, as this would result in the highest score. This is illustrated in figure 2.3b where two entities *"Sensor 1"* and *"Temperature"* and the relation *"measures"* are taken from the embedding space shown in figure 2.3a. If we now construct the triple ("Sensor 1", "measures", "Temperature") this would result in a very high score[10] using the TransE interaction model, as the combined $\mathbf{h} + \mathbf{r}$ embedding vector almost perfectly lines up with the $\mathbf{t}$ embedding vector, and thus the two have a small distance in the latent space, which means that the model has understood that "Sensor 1" measures the "Temperature".



(a) Embeddings created by KGEM

(b) Interaction Function TransE

Figure 2.3: (a) Exemplary embedding created by a KGEM from a KG and (b), a subset of those embeddings combined with the TransE interaction function.

The more intricate question we are left with is how to obtain these embeddings coming from a KG. As KGs only contain known facts, as specified by [125], KGs are usually extremely sparse, as can be observed in figure 2.2b even for this small example. Given the natural tensor structure there have been some approaches to solve this through straight tensor factorization techniques such as Canonical Decomposition/PARAFAC [42], but specifically designed KGEMs that contain additional steps seem to best preserve the structure of KGs [98, 129, 60].

[9] In the TransE publication the norm of the function is a tunable hyperparameter that also can be set to 1.

[10] Please note the negativity of the score, i.e., 0 is the highest possible score.

Hereby a KGEM should be considered as the entire pipeline that defines how to get from KG to obtain the embeddings given an interaction model and not just the interaction model itself. This pipeline can be considered non-trivial since it is composed of multiple components beyond the interaction model itself. [98] has established a solid groundwork with their excellent review of this field and others such as [129] and [60] build on this by analyzing and categorizing the logic and behaviour of the various interaction models available in the literature.

Nevertheless, in [6] it was observed that many publications in the literature proposing new KGEMs focus solely on the interaction model and neglect the performance impact of other KGEM components when stating new SOTA results. To substantiate this claim, [5] has shown that removing symmetric triples, i.e. triples where $(h, r, t) \in \mathcal{K} \implies (t, r, h) \in \mathcal{K}$ from the benchmarking datasets, which typically are used in the literature, has a significant impact on the resulting performance of the KGEMs. [61] showed that a fine-tuned re-implementation of DistMult[137], which is one of the older interaction models, can match and even outperform newer SOTA models' results. Similar observations have been made by [91] that focused solely on loss functions[11] and could observe that the loss function has a significant impact on all models. The concurrent benchmarking study in [109] with a less comprehensive scope than the research contribution in appendix A, investigated five different interaction models and has observed that older models can obtain and beat the SOTA results published by newer articles based on newly proposed interaction models simply by changing the KGEM components around the older interaction models. As this incomplete conception of KGEMs hinders the progress of the development of better models in academia, and the lack of a holistic framework impedes applied practitioners and researches in making the right decision when choosing a KGEM, the research contribution of this thesis presented in appendix A set out to deliver this missing contribution to the field of KGEM research.

The following works covering KGEMs were established during an external research stay at the Ludwig-Maximilians-Universität München with the research group of Prof. Dr. Volker Tresp and took place in tight collaboration with the research group of Prof. Dr. Jens Lehmann of the Smart Data Analytics Group at the University of Bonn.

[11] Loss functions are covered in more detail in section 2.2.2.1

2.2.2.1 Contribution: Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework

A plethora of new KGEMs is being proposed every year often focusing on introducing new architectures and mathematical principles to the interaction model while neglecting other components of KGEMs such as the training approach and loss function, which has led other researchers to

> **"[...] cast doubt on the claim that the performance improvements of recent models are due to architectural changes as opposed to hyper-parameter tuning or different training objectives"** Kadlec *et al.* in [61]

As shown in the previous section there have been many relevant research contributions that substantiate this claim and show that a KGEM and its performance are contingent on much more than just the interaction model itself. Therefore, the research contribution of this thesis presented in appendix A sets out to isolate the modular components of KGEMs found in the literature, unify them in one framework, and perform a large-scale evaluation of the possible KGEM configuration combinations to give insights into how relevant each component actually is as well as shine light on interactions between those components.

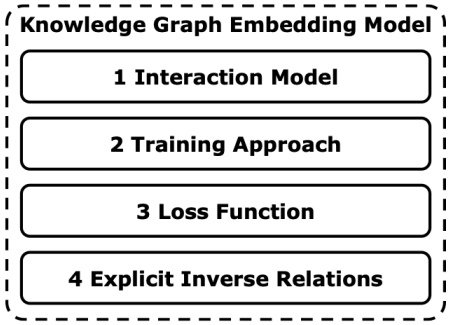


Figure 2.4: The four components of a KGEM.

Leaving the details to the publication itself, we will touch upon the main underlying KGEM framework proposed to give a broad understanding in the light of this thesis and start with the four main components of KGEMs that the contribution has identified as shown in figure 2.4, which can be described as follows:

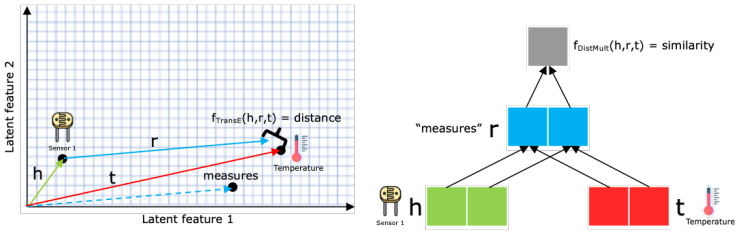
1. **Interaction Model**

   In [108] it was shown that certain interaction model architectures are more capable than others to model different relation types. This might be ob-

vious in cases like the Unstructured Model[19]

$$f(h,t) = -\|\mathbf{h} - \mathbf{t}\|_2^2 \tag{2.3}$$

that, by comparison to the earlier introduced TransE model, does not take the relation embedding into account and thus is indifferent the impact of relations. However, for more extensive interaction models this ability often becomes interdependent on several KGEM components and is thus less straightforward [108]. The works presented by [129] and [60] analyzed the available interaction models in detail and categorized them into two basic categories as shown in figure 2.5.



(a) Translational distance based scoring function of TransE.

(b) Semantic matching based scoring function of DistMult.

Figure 2.5: Comparing translational distance and semantic matching scoring functions based on TransE[18] and DistMult[137].

Whereas translational distance based models project the vectors into the latent space and measure the distance between those vectors, semantic matching models multiply the features of the latent space to derive a similarity metric.

2. Training Approach

In the RDF semantics description defined by [125] a KG should always consider the Open World Assumption (OWA) to ensure monotonicity of the triples entailed. The OWA hereby means that the triples not contained in our KG, i.e. $(h,r,t) \notin \mathcal{K}$, should be considered unknown. Contrary to this is the Closed World Assumption (CWA) where we assume that every triple not contained in the KG is *false*. However, this violates the monotonicity principle defined in [125], which means that whenever we attain new knowledge, and thus new triples, this knowledge should not invalidate prior available knowledge. This is by definition the case in the CWA setting where every time a new triple $(h,r,t) \notin \mathcal{K}$ is observed and thus considered *true* it must have been *false* before and thus violate the monotonicity. Unfortunately, the OWA does not allow us to train a KGEM

since it will encourage the model to overgeneralize by always only predicting *true*[12] [98], which would defeat the purpose of our undertaking. In order to obtain so called 'negative triples' there are two approaches, which are not always termed correctly in the literature. In this research contribution they are termed as Local Closed World Assumption (LCWA) and stochastic Local Closed World Assumption (sLCWA), which is descriptive looking at the following explanation.

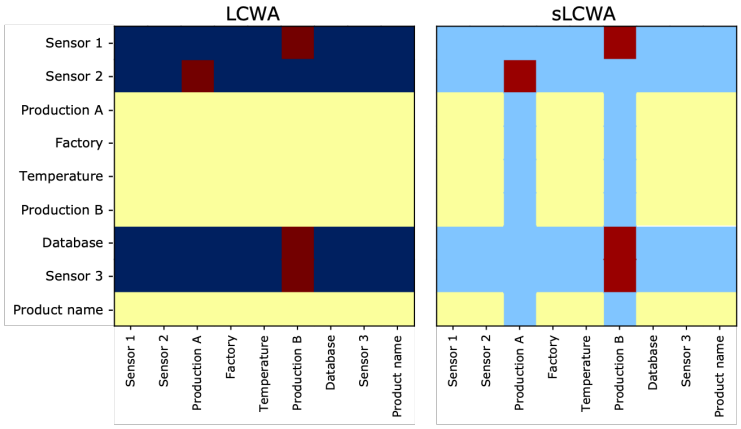


Figure 2.6: The considered negative triples for the two different training approaches for the relation *part_ of* of the KG in the figure 2.2a. The red colored triples are the ones already contained in the KG. The dark blue triples are considered under the LCWA approach and the light blue triples are considered under the sLCWA approach. All other triples are not considered.

In the figure 2.6 the two approaches are compared similar to [6], but adjusted for the example of this thesis. In the LCWA approach we take a true triple during training, such as *(Sensor 2, part_ of, Production A)* and exchange the tail t by every other entity $e \in \mathcal{E}$ and assume those triples to be *false*. The name is derived by the fact that this is the CWA, but only locally under the context of the given (h, r) tuple. As we can see already in this small example, this approach often produces significantly more nega-

[12]The intuition behind this is that a KGEM obtains all its knowledge from the KG. If all that the KGEM ever sees is *true*, it will be tempted to assume that any possible statement (h, r, t) will always be true. So in order to learn the actual logic of the KG at hand it also needs to be presented with knowledge about statements that are *false*.

tive triples than the positive ones contained in the KG. Contrary to that the sLCWA exchanges both the head h and the tail t by any entity $e \in \mathcal{E}$, but instead of assuming all those samples to be *false*, it only samples with a pre-defined distribution[13] a certain number of negative triples from this locally closed world, hence the name sLCWA. Using sLCWA the number of negative triples can be controlled regardless the size of the KG whereas with LCWA the number of negative triples is dependent on the number of entities $|\mathcal{E}|$ in the KG.

3. Loss Function

Now that we have covered the interaction model and also found a way to create negative triples, the question is how to derive a loss function that enables us to train the KGEM. Defining f as the interaction model, t_i^+ as a positive triple of our KG as $t_i^+ \in \mathcal{K}$ with $l_i^+ = 1$ as the label, and t_i^- as a negative triple that was obtained using one of the above mentioned training approaches with the label $l_i^- = 0$, we can consider following loss functions

- **pointwise**
  In pointwise loss functions the scoring of one triple is compared to the labels as e.g. in the Square Error Loss [91]

$$L(t_i, l_i) = \frac{1}{2}(f(t_i) - l_i)^2 \tag{2.4}$$

  The pointwise loss functions used in the research contribution were Binary Cross Entropy Loss (BCEL)[40], and Soft Plus Loss (SPL)[91].

- **pairwise**
  Compared to pointwise loss functions the pairwise loss functions are more natural to the OWA, since they do not assume *false* triples, but only less known ones by comparing known/true triples to unknown triples with the only condition that the score of the true triple has to be higher than the unknown triple, such as with the Margin Ranking Loss (MRL)/pairwise hinge loss [91]

$$L(t_i^+, t_i^-) = max(0, \lambda + f(t_i^-) - f(t_i^+)) \tag{2.5}$$

  Here λ is the margin parameter, defining by which margin the score for the positive triple t_i^+ has to be higher than for the negative triple t_i^-.

[13] This is usually referred to as negative sampling. A common case is to sample n entities $e \in \mathcal{E}$ using a uniform distribution[18], i.e. every entity is sampled with the same probability. Another common approach is using the Bernoulli distribution[130] to offset the likelihood of observing more heads than tails for a given relation $r \in \mathcal{R}$ or vice versa.

- **setwise**

  The setwise loss functions can be considered an extension of the pairwise loss functions in that the true triple is not only compared to one other triple, but to multiple negative triples as e.g. the Cross Entropy Loss (CEL) [61, 109]. Here we consider a set of N negative triples l_i^- defined as $\mathcal{T}^-$ where the score of the true triple t^+ is normalized with the softmax as follows

  $$p^+ = \frac{exp(f(t^+))}{\sum_{t \in \mathcal{T}^- \cup t^+} exp(f(t))} \tag{2.6}$$

  Afterwards, the loss is simply calculated by taking the log of the normalize scores p^+

  $$L = -log(p^+) \tag{2.7}$$

  In addition to the CEL the self-adversarial negative sampling loss (NSSAL)[116] was used.

4. **Explicit Inverse Relations**

   When using Explicit Inverse Relations that were introduced by [63] and [71] the set of relations $\mathcal{R}$ is extended by a set of inverse relations $r_{inv} \in \mathcal{R}_{inv}$ with $\mathcal{R}_{inv} \cap \mathcal{R} = \emptyset$. This is achieved by training an inverse triple (t, r_{inv}, h) for each triple $(h, r, t) \in \mathcal{K}$ and can be considered a vital component of a KGEM. The reason for this is twofold. First, it effectively doubles the relation space for every model that uses embeddings for the relations since there now are twice as many relations. Second, in the case of training with the LCWA presented above, the models are normally only trained on predicting tail entities t given (h, r). However, using explicit inverse relations implicitly adds head entity prediction to the training by changing the task of predicting the head entity h given a tuple (r, t) to instead predicting h given a tuple (t, r_{inv}), which has been shown to be an essential component for some KGEMs such as the work in [40].

In order to evaluate the proposed KGEM framework, 21 different interaction models, the LCWA and sLCWA training approaches, 5 different loss functions, and the use of explicit inverse triples were evaluated on 4 different commonly used KGEM benchmarking datasets, i.e., Kinships[39], WN18RR[120], FB15k-237[120] and YAGO3-10[85]. In total **73,683** experiments were carried out, which took **24,804** GPU hours in total to compute. The results of the experiments delivered ample information that is analyzed in great detail in the research contribution itself. In this section we will focus on the main takeaways with regards to the four components we just introduced.

- **Interaction Model**

  The initial claim could be confirmed as changing the above mentioned KGEM components and tuning the hyper-parameters allowed us to achieve new SOTA results for several interaction models in line with findings of other publications [61, 109]. This showed that the interaction model has to be seen as just one component of a KGEM. This means that in the future publications proposing new interaction models should also test existing interaction models with their KGEM components to see whether the effect is attributable to the interaction model itself. It was also observed that a bigger model size does not imply better model performance, but that the configuration of the KGEM is much more important. A notable mention is that RotatE [116] was the only interaction model to be among the top-ten interaction models across all datasets, whose robust performance also has been observed by other studies [108]. The model is defined as

  $$f(h, r, t) = -\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\|, \tag{2.8}$$

  where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$, d the latent embedding dimension size, and $\odot$ being the Hadamard product. The $\mathbf{t} = \mathbf{h} \odot \mathbf{r}$ operation is a rotation from head to tail entities in the complex space, which due to restricting the modulus of the complex elements of $\mathbf{r}$ to 1, i.e. $|r_i| = 1$, can be seen as a rotation in the complex plane as $e^{i\theta_{r,i}}$ and is able to model all relational patterns *symmetry*, *antisymmetry*, *inversion*, and *composition* [116].

- **Training approach**

  In most cases the LCWA allows to obtain slightly better performance, on the FB15k-237 dataset significantly better, with at the same time higher variance. This indicates that it is harder to find a good configuration. At the same time the choice of interaction model has shown to be an important factor. Whereas the previously introduced RotatE was able to achieve good results with both the LCWA and sLCWA, other interaction models were heavily impacted by one or the other training approach, such as e.g. ConvE that usually only performs well under the LCWA.

- **Loss function**

  In general the MRL has performed worst over all datasets. Even though most of the results were still in a reasonable performance range the main downside of it is that the scores have no natural reference, since positive triples only have to score higher than another false triple, which does not imply that all positive triples should have a higher score than other negative triples. The BCEL, NSSAL and SPL functions were able to achieve good results across all datasets, while the pointwise loss functions BCEL and SPL usually could obtain slightly better results under the LCWA.

However, this again coincides with the LCWA being more sensitive to the overall configuration, which has shown itself in the higher variance of the results under this approach.

- **Explicit inverse triples**

  As expected using explicit inverse triples leads in most cases to better results when training under the LCWA, which does not apply for the sLCWA.

Overall, there could not be identified a general configuration that works for all datasets, which emphasizes that finding the right KGEM is not a trivial task, but requires careful considerations not only with regards to the components, but also considering the KG at hand. With this in mind the ability of the best KGEMs created during the experiments to model different kinds of relationships was investigated according to [116] with the following relation types:

- **symmetric relations**, i.e. $(h, r, t) \in \mathcal{K} \implies (t, r, h) \in \mathcal{K}$
- **anti-symmetric relations**, i.e. $(h, r, t) \in \mathcal{K} \implies (t, r, h) \notin \mathcal{K}$
- **composite relations**, i.e. $(h, r_1, b) \in \mathcal{K} \wedge (b, r_2, t) \in \mathcal{K} \implies (h, r, t) \in \mathcal{K}$

Here it was observed that symmetric relations in general are the easiest to model, which is in line with the findings in [5]. Modelling anti-symmetric and composite relations seems to be much harder, which is plausible given that some models, e.g., DistMult, cannot model anti-symmetric relations, since it implies $f(h, r, t) = f(t, r, h)$. However, the degree to which the KGEMs could model these three types of relations also varied between the datasets, especially for the latter two relation types, implying that the composition of the KG at hand has a strong impact on the ability of a KGEM to model anti-symmetric and composite relations.

Another big motivation of the research contribution in appendix A was to boost the reproducibility in the KGEM field. Having identified the components of a KGEM the question was whether the published results of the KGEMs included in the evaluation study could be reproduced using the exact same settings as provided by their accompanying publications. Besides the fact that some publications do not provide the full setup required to conduct the experiment, it was discovered that most results could not be reproduced given the provided setup of the accompanying publication. It is important to stress that this doesn't mean the results were falsely reported. This became clear during the Hyper Parameter Optimization (HPO) study, where the published results could be obtained,

albeit using different hyper-parameters. Considering the many custom KGEM implementations that were only created for the purpose of one publication it is standing to reason that small deviations in the implementation cause this deviation. Given that a KGEM is a complex architecture that consists of many interacting components adds another layer of complexity considering the software required to conduct these experiments, which is not trivial. As this study itself required such a big and comprehensible software package to conduct the above-mentioned experiments, it was decided to develop **PyKEEN 1.0** along with the newly proposed KGEM framework. PyKEEN 1.0 is an open-source KGEM library with a focus on usability and composability of KGEMs to foster reproducibility and research in the KGEM community also outside academia, which is covered in the next research contribution.

2.2.2.2 Contribution: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

As shown in the previous section, and documented by the research contribution of this thesis in appendix A, there is a reproducibility crisis in the KGEM community often due to the many custom KGEM software implementations. This is an apparent issue to researchers in this field, and at the same time it hampers the advancement of KGEM research, as shown by the comment of one of our reviewers during the review process of this publication:

> **"The last two journal reviews that I wrote in this sub-field were two papers that were in review for 6 to 9 months because reviewers were unable to trust the results because the authors did not use a standardized research library."**
> Reviewer#3 during the review of the research contribution in appendix B at the Journal of Machine Learning Research

To thwart this ongoing issue the research contribution of this thesis, presented in appendix B, developed PyKEEN (Python KnowlEdge EmbeddiNgs) 1.0, which is an entirely configurable KGEM framework that was developed jointly with the research contribution presented in appendix A. As argued earlier, KGEM software is non-trivial in the sense that it involves many components that rely on each other, e.g., the training approach has to integrate with the loss function that again needs additional considerations when modelling explicit inverse triples. Even though other KGEM frameworks exist, these are usually not fully composable, i.e. different KGEM components cannot be freely combined, and often have a limited scope with regards to the amount of components available as analyzed in this research contribution. In addition, the large scale of

the experiments conducted required a seamless integration with optimizers, i.e. HPO libraries, while at the same time ensuring the full reproducibility and documentation of experiments as e.g., setting and documenting the random seeds of each experiment to ensure integrity of the results. Another advancement of PyKEEN 1.0 is the introduction of *Automatic Memory Optimization*, which was born out of necessity conducting the HPO study. With varying latent space dimensions and training approaches for KGEMs the hardware requirements usually vary considerably too, which often leads to Out Of Memory situations that cause the computer running the experiments to crash. To avoid this *Automatic Memory Optimization* finds the maximum permissible settings for the hardware at hand and uses sub-batching to guarantee the correctness of the results.

Table 2.1: PyKEEN current statistics (July 2021)

Metric	Value
Datasets	27
Interaction Models	31
Training approaches	2
Loss Functions	9
Negative Samplers	3
Optimizers	6
Regularizers	5
Early Stoppers	2
Evaluators	2
Metrics	16
HPO samplers	3

To promote reproducibility even further the research contribution is accompanied by a rich and detailed documentation that goes beyond the technicality of the proposed software framework itself to allow new researchers and practitioners to understand design decisions based on the underlying theory. As GitHub has shown to be a very successful collaboration platform for developers and researchers [35], the software framework is hosted there to allow other researchers and practitioners to contribute. This has been shown to be very effective as PyKEEN 1.0 has an active community that consists among others of commercial pharmaceutical research departments that use PyKEEN 1.0 for e.g. drug discovery research [17]. A big advantage of this community effort approach has proven to be the continuous transparent development where the good ideas and contributions of many are unified. As of writing this thesis the PyKEEN 1.0 framework has already expanded significantly, as shown in table 2.1. Lastly, it could be observed that it is crucial for the quality control that the software framework is used and tested by a diverse group of researchers and practitioners, as some scarce errors only become apparent in less common use-cases. This allowed us to detect errors that would otherwise go unnoticed and avoids other researches and practitioners making the same mistakes, which is especially relevant for the research contribution presented below.

2.2.2.3 Contribution: Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods

While working on the practical application of KGEMs for data integration during the course of this PhD project it was discovered that the ConvE interaction model [40] in certain setups can lead to a trained model that gives the exact same score to many different entities when performing link prediction, especially with triples $(h, r, t) \notin \mathcal{K}$. The same effect has been observed by [108] for the CrossE[138], ConvKB[96] and CapsE[97] models. [117] has evaluated this problem in more detail and has shown that the ConvKB, CapsE, and ConvE models often have high ratios of triples that receive the exact same score. At the same time it was discovered that the evaluation metrics in the field of KGEMs were fundamentally flawed, which is covered in the research contribution presented in appendix C.

When evaluating KGEMs we usually split a KG into a training KG $\mathcal{K}_{train} \subseteq \mathcal{K}$ and a test KG $\mathcal{K}_{test} = \mathcal{K} \setminus \mathcal{K}_{train}$ that contains $n = |\mathcal{K}_{test}|$ triples. Now for every of those n triples $(h, r, t) \in \mathcal{K}_{test}$ we want to see how well the trained model understood the KG at hand by computing the score for every possible entity $\{(h, r, e) \mid e \in \mathcal{E}\}$[15] and ignore existing triples in the KG $(h, r, e) \in (\mathcal{K}_{train} \cup \mathcal{K}_{test})$, so the scores do not deteriorate when there are many triples already included that usually achieve high scores. Accordingly we define the set of candidate triples as $\mathcal{C} = \{(h, r, e) \mid e \in \mathcal{E}\} \setminus \{(h, r, e) \in (\mathcal{K}_{train} \cup \mathcal{K}_{test})\}$ and derive the number of candidates as $C = |\mathcal{C}|$. Evaluating each of the triples $c_i \in \mathcal{C}$ such that $s_i = f_{KGEM}(c_i)$ we obtain a list of scores $\mathcal{S} = [s_1, \ldots, s_C]$ and compare these scores to the true triple with the entity e_{true} and the score $s_{true} = f_{KGEM}(h, r, e_{true})$. In cases where the KGEM gives the exact same score to many different triples, i.e. $f_{KGEM}(\mathcal{C}_i) = \cdots = f_{KGEM}(\mathcal{C}_{i+m}) = f_{KGEM}(h, r, e_{true})$, this leads to situations that can best be explained with figure 2.7.

Candidate	Score	Rank
e_1	0	1
e_2	0	1
e_3	0	1
...	0	1
e_{true}	0	1
e_C	0	1

Figure 2.7: The ranking problem.[14]

[14] Please note that the 'true' candidate is inserted into the list of C candidates.

[15] This case describe the so called right side prediction, i.e. predicting the right tail t given the head and relation tuple (h, r). Analogue to this is the left side prediction where the goal is to predict the head entity h given the relation and tail tuple (r, t)

In this example all entities have the score of 0. When our goal would be to use this KGEM for link prediction, this model can be considered as useless, since the number of candidates C in real KGs such as YAGO3-10 can be much higher than 100,000. This means that if we want to find the most likely candidate, we effectively have to randomly pick one of the possible candidates, which is not helpful for the link prediction task. The main issue lies in the fact that, in these kind of situations, the models achieve very good evaluation metrics that are used to deem the performance of the models. In figure 2.7 we can see that all candidates get the rank 1, which might look familiar to people often looking at scoring tables in sports, where teams that have the exact same score get assigned the same rank. Returning to our notation above, this type of scoring can be defined as having a candidate score s_{cand} and the 'true' triple score s_{true} that allows us to derive the rank of the triple (h, r, t) at hand by calculating

$$r^{+}_{true} = |\{s_{cand} \in \mathcal{S} \mid s_{cand} > s_{true}\}| + 1 \tag{2.9}$$

where r^{+}_{true} is the rank of our true triple, which is 1 in this case. We call this the optimistic rank, because it assumes the best possible case. As this might seem fair in sports, during link prediction this poses a big problem, because it does not reflect the true capability of the model. Contrary to this we have the pessimistic rank defined as

$$r^{-}_{true} = |\{s_{cand} \in \mathcal{S} \mid s_{cand} \geq s_{true}\}| + 1 \tag{2.10}$$

Here we assume the worst possible case, which would in this example result in the rank $C + 1$. Considering both the optimistic rank r^{+}_{true} and the pessimistic rank r^{-}_{true} we can now define the realistic rank.

$$r_{true} = \frac{1}{2}(r^{-}_{true} + r^{+}_{true}) \tag{2.11}$$

which gives a reasonable impression of the KGEMs performance. Assuming that in the example above we have $C = 998$ candidates, we would obtain the realistic rank $r_{true} = 500$. This makes sense from the perspective that if we randomly have to pick one out of 999 possibilities (s_{true}), which all have the same likelihood, the rank we obtain in average would be the expected value of the uniform distribution. As observed in [14] all three types of ranking can be found in KGEM implementations as well as the non-deterministic ranking approach, where the true triple is randomly placed in the range of equal scores.

As we have only looked at ranks for single triples we can expand the above-mentioned notation to multiple triples. We repeat this procedure $n = |\mathcal{K}_{test}|$ times, and thus obtain a list of scores lists $\mathcal{S}$, where $\mathcal{S}_i = [s_1, \ldots, s_{C_i}], i \in \{1, \ldots, n\}$ from which we can derive a list of n ranks r defined as $\mathcal{I}$. This can be used to calculate the Mean Rank (MR), which is one of the most used metrics

for KGEMs, as follows:

$$MR = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r \tag{2.12}$$

The benefit of the MR is that it reflects the overall performance in one single score while at the same time reflecting any change in performance, i.e. if a model improves this will be reflected by the MR. Unfortunately, it also has a main disadvantage, i.e. it does not account for the number of candidates. Having a MR of 10 is considerably better on a dataset with 1 million entities than on a dataset with only 20 possible entities. In addition, even on KGs with the same number of entities, it is easier to achieve a good MR when the KG is denser than on sparser KGs, since we filter out all triples contained in the KG during evaluation as defined above and thus reduce the number of possible candidates C. This means that the MR is not comparable between datasets and in most cases only makes sense when the dataset is known in detail. This is a strong limitation especially in the light of data integration, since we usually are looking into new datasets that have not been analyzed before and thus are unknown. Therefore, the research contribution in appendix C sets out to create a fair metric inspired by the Adjusted Rand Index [105], which adjusts the chance that a random model without predictive performance achieves the same result. In the following we briefly go through the proposed metric. A more detailed analysis and reflections upon other metrics can be found in the research contribution in appendix C. We start with the expected mean rank

$$\mathbb{E}[MR] = \frac{1}{2n} \sum_{i=1}^{n} (|\mathcal{S}_i| + 1), \tag{2.13}$$

which exhibits the expected MR that would be achieved by a fully random model. This can now be combined with the normal MR from equation 2.12 to obtain Adjusted Mean Rank (AMR) as follows:

$$AMR = \frac{MR}{\mathbb{E}[MR]} = \frac{2 \sum_{i=1}^{n} r_i}{\sum_{i=1}^{n} (|\mathcal{S}_i| + 1)} \tag{2.14}$$

The AMR is adjusted by the chance per ranked triple. In order to obtain a range that is easier to interpret the AMR was changed to the Adjusted Mean

Rank Index (AMRI) as follows:

$$AMRI = 1 - \frac{MR - 1}{\mathbb{E}[MR - 1]} = \frac{2\sum_{i=1}^{n}(r_i - 1)}{\sum_{i=1}^{n}(|\mathcal{S}_i|)}, \tag{2.15}$$

which has a symmetric range [-1,1] with $AMRI = 0$ expressing a fully random model, $AMRI = 1$ depicting a model with perfect prediction performance, and $AMRI < 0$ indicating that the model performs worse than random.

The proposed metrics were included in the benchmarking study in appendix A and are also implemented in PyKEEN 1.0 shown in appendix B. In appendix A it was shown that the different ranking approaches, i.e. *optimistic, pessimistic, realistic*, can lead to significant differences in the achieved scores of a model and [14] has shown that using the AMRI can have a considerable impact on the performance for entity alignment tasks, which is another research direction that involves aligning KGEMs that were trained on different KGs. Overall, we can conclude that the realistic ranking approach is a considerably more meaningful approach and should be the only one considered in all cases and that the AMRI is a much more robust and suitable metric to compare the performance of KGEMs.

2.2.3 Discussion

In this section two diverse ways of modelling relations were covered. The Bayesian Cut and KGEMs. With the Bayesian Cut a fully Bayesian version of a SBM that combined degree correction and a community enforcing constraint was proposed. This model has shown superior performance on the benchmark community datasets compared to other available SBMs. This achievement can be attributed to the community constraint together with the sampling characteristics of the Bayesian model. On the downside it is computationally expensive due to the required Markov Chain Monte Carlo sampling. More efficient versions that use Maximum Likelihood Estimation (MLE) approach such as [62] exist. However, the MLE approach does not exhibit information about the certainty of the found parameters, which is unfortunate given that SBMs without community constrains suffer from finding inadequate solutions as shown in the research contribution. Another drawback is that the number of clusters has to be known upfront or it has to be inferred through a computationally expensive inference [48]. Even though multilayer SBMs exist [121], most models are focused on one relation as is the proposed Bayesian Cut and despite the degree-correction that has been added to SBMs to model individual node behaviour, the focus of SBMs is to model clusters/blocks as the name suggests. This makes it a very suitable

model for more homogeneous relations and nodes where one is interested in the structure of a network at hand as well as the statistics that describe connection behaviour of each cluster such as in brain connectivity research [93] and social network community structures [95], but with regards to the application for data integration tasks it seems less applicable.

On the contrary we have KGEMs that are especially designed to preserve the three-dimensional nature of KGs constituting of entity to entity connections across many relations. With the contribution Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework a unified framework is proposed and the large-scale benchmarking study supplies rich information that allows insights into the impact of KGEM design choices. Built upon the requirements and experience of this study the contribution PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings set out to develop an open source KGEM software framework. PyKEEN 1.0 has shown that the complexity of KGEMs can be abstracted to an easy-to-use and composable software framework that is appealing to both researchers and practitioners, and at the same time fosters reproducibility in the KGEM community. With Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods another contribution to this field was made by proposing a new metric that allows a more realistic and objective way of comparing the performance of KGEMs.

In general, KGEMs have been shown to be a very versatile and capable approach to model KGs. As the implications of the covered research contributions to the purpose of data integration was broached throughout this chapter, it can easily be observed that KGEMs pose a very suitable approach for challenges associated with data integration. As a matter of fact, the existing applications, such as the Google Knowledge Vault presented in this chapter, have been shown to be extremely scalable whilst serving a similar purpose that is namely to integrate new knowledge with existing knowledge from an ever-evolving data source. How this approach can be used specifically for the data integration task in big data applications in industrial settings will be covered in the next chapter.

Chapter 3

Practical application

In this chapter we will analyze a data integration problem in an industrial setting and construct a data integration framework for industrial big data analytics. The underlying machine learning understanding of this chapter will build on the previous chapter where we have looked at the capability and principles behind MBPLS as a data fusion algorithm and RML to link data from a relational perspective. Here, especially the KGEMs showed promising characteristics and we will see how its ability can be used in the following sections. Therefore, we will start with analyzing the data integration challenges encountered in the case study introduced section 2.1.1.1. Afterwards, we will formulate a data integration framework to effectively counteract these challenges, and subsequently apply this framework on a simulated case study that portrays common challenges in the industry.

3.1 The case study, continued

In section 2.1.1.1 we have analyzed industrial fermentation process data, and have touched upon the data integration challenges that were faced before the actual analysis of the process data could start. To recap, there were about 400 extracted files split into 200 files for the seed fermentation and 200 files for the

main fermentation. Here we focus on the main fermentation, where each file represents the production of one batch, consisting of at least 1500 time points over circa 200 variables, i.e. sensors and settings that are tracked during the production.

In order to better understand where these files originate and how they usually are created, we can look at figure 3.1.[1] To start we consider the production of interest as the real-world object at hand. Even just focusing on data related parts, the production usually is a large heterogeneous system that consists of many sub-systems and different hardware components combined with various software solutions to record process data. Although companies usually try to reduce the amount of databases and storage systems related to one production system by combining the data recorded of several sensor systems into one database, the result is usually that at least a few heterogeneous storage systems and databases remain. Going from the production itself to the storage system we cross the first knowledge and technical barrier, i.e., the responsibilities go over to other departments and/or persons that does not share the same knowledge and in addition, the technical system is changed. Considering the technical solutions, a common combination is an Enterprise Resource Planning System, which usually is more focused on metadata such as when, how much, with which equipment should be produced, and a more process focused Manufacturing Execution System, which stores the actual process data and production execution steps in real-time. However, it should be noted that the covered parts between these two exemplary systems might change from use-case to use-case even with the same company, since the software suppliers usually try to cover both systems' functionalities to some extent.

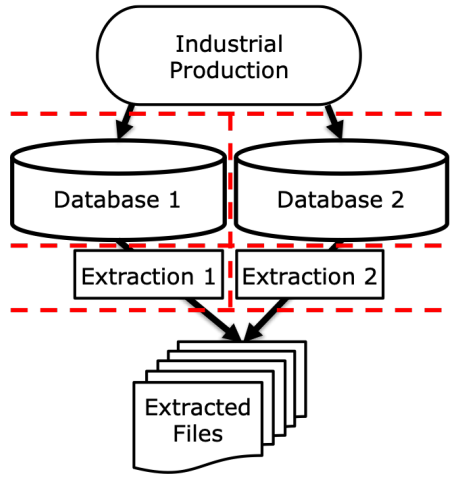


Figure 3.1: Knowledge and technical barriers in industrial settings. The red-dashed lines indicate that the responsibility for handling the task is given to different departments and/or persons, and that the technical system is different.

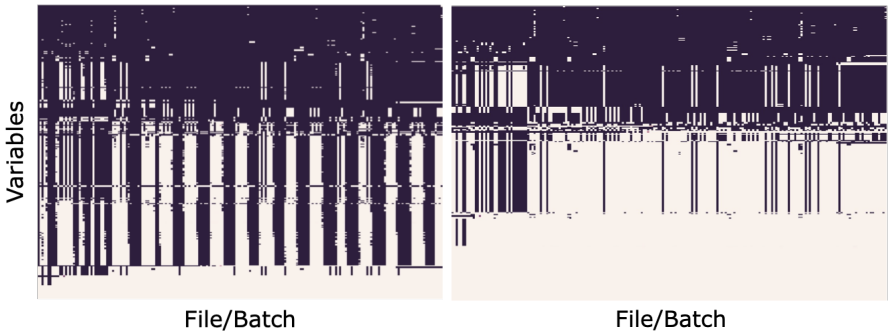
[1] Please note that this is a greatly simplified view of industrial settings, which in reality have even more knowledge and technical barriers.

The different systems used to store and manage the data go hand in hand with the persons and departments knowing those diverse systems. If the business owner of the production process has the interest to analyze the production data, other departments and persons will be involved in the data extraction process, not uncommonly even different departments/persons for the extraction from each system. These extracted files are then usually given back to process specialists of the business unit responsible for the production. However, whereas a chemist or biologist could perform the basic statistical data analysis without help in the old days, now the extreme amount of data requires the process specialist to collaborate with data scientists that have a background in machine learning to utilize the knowledge hidden in the data.

What follows next for the data scientist is termed *data profiling*, and probably known to every person that has ever worked with real-world datasets, which can be described best as "eye-balling" the dataset at hand [4]. The goal of this activity is to understand the dataset and check for possible peculiarities, which is described in great detail in [2]. To become familiar with the dataset in this case study, we can start by analyzing what these 200 main fermentation files contain and define the set of batches as $\mathcal{F}$ and the union of all variables in those batches as $\mathcal{D} = \bigcup_{B \in \mathcal{F}} B$, which we use to define the Jaccard similarity of the set of variables in one batch B compared to the union of all variables in all files $\mathcal{D}$ as follows

$$J(B, \mathcal{D}) = \frac{|B \cap \mathcal{D}|}{|\mathcal{D}|}. \tag{3.1}$$

The Jaccard similarity tells us how similar the set of variables in one batch B is to the union of all sets of variables in all batches, which in this case revealed that many batches only have $10 - 20\%$ in common with the overall dataset at hand. This is especially noteworthy, since this is data from a highly standardized process that has the same hardware and documentation requirements across all plants and countries within the organization. To make this problem visible, we can look at figure 3.2. Here, one batch is shown as a pixel column and a "type" of variable as pixel row. A light colored pixel indicates that the specific variable is available in the batch and a dark colored pixel that the type of variable is missing. In figure 3.2a, which shows the situation before all data quality issues were resolved, we can see that only very few variables are available within most batches and that there seems to be a peculiar pattern of variables that are missing for roughly half the available batches. Accordingly, the suspicious data pattern triggered a lengthy analysis process, which finally led to the data availability shown in figure 3.2b.



(a) Data sparsity before **(b)** Data sparsity after

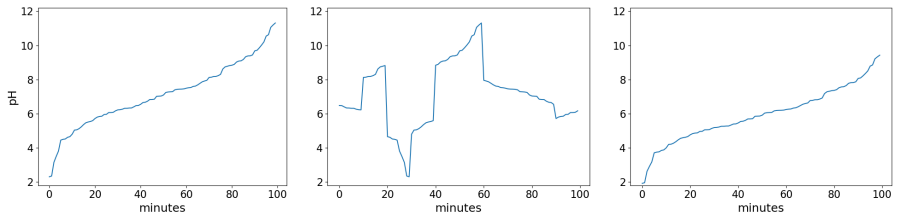
Figure 3.2: Data sparsity presented by each dark pixel representing that a particular variable was **not** found in a file/batch for (a) before finding data quality issues, and (b), after fixing the query system.

After successfully loading the data a data scientist can create many metrics about the data at hand, but often lacks the process understanding that allow to make conclusions, as pointed out by other authors stating that this challenge

> **"[...] is arguably the most difficult, namely meaningfully interpreting the data profiling results."**
> Abedjan *et al.* in [2]

As an example of this we can take the time series for pH curves in a batch production as shown in figure 3.3. A popular way to profile real valued data fields is to calculate the normal distribution for a given variable. However, this can lead to wrong conclusions as in this case the mean and standard deviation suggest that **(a)** and **(b)** are the same, while actually **(a)** and **(c)** are two variations of a normal fermentation cycle and **(b)** represents a cleaning cycle.

This is a case that is often found in the process industry, because most values are a part of time series, i.e., a stream of values that not only for themselves have a logic, but that also have a hidden logic over time as shown in figure 3.3. Even though the values in figure 3.3b neither violate the range constraint of pH values, i.e. $pH \in [0, 14]$, nor violate the parametrized domain range, they do not represent a valid fermentation cycle. At this point it is important to remember that the time series are not limited to real valued data points, but also can consist of all kinds of data, e.g., log-entries, which equivalently represent certain patterns over time. This is an added meta-layer of complexity, contrary to single values that can be checked by the database management systems with regards to



(a) $\mu = 7.1, \sigma = 1.9$ **(b)** $\mu = 7.1, \sigma = 1.9$ **(c)** $\mu = 5.9, \sigma = 1.6$

Figure 3.3: Three different time series of a pH-curve parametrized by a normal distribution. According to the normal distribution graph **(a)** and **(b)** are identical, while graph **(c)** is an outlier.

the range, syntax, the value itself or in combination with other values. Therefore, the profiling and validation of time series data often requires plotting and the visual inspection by a domain specialist, since the data scientist profiling the data usually does not possess this knowledge. Understandably, this is a very expensive undertaking that does not scale and thus is often omitted or only performed spot-wise. This can in many cases have detrimental effects on the ML models, as structural biases and errors harm the model performance [78], and might lead to viable big data applications being dropped. If we consider the above-mentioned example of having a cleaning cycle mistaken for a fermentation cycle, even if the cleaning cycles just represent 10-20% of the overall data, this has a strong impact on model performance.

An important notion is that we are not interested in completely error free data, e.g., sensors in the process industry can often produce outlier values or might lose connection and not transmit values at all for some time. This data should be considered *in domain* and should also be considered when designing ML applications or analyzing the data. Therefore, the pressing issue is to find structural errors, whose detection usually requires domain experts to perform manual data inspection while also considering that in big corporations the data consumed by single ML applications can easily span over the domain expertise of several experts. To determine how this issue can be solved we will analyze the concomitant requirements and construct a data integration framework in the next section.

3.2 Formulation of a Data Integration Framework

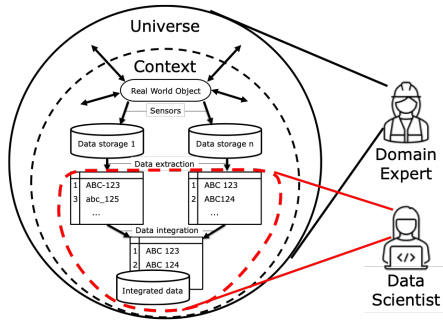


Figure 3.4: The different perspectives of a domain expert and a data scientist when being confronted with process data.

To understand why domain experts seemingly have such supernatural powers we take a big step back and look at the context we actually operate in, which is illustrated in figure 3.4. Here we see the data integration context presented earlier in this thesis in figure 1.1 and the therewith related perspectives of a data scientist and a domain expert on the overall context. It becomes obvious that the data scientist is constrained by a very limited view, while the domain expert not only sees the entire data creation context, but in addition also can relate the real-world object to other objects in the company as well as outside. To elaborate on this, a data scientist might be aware of the fact that the value of a pH sensor should not exceed the range of $[0, 14]$, however, the knowledge that the lower bound of the pH in this particular case is limited to 2.5, due to the use of vinegar, is only visible to the domain expert. In another case the protein content of barley used for the beer fermentation might be dependent on the weather conditions of that season, which shows that the required knowledge expands across the context of the production itself into the universe that surrounds us, which the domain expert is aware of. It is important to notice that we do not argue that the data scientist is not capable of understanding these relations, we just state that the logical relations of how the real-world object that created the data is linked to other real-world objects, and information that would allow the data scientist to establish the required logic, is not available for the data scientist.

To emphasize the relational nature of building the required knowledge we can imagine how we would approach this problem ourselves. The domain specialist certainly does not remember the correlations between the season and the protein content of barley as actual full data, but rather remembers that there was a specific relation. Cognitive science shows us that the ability of humans to learn from very few, or sometimes no examples, is the ability to derive conclusions based on the similarity of objects [72], and thus not remembering correlations matrices to derive conclusions. As announced earlier in chapter 2 we would like to build our own "artificial domain expert". As the excellent ability to

understand data of the actual domain expert can be attributed to being human [72], we will have to understand how humans work.

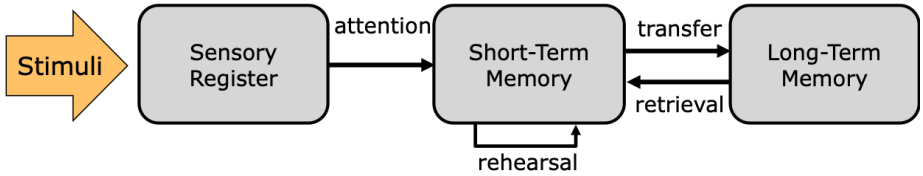


Figure 3.5: The Multi-Store Model proposed by [8].

Figure 3.5 shows a model of how the human memory works as proposed by [8]. According to [8] these three components can be explained as follows:

- **Sensory Register**

  The sensory register handles all environmental input that our senses encounter. Thereby it processes the input for each sense separately and holds it available for the short-term memory for a very short time after which it decays.

- **Short-Term Memory**

  The short-term memory can only hold a very small amount of data, which is assumed to be about 7 pieces of information. This is why it uses the attention mechanism illustrated in the figure, i.e. it chooses the information that was made available from the sensory register. When the short-term memory encounters information it is believed to automatically transfer this information to the long-term memory. However, the short-term memory can also choose to rehearse information, which will increase the amount of this particular information transferred to the long-term memory. Other research has proven that this assumption is too simple and that the short-term memory, also called working memory, is more complex as it continuously relies on information already known and stored in the long-term memory and uses it combined with the context to steer the attention [34]. In line with this notion we use our own interpretation, which is that the rehearsal can be considered as a type of reasoning, where the information presented is compared to the logic available in the long-term memory, which activate the short-term memory once the presented objects are considered important. This can be seen as the short-term memory using the long-term memory to decide where to pay attention. Furthermore, unknown objects can get attention as well, but require a more elaborate rehearsal as these objects have to be compared to already known objects and the currently established logic.

- **Long-Term Memory**

  The long-term memory serves as an unlimited storage for information in a more abstract sense such that it does not store the full data presented to the sensors. The long-term memory saves the information presented by the short-term memory and allows it to retrieve past information. As the literature has engaged in a lengthy discussion on how long-term and short-term memory are different [34], for this thesis we consider the long-term memory an unlimited storage for objects that were reasoned by the short-term memory and make the important distinction that the long-term memory, contrary to the short-term memory, does not engage in reasoning or rehearsal as described under short-term memory.

Based on these insights we will now formulate a data integration framework that allows us build a human like artificial domain expert.

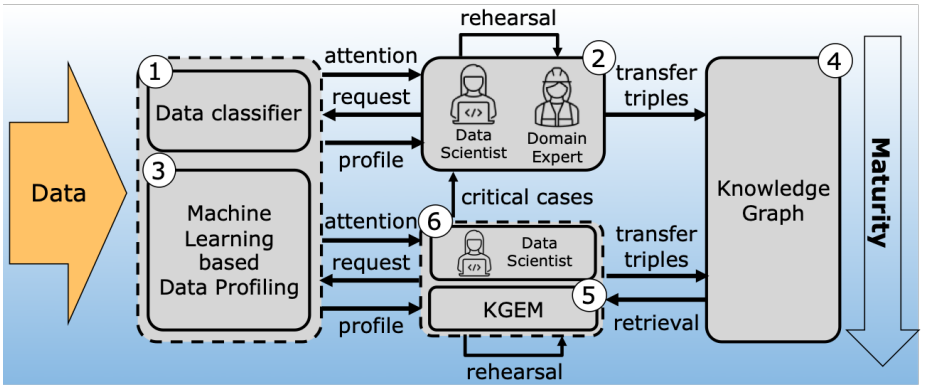


Figure 3.6: A data integration framework.

In figure 3.6 the proposed data integration framework based on the multi-store model by [8] is shown. In the following we will elaborate on the individual components in more detail:

(1) In the first step the data is classified regarding its type, e.g., textual data, time series data, image data. This classification can be done based on the filetype, which is very suitable for image files, but for data stored in a database it might require further rule based checks, e.g., small auto-correlation samples to determine whether a column is containing a time series. In addition, the data classifier is considered to consist of the accompanying metadata found in the industry, i.e., database, factory, sensor,

production unit etc. that identifies where the data is coming from or what object it is describing, which is different from metadata created through data profiling as found in the literature [84, 123] that will be covered in point 3. The intention of data classifiers is to decide how the data should be analyzed optimally to get a thorough understanding of the data at hand and therefore, these operations are computationally very inexpensive and can be based on simple rules as well as just passing the context related metadata.

(2) The second stage starts with consuming the data classifiers and deciding how this data should be profiled. The reason that the data is not profiled directly is to save computational complexity and not compute meaningless metrics for all incoming data. This can be thought of as the attention mechanism of this framework. As shown in the figure this is work to be carried out by the data scientist and the domain expert in collaboration, as the domain expert holds the knowledge about how data can be profiled in the most meaningful way, which is closely related to the principle presented in [36] and [29]. The important part is that this knowledge is converted to triples and stored in the KG, e.g., *("dataclass_1", "requires_profiling", "mean_and_std")*, which allows us to remember how the domain specialist profiled the data. Now these profiling requests, which we elaborate under point 3, are created and after completion are returned to the data scientist and the domain expert. Based on the profiling of the data the domain specialist might agree that the data is representative, in which case we save this knowledge as triples into the KG. An important notion is that we follow the mantra that Google introduced with their KG in [112], which is to store "things, not strings". This means that we consider one column storing data a "thing". Thus, data representing one conceptual object, often represented by one column in a database, is stored as a unique concept and thus the triples are saved in relation to this. To put it in a more explicit way, this means that we call the pH data created by a specific sensor a "thing". From this we derive many factual triples such as *("thing_pH_123", "has_dataclass", "dataclass_1")* and *("thing_pH_123", "has_profile", "mean_and_std_123")*. Please note that these triples are just storing abstract facts in the KG, whereas the relating embodiment of the actual numbers is handled by part (1) and (3). This abstract construction of relational facts allows us much more. First, this allows us to explicitly state that the same data might be saved in several databases and storage systems. A domain expert can for example put knowledgeable facts into the KG, e.g., *("unit_pH", "has_profile", "pH_range")*, which stores the permissible range of the pH unit. This again allows an indirect linking over the KG by knowing that our "thing" has the unit pH by storing the triple *("thing_pH_123", "has_unit", "unit_pH")*. Second, this allows us to store required data cu-

ration, pre-processing and cleaning procedures as facts in the KG. This is very much in line with the works that have called for and shown that including such knowledge about the data at hand can be very beneficial to handle the data in a more correct way [122, 123]. Also, this aligns with the works in [84] and [83] that have used a metadata based feature vector to train ML models to predict how to detect data errors and which cleaning workflows should be used. However, it should be noted that we in this case use KG and thus not a feature vector based approach. This allows us to follow the OWA of KGs, which means that we only look at the facts deemed necessary by the domain expert, which improves the computational efficiency, since we do not have to calculate every feature every time.

(3) In the third stage the data is profiled in a bespoke way as requested by the data scientist in collaboration with the domain expert. The work in [2] gives a great overview of data profiling tasks and the work [123] has analyzed the connections between the data profiles, called metadata in their work, and data quality issues. In general, this task aims at creating structured information that represents the data at hand in a more meaningful way that makes it easier to manage the data. As an example we can take the body height measurement of one million people. This data is much easier and more meaningful to describe with a mean and standard deviation than to look at the one million values simultaneously. However, as promoted in the introduction of this thesis we want to opt for a pure machine learning approach even though we do not excluded hand crafted features, if meaningful. To elaborate on this statement, the problem with hand crafted features, such as the mean and std, is that they often are simple and subject to not covering the entire truth. Especially in the process industry that works with many time-series this is a known problem. Therefore, we should use unsupervised non-linear machine learning models. Especially for time series we can use a combination of a Variational AutoEncoder (VAE) proposed by [66], and a Convolutional Neural Network (CNN) as shown in [70] that creates abstractions of patterns similar to how humans perceive vision. The benefit of such a combination is that the VAE embeds the CNN in an approximate standard normal distributed space as shown in figure 3.7. Here we can see how these very complex time series are embedded in the latent space in a principled statistical way by the VAE. Therefore, we can make use of non-parametric clustering algorithms such as DBSCAN[44] without the need to tune parameters. This allows us to establish domain specific clusters as shown in figure 3.7. From this the data scientist can capture the entire complex domain of a production specific time series with the profile *"embedding_class_123"* and save the triple *("thing_pH_123", "has_profile", "embedding_class_123")* to the KG as discussed in point (2). This shows how powerful unsupervised NN can be to capture complex data and the good things is that this principle can basically be used for any

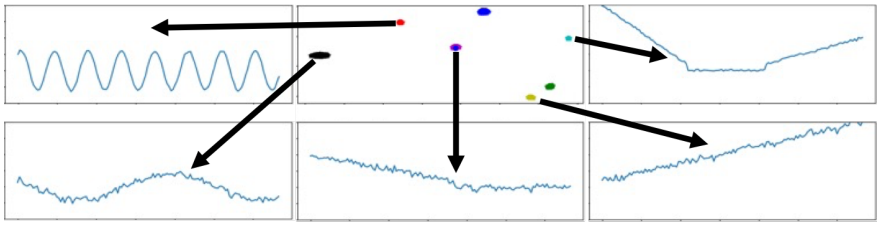


Figure 3.7: Latent space embeddings created by a Convolutional Variational AutoEncoder (CVAE) with 16 latent dimensions, shown in the two-dimensional space reduced by PCA. The five shown time series are sampled from the model.

kind of data, as e.g., shown in [31].

(4) The fourth component is a less active part as it is the long-term storage. In the previous stages we have defined how the data scientist and domain expert create triples that classify the data at hand, define which data profiles should be created for specific classes of data, and how data has to be curated, cleaned and pre-processed. This information is stored in the KG representing the relational facts between these concepts and objects. At this time we already have a system similar to the concept of the Enterprise KG presented in [25], whose objective was to profile all company data dataset by dataset in order to create a corporate KG that shows the relations between datasets and their similarities. This in return allows to query the KG similar to other databases. In our case we could query the KG for all variables that share a specific latent embedding as introduced above by asking for all things as *(?, "has_profile", "embedding_class_123")*. From this point the KG can be seen as a natural integrator of knowledge [33], which combines various data sources and puts them in a global context. However, contrary to [25], we at this point still rely on the domain specialist instead of automated data profiling, which is not a very scalable approach, but not without a reason. The approach so far resembles the curated approach explained in section 2.2.2, where experts create a KG. Our ambition so far has been to characterize how a domain expert understands and profiles data, which is materialized and formulated by the data scientist and stored in the KG in the form of triples. As indicated in figure 3.6 we can progress to stage 5 once we reach a certain maturity and have established a baseline of knowledge in our KG.

(5) As the KG in stage 4 represents the relational abstraction of how a domain expert sees the data at hand, we can finally start to build our "artificial domain expert". As elaborated in section 2.2.2, KGEMs are a very suitable

approach to learn and understand the relational data represented in a KG. To do this, we can train a KGEM based on the triples retrieved from the KG. The intriguing part about this is that with this approach we start to learn to think and reason as the domain experts, i.e. all experts across the entire organization, since step (1) to (3) can be carried out by multiple domain experts, which allows us to build a global enterprise domain expert model. Once the KGEM is trained it can effectively replace the domain expert and allow a collaboration with the data scientist or even an automation as we will elaborate in step (6).

(6) Together with the KGEM the data scientist can now resolve most problems without the help of the domain expert. Please note that the goal is basically to completely automate this task with the KGEM and only in cases that the KGEM cannot solve without help, the data scientist can have a closer look and also contact the domain expert in process (2) to resolve unresolved questions in an active learning fashion, similar to [83]. Since the KGEM has learned from the domain expert in an early state which data classes should be profiled in which way, the KGEM can also without help reason whether newly presented data is acting as to be expected based on the metadata, which kind of errors it has or which data pre-processing steps it should undergo. To automate this, the KGEM can essentially crawl over all databases in the company, since it has learned to profile the data, store new insights that align strongly with the prior knowledge already stored in the KG. Another great functionality of the KGEM is that the data scientist can now use it as an oracle. In cases where the data scientist wants to investigate a dataset, all relevant questions about the data at hand can be answered by the KGEM, while only unresolved requests have to be clarified with a domain expert, which is the main issue for data scientists when creating big data applications.

Overall, the here presented network has several nice merits. First of all, it allows to create an artificial domain expert that learns to interpret and evaluate the data as the real domain experts, while it still allows the interaction with data scientists and domain experts subsequently to support lifelong learning of the model and can be integrated in active learning approaches. Second, especially the fact that it is based on triples under the OWA makes it superior to fixed vector based ML approaches, i.e., each "object/thing" we want to describe can have from zero to thousands of related pieces of information and the KGEM naturally handles the sparsity and utilizes the underlying logic of the KG. This is also the reason why this approach is very scalable, both in the number of data profiles as well as the number of objects it can handle, since it only calculates the data profiles that are deemed necessary based on the data classification, which is regardless of the number of available data profiling approaches and

objects globally stored in the KG. At the same time the data classification and profiling mechanisms in steps (1) and (3) can be simultaneously utilized by both the experts in step (2) and the KGEM in step (5). During these steps new triples might be created and can be added to the KG at runtime. Third, the computationally expensive task of relating all obtained features between all objects is only occurring when the KGEM is trained, which might be performed periodically and can in large-scale settings easily be distributed as shown in [77], where a distributed training setup was used to train a KGEM on the full Freebase KG that contains 121,216,723 entities, 25,291 relations and 2,725,070,599 triples. Having covered the principles behind this framework, we will now look at some experiments to see how it can be deployed to solving issues as discussed earlier.

3.3 Experiments

In the following we will conduct a few experiments to show the general concept of how the data integration framework based on a KGEM can be utilized. We use synthetic data to focus on the theoretical implications that allow us to understand the architecture behind this approach.

As a starting point we have a typical use case that can be found in the processing industry, which is data stored in a data lake containing 400 files exhibiting 400 batches of a production of *"Product A"* in *"Factory A"*. Each file contains 10 variables displaying the time series of one particular sensor for each batch. In the following we can see a few samples of the 10 variables

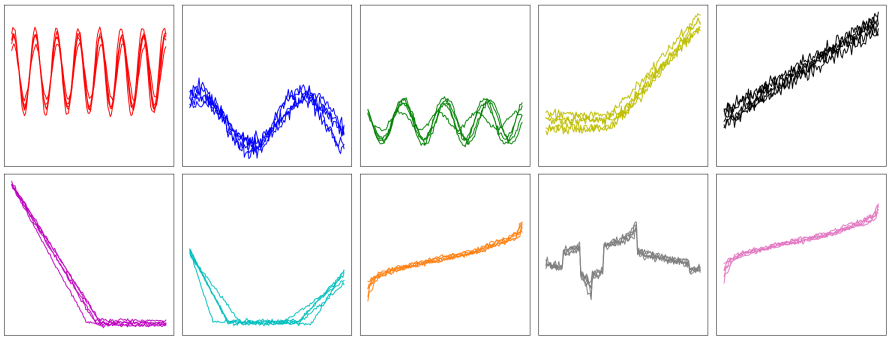


Figure 3.8: 5 randomly sampled time series for each of the 10 variables. The variables are for simplicity called (●) "Var 1", (●) "Var 2", (●) "Var 3", (●) "Var 4", (●) "Var 5", (●) "Var 6", (●) "Var 7", (●) "Var 8", (●) "Var 9", and (●) "Var 10".

Considering the time series shown above, it would not make sense to look at single values in time, but rather at the time series as a whole. In general it is important to understand how the domain expert characterizes this data. Domain experts in the process industry can usually quickly tell time series of different sensors apart and in addition characterize these as e.g. outlier or incorrect data, even though they cannot recall every exact absolute value at each point in time for such a time series of a specific sensor. As the goal of the stages 1-3 is to teach the KGEM how a domain expert perceives and analyzes data, we have to rely on machine learning models that behave similar to humans in profiling data without the need to be explicit. One way to mimic this human ability is to project the time series into a latent space by using a CVAE.

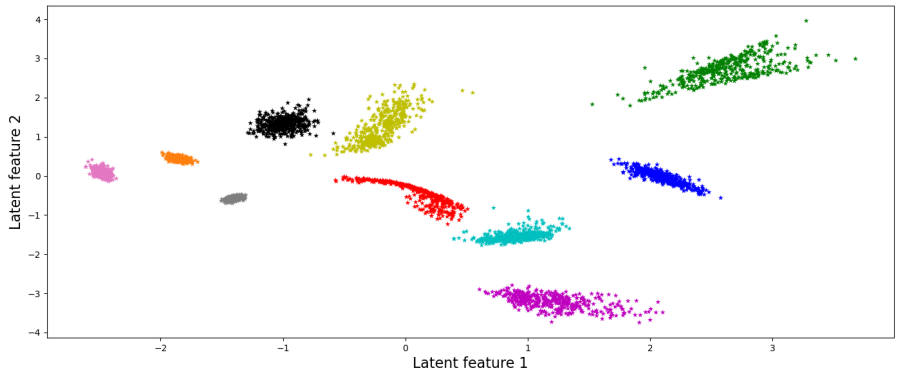


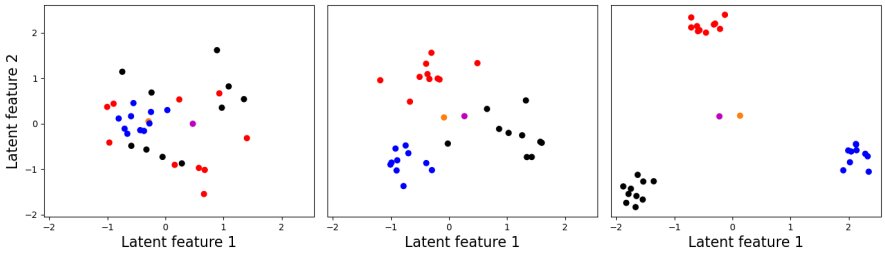
Figure 3.9: All 4,000 time series from all 400 files embedded in a 16 dimensional latent space by a CVAE, shown in a two-dimensional space reduced by PCA. The group assignments are based on DBSCAN clustering on all 16 dimensions with the colors aligned with figure 3.8.

As shown in figure 3.9, the CVAE puts similar looking time series close together in the latent space. The tricky part is how to construct meaningful triples out of these points in the latent space. One suitable way is to use a density based clustering approach, such as DBSCAN[44], to cluster similar time series into "embedding classes". The derived embedding classes can now be saved as triples, e.g., *("Var 2", "has_profile", "embedding_class_2")*. Given that we assume this to be a suitable approach, we also have to save the classification for each variable, e.g., *("Var 2", "is_class", "time_series")* as well as the chosen profiling approach, e.g., *("Var 2", "profiled_with", "CVAE_DBSCAN")* and *("time_series", "profiled_with", "CVAE_DBSCAN")*. This will later on allow the KGEM to learn which approach should be taken to profile certain data classes, e.g., time series, textual data, images etc.. The reason for this approach of reducing complex signals to concise features is to resemble the human behaviour of storing knowledge.

If a human sees a red car, it does not save an actual picture including all details of the car, but rather stores the features of the car in the brain. In reverse, if I might tell you that I'm thinking of a red Italian race car there is a good chance that you as a reader can envision a picture of such a car. What is even more fascinating, is that you very likely also can imagine the sound it makes without further information. We can assume that the two parts, i.e. the sound and the visuals of the car, come from two different embedding models, i.e. your eyes and your ears. It is only that your brain connects these two models in a relational

fashion that allows you to connect the two distinct signals. For that reason the proposed data integration framework also uses embedding models to consume data in a useful way to construct triples out of them that allow the KGEM to construct meaningful relations between the data at hand.

To come back to the above-mentioned use case, we use this approach to automatically parse the 400 files and create triples. The assumption is that these files are all homogeneous, with the column names being {*"Var 1", ..., "Var 10"*}, resulting in the respective embedding classes {*"embedding_class_1", ..., "embedding_class_10"*}. Furthermore, we assume that the variables are created by the respective sensors {*"Sensor 1", ..., "Sensor 10"*} and that each variable is created during the production of *"Product A"* in *"Factory A"*. As an example this results in the following triples for *"Var 2"*: *("Var 2", "has_profile", "embedding_class_2")*, *("Var 2", "created_by", "Sensor 2")*, *("Var 2", "for_product", "Product A")*, *("Var 2", "part_of", "Factory A")*. In addition we store one triple *("Factory A", "produces", "Product A")*. With these triples saved in our KG, we can train our KGEM. Based on the KGEM framework presented in section 2.2.2.1, we train our KGEM with the following four components, i.e., the RotatE interaction model shown in equation 2.8, the LCWA training approach, the SPL loss function and no explicit inverse triples.



(a) 50 epochs **(b)** 100 epochs **(c)** 200 epochs

Figure 3.10: The entity embeddings of the KGEM during training. (●) "Sensors", (●) "Variables", (●) "Embedding classes", (●) "Product A" and (●) "Factory A". The embeddings are reduced to the two-dimensional space based on PCA.

In figure 3.10 the entity embeddings of the above-described KGEM are shown during training. Here we can observe how the model slowly builds a logic structure of the entities as it converges. In figure 3.10c we can see how the model has grouped the sensors, variables and embedding classes as categories for themselves. This is fascinating since the triples, as shown above, do not contain these links directly and thus are reasoned by the model itself.

This capability is a consequence of the KGEM creating similar embeddings for entities with similar relations. To see how we can utilize this fact for challenging data integration tasks we slightly adjust our above-mentioned use case and consider the following example. The 400 files described above actually come from two factories, one in Denmark and another one in the United States. Here it might be the case that the temperature in Denmark is saved in the degree Celsius unit, while in the United Stated it is saved in degree Fahrenheit. Both variables measure the same physical phenomena, but their absolute signal is offset by a factor plus an absolute constant, but both variables have the same name, i.e., "Var 8" in our case. In figure 3.8 "Var 8" and "Var 10" show such phenomenon. They look very similar, but there is a structural difference between the two. With regards to our use case, this means that "Var 8" contains "embedding_class_8" for files from the Danish factory and "embedding_class_10" for files from the U.S. factory, while the actual column "Var 10" is not contained in any of the files and we wrongfully assume that all files come from the Danish factory only.

In figure 3.11 the above-mentioned KGEM that was used for figure 3.10 is trained on the just mentioned data with *"Var 8"* that erroneously contains the data of another variable in 200 of the 400 files. This ratio emphasizes that this is no outlier, as half of the data under *"Var 8"* actually is wrong. Nonetheless, the model seems to notice that *"Var 8"* is different from the other variables. As we are still in the unsupervised setting, i.e. the data scientist is working without the supervision of the domain expert, this functionality can be very helpful for the data scientist to direct the right questions to the domain expert. The remaining problem is how to identify the variables that seem to behave abnormal. From a visual perspective clustering might seem to be a suitable approach, but since the embeddings often have hundreds of dimensions clustering itself can be a challenge. To allow us to do this in an efficient way, we have to follow a design guideline, which is that any data profiler is only allowed to create one triple per file inspected, e.g., *("Var 2", "has_profile", "embedding_class_2")*, i.e.

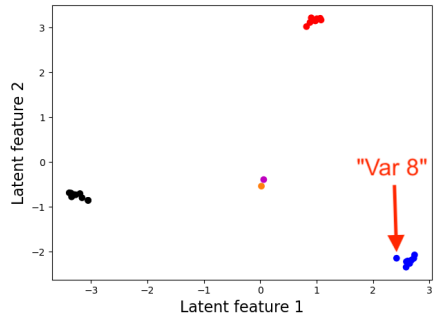


Figure 3.11: Training the KGEM with *"Var 8"* containing two different signals under the same column name. Colors are aligned with figure 3.10. The embeddings are reduced to the two-dimensional space based on PCA.

having used ten profiles would result in ten *"has_profile"* triples. Under this assumption, the KGEM should only have learned to predict one true triple per used profiler. The more heterogeneous the inspected datasets are the more true triples will likely be predicted. Whether a triple is true or not is often defined by using a threshold value, which aside desired sensitivity is highly dependent on the chosen interaction model and loss function. In our case, i.e. using the RotatE interaction function together with the SPL loss function wrapped in the sigmoid function, we set it to 0.25, which is exactly the middle of the value range that this particular KGEM can predict. Doing this results in detecting *"Var 8"* as an odd variable.

Approaching these problems with the above-described approach is a great way for the data scientist to make interesting findings that can be presented to the domain expert. Once the findings are verified by the domain expert, only the correct triples remain in the KG. With the now corrected knowledge the KGEM can be retrained and used for immediate verification without the domain expert when new datasets are investigated and converted into proposed triples. Every time there is a disagreement between the KGEM and an proposed triple, the data scientist can go with this particular piece of information to the domain expert and ask for verification, which will allow for an ever-growing knowledge base about the entire data available in a company.

3.4 Discussion

In the previous sections we have covered typical data integration problems in the process industry and have looked at how a data integration framework based on KGEMs can help us to solve these challenges. As the experiments section focused on the principles behind this approach and did not perform large-scale experiments that would be found in the industry, the experiments should be considered a proof of concept. Hereby, one of the remaining challenges is how to construct the triples when profiling datasets and saving metadata. From the research contribution in appendix A we know that symmetric relations are significantly easier to learn for KGEMs than anti-symmetric and composite relations. Nevertheless, the abstraction of complex relations in a pure machine learning approach that does not require to pre-define rules or feature engineering is a very promising approach to build artificial domain experts that can help companies to solve complex data integration problems that go beyond single values or columns alone. Hereby, the KGEM plays the integrator role that combines relational knowledge from the profiled data in a manner similar to humans, which allows for a more principled approach than the fixed vector based machine learning frameworks.

Chapter 4

Discussion

In the previous chapters we have looked at the data integration problem and have investigated ML approaches that potentially could solve this challenge. As the discussions in the previous chapters were more related to technical aspects and the research contributions themselves, we will now reflect upon the overall aspect of data integration. Overall, this thesis argues in line with the literature that the hardest problem in data integration is to combine findings about data points with their respective relations to other pieces of information that eventually allow a meaningful interpretation of these. Usually, the literature guides towards the reliance upon domain experts that, contrary to solely data-based approaches, are able to resolve these complex problems.

In the first research question we set out to investigate *"Which ML methods are suitable for data integration"*. Section 2.1 investigated data fusion as a sub-part of data integration. The case study in appendix F showed that the MBPLS algorithm is a very suitable approach to merge multiple data sources, but at the same time is very susceptible to many data integration problems. The findings from the case study confirmed that relational knowledge about the data at hand is required to resolve the data integration challenges in a meaningful way. To cover this aspect this thesis has provided multiple research contributions in the area of RML in section 2.2, where the focus has been on KGEMs that have been proven to be a very suitable way of modelling relational patterns. The motivation for using KGEMs for data integration is that they have been proven

to be a good integrator of a vast amount of knowledge [41]. The KGEMs ability of predicting new knowledge based on existing facts is a very powerful approach that allows to build a human-like artificial domain expert to solve the hardest problem in data integration as outlined above.

One aspect the presented research contributions did not cover is the integration of literals, i.e., triples that are accompanied by a value, which allow a more realistic embedding of number ranges [69], e.g., one age entity combined together with an age value instead of creating one entity per age value. In addition, the focus of this thesis has been on the transductive setup, i.e., models that only can predict links between entities and relations that were known during training. Contrary to this, inductive approaches, such as [56] and [119], can generalize to unseen entities by deriving similarities to other entities without requiring the retraining of the KGEM. However, both approaches are rather an add-on to the existing KGEMs and can be added as desired at the expense of additional computational requirements and complexity. The downside of not including these approaches is that the training of the KGEM is required more frequently especially in the beginning, i.e., we convert the inductive task to a transductive task. Nevertheless, as the KG is very small in the beginning, the training will also be computationally very inexpensive. Once the KG contains a large number of triples, the training will become more expensive, but at the same time it will become less likely that we encounter unseen relations and entities, which naturally mitigates the downside of the transductive setup in this case.

Considering KGEMs as the knowledge integrator to replace the domain expert we look at the second research question *"What is a suitable generic framework of data integration under the pure ML approach"*. In chapter 3 the data integration problem from the above-mentioned case study was revived. Here it was shown that especially in the process industry the main problem is to know what the data at hand actually means and what exactly it represents. To overcome this challenge this thesis has proposed a data integration framework that is build around KGEMs which embody the artificial domain expert. To demonstrate the pure ML approach, the experiments have shown a proof-of-concept of how recent advances in unsupervised ML algorithms can be combined with KGEMs under this framework to build a fully autonomous ML system that can handle data integration challenges with a focus on the process industry. Regardless, this approach potentially could be used for any industry and offers great versatility and extensibility for the following reasons. First, while the KG is likely to be based on company data, it can continuously be enriched with facts from knowledge sources such as Wikidata [124]. Second, using entity alignment methods the KGs learned at different subsidiaries of a company can be joined globally [12], which means that it does not necessarily require a centralized approach. Third, during the case study presented in appendix F it was noticed that in large corporations the reconciliation of data integration problems often requires

multiple domain experts, as the vast amount of big data exceeds the capability and horizon of individual domain experts, which is another point that shows the necessity of this approach in the age of big data, as KGEMs can easily exceed this boundary and thus allows the integration of knowledge that far exceeds the capabilities of humans.

Overall, it should be noted that this is not a mere error or outlier detection system. The baseline of this approach is to build an artificial domain expert that integrates all knowledge about the data in a company, which first and foremost allows us to resolve data integration challenges by understanding how to interpret the data at hand. But beyond this, the framework can be used for data discovery, similar to [25], but instead of explicitly having to query for similarities we can rely predicting the similarities of datasets with the help of KGEMs. To illustrate this we can take the triple *("Var 2", "has_profile", "embedding_class_2")* from the previous experiments section. Once the KGEM learns which variables are similar to *"Var 2"* it can also predict which variables most likely will have the profile *"embedding_class_2"* even though the data of those variables has not been profiled yet.

Last but not least, it should be noted that this is not a plug & play system at the current state. As indicated by the proof-of-concept, one of the open issues is how to create triples that allow for the most efficient learning of the KGEM, which this thesis leaves for future work.

Chapter 5

Conclusion

In this project, we addressed the problem of data integration for industrial big data applications by investigating suitable ML approaches. Overall, this thesis has a threefold contribution as shown in figure 5.1.

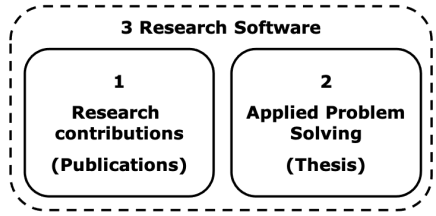


Figure 5.1: Contributions

First, this thesis has provided research contributions that investigated algorithms in the area of data fusion and RML with a focus on their suitability to solve data integration challenges. With regards to this, we looked at a case study of an enzyme producer in the process industry that involved multiple data sources. In line with the literature, this case study evidently showed that before the actual task of a big data project can be performed, such as exploratory data analysis or building a ML solution, multiple data sources require careful data integration steps which are inherently hard. The necessity of involving domain experts shows that data integration requires relational knowledge of the data at hand that often goes beyond the data source itself or is not even contained in the data itself, which is the hardest task from a machine learning perspective. Justifiably, the lit-

erature suggests to rely on humans. These findings confirmed the need for a more human-like ML approach to alleviate the dependency on domain experts to solve data integration tasks. This thesis argues that the ability of the domain experts to solve these data integration tasks relies on the ability to comprehend and remember the relations between the different data objects as well as their real-world connections to each other that might not be stored in the data.

To accommodate this requirement of understanding relational data, we looked at models that fall under the category of RML. Aside the Bayesian Cut, which is a specialized model for community detection in graphs, this thesis has investigated KGEMs in greater detail. In particular, we have developed a KGEM framework that enables composability and reproducibility and was accompanied by a large-scale benchmarking study, which made the contribution of individual components transparent, such that the necessary design decisions are more comprehensible and accessible. Furthermore, we proposed a new measure that alleviates the conceptually flawed way of measuring the performance of KGEMs, which is vital when relying on KGEMs in big data applications. The ability of KGEMs to handle diverse relations and entities that can be used for entity resolution and link prediction make it a suitable integrator for the relational knowledge that is required to match the ability of actual domain experts.

Second, to utilize the power of KGEMs, this thesis proposed a data integration framework purely based on machine learning that combines other ML approaches with KGEMs as an artificial domain expert. In the proof-of-concept experiments that were covered in this thesis, this approach has been shown to be a very suitable approach that has promising merits. These merits include that it can scale vastly while it can accommodate various existing data profiling approaches and various types of information that are handled through the KGEM in a relational fashion. Thereby, the KGEM is learning the overall data structure and logic, similar to a domain expert, which allows it to solve data integration challenges. In addition, the proposed approach can also be used for further applications, e.g., finding similar products/process data in the databases.

Third, all research contributions as well as the experiments are based on research software that was created as a part of this thesis, which is openly available, accompanied by journal publications and thoroughly documented, and thus fosters future research in the covered research areas and beyond as well as applications that build on it, such as the data integration framework proposed in this thesis.

Appendix A

Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework

---

Mehdi Ali, Max Berrendorf*, Charles Tapley Hoyt*, **Laurent Vermue***, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework. *Under review at IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. *Equal contribution

Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework

Mehdi Ali, Max Berrendorf†, Charles Tapley Hoyt†, Laurent Vermue†, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann

Abstract—The heterogeneity in recently published knowledge graph embedding models' implementations, training, and evaluation has made fair and thorough comparisons difficult. To assess the reproducibility of previously published results, we re-implemented and evaluated 21 models in the PyKEEN software package. In this paper, we outline which results could be reproduced with their reported hyper-parameters, which could only be reproduced with alternate hyper-parameters, and which could not be reproduced at all, as well as provide insight as to why this might be the case.

We then performed a large-scale benchmarking on four datasets with several thousands of experiments and 24,804 GPU hours of computation time. We present insights gained as to best practices, best configurations for each model, and where improvements could be made over previously published best configurations. Our results highlight that the combination of model architecture, training approach, loss function, and the explicit modeling of inverse relations is crucial for a model's performance and is not only determined by its architecture. We provide evidence that several architectures can obtain results competitive to the state of the art when configured carefully. We have made all code, experimental configurations, results, and analyses available at https://github.com/pykeen/pykeen and https://github.com/pykeen/benchmarking.

Index Terms—Knowledge Graph Embeddings, Link Prediction, Reproducibility, Benchmarking

1 Introduction

As the usage of knowledge graphs (KGs) becomes more widespread, their inherent incompleteness can pose a liability for typical downstream tasks that they support, e.g., question answering, dialogue systems, and recommendation systems [1]. Knowledge graph embedding models (KGEMs) present an avenue for predicting missing links. However, the following two major challenges remain in their application.

First, the reproduction of previously reported results turned out to be a major challenge — there are even examples of different results reported for the same combinations of KGEMs and datasets [2]. In some cases, the lack of availability of source code for KGEMs or the usage of different frameworks and programming languages inevitably introduces variability. In other cases, the lack of a precise specification of hyper-parameters introduces variability.

Second, the verification of the novelty of previously reported results remains difficult. It is often difficult to attribute the incremental improvements in performance reported with each new state of the art model to the model's architecture itself or instead to the training approach, hyper-parameter values, or specific prepossessing steps, e.g., the explicit modeling of inverse relations. It has been shown that baseline models can achieve competitive performance to more sophisticated ones when optimized appropriately [3], [2]. Additionally, the variety of implementations and interpretations of common evaluation metrics for link prediction makes a fair comparison to previous results difficult [4].

This paper makes two major contributions towards addressing these challenges:

1) We performed a reproducibility study in which we tried to replicate reported experimental results in the original papers (when sufficient information was provided).
2) We performed an extensive benchmark study on 21 KGEMs over four benchmark datasets in which we evaluated the models based on different hyper-parameter values, training approaches (i.e. training under the *local closed world assumption* and *stochastic local closed world assumption*), loss functions, optimizers, and the explicit modeling of inverse relations.

Previous studies have already investigated important aspects for a subset of models: Kadlec *et al.* [3] showed that a fine-tuned baseline (DistMult [5]) can outperform

†*Equal contribution.*
Mehdi Ali is affiliated with Smart Data Analytics (University of Bonn), Germany, & Fraunhofer IAIS, Sankt Augustin and Dresden, Germany.
Max Berrendorf is affiliated with Ludwig-Maximilians-Universität München, Munich, Germany.
Charles Tapley Hoyt is affiliated with Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA.
Laurent Vermue is affiliated with the Technical University of Denmark, Kongens Lyngby, Denmark.
Mikhail Galkin is affiliated with Mila & McGill University, Montreal, Canada
Sahand Sharifzadeh is affiliated with Ludwig-Maximilians-Universität München, Munich, Germany.
Asja Fischer is affiliated with the Ruhr University Bochum, Germany.
Volker Tresp is affiliated with Ludwig-Maximilians-Universität München & Siemens AG, Munich, Germany.
Jens Lehmann is affiliated with Smart Data Analytics (University of Bonn), Bonn, Germany, & Fraunhofer IAIS, Sankt Augustin and Dresden Germany.

more sophisticated models on FB15K. Akrami *et al.* [2], [6] examined the effect of removing faulty triples from KGs on the model's performance. Mohamed *et al.* [7] studied the influence of loss functions on the models' performances for a set of KGEMs. Concurrent to the work on this paper, Rufinelli *et al.* [8] performed a benchmarking study in which they investigated five knowledge graph embedding models. After describing their benchmarking [8], they called for a larger study that extends the search space and incorporates more sophisticated models. Our study answers this call and realizes a fair benchmarking by completely re-implementing KGEMs, training pipelines, loss functions, and evaluation metrics in a unified, open-source framework. Inspired by their findings, we have also included the cross entropy loss (CEL) function, which has been previously used by Kadlec *et al.* [3]. Our benchmarking can be considered as a superset of many previous benchmarkings — to the best of our knowledge, there exists no study of comparable breadth or depth. A further interesting study with a different focus is the work of Rossi *et al.* [9] in which they investigated the effect of the structural properties of KGs on models' performances, instead of focusing on the combinations of different model architectures, training approaches, and loss functions.

This article is structured as follows: in Section 2, we introduce our notation of KG and the link prediction task and introduce an exemplary KG to which we refer in examples throughout this paper. In Section 3, we present our definition of a KGEM and review the KGEMs that we investigated in our studies. In Section 4, we describe and discuss established evaluation metrics as well as a recently proposed one [10]. In Section 5, we introduce the benchmark datasets on which we conducted our experiments. In Section 6 and Section 7, we present our respective reproducibility and benchmarking studies. Finally, we provide a discussion and an outlook for our future work in Section 8.

2 Knowledge Graphs

For a given set of entities $\mathcal{E}$ and set of relations $\mathcal{R}$, we consider a knowledge graph $\mathcal{K} \subseteq \mathbb{K} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ as a directed, multi-relational graph that comprises triples $(h, r, t) \in \mathcal{K}$ in which $h, t \in \mathcal{E}$ represent a triples' respective head and tail entities and $r \in \mathcal{R}$ represents its relationship. Figure 1 depicts an exemplary KG. The direction of a relationship indicates the roles of the entities, i.e., head or tail entity. For instance, in the triple *(Sarah, CEO_Of, Deutsche_Bank)*, *Sarah* is the head and *Deutsche_Bank* is the tail entity. KGs usually contain only true triples corresponding to available knowledge.

In contrast to triples in a KG, there are different philosophies, or *assumptions*, for the consideration of triples *not* contained in a KG [11], [12]. Under the closed world assumption (CWA), all triples that are not part of a KG are considered as false. Based on the example in Figure 1, the triple *(Sarah, lives_in, Germany)* is a false fact under the CWA since it is not part of the KG. Under the open world assumption (OWA), it is considered unknown as to whether triples that are not part of the KG are true or false. The construction of KGs under the principles of the semantic web (and RDF) rely

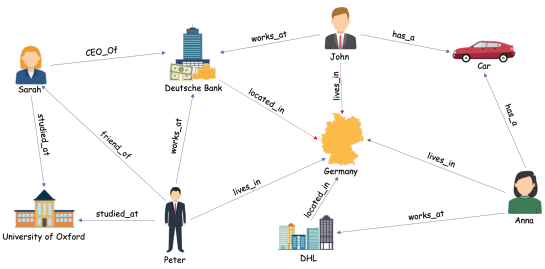


Fig. 1. Exemplary KG: nodes represent entities and edges their respective relations.

on the OWA as well as most of the relevant works to this paper [13], [11].

Because KGs are usually incomplete and noisy, several approaches have been developed to predict new links. In particular, the task of link prediction is defined as predicting the tail/head entities for $(h, r)/(r, t)$ pairs. For instance, given queries of the form *(Sarah, studied_at, ?)* or *(?, CEO_of, Deutsche Bank)*, the task is the correctly detect the entities that answer the query, i.e. *(Sarah, studied_at,* ***University of Oxford****)* and *(****Sarah****, CEO_of, Deutsche Bank)*. While classical approaches have relied on domain-specific rules to derive missing links, they usually require a large number of user-defined rules in order to generalize [11]. Alternatively, machine learning approaches learn to predict new links based on the set of existing ones. It has been shown that especially relational-machine learning methods are successful in predicting missing links and identifying incorrect ones, and recently knowledge graph embedding models have gained significant attention [11].

3 Knowledge Graph Embedding Models

Knowledge graph embedding models (KGEMs) learn latent vector representations of the entities $e \in \mathcal{E}$ and relations $r \in \mathcal{R}$ in a KG that best preserve its structural properties [1], [11], [14]. Besides for link prediction, they have been used for tasks such as entity disambiguation, and clustering as well as for downstream tasks such as question answering, recommendation systems, and relation extraction [1]. Figure 2 shows an embedding of the entities and relations in $\mathbb{R}^2$ from the KG from Figure 1.

Here, we define a KGEM as four components: an *interaction model*, a *training approach*, a *loss function*, and its usage of *explicit inverse relations*. This abstraction enables investigation of the effect of each component individually and in combination on each KGEMs' performance. Each are described in detail in their following respective subsections 3.1, 3.2, 3.3, and 3.4. We focus on *shallow* embedding approaches [15] in this work, i.e., matrix lookups represent the entity and relation encoders. Recently, several graph neural network (GNN)-based approaches for learning representations of KGs have been developed. GNNs encode entities and relations by neighbor aggregation. We refer interested readers to [14], [15]. Furthermore, learning representation for temporal KGs has gained increased interest. Because learning representation for temporal KGs is a

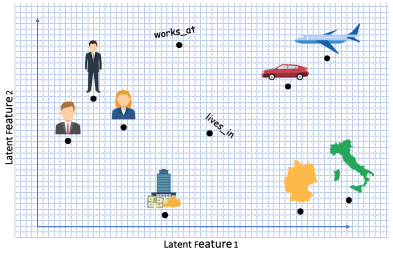


Fig. 2. An example embedding of the entities and relations from the knowledge graph portrayed by Figure 2.

distinct line of research with its own benchmarking datasets, we do not discuss temporal KGEMs in this work. Instead, we refer interested readers to [16].

In this paper, we use a boldface lower-case letter $\mathbf{x}$ to denote a vector, $\|\mathbf{x}\|_p$ to represent its l_p norm, a boldface upper-case letter $\mathbf{X}$ to denote a matrix, and a fraktur-font upper-case letter $\mathfrak{X}$ to represent a three-mode tensor. Furthermore, we use $\odot$ to denote the Hadamard product $\odot : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$:

$$[\mathbf{a} \odot \mathbf{b}]_i = \mathbf{a}_i \cdot \mathbf{b}_i \tag{1}$$

Finally, we use $\overline{x}$ to denote the conjugate of a complex number $x \in \mathbb{C}$.

3.1 Interaction Models

An interaction model $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ computes a real-valued score representing the plausibility of a triple $(h, r, t) \in \mathbb{K}$ given the embeddings for the entities and relations. In general, a larger score indicates a higher plausibility. The interpretation of the score value is model-dependent, and usually, it cannot be directly interpreted as a probability. We follow [1], [14] and categorize interaction models into *translational distance* based and *semantic matching* based interaction models. Translational distance interaction models compute the plausibility of triples based on a distance function, e.g., Euclidean distance between (projected) entities, and semantic similarity matching models exploit the similarity of the latent features usually induced by inner a product formulation.

3.1.1 Translational Distance Interaction Models

Unstructured Model [?] The Unstructured Model (UM) [17] scores a triple by computing the distance between the head and tail entity

$$f(h, t) = -\|\mathbf{h} - \mathbf{t}\|_2^2 \ , \tag{2}$$

where $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the embeddings of head and tail entity, respectively. A small distance between these embeddings indicates a plausible triple. In the UM, relations are not considered, and therefore, it cannot distinguish between different relationship types. However, the model can be beneficial for learning embeddings for KGs that contain only a single relationship type or only equivalent relationship types, e.g. *GrandmotherOf* and *GrandmaOf*. Moreover, it may serve as a baseline to interpret the performance of relation-aware models.

Structured Embedding Structured Embedding (SE) [18] models each relation by two matrices $\mathbf{M}_r^h, \mathbf{M}_r^t \in \mathbb{R}^{d \times d}$ that perform relation-specific projections of the head and tail embeddings:

$$f(h, r, t) = -\|\mathbf{M}_r^h \mathbf{h} - \mathbf{M}_r^t \mathbf{t}\|_1 \ . \tag{3}$$

As before, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are the embeddings of head and tail entity, respectively. By employing different projections for the embeddings of the head and tail entities, SE explicitly distinguishes between the subject- and object-role of an entity.

TransE TransE [19] models relations as a translation of head to tail embeddings, i.e. $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Thus, the interaction model is defined as:

$$f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p \ , \tag{4}$$

with $p \in \{1, 2\}$ is a hyper-parameter. A major advantage of TransE is its computational efficiency which enables its usage for large scale KGs. However, it inherently cannot model 1-N, N-1, and N-M relations: assume $(h, r, t_1), (h, r, t_2) \in \mathcal{K}$, then the model adapts the embeddings in order to ensure $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_1$ and $\mathbf{h} + \mathbf{r} \approx \mathbf{t}_2$ which results in $\mathbf{t}_1 \approx \mathbf{t}_2$.

TransH TransH [20] is an extension of TransE that specifically addresses the limitations of TransE in modeling 1-N, N-1, and N-M relations. In TransH, each relation is represented by a hyperplane, or more specifically a normal vector of this hyperplane $\mathbf{w}_r \in \mathbb{R}^d$, and a vector $\mathbf{d}_r \in \mathbb{R}^d$ that lies in the hyperplane. To compute the plausibility of a triple $(h, r, t) \in \mathbb{K}$, the head embedding $\mathbf{h} \in \mathbb{R}^d$ and the tail embedding $\mathbf{t} \in \mathbb{R}^d$ are first projected onto the relation-specific hyperplane: $\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. Then, the projected embeddings are used to compute the score for the triple (h, r, t):

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{d}_r - \mathbf{t}_r\|_2^2 \ . \tag{5}$$

TransR TransR [21] is an extension of TransH that explicitly considers entities and relations as different objects and therefore represents them in different vector spaces. For a triple $(h, r, t) \in \mathbb{K}$, the entity embeddings, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$, are first projected into the relation space by means of a relation-specific projection matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$: $\mathbf{h}_r = \mathbf{M}_r \mathbf{h}$ and $\mathbf{t}_r = \mathbf{M}_r \mathbf{t}$. Finally, the score of the triple (h, r, t) is computed:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 \tag{6}$$

where $\mathbf{r} \in \mathbb{R}^k$.

TransD TransD [22] is an extension of TransR that, like TransR, considers entities and relations as objects living in different vector spaces. However, instead of performing the same relation-specific projection for all entity embeddings, entity-relation-specific projection matrices $\mathbf{M}_{r,h}, \mathbf{M}_{t,h} \in \mathbb{R}^{k \times d}$ are constructed. To do so, all head entities, tail entities, and relations are represented by two vectors, $\mathbf{h}, \mathbf{h}_p, \mathbf{t}, \mathbf{t}_p \in \mathbb{R}^d$ and $\mathbf{r}, \mathbf{r}_p \in \mathbb{R}^k$, respectively. The first set of embeddings is used for calculating the entity-relation-specific projection matrices: $\mathbf{M}_{r,h} = \mathbf{r}_p \mathbf{h}_p^T + \tilde{\mathbf{I}}$ and $\mathbf{M}_{r,t} = \mathbf{r}_p \mathbf{t}_p^T + \tilde{\mathbf{I}}$, where $\tilde{\mathbf{I}} \in \mathbb{R}^{k \times d}$ is a $k \times d$ matrix with ones on the diagonal and zeros elsewhere. Next, $\mathbf{h}$ and $\mathbf{t}$ are

projected into the relation space by means of the constructed projection matrices: $\mathbf{h}_r = \mathbf{M}_{r,h}\mathbf{h}$ and $\mathbf{t}_r = \mathbf{M}_{r,t}\mathbf{t}$. Finally, the plausibility score for $(h, r, t) \in \mathbb{K}$ is given by:

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 \ . \tag{7}$$

RotatE RotatE [23] models relations as rotations from head to tail entities in the complex space: $\mathbf{t} = \mathbf{h} \odot \mathbf{r}$, where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$ and $|r_i| = 1$, that is the complex elements of $\mathbf{r}$ are restricted to have a modulus of one. Because of the latter, r_i can be represented as $e^{i\theta_{r,i}}$, which corresponds to a counterclockwise rotation by $\theta_{r,i}$ radians. The interaction model is then defined as:

$$f(h, r, t) = -\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\| \ , \tag{8}$$

which allows to model *symmetry*, *antisymmetry*, *inversion*, and *composition* [23].

MuRE MuRE [24] is the Euclidean counterpart of MuRP, a hyperbolic interaction model that is capable of effectively modeling hierarchies in KG. Its interaction model involves a distance function:

$$f(h, r, t) = -\|\mathbf{R}\mathbf{h} - \mathbf{t} + \mathbf{r}\|_2^2 + \mathbf{b_h} + \mathbf{b_t} \tag{9}$$

where the head entity is transformed by the diagonal matrix $\mathbf{R} \in \mathbf{R}^{d \times d}$ and the tail entity by the relation r. $\mathbf{b_h}$ and $\mathbf{b_t}$ represent scalar offsets.

KG2E KG2E [25] aims to explicitly model (un)certainties in entities and relations (e.g. influenced by the number of triples observed for these entities and relations). Therefore, entities and relations are represented by probability distributions, in particular by multi-variate Gaussian distributions $\boldsymbol{\mathcal{N}}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ where the mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ denotes the position in the vector space and the diagonal variance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$ models the uncertainty. Inspired by the TransE model, relations are modeled as transformations from head to tail entities: $\boldsymbol{\mathcal{H}} - \boldsymbol{\mathcal{T}} \approx \boldsymbol{\mathcal{R}}$ where $\boldsymbol{\mathcal{H}} \sim \boldsymbol{\mathcal{N}}_h(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$, $\boldsymbol{\mathcal{H}} \sim \boldsymbol{\mathcal{N}}_t(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, $\boldsymbol{\mathcal{R}} \sim \boldsymbol{\mathcal{P}}_r = \boldsymbol{\mathcal{N}}_r(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ and $\boldsymbol{\mathcal{H}} - \boldsymbol{\mathcal{T}} \sim \boldsymbol{\mathcal{P}}_e = \boldsymbol{\mathcal{N}}_{h-t}(\boldsymbol{\mu_h} - \boldsymbol{\mu_t}, \boldsymbol{\Sigma_h} + \boldsymbol{\Sigma_t})$ (since head and tail entities are considered to be independent with regards to the relations). The interaction model measures the similarity between $\boldsymbol{\mathcal{P}}_e$ and $\boldsymbol{\mathcal{P}}_r$ by means of the Kullback-Leibler (KL) divergence:

$$\begin{aligned} f(h, r, t) &= \mathcal{D}_{\mathcal{KL}}(\boldsymbol{\mathcal{P}}_e, \boldsymbol{\mathcal{P}}_r) \\ = \frac{1}{2}\Big\{ & tr(\boldsymbol{\Sigma}_r^{-1}\boldsymbol{\Sigma}_e) + (\boldsymbol{\mu}_r - \boldsymbol{\mu}_e)^T \boldsymbol{\Sigma}_r^{-1}(\boldsymbol{\mu}_r - \boldsymbol{\mu}_e) \\ & - log(\frac{det(\boldsymbol{\Sigma}_e)}{det(\boldsymbol{\Sigma}_r)}) - d \Big\} \ . \end{aligned} \tag{10}$$

Besides the asymmetric KL divergence, the authors propose a symmetric variant which uses the expected likelihood.

3.1.2 Semantic Matching Interaction Models

RESCAL RESCAL [26] is a bilinear model that models entities as vectors and relations as matrices. The relation matrices $\mathbf{W}_r \in \mathbb{R}^{d \times d}$ contain weights $w_{i,j}$ that capture the amount of interaction between the i-th latent factor of $\mathbf{h} \in \mathbb{R}^d$ and the j-th latent factor of $\mathbf{t} \in \mathbb{R}^d$ [11], [26]. Thus, the plausibility score of $(h, r, t) \in \mathbb{K}$ is given by:

$$f(h, r, t) = \mathbf{h}^T \mathbf{W}_r \mathbf{t} = \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}^{(r)} h_i t_j \tag{11}$$

DistMult DistMult [5] is a simplification of RESCAL where the relation matrices $\mathbf{W}_r \in \mathbb{R}^{d \times d}$ are restricted to diagonal matrices:

$$f(h, r, t) = \mathbf{h}^T \mathbf{W}_r \mathbf{t} = \sum_{i=1}^{d} \mathbf{h}_i \cdot diag(\mathbf{W}_r)_i \cdot \mathbf{t}_i \ . \tag{12}$$

Because of its restriction to diagonal matrices DistMult is computational more efficient than RESCAL, but at the same time less expressive. For instance, it is not able to model anti-symmetric relations, since $f(h, r, t) = f(t, r, h)$.

ComplEx ComplEx [27] is an extension of DistMult that uses complex valued representations for the entities and relations. Entities and relations are represented as vectors $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$, and the plausibility score is computed using the Hadamard product:

$$f(h, r, t) = Re(\mathbf{h} \odot \mathbf{r} \odot \mathbf{t}) \tag{13}$$

where $Re(\mathbf{x})$ denotes the real component of the complex valued vector $\mathbf{x}$. Because the Hadamard product is not commutative in the complex space, ComplEx can model anti-symmetric relations in contrast to DistMult.

QuatE QuatE [28] learns hypercomplex valued representations (quaternion embeddings) for entities and relations, i.e., $\mathbf{e_i}, \mathbf{r_j} \in \mathbb{H}^d$. Hypercomplex representations extend complex representations by representing each number with one real and three imaginary components. In QuatE, relations are modelled as rotations in the hypercomplex space. More precisely, the relation is used to rotate the head entity: $\mathbf{h_r} = \mathbf{h} \otimes \mathbf{r}$, where in this context $\otimes$ represents the Hamilton product. The final score is obtained by computing the inner product between the rotated head and the the tail entity:

$$f(h, r, t) = \mathbf{h_r} \cdot \mathbf{t} \tag{14}$$

In contrast to ComplEx, QuatE is capable of modeling *composition* patterns.

SimplE SimplE [29] is an extension of **canonical polyadic (CP)** [29], one of the early tensor factorization approaches. In CP, each entity $e \in \mathcal{E}$ is represented by two vectors $\mathbf{h}_e, \mathbf{t}_e \in \mathbb{R}^d$ and each relation by a single vector $\mathbf{r} \in \mathbb{R}^d$. Depending whether an entity participates in a triple as the head or tail entity, either $\mathbf{h}_e$ or $\mathbf{t}_e$ is used. Both entity representations are learned independently, i.e. observing a triple (e_1, r, e_2), the method only updates $\mathbf{h}_{e_1}$ and $\mathbf{t}_{e_2}$. In contrast to CP, SimplE introduces for each relation r the inverse relation r', and formulates the interaction model based on both:

$$f(h, r, t) = \frac{1}{2}\left(\langle \mathbf{h}_{e_i}, \mathbf{r}, \mathbf{t}_{e_j} \rangle + \langle \mathbf{h}_{e_j}, \mathbf{r}', \mathbf{t}_{e_i} \rangle\right) \ . \tag{15}$$

Therefore, for each triple $(e_1, r, e_2) \in \mathbb{K}$, both $\mathbf{h}_{e_1}$ and $\mathbf{t}_{e_2}$ as well as $\mathbf{h}_{e_2}$ and $\mathbf{t}_{e_1}$ are updated [29].

TuckER TuckER [30] is a linear model that is based on the tensor factorization method Tucker [31] in which a three-mode tensor $\mathfrak{X} \in \mathbb{R}^{I \times J \times K}$ is decomposed into a set of factor matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$ and a core tensor $\mathfrak{Z} \in \mathbb{R}^{P \times Q \times R}$ (of lower rank): $\mathfrak{X} \approx \mathfrak{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$, where $\times_n$ is the tensor product, with n denoting along which mode the tensor product is computed. In TuckER, a KG is considered as a binary tensor which is factorized using the Tucker factorization where

$\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ denotes the entity embedding matrix, $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$ represents the relation embedding matrix, and $\mathfrak{W} = \mathfrak{Z} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the *core tensor* that indicates the extent of interaction between the different factors. The interaction model is defined as:

$$f(h,r,t) = \mathfrak{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t} \ , \tag{16}$$

where $\mathbf{h}, \mathbf{t}$ correspond to rows of $\mathbf{E}$ and $\mathbf{r}$ to a row of $\mathbf{R}$.

ProjE ProjE [32] is a neural network-based approach with a *combination* and a *projection* layer. The interaction model first combines h and r by a combination operator [32]: $\mathbf{h} \otimes \mathbf{r} = \mathbf{D}_e\mathbf{h} + \mathbf{D}_r\mathbf{r} + \mathbf{b}_c$, where $\mathbf{D}_e, \mathbf{D}_r \in \mathbb{R}^{k \times k}$ are diagonal matrices which are used as shared parameters among all entities and relations, and $\mathbf{b}_c \in \mathbb{R}^k$ represents the candidate bias vector shared across all entities. Next, the score for the triple $(h,r,t) \in \mathbb{K}$ is computed:

$$f(h,r,t) = g(\mathbf{t}\, z(\mathbf{h} \otimes \mathbf{r}) + \mathbf{b}_p) \ , \tag{17}$$

where g and z are activation functions, and $\mathbf{b}_p$ represents the shared projection bias vector.

HolE Holographic embeddings (HolE) [33] make use of the circular correlation operator to compute interactions between latent features of entities and relations:

$$f(h,r,t) = \sigma(\mathbf{r}^T(\mathbf{h} \star \mathbf{t})) \ . \tag{18}$$

where the circular correlation $\star : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is defined as $[\mathbf{a} \star \mathbf{b}]_i = \sum_{k=0}^{d-1} \mathbf{a}_k * \mathbf{b}_{(i+k) \ mod \ d}$. By using the correlation operator each component $[\mathbf{h} \star \mathbf{t}]_i$ represents a sum over a fixed partition over pairwise interactions. This enables the model to put semantic similar interactions into the same partition and share weights through $\mathbf{r}$. Similarly irrelevant interactions of features could also be placed into the same partition which could be assigned a small weight in $\mathbf{r}$.

ERMLP ERMLP [34] is a multi-layer perceptron based approach that uses a single hidden layer and represents entities and relations as vectors. In the input-layer, for each triple the embeddings of head, relation, and tail are concatenated and passed to the hidden layer. The output-layer consists of a single neuron that computes the plausibility score of the triple:

$$f(h,r,t) = \mathbf{w}^T g(\mathbf{W}[\mathbf{h};\mathbf{r};\mathbf{t}]), \tag{19}$$

where $\mathbf{W} \in \mathbb{R}^{k \times 3d}$ represents the weight matrix of the hidden layer, $\mathbf{w} \in \mathbb{R}^k$, the weights of the output layer, and g denotes an activation function such as the hyperbolic tangent.

Neural Tensor Network The Neural Tensor Network (NTN) [35] uses a bilinear tensor layer instead of a standard linear neural network layer:

$$f(h,r,t) = \mathbf{u}_r^T \cdot \tanh(\mathbf{h}\mathfrak{W}_r\mathbf{t} + \mathbf{V}_r[\mathbf{h};\mathbf{t}] + \mathbf{b}_r) \ , \tag{20}$$

where $\mathfrak{W}_r \in \mathbb{R}^{d \times d \times k}$ is the relation specific tensor, and the weight matrix $\mathbf{V}_r \in \mathbb{R}^{k \times 2d}$, the bias vector $\mathbf{b}_r$, and the weight vector $\mathbf{u}_r \in \mathbb{R}^k$ are the standard parameters of a neural network, which are also relation specific. The result of the tensor product $\mathbf{h}\mathfrak{W}_r\mathbf{t}$ is a vector $\mathbf{x} \in \mathbb{R}^k$ where each entry x_i is computed based on the slice i of the tensor $\mathfrak{W}_r$: $\mathbf{x}_i = \mathbf{h}\mathfrak{W}_r^i\mathbf{t}$ [35]. As indicated by the interaction model, NTN defines for each relation a separate neural network which makes the model very expressive, but at the same time computationally expensive.

ConvKB ConvKB [36] uses a convolutional neural network (CNN) whose feature maps capture global interactions of the input. Each triple $(h,r,t) \in \mathbb{K}$ is represented as a input matrix $\mathbf{A} = [\mathbf{h};\mathbf{r};\mathbf{t}] \in \mathbb{R}^{d \times 3}$ in which the columns represent the embeddings for h, r and t. In the convolution layer, a set of convolutional filters $\boldsymbol{\omega}_i \in \mathbb{R}^{1 \times 3}, i = 1, \ldots, \tau$, are applied on the input in order to compute for each dimension global interactions of the embedded triple. Each $\boldsymbol{\omega}_i$ is applied on every row of $\mathbf{A}$ creating a feature map $\mathbf{v}_i = [v_{i,1}, ..., v_{i,d}] \in \mathbb{R}^d$:

$$\mathbf{v}_i = g(\boldsymbol{\omega}_j\mathbf{A} + \mathbf{b}) \ , \tag{21}$$

where $\mathbf{b} \in \mathbb{R}$ denotes a bias term and g an activation function which is employed element-wise. Based on the resulting feature maps $\mathbf{v}_1, \ldots, \mathbf{v}_\tau$, the plausibility score of a triple is given by:

$$f(h,r,t) = [\mathbf{v}_i; \ldots; \mathbf{v}_\tau] \cdot \mathbf{w} \ , \tag{22}$$

where $[\mathbf{v}_i; \ldots; \mathbf{v}_\tau] \in \mathbb{R}^{\tau d \times 1}$ and $\mathbf{w} \in \mathbb{R}^{\tau d \times 1}$ is a shared weight vector. ConvKB may be seen as a restriction of ER-MLP with a certain weight sharing pattern in the first layer.

ConvE ConvE [37] is a CNN-based approach. For each triple (h,r,t), the input to ConvE is a matrix $\mathbf{A} \in \mathbb{R}^{2 \times d}$ where the first row of $\mathbf{A}$ represents $\mathbf{h} \in \mathbb{R}^d$ and the second row represents $\mathbf{r} \in \mathbb{R}^d$. $\mathbf{A}$ is reshaped to a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ where the first $m/2$ half rows represent $\mathbf{h}$ and the remaining $m/2$ half rows represent $\mathbf{r}$. In the convolution layer, a set of *2-dimensional* convolutional filters $\Omega = \{\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_i \in \mathbb{R}^{r \times c}\}$ are applied on $\mathbf{B}$ that capture interactions between $\mathbf{h}$ and $\mathbf{r}$. The resulting feature maps are reshaped and concatenated in order to create a feature vector $\mathbf{v} \in \mathbb{R}^{|\Omega| rc}$. In the next step, $\mathbf{v}$ is mapped into the entity space using a linear transformation $\mathbf{W} \in \mathbb{R}^{|\Omega| rc \times d}$, that is $\mathbf{e}_{h,r} = \mathbf{v}^T\mathbf{W}$. The score for the triple $(h,r,t) \in \mathbb{K}$ is then given by:

$$f(h,r,t) = \mathbf{e}_{h,r}\mathbf{t} \ . \tag{23}$$

Since the interaction model can be decomposed into $f(h,r,t) = \langle f'(\mathbf{h},\mathbf{r}), \mathbf{t} \rangle$, the model is particularly designed to 1-N scoring, i.e. efficient computation of scores for (h,r,t) for fixed h, r and many different t.

3.2 Training Approaches

Because most KGs contain only positive examples, we require training approaches involving techniques such as negative sampling to avoid over-generalization to true facts. Here, we describe two common training approaches found in the literature: the local closed world assumption (LCWA) and the stochastic local closed world assumption (sLCWA). It should be noted that the LCWA and the sLCWA do not affect the evaluation.

3.2.1 *Local closed world assumption*

The LCWA was introduced by [34] and used in subsequent works as an approach to generate negative examples during training [37], [30]. In this setting, for any triple $(h,r,t) \in \mathcal{K}$ that has been observed, a set $\mathcal{T}^-(h,r)$ of negative examples is created by considering all triples $(h,r,t_i) \notin \mathcal{K}$ as false. Therefore, for our exemplary KG (Figure 1) for the

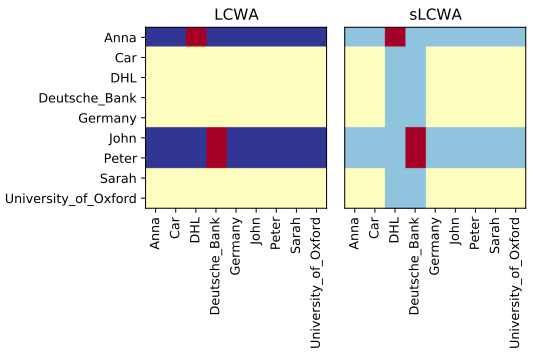


Fig. 3. Visualization of different training approaches for the relation `works_at` in the KG in Figure 1. Red color indicates positive examples, i.e. true triples present in the KG. Dark blue color denotes triples used as negative examples in LCWA. Light blue color sampling candidates for negative examples in sLCWA. Yellow color indicates triples that are not considered.

pair *(Peter, works_at)*, the triple *(Peter, works_at, DHL)* is a false fact since for this pair only the triple *(Peter, works_at, Deutsche Bank)* is part of the KG. Similarly, we can construct $\mathcal{H}^-(r,t)$ based on all triples $(h_i,r,t) \notin \mathcal{K}$, or $\mathcal{R}^-(h,t)$ based on the triples $(h,r_i,t) \notin \mathcal{K}$. Constructing $\mathcal{R}^-(h,t)$ is a popular choice in visual relation detection domain [38], [39]. However, most of the works in knowledge graph modeling construct only $\mathcal{T}^-(h,r)$ as the set of negative examples, and in the context of this work refer to $\mathcal{T}^-(h,r)$ as the set of negatives examples when speaking about LCWA.

3.2.2 Stochastic local closed world assumption

Under the stochastic local closed world assumption (sLCWA), instead of considering all possible triples $(h,r,t_i) \notin \mathcal{K}$, $(h_i,r,t) \notin \mathcal{K}$ or $(h,r_i,t) \notin \mathcal{K}$ as false, we randomly take samples of these sets.

Two common approaches for generating negative samples are uniform negative sampling (UNS) [19] and Bernoulli negative sampling (BNS) [20] in which negative triples are created by corrupting a positive triple $(h,r,t) \in \mathcal{K}$ by replacing either h or t. We denote with $\mathcal{N}$ the set of all potential negative triples:

$$\mathcal{T}(h,r) = \{(h,r,t') \mid t' \in \mathcal{E} \wedge t' \neq t\} \quad (24)$$

$$\mathcal{H}(r,t) = \{(h',r,t) \mid h' \in \mathcal{E} \wedge h' \neq h\} \quad (25)$$

$$\mathcal{N} = \bigcup_{(h,r,t)\in\mathcal{K}} \mathcal{T}(h,r) \cup \mathcal{H}(r,t) \ . \quad (26)$$

Theoretically, we would need to exclude all positive triples from this set of candidates for negative triples, i.e., $\mathcal{N}^- = \mathcal{N} \setminus \mathcal{K}$. In practice, however, since usually $|\mathcal{N}| \gg |\mathcal{K}|$, the likelihood of generating a false negative is rather low. Therefore, the additional filter step is often omitted to lower computational cost. It should be taken into account that a corrupted triple that is *not part* of the KG can represent a true fact.

UNS and BNS differ in the way they define sample weights for (h',r,t) or (h,r,t'):

Uniform negative sampling With uniform negative sampling (UNS) [19], the first step is to randomly (uniformly) determine whether h or t shall be corrupted for a positive triple $(h,r,t) \in \mathcal{K}$. Afterwards, an entity $e \in \mathcal{E}$ is uniformly sampled and selected as the corrupted head/tail entity.

Bernoulli negative sampling With Bernoulli negative sampling (BNS) [20], the probability of corrupting h or t in $(h,r,t) \in \mathcal{K}$ is determined by the property of the relation r: if the relation is a *one-to-many* relation (e.g. *motherOf*), BNS assigns a higher probability to replace h, and if it is a *many-to-one* relation (e.g. *bornIn*) it assigns a higher probability to replace t. More precisely, for each relation $r \in \mathcal{R}$ the average number of tails per head (*tph*) and heads per tail (*hpt*) are first computed. These statistics are then used to define a Bernoulli distribution with parameter $\frac{tph}{tph+hpt}$. For a triple $(h,r,t) \in \mathcal{K}$ the head is corrupted with probability $\frac{tph}{tph+hpt}$ and the tail with probability $\frac{hpt}{tph+hpt}$. The described approach reduces the chance of creating corrupted triples that represent true facts [20].

3.3 Loss Functions

The loss function can have a significant influence on the performance of KGEMs [7]. In the following, we describe *pointwise*, *pairwise*, and *setwise* loss functions that have been frequently be used within KGEMs. For additional discussion and a slightly different categorization we refer to the work of Mohamed *et al.* [7].

3.3.1 Pointwise Loss Functions

Let f denote the interaction model of a KGEM. With t_i, we denote a triple (i.e. $t_i \in \mathbb{K}$), and with $l_i \in \{0,1\}$ or $\hat{l}_i \in \{-1,1\}$ its corresponding label, where 1 corresponds to the label of the positive triples, and 0 / -1 to the label of the negative triples. Pointwise loss functions compute an independent loss term for each triple-label pair, i.e. for a batch $B = \{(t_i,l_i)\}_{i=1}^{|B|}$, the loss is given as

$$\mathcal{L} = \frac{1}{|B|} \sum_{(t_i,l_i)\in B} L(t_i,l_i) \quad (27)$$

In the following, we describe four different pointwise losses: The *square error loss*, *binary cross entropy loss (BCEL)*, *pointwise hinge loss*, and *logistic loss*.

Square Error Loss The square error loss function computes the squared difference between the predicted scores and the labels $l_i \in \{0,1\}$ [7]:

$$L(t_i,l_i) = \frac{1}{2}(f(t_i) - l_i)^2 \quad (28)$$

The squared error loss strongly penalizes predictions that deviate considerably from the labels, and is usually used for regression problems. For simple models it often permits more efficient optimization algorithms involving analytical solutions of sub-problems, e.g. the Alternating Least Squares algorithm used by [26].

Binary cross entropy loss The binary cross entropy loss is defined as [37]:

$$\begin{aligned} L(t_i,l_i) = &- (l_i \cdot \log(\sigma(f(t_i))) \\ &+ (1-l_i) \cdot \log(1-\sigma(f(t_i)))), \end{aligned} \quad (29)$$

where $l_i \in \{0, 1\}$ and σ represents the logistic sigmoid function. Thus, the problem is framed as a binary classification problem of triples, where the model's outputs are regarded as logits. The loss is not well-suited for translational distance models because these models produce a negative distance as score and cannot produce positive model outputs. ConvE and TuckER were originally trained in a multi-class setting using the binary cross entropy loss where each (h, r)-pair has been classified against $e \in \mathcal{E}$ simultaneously, i.e., if $|\mathcal{E}| = n$, the label vector for each (h, r)-pair has n entries indicating whether the triple (h, r, e_i) is (not) part of the KG, and along each dimension of the label vector a binary classification is performed. It should be noted that there exist different implementation variants of the binary cross entropy loss that address numerical stability. ConvE and TuckER employed a numerically unstable variant, and in the context of this work, we refer to this variant when referring to the binary cross entropy loss.

Pointwise Logistic Loss/Softplus loss An alternative, but equivalent formulation of the binary cross entropy loss is the pointwise logistic loss (or Softplus loss (SPL)):

$$L(t_i, l_i) = \log(1 + \exp(-\hat{l}_i \cdot f(t_i)) \tag{30}$$

where $\hat{l}_i \in \{-1, 1\}$ [7]. It has been used to train ComplEx, ConvKB, and SimplE. We consider both variants separately because both have been used in different model implementations, and their implementation details might yield different results (e.g., to numerical stability).

Pointwise Hinge Loss The pointwise hinge loss sets the score of positive examples larger than a margin parameter λ while reducing the scores of negative examples to values below $-\lambda$:

$$L(t_i, l_i) = \max(0, \lambda - \hat{l}_i \cdot f(t_i)) \tag{31}$$

where $\hat{l}_i \in \{-1, 1\}$. The loss penalizes scores of positive examples which are smaller than λ, but does not impose any restriction on values $> \lambda$. Similarly, negative scores larger than $-\lambda$ contribute to the loss, whereas all values smaller than $-\lambda$ do not have any loss contribution [7]. Thereby, the model is not encouraged to further optimize triples which are already predicted well enough (according to the margin parameter λ).

3.3.2 *Pairwise Loss Functions*

Next, we describe widely applied pairwise loss functions that are used within KGEMs, namely the *pairwise hinge loss* and the *pairwise logistic loss*. They both compare the scores of a positive triple t^+ and a negative triple t^-. The negative triple in a pair is usually obtained by corrupting the positive one. Thus, the pairs often share common head or tail entities and relations. For a batch of pairs $B = \{(t_i^+, t_i^-)\}_{i=1}^{|B|}$, the loss is given as

$$\mathcal{L} = \frac{1}{|B|} \sum_{(t_i^+, t_i^-) \in B} L(f(t_i^-) - f(t_i^+)) \ . \tag{32}$$

Hence, the loss function evaluates the difference in scores $\Delta = f(t_i^-) - f(t_i^+)$ between a positive and a negative triple, rather than their absolute scores. This is in accordance to the OWA assumption, where we do not assume to have negative labels, but just "less positive" ones.

Pairwise Hinge Loss/Margin ranking loss The pairwise hinge loss or margin ranking loss (MRL) is given by

$$L(\Delta) = \max(0, \lambda + \Delta) \ . \tag{33}$$

Pairwise Logistic Loss The pairwise logistic loss is defined as [7]:

$$L(\Delta) = \log(1 + \exp(\Delta)) \ . \tag{34}$$

Thus, it can be seen as a soft-margin formulation of the pairwise hinge loss with a margin of zero.

3.3.3 *Setwise Loss Functions*

Setwise loss functions neither compare individual scores, or pairs of them, but rather more than two triples' scores. Here, we describe the self-adversarial negative sampling loss (NSSAL) and the cross entropy loss (CEL) as examples of such loss functions that have been applied within KGEMs [23], [7].

Self-adversarial negative sampling loss The Self-adversarial negative sampling loss (NSSAL) addresses the limitation that many negative examples are trivial and do not provide helpful information. The authors of [23] propose to overcome this limitation by sampling negative samples according to the scores predicted by the interaction model [23]:

$$p((h_i', r, t_i')|(h_i, r_i, t_i)) = \frac{\exp(\alpha f(h_i', r, t_i'))}{\sum_{j=1}^{n} \exp(\alpha f(h_j', r, t_j'))} \ , \tag{35}$$

where $(h_i, r_i, t_i) \in \mathcal{K}$ denotes a true triple, $\{(h_i', r, t_i')\}_{i=1}^{K}$ it's set of negative samples generated, and $\alpha \in \mathbb{R}$ a temperature parameter. Because sampling from this distribution may be computationally expensive, the probabilities obtained by Equation 35 are used to weight the generated negative examples in the loss function [23].

$$\begin{aligned} \mathcal{L} = &- \log(\sigma(\gamma + f(h, r, t))) \\ &- \sum_{i=1}^{K} p((h', r, t')) \cdot \log(\sigma(-(\gamma + f(h_i', r, t_i')))) \ . \end{aligned} \tag{36}$$

Thus, negative samples for which the model predicts a high score relative to other samples are weighted stronger.

Cross entropy loss The cross entropy loss (CEL) has been successfully applied together with 1-N scoring, i.e., predicting for each (h, r)-pair simultaneously a score for each possible tail entity, and framing the problem as a multi-class classification problem [3], [8]. To apply the CEL, first, the labels are normalized in order to form a proper probability distribution. Second, the predicted scores for the tail entities of (h, r)-pair are normalized by a softmax:

$$p(t \mid h, r) = \frac{\exp(f(h, r, t))}{\sum_{t' \in \mathcal{E}} \exp(f(h, r, t'))} \ . \tag{37}$$

Finally, the cross entropy between the distribution of the normalized scores and the normalized label distribution is computed:

$$\mathcal{L} = -\sum_{t' \in \mathcal{E}} \mathbb{I}[(h, r, t') \in \mathcal{K}] \cdot \log(p(t \mid h, r)) \ , \tag{38}$$

where $\mathbb{I}$ denotes the indicator function. Note that this loss differs from the multi-class binary cross entropy as it applies a softmax normalization implying that this is a *single-label* multi-class problem.

3.4 Explicitly Modeling Inverse Relations

Inverse relations introduced by [29] and [40] are explicitly modeled by extending the set of relations $\mathcal{R}$ by a set of inverse relations $r_{inv} \in \mathcal{R}_{inv}$ with $\mathcal{R}_{inv} \cap \mathcal{R} = \emptyset$. This is achieved by training an inverse triple (t, r_{inv}, h) for each triple $(h, r, t) \in \mathcal{K}$. Equipping a KGEM with inverse relations implicitly doubles the relation embedding space of any model that has relation embeddings. The goal is to alter the scoring function, such that the task of predicting the head entities for (r, t) pairs becomes the task of predicting tail entities for (t, r_{inv}) pairs. The explicit training of the implicitly known inverse relations can lead to better model performance [40] and can for some models increase the computational efficiency [37].

4 Evaluation Metrics for KGEMs

KGEMs are usually evaluated based on link prediction, which is on KG defined as predicting the tail/head entities for $(h, r)/(r, t)$ pairs. For instance, given queries of the form *(Sarah, studied_at, ?)* or *(?, CEO_of, Deutsche Bank)* the capability of a link predictor to predict the correct entities that answer the query, i.e. *(Sarah, studied_at,* ***University of Oxford****)* and *(****Sarah****, CEO_of, Deutsche Bank)* is measured.

However, given the fact that usually true negative examples are not available, both the training and the test set contain only true facts. For this reason, the evaluation procedure is defined as a ranking task in which the capability of the model to differentiate corrupted triples from known true triples is assessed [19]. For each test triple $t^+ = (h, r, t) \in \mathcal{K}_{test}$ two sets of corrupted triples are constructed:

1) $\mathcal{H}(r, t) = \{(h', r, t) \mid h' \in \mathcal{E} - \{h\}$ which contains all the triples where the head entity has been corrupted, and
2) $\mathcal{T}(h, r) = \{(h, r, t') \mid t' \in \mathcal{E} - \{t\}\}$ that contains all the triples with corrupted tail entity.

For each t^+ and its corresponding corrupted triples, the scores are computed and the entities sorted accordingly. Next, the rank of every t^+ among its corrupted triples is determined, i.e. the position in the score-sorted list.

Among the corrupted triples in $\mathcal{H}(r, t)$ / $\mathcal{T}(h, r)$, there might be true triples that are part of the KG. If these false negatives are ranked higher than the current test triple t^+, the results might get distorted. Therefore, the *filtered* evaluation setting has been proposed [19], in which the corrupted triples are filtered to exclude known true facts from the train and test set. Thus, the rank does not decrease when ranking another true entity higher.

Moreover, we want to draw attention to the fact that the metrics can be further be distorted by *unknown false negatives*, i.e., true triples that are contained in the set of corrupted triples but are not part of the KG (and therefore cannot be filtered out). Therefore, it is essential to investigate the predicted scores of a KGEM and not solely rely on the computed metrics.

Based upon these individual ranks, the following measures are frequently used to summarize the overall performance:

Mean rank The mean rank (MR) represents the average rank of the test triples, i.e.

$$\text{MR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} rank(t) \tag{39}$$

Smaller values indicate better performance.

Adjusted mean rank Because the interpretation of the MR depends on the number of available candidate triples, comparing MRs across different datasets (or inclusion of inverse triples) is difficult. This is sometimes further exacerbated in the filtered setting because the number of candidates varies. Therefore, with fewer candidates available, it becomes easier to achieve low ranks. The adjusted mean rank (AMR) [10] compensates for this problem by comparing the mean rank against the expected mean rank under a model with random scores:

$$\text{AMR} = \frac{MR}{\frac{1}{2} \sum\limits_{t \in \mathcal{K}_{test}} (\xi(t) + 1)} \tag{40}$$

where $\xi(t)$ denotes the number of candidate triples against which the true triple $t \in \mathcal{K}_{test}$ is ranked. In the unfiltered setting we have $\xi(t) = |\mathcal{E}| - 1$ for all $t \in \mathcal{K}_{test}$. Thereby, the measure also adjusts for chance, as a random scoring achieves an expected adjusted mean rank of 1. The AMR has a fixed value range from 0 to 1, where smaller values (AMR $\ll 1$) indicate better performance.

Mean reciprocal rank The mean reciprocal rank (MRR) is defined as:

$$\text{MRR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} \frac{1}{rank(t)} \tag{41}$$

where $\mathcal{K}_{test}$ is a set of test triples, i.e. the MRR is the mean over reciprocal individual ranks. However, the MRR is flawed since the reciprocal rank is an ordinal scale and not an interval scale, i.e. computing the arithmetic mean is statistically incorrect [41], [42]. Still, it is often used for early stopping since it is a smooth measure with stronger weight on small ranks, and less affected by outlier individual ranks than the mean rank. The MRR has a fixed value range from 0 to 1, where larger values indicate better performance.

Hits@K Hits@K denotes the ratio of the test triples that have been ranked among the top *k* triples, i.e.,

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid rank(t) \leq k\}|}{|\mathcal{K}_{test}|} \tag{42}$$

Larger values indicate better performance.

Additional Metrics Further metrics that might be relevant are the area under the Receiver Operating Characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR) [11]. However, these metrics require the number of true positives, false positives, true negatives, and false negatives, which in most cases cannot be computed since the KGs are usually incomplete.

5 Existing Benchmark Datasets

In this section, we describe the benchmark datasets that have been established to evaluate KGEMs. A summary is also given in Table 1.

FB15K Freebase is a large cross-domain KG consisting of around 1.2 billion triples and more than 80 million entities. Bordes *et al.* [19] extracted a subset of Freebase, which is used as a benchmark dataset and named it FB15K. It contains 14,951 entities, 1,345 relations, as well as more than half a million triples describing facts about movies, actors, awards, sports, and sports teams [37].

FB15K-237 FB15K has a test-leakage, i.e. a major part of the test triples (∼81%) are inverses of triples contained in the training set: for most of the test triples of the form (h, r, t), there exists a triple (h, r', t) or (t, r', h) in the training set. Therefore, Toutanova and Chen [43] constructed FB15K-237 in which inverse relations were removed [43]. FB15K-237 contains 14,541 entities and 237 relations.

WN18 WordNet[1] is a lexical knowledge base in which entities represent terms and are called *synsets*. Relations in WordNet represent conceptual-semantic and lexical relationships (e.g. hyponym). Bordes *et al.* [17] extracted a subset of WordNet named WN18 that is frequently used to evaluate KGEMs. It contains 40,943 synsets and 18 relations.

WN18RR Similarly to FB15K, WN18 also has a test-leakage (of approximately 94%) [43]. For instance, for most of the test triples of the form *(h, hyponym, t)*, there exists a triple *(t, hypernym, o)* in the training set. Dettmers *et al.* [37] have shown that a simple rule-based system can obtain results competitive to the state of the art results on WN18. For this reason, they constructed WN18RR by removing inverse relations similarly to the procedure applied to FB15K. WN18RR contains 40,943 entities and 11 relations.

Kinships The Kinships [44] dataset describes relationships between members of the Australian tribe *Alyawarra* and consists of 10,686 triples. It contains 104 entities representing members of the tribe and 26 relationship types that represent kinship terms such as *Adiadya* or *Umbaidya* [17].

Nations The Nations [45] dataset contains data about countries and their relationships with other countries. Exemplary relations are *economic_aid* and *accusation* [17].

Unified Medical Language System [46] The Unified Medical Language System (UMLS) [46] is an ontology that describes relationships between high-level concepts in the biomedical domain. Examples of contained concepts are *Cell*, *Tissue*, and *Disease*, and exemplary relations are *part_of* and *exhibits* [17], [46].

YAGO3-10 Yet Another Great Ontology (YAGO) [47] is a KG containing facts that have been extracted from Wikipedia and aligned with WordNet in order to exploit the large amount of information contained in Wikipedia and the taxonomic information included in WordNet. It contains general facts about public figures, geographical entities, movies, and further entities, and it has a taxonomy for those concepts. YAGO3-10 is a subset of YAGO3 [48] (which is an extension of YAGO) that contains entities associated with at least ten different relations. In total, YAGO3-10 has 123,182 entities and 37 relations, and most of the triples describe attributes of persons such as citizenship, gender, and profession [37].

1. https://wordnet.princeton.edu/

TABLE 1
Existing Benchmark Datasets.

Dataset	Triples	Entities	Relations
FB15K	592,213	14.951	1,345
FB15K-237	272,115	14,541	237
WN18	151,442	40,943	18
WN18RR	93,003	40,943	11
Kinships	10,686	104	26
Nations	11,191	14	56
UMLS	893,025	135	49
YAGO3-10	1,079,40	132,182	37

6 Reproducibility Studies

The goal of the reproducibility studies was to investigate whether it is possible to replicate experiments based on the information provided in each model's accompanying paper. If specific information was missing, such as the number of training epochs, we tried to find this information in the accompanying source code if it was accessible. For our study, we focused on the two most frequently used benchmark datasets, FB15K and WN18, as well as their respective subsets FB15K-237 and WN18RR. Table 3 (Appendix A1) illustrates for which models results were reported (in the accompanying publications) for the considered datasets. A checkmark denotes that results were reported, and green background indicates that the entire experimental setup for the corresponding dataset was described. Results have not been reported for every model for every dataset because some of the benchmark datasets were created after the models were published. Therefore, these models have been excluded from our reproducibility study.

Experimental Setup For each KGEM, we applied identical training and evaluation settings as described in their concomitant papers. We ran each experiment four times with random seeds to measure the variance in the obtained results. We evaluated the models based on the ranking metrics MR, AMR, MRR, and Hits@K. As discussed in [4], [10], the exact computation of ranks differs across different codebases, and can lead to significant differences [4]. We follow the nomenclature of Berrendorf *et al.* [10], and report scores based on the optimistic, pessimistic, and realistic rank definitions.

Tables 8-11 (Appendix A5-A6) represent the results for FB15K, FB15K-237, WN18, and WN18RR where experiments highlighted in black were reproducible, in blue soft-reproducible experiments (i.e., could be reproduced by a margin $\leq$ 5%), and experiments highlighted in orange could not be reproduced. In the following, we discuss the observations that we made during our experiments.

6.1 Reproductions Requiring Alternate Hyper-Parameters

One of the observations we made is that for some experiments, results could only be reproduced with a different set of hyper-parameter values. For instance, the results for TransE could only be reproduced by adapting the batch

size and the number of training epochs. We trained TransE on WN18 for 4000 epochs compared to a reported number of 1000 epochs in order to obtain comparable results. Furthermore, for RotatE on FB15K and WN18, we received better results when adapting the learning rate. The reason for these differences might be explained by the implementation details of the underlying frameworks which have been used to train the models. Authors of early KGEMs often implemented their training algorithms themselves or used frameworks that were popular at the respective time but are not used anymore. Therefore, differences between the former and current frameworks may require an adaption of the hyper-parameter values. Even within the same framework, bug fixes or optimizations of the framework can lead to different results based on the used version. Our benchmarking study highlights that with adapted settings, results can be reproduced and even improved.

6.2 Unreported Hyper-parameters Impedes Reproduction

Some experiments did not report the full experimental setup impeding the reproduction of results. For example, the embeddings in the ConvKB experiments have been pre-trained based on TransE. However, the batch size for training TransE has not been reported, which can significantly affect the results, as previously discussed. Furthermore, we obtained a high deviation for the reported results for HolE on FB15K. The apparent reason is that we could not find the hyper-parameter setting for FB15K, such that we used the same setting as for WN18, which we found in the accompanying implementation.

6.3 Two Perspectives: Publication versus Implementation

While preparing our experiments, we observed that for some experiments, essential aspects, which are part of the released source code, have not been discussed in the paper. For instance, in the publication describing ConvE, it is not mentioned that inverse triples have been added to the KGs in a pre-processing step. This step seems to be essential to reproduce the results. A second example is SimplE, for which the predicted scores have been clamped to the range of $[-20, 20]$. This step was not mentioned in the publication, but it can have a significant effect when the model is evaluated based on an optimistic ranking approach, which is the case for SimplE.

6.4 Lack of Official Implementations Impedes Reproduction

During our experiments, we observed that for DistMult and TransD, we were able to reproduce the results on WN18, but not on FB15K. A reason might be differences in the implementation details of the frameworks used to train and evaluate the models. For example, the initialization of the embeddings or the normalization of the loss values could have an impact on the performance. Since there exists no official implementation (see Table 3 in Appendix A1) for DistMult and TransD, it is not possible to check the above-mentioned aspects. Furthermore, we were not able to reproduce the results for TransH for which also no official implementation is available. There exist reference implementations[2], which slightly differ from the model initially proposed.

6.5 Reproducibility is Dependent on The Ranking Approach

As discussed in [4], [10], the ranking metrics have been implemented differently by various authors. In our experiments, we report results based on three common implementations of the ranking metrics: i.) realistic, ii.) optimistic and iii.) pessimistic ranking (Section 4). If a model predicts the same score for many triples, there will be a large discrepancy between the three ranking approaches. We could observe such a discrepancy for SimplE for which the results on FB15K (Table 8 in Appendix A5) and WN18 (Table 10 in Appendix A6) were almost 0% based on the realistic ranking approach, but were much higher based on the optimistic ranking approach. Similar observations for other KGEM have been made in [4].

7 Benchmarking

In our benchmarking studies, we evaluated a large set of different combinations of interaction models, training approaches, loss functions, and the effect of explicitly modeling inverse relations. Additionally, we evaluated how well the interaction models can model symmetry, anti-symmetry and composition patterns (Appendix 8). In particular, we investigated 21 interaction models, two training approaches, and five loss functions on four datasets. We refer to a specific combination of interaction model, training approach, loss function, and whether inverse relations are explicitly modeled as a *configuration*, e.g., RotatE + LCWA + SPL + inverse relations. We do *not* refer to different hyper-parameter values such as batch size or learning rate when we use the term configuration. For each configuration, we used random search to perform the hyper-parameter optimizations over all other hyper-parameters and applied early stopping on the validation set. Each hyper-parameter optimization experiment lasted for a maximum of 24 hours or 100 iterations, in which new hyper-parameters have been sampled in each iteration. Overall, we performed individual hyper-parameter optimizations for more than 1,000 configurations. We retrain the model with the best hyper-parameter setting and report evaluation results on the test set.

Before presenting our results, we provide an overview of the experimental setup, comprising the investigated interaction models, training approaches, loss functions, negative samplers, and datasets. We used the sLCWA and LCWA as training approaches. For the sLCWA we applied a *1:k*-Scoring as usually done throughout the literature [19], [27], where k denotes the number of negative examples for each positive. For the LCWA, we applied a *1:N*-Scoring, i.e., we sample each batch against all negatives examples as typically done for training with the LCWA [37]. Table 4 (Appendix A1) shows the hyper-parameter ranges for the sLCWA and the LCWA assumptions.

2. https://github.com/thunlp/OpenKE

Datasets We performed experiments on the following four datasets: WN18RR, FB15K-237, Kinships and YAGO3-10. We selected WN18RR and FB15K-237 since they are widely applied benchmarking datasets. We chose Kinships and YAGO3-10 to investigate the performance of KGEMs on a small and a larger dataset.

Interaction Models We investigated all interaction models described in Section 3.1. Because of our vast experimental setup and the size of YAGO3-10, we restricted the number of interaction models on YAGO3-10 as otherwise, the computational effort would be prohibitive. Based on their variety of model types as described in Section 3.1, we selected the following interaction models: ComplEx, ConvKB, DistMult, ERMLP, HolE, MuRE, QuatE, RESCAL, RotatE, SE, TransD, and TransE.

Training Approaches We trained the interaction models based on the sLCWA (Section 3.2.2) and the LCWA (Section 3.2.1) training approaches. Due of the extent of our benchmarking study and the fact that YAGO3-10 contains more than 132,000 entities, which makes the training based on the LCWA with 1-n scoring expensive, we restricted the training approach to the sLCWA for YAGO3-10.

Loss Functions We investigated MRL, BCEL, SPL, NSSAL, and CEL since they represent the variety of types described in Section 3.3 and because they have been previously shown to yield good results. MRL has not been historically used in the 1-N scoring setting likely due to the fact that in 1-N scoring, the number of positive and negative scores in each batch is not known in advance and dynamic. Thus, the number of possible pairs varies as well ranging from $N-1$ to $(N/2)^2$ for each (h,r) combination. The accompanying variance in memory requirements for each batch thus poses practical challenges. Therefore, we did not use the MRL in combination with the 1-N scoring setting.

Negative Sampler When using the sLCWA, we generated negative samples with UNS. When training with the LCWA and 1-N scoring, no explicit negative sampling was required.

Early Stopping We evaluated each model every 50 epochs and performed early stopping with a patience of 100 epochs on all datasets except for YAGO3-10. There, considering the larger number of triples seen in each epoch we evaluated each model every 10 epochs and performed early stopping with a patience of 50 epochs.

Below, we describe the results of our benchmarking study. In the four following subsections, we summarize the results for each dataset (i.e., Kinships, WN18RR, FB15K-237, YAGO3-10) along with a discussion of the effect of the models' individual components (i.e., training approaches, loss functions, the explicit modeling of inverse relations) and optimizers on the performance. Finally, we compare the model complexity versus performance. In the appendix, we provide further results. In particular, we provide for each model the results of all tested combinations of interaction model, training approach, and loss function.

7.1 Results on the Kinships Dataset

Investigating the model performances on Kinhsips is interesting because it is a comparatively small KG and thus permits for each configuration a large number of HPO iterations for all interaction models. Figure 4 provides a general overview of the results, i.e., performance of the interaction models, loss functions, training approach, the effect of modeling inverse relations, and the effect of the optimizers. Overall, it can be observed that for most interaction models, several well-performing configurations can be determined. However, some interaction models heavily depend on specific configurations such as KG2E and QuatE. Although link prediction on Kinships seems to be relatively easy, there are several translational distance-based interaction models that perform relatively poor (i.e., TransD, TransE, TransH, TransR, and UM). The poor performance of UM is not surprising considering that it omits the multi-relational information of the data. Finally, the results illustrate that Adam outperforms Adadelta (in many cases with high margin). Therefore, we decided to progress only with Adam as optimizer for the remaining datasets in order to reduce the computational costs.

Impact of the Training approach Figure 5 and Figure 6 depict the effect of the training approaches. The former summarizes the results over the interaction models, and the latter differentiates between them. At this point, we focus only on the BCEL and the SPL (which is equivalent to BCEL, but numerical more stable, see Section 3.3.1) since they have been trained with both training approaches. It can be observed that the training approach has an effect on the performance. There are more sLCWA-well performing configurations than LCWA ones.

Impact of the Loss Function Figure 5 highlights that selecting the appropriate loss function is crucial also for relatively small dataset such Kinships. Although all five loss functions achieve high performance, all except the MRL exhibit high variance. Comparing an interaction model that has been trained with the MRL with an interaction model that has been trained with a different loss function can lead to misleading conclusions since finding a suitable configuration for the loss functions except for the MRL is more difficult.

Impact of Explicitly Modeling Inverse Relations Figures 7-9 present the effect of explicitly modeling inverse relations. It can be observed that in general, the LCWA benefits from explicit usage of inverse relations in terms of robustness. This is to be expected since in the LCWA, the model only learns to perform tail predictions, and without explicitly modeling inverse relations, the model might have difficulties in correctly predicting head entities. However, when explicitly modeling inverse relations, the head predictions are obtained by predicting the tail entities of the corresponding inverse triples (see Section 3.4)

Interestingly, MRL and NSSAL-based configurations, which are both only trained with the sLCWA (i.e., the model already learns to perform head and tail predictions) are more robust when trained with inverse relations. Therefore, depending on the dataset, it might be helpful to employ inverse relation for these loss functions even though they might be trained with sLCWA.

Model Complexity versus Performance Figure 28 (Appendix A11) plots the model size against the obtained performance. The results highlight that there is no strong correlation between model size and performance, i.e., models with a small number of parameters can perform equally

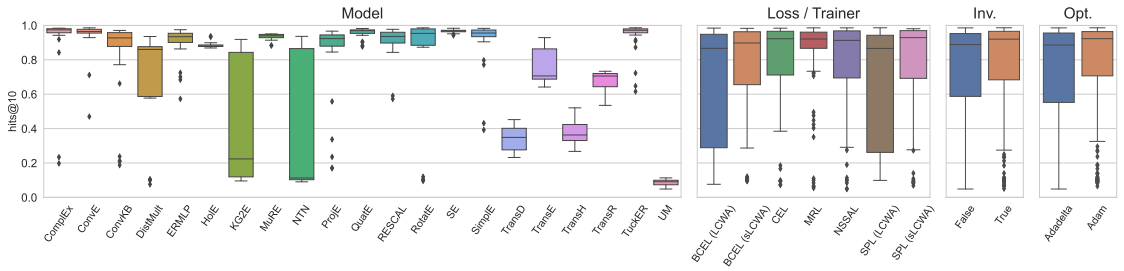


Fig. 4. Overall hits@10 results for Kinships where box-plots summarize the best results across different configurations, i.e., combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

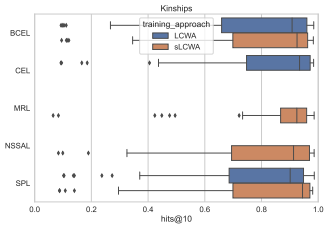


Fig. 5. Impact of the training approach on the performance for a fixed loss function for the Kinships dataset based on Adam.

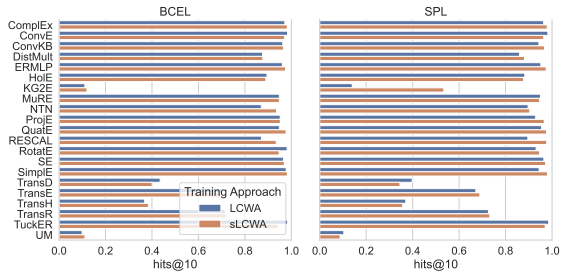


Fig. 6. Impact of training approach on the performance for a fixed interaction model and loss function for the Kinships dataset based on Adam.

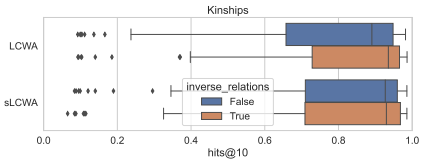


Fig. 7. Impact of explicitly modeling inverse relations on the performance for a fixed training approach for the Kinships dataset based on Adam.

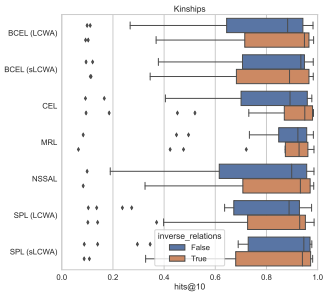


Fig. 8. Impact of explicitly modeling inverse relations on the loss function based on Adam for Kinships.

well as large models on the Kinships data set. The skyline comprises small UM models, some intermediate HolE and ProjE models, and larger RotatE and TuckER models. A full list is provided in Table 14 in Appendix A8.

7.2 Results on the WN18RR Dataset

Figure 10 depicts the overall results over WN18RR. A detailed overview of all configurations can be found in Figure 31 in Appendix A14. The results highlight that there are several combinations of interaction models, loss functions, and training approaches that obtain hits@10 results that are competitive with state-of-the-art results[3]. In particular, ComplEx (53.74%), ConvE (56.33% compared to 52.00% in the original paper [37]), DistMult (52.62%), MuRE (57.90% compared to 55.50% in the original paper [24]), KG2E (52.30%), ProjE (51,73%), TransE (56.98%), RESCAL (53.92%), RotatE (60.09% compared to 56.61% in the original paper [23]), SimplE (50.89%), and TuckER (56.09% compared to 52.6% in the original paper [30]) obtained high performance. Especially the result obtained by TransE is impressive since with a suitable configuration, it beats most of the published state-of-the-art results. The results highlight that determining an appropriate combination of interaction model, loss function, training approach, and the decision to explicitly modeling inverse relation is fundamental since many interaction models such as ConvE and KG2E reveal a high variance across different configurations. The results for ComplEx and RESCAL further underpin this observation. They reveal competitive results with very specialized configurations that represent outliers. Another interesting observation is the performance of UM, which does not model relations, but can still compete with some of the other

3. https://paperswithcode.com/sota/link-prediction-on-wn18rr

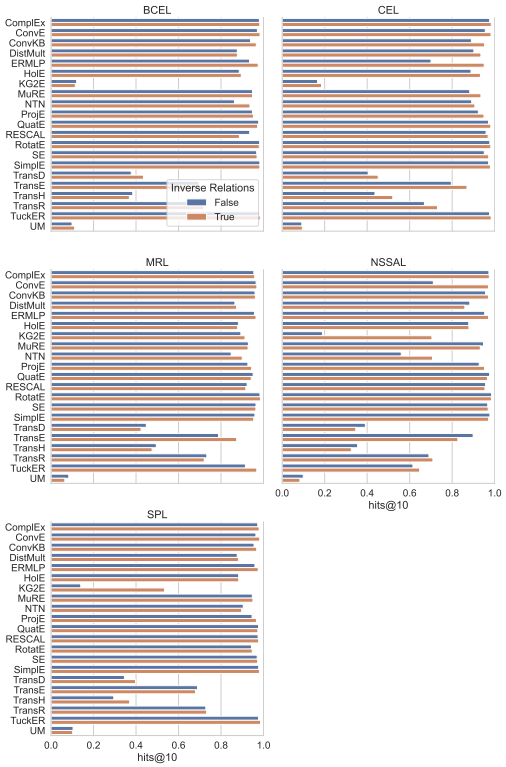


Fig. 9. Impact of explicitly modeling inverse relations on the performance for a fixed interaction model and loss function for the Kinships dataset based on Adam.

interaction models on WN18RR. This observation might indicate that the relational patterns in WN18RR are not too diverse across relations.

Impact of the Training Approach Figure 11 depicts the impact of the training approach for a fixed loss function. Again, we focus only on BCEL and SPL since they have been trained under both the sLCWA and LCWA. The figure highlights that for both realizations of the binary cross entropy loss, the LCWA achieves higher maximum performance, but at the same time, it reveals a larger variance on both loss functions. Consequently, it may be more difficult to find configurations that obtain high performance. The overall lower variance of SPL can be explained by the fact that it is numerically more stable than the BCEL.

Figure 12 shows the impact of the training approaches for fixed interaction models and used loss functions. The results indicate that for some combinations of interaction models and loss functions, the training approach's choice has a significant impact on the results. For instance, ConvE, RotatE, TransE and TuckER reveal stronger performance when trained with the LCWA whereas TransH suffer under the LCWA.

Impact of the Loss Function Figures 10 and 11 depict the performance of the different loss functions. State-of-the-art results for WN18RR are currently between 50% and 60%, and for each loss function, at least 50% could be achieved (Figure 31 in Appendix A14). However, the MRL is comparably less competitive than the other loss functions. This observation is especially important considering that early KGEMs have often been trained with the MRL. The results highlight that there is a trade-off between highest performance and robustness, i.e., SPL and BCEL achieve the highest performance (when trained under the LCWA), but also have high variance across different configurations (especially BCEL + LCWA).

Figure 35 (Appendix A18) reveals that some interaction models can obtain a further performance boost when configured with specific loss functions. For instance, the performance of ComplEx, ProjE and RESCAL can be increased by a significant margin when composed together with the CEL.

Impact of Explicitly Modeling Inverse Relations Figures 13 and 14 depict that overall, it is easier too find a strong performing sLCWA-configurations when trained without inverse relations. Surprising is that for LCWA based configurations, the interaction models are still competitive when trained without inverse relations. This observation is surprising because KGEMs that are configured with the LCWA and without inverse relations are not explicitly trained to predict the head entities of triples.

Figure 15 highlights that the choice of employing inverse relation can have an important impact on the model performance of an interaction model. For instance, TransE expresses quite different behaviors on the evaluated loss functions when inverse relations are explicitly modeled or not: on BCELs, CELs inverse relations improve the performance of TransE, whereas on SPLs the performance declines. On MRLs and NSSALs, the performance is very robust regardless of the choice of modeling inverse relations.

Model Complexity vs. Performance Figure 28 (Appendix A11) highlights that there is no significant correlation between model size and performance. Instead, the results show that with an appropriate configuration, the model complexity can be significantly reduced (Table 15 in Appendix A8). For instance, for RotatE, several high-performing configurations have been found (Figure 31 in the Appendix A14), and the second-best configuration achieved a hits@10 value of 58.33% while trained with an embedding dimension of *64* (in the complex space). This is especially interesting considering that RotatE originally obtained a performance of 57.1% hits@10 [23] with an embedding dimension of 500 (in the complex space) using the sLCWA as training approach and the NSSAL as loss function [4]. By changing the training approach and the loss function, the embedding dimension could be reduced significantly while getting at the same time an improvement in the hits@10 score.

7.3 Results on the FB15K-237 Dataset

Figure 16 provides an overall overview of the results obtained on FB15K-237. For the results for each individual configuration, we refer to Figure 32 in Appendix A15. We can observe that TuckER outperforms the other interaction

4. https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding

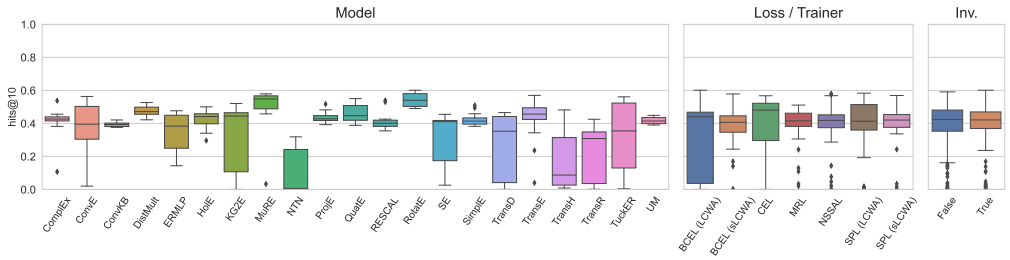


Fig. 10. Overall hits@10 results for WN18RR where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

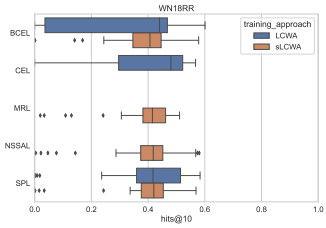


Fig. 11. Impact of the training approach on the performance for a fixed loss function for the WN18RR dataset.

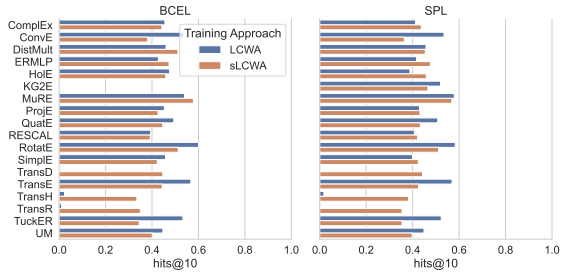


Fig. 12. Impact of training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset.

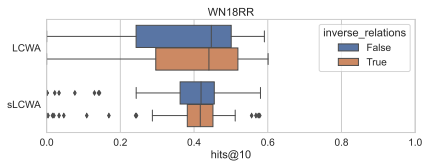


Fig. 13. Impact of explicitly modeling inverse relations on the performance for a fixed training approach for the WN18RR dataset.

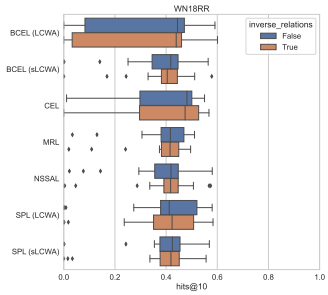


Fig. 14. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the WN18RR dataset.

models followed by RotatE. DistMult again obtains surprisingly good results (Table 32 in Appendix A15) considering that the interaction model enforces symmetric relations. The results illustrate again that choosing a suitable composition is essential for the performance of an interaction model. For instance, TuckER and QuatE perform well only with dedicated compositions. A further example is DistMult, which again obtains surprisingly good results (Table 32 in Appendix A15) considering that the interaction model enforces symmetric relations. DistMult, however, achieves a strong performance only when composed with the LCWA and the CEL (Table 17 in Appendix A9), highlighting that a simple interaction model can obtain strong performance when composed beneficially.

Impact of the Training Approach Figure 17 shows that for both BCEL and SPL (we focus here only on these two loss functions since they have been trained with both training approaches), the LCWA obtains significantly higher results, but they express a high variance at the same time. Figures 18 and 36 (Appendix A19) illustrate that some interaction models are extremely sensitive to the choice of the training approaches. For instance, it can be observed that RotatE, TransE, and TuckER suffer when trained together with the sLCWA for both loss functions. Table 17 (Appendix A9) shows that most of the interaction models obtain their best performance on FB15K-237 when trained together with the LCWA.

Impact of the Loss Function Figures 16 and 17 illustrate that the BCEL and SPL outperform the other loss functions, but they also exhibit higher variance. Figure 36 (Appendix A19) expresses that some interaction models seem to be more sensitive to the usage of different loss function. For instance, ConvE and TuckER suffer from the MRL and the NSSAL, DistMult together with the CEL outperforms the other loss functions. However, TransE performs similarly

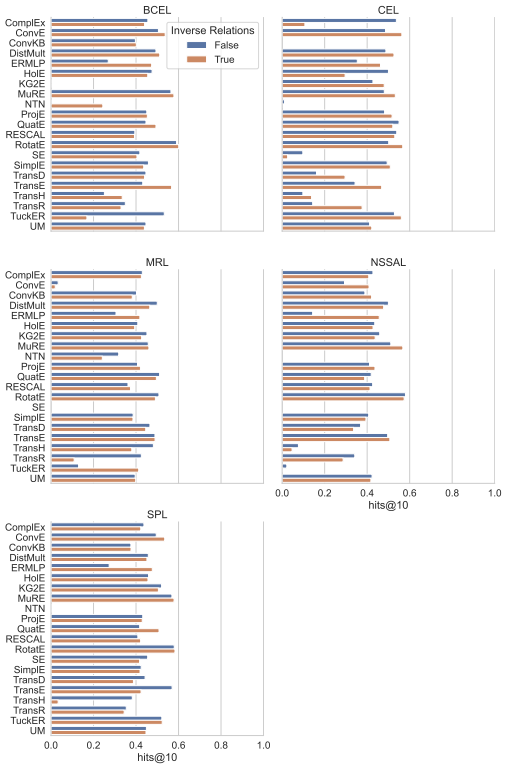


Fig. 15. Impact of explicitly modeling inverse relations on the performance for a fixed interaction model and loss function for the WN18RR dataset.

for all loss functions except the NSSAL.

Impact of Explicitly Modeling Inverse Relations Figure 19 reveals, as for the previous datasets, that in general, the usage of inverse relations is beneficial for the training based on the LCWA approach. Different from the results obtained for WN18RR (Figure 13), the LCWA is not competitive when trained without inverse relations, and that there are several sLCWA-configurations that benefit from inverse relations, too. Figure 20 underlines these observations: loss functions that are trained with the LCWA obtain more high performing configurations, and the best LCWA-configurations outperform the best sLCWA-configurations. The NSSAL obtains its best performance when trained together with inverse relations, although it has only been trained together with the sLCWA whereas the MRL suffers from inverse relations.

Figure 21 depicts the effect of explicitly modeling inverse relations on the combinations of interaction models and loss functions. There are specific combinations of interaction models and loss functions that benefit from the addition of inverse relations. Among those are, for instance, ERMLP, RESCAL, and TuckER when trained based on BCEL. A second example can be seen for models that have been trained based on SPL for which ComplEx improve when trained without inverse relations.

Model Complexity vs. Performance Figure 28 (Appendix A11) illustrates that for FB15K-237, there is no clear correlation between model size and performance. Tiny models can already obtain similar performance as larger models. The skyline comprises an intermediate UM, TransE and DistMult models, and a larger TuckER model. A full list is provided in Table 13 (Appendix A8).

7.4 Results on the YAGO3-10 Dataset

YAGO3-10 is the largest benchmark dataset in our study. Therefore, it is of interest to investigate how the different interaction models perform on a larger KG. As mentioned in the introduction of this chapter, we reduced the experimental setup for YAGO3-10 in order to reduce the computational complexity of our entire study. Figure 22 depicts the overall results obtained for YAGO3-10. Detailed results for all configurations are illustrated in Figure 33 in Appendix A16.

The results highlight the previous observation that the performance of many KGEMs heavily depends on the choice of its components and is dataset-specific. For instance, MuRE, the best-performing interaction model, and especially RotatE, which is among the top-performing interaction models, exhibit high variance across their configurations. TransE, which was among the top-performing interaction models on WN18RR, performed poorly on YAGO3-10. One might conclude that TransE performs better on smaller KGs, but the results obtained on Kinships do not support this assumption. It should be taken into account that some interaction models might benefit from being trained with the LCWA on YAGO3-10 as observed for TransE on WN18RR. Therefore, TransE might perform much better when trained with the LCWA approach. Remarkably, ComplEx and QuatE seem to be robust for all sLCWA configurations. With regards to the loss functions, all loss functions except MRL obtain comparable results. Though, the MRL is more robust than other loss functions.

Impact of the Loss Function Figure 23 depicts the impact of the loss functions for fixed interaction models. As observed for the previous datasets, some interaction models perform better with specific loss functions. For instance, RotatE and TransE obtain better performance when trained with MRL and NSSAL, whereas QuaE benefits from the SPL.

Impact of Explicitly Modeling Inverse Relations Figure 24 shows the effect of explicitly modeling inverse relations for fixed loss functions. Three key aspects can be observed. First, the BCEL performs better when trained without inverse relations both in terms of maximum performance and robustness. Second, the NSSAL is the only loss function for which the addition of inverse relations is beneficial for both performance and robustness. However, it should be considered that the median performance is worse compared to the setting in which inverse relations are omitted. Third, the MRL benefits from explicitly modeling inverse relations.

Figure 25 illustrates the effect of the usage of inverse relations on the interaction models. The figure reveals that the combinations of some interaction models and loss functions are not impacted by explicitly modeling inverse relations (e.g., ERMLP for all loss functions except MRL). In contrast,

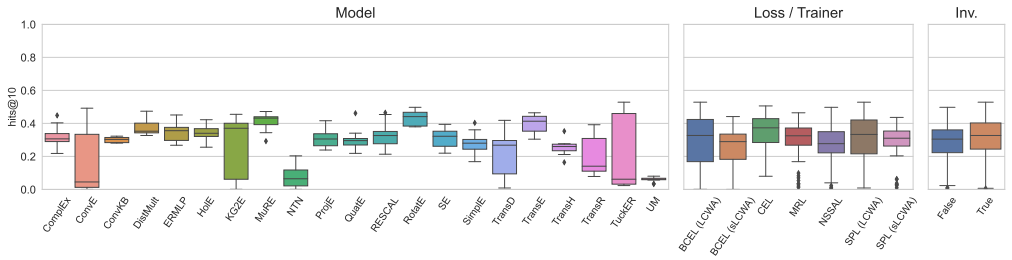


Fig. 16. Overall hits@10 results for FB15K-237 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations.

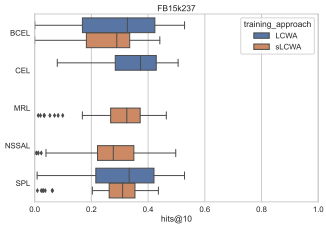


Fig. 17. Impact of the training approach on the performance for a fixed loss function for the FB15K-237 dataset.

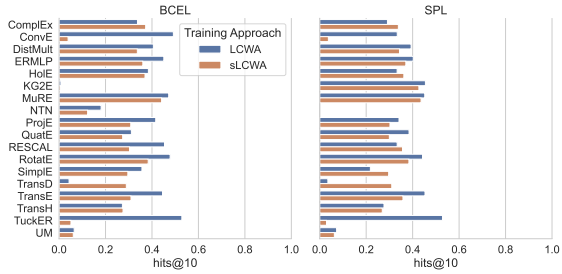


Fig. 18. Impact of training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset.

others have clear preferences, e.g., ConvKB suffers from the usage of inverse relations for all loss functions except SPL.

Model Complexity vs. Performance Figure 28 (Appendix A11) expresses that there is a low correlation between model size and performance for YAGO3-10. However, the improvement is tiny compared to the differences in model size. It should be taken into account that for KGEMs, the model size is usually dependent on the number of entities and relations. Therefore, dependent on the space complexity of the interaction model (Table 2 in Appendix A1), the size can grow fast for large KGs. The skyline comprises an intermediate TransE, DistMult and ConvKB model, and a larger MuRE model. A full list is provided in Table 16 (Appendix A8).

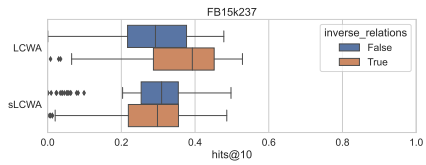


Fig. 19. Impact of explicitly modeling inverse relations on the performance for a fixed training approach for the FB15K-237 dataset.

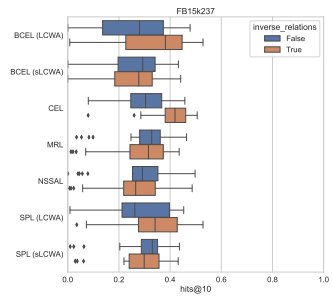


Fig. 20. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the FB15K-237 dataset.

8 Discussion & Future Work

Table 5 (Appendix A1) illustrates the extent of our studies and Table 6 (Appendix A2) summarizes the main findings our work. Although the re-implementation of all machine learning components into a unified, fully configurable framework was a major effort, we believe it is essential to analyze reproducibility and obtain fair results on benchmarking. In particular, we were able to address the issue of incompatible evaluation procedures and preprocessing steps in previous publications that are not obvious.

During our reproducibility study, we found that the reproduction of experiments is a major challenge and, in many cases, not possible with the available information in current publications. In particular, we observed the following four main aspects:

- For a set of experiments, the results can sometimes only be reproduced with a different set of hyperparameter values.

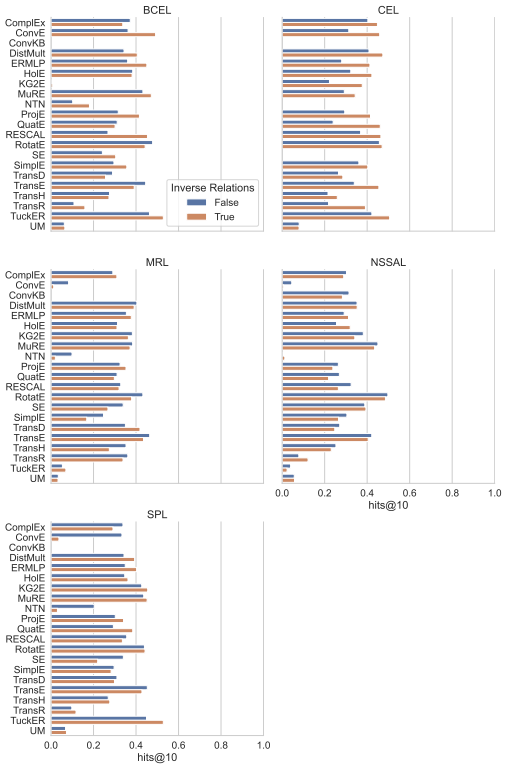


Fig. 21. Impact of explicitly modeling inverse relations on the performance for a fixed interaction model and loss function for the FB15K-237 dataset.

- For some experiments, the entire experimental setup was not provided, impeding the reproduction of experiments.
- The lack of an official implementation hampers the reproduction of results.
- Some results are dependent on the utilized ranking approach (average, optimistic, and pessimistic ranking approach). For example, the optimistic rank may lead to incorrect conclusions about the model's performance.

Our benchmarking study shows that the term KGEM should be used with caution and should be differentiated from the actual interaction model since our results highlight that the specific *combination* of the interaction model, training approach, loss function, and the usage of explicit inverse relations is often fundamental for the performance.

No configuration performs best across all datasets. Depending on the dataset, several configurations can be found that achieve comparable results (Tables 17-20 in Appendix A9-A10, and Figures 30-33 in Appendix A13-A16). Moreover, with an appropriate configuration, the model size can significantly be compressed (see Pareto-optimal configurations in Tables 13-16 in Appendix A8) that has especially a practical relevance when looking for a trade-off between required memory and performance.

The results also highlight that even interaction models such as TransE that have been considered as baselines can outperform state-of-the-art interaction models when trained with an appropriate training approach and loss function. This raises the question of the necessity of the vast number of available interaction models. However, for some interaction models such as RotatE, MuRE or TuckER, we can observe a good performance across all datasets (note: TuckER has not been evaluated on YAGO3-10). For RotatE, we even obtained the state-of-the-art results on WN18RR (similar results were obtained by Graph Attenuated Attention Networks [49]), and for ConvE, MuRE, and TuckER, we obtained results superior to the originally published ones. ComplEx proved to be a very robust interaction model across different configurations. This can, in particular, be observed from the results obtained on YAGO3-10 (Figure 22).

We discovered that no loss function consistently achieves the best results. Instead it can be seen that with different loss functions such as the BCEL, NSSAL, and SPL good results can be obtained across all datasets. Remarkably, the MRL is overall the worst-performing loss function. However, one might argue that the MRL is the most compatible loss function with the sLCWA since it does not assume artificially generated negative examples to be actually false in contrast to the other loss functions used. The MRL only learns to score positive examples higher than *corresponding* negative examples, but it does not ensure that a negative example is scored lower than every other positive example. Thus, the absolute score values are not interpretable and cannot be used to compared triples without common head/tail entities. They can only be interpreted relatively, and only when comparing scores for triples with the same $(hr)/(rt)$. Although loss functions such as BCEL or SPL treat generated negative triples as true negatives that actually contain also unknown positive examples, they obtain good performance. This might be explained by the fact that usually the set of unknown triples are dominated by false triples. Therefore, it is likely that a major part of the generated triples are actually negative. Consequently, the KGEM learns to distinguish better positive from negative examples.

Considering the explicit usage of inverse relations, we found out that the impact of inverse relations can be significant, especially when the interaction model is trained under the LCWA. This might be explained by the fact that based on the LCWA-training, the KGEM only learns to perform *one-side predictions* (i.e., it learns to either predict head or tail entities), but during the evaluation, it is asked to perform *both-side predictions*. Through the inclusion of inverse relations, the model learns to perform both-side predictions based on one side, i.e., $(*, r, t)$ can be predicted through $(t, r_{inverse}, *)$. Overall, our results indicate that further investigations on FB15K-237 and YAGO3-10 might lead to results that are competitive to the state-of-the-art.

Looking forward, it would be of great interest to re-investigate previously performed studies that analyze the relationship between the performance of KGEMs and the properties of the underlying KGs to verify that their findings indeed can be attributed to the *interaction model* alone, rather than the exact configuration including the loss function, the training approach and the explicit modeling of inverse relations. Further, the effect of explicitly modeling inverse

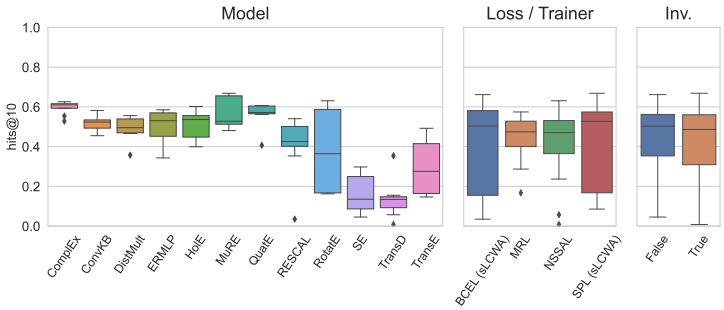


Fig. 22. Overall hits@10 results for YAGO3-10 where box-plots summarize the results across different combinations of interaction models, training approaches, loss functions, and the explicit usage of inverse relations. In contrast, to the previous datasets, the models have only been trained based on the stochastic local closed world assumption.

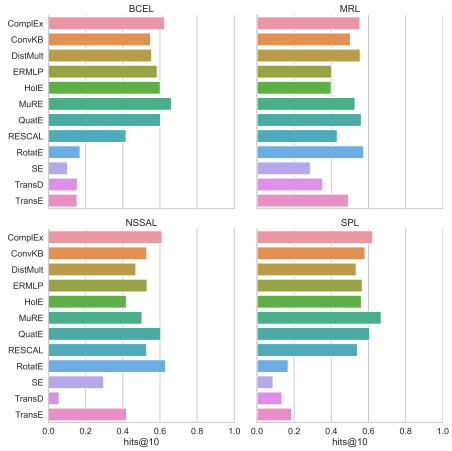


Fig. 23. Impact of the loss functions on the performance for a fixed interaction models for the YAGO3-10 dataset. Because this dataset was only trained under the sLCWA, the boxers are not split like for the other datasets.

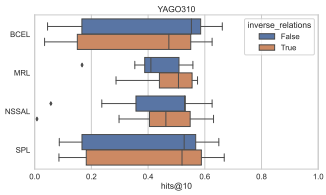


Fig. 24. Impact of explicitly modeling inverse relations on the performance for a fixed loss function for the YAGO3-10 dataset.

relations has not been analyzed in depth, in particular how the learned representations of a relation and its inverse are related to each other. Ultimately, we believe our work provides an empirical foundation for such studies and a practical tool to execute them.

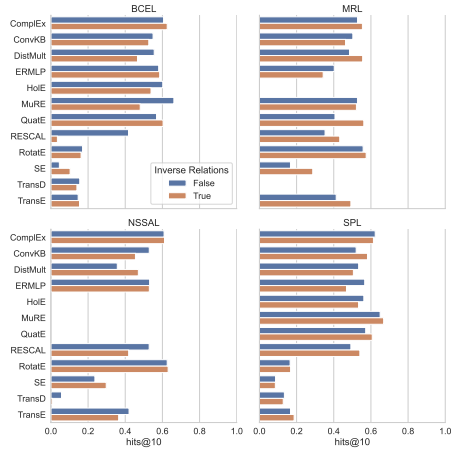


Fig. 25. Impact of explicitly modeling inverse relations on the performance for a fixed interaction model and loss function for the YAGO3-10 dataset.

Acknowledgment

We want to thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocations of computer time and the Technical University of Denmark for providing us access to their DTU Compute GPU cluster that enabled us to conduct our studies. This work was funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and Grant No. 01IS18050D (project "MLWin"), the Innovation Fund Denmark with the Danish Center for Big Data Analytics driven Innovation (DABAI), and the Defense Advanced Research Projects Agency (DARPA) Automating Scientific Knowledge Extraction (ASKE) program under grant HR00111990009.

References

[1] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017.

[2] F. Akrami, L. Guo, W. Hu, and C. Li, "Re-evaluating embedding-based knowledge graph completion methods," in *CIKM*. ACM, 2018, pp. 1779–1782.

[3] R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," in *Rep4NLP@ACL*. Association for Computational Linguistics, 2017, pp. 69–74.

[4] Z. Sun, S. Vashishth, S. Sanyal, P. P. Talukdar, and Y. Yang, "A re-evaluation of knowledge graph completion methods," in *ACL*. Association for Computational Linguistics, 2020, pp. 5516–5522.

[5] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR (Poster)*, 2015.

[6] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, "Realistic re-evaluation of knowledge graph completion methods: An experimental study," in *SIGMOD Conference*. ACM, 2020, pp. 1995–2010.

[7] S. K. Mohamed, V. Novácek, P. Vandenbussche, and E. Muñoz, "Loss functions in knowledge graph embedding models," in *DL4KG@ESWC*, ser. CEUR Workshop Proceedings, vol. 2377. CEUR-WS.org, 2019, pp. 1–10.

[8] D. Ruffinelli, S. Broscheit, and R. Gemulla, "You CAN teach an old dog new tricks! on training knowledge graph embeddings," in *ICLR*. OpenReview.net, 2020.

[9] A. Rossi, D. Firmani, A. Matinata, P. Merialdo, and D. Barbosa, "Knowledge graph embedding for link prediction: A comparative analysis," *CoRR*, vol. abs/2002.00819, 2020.

[10] M. Berrendorf, E. Faerman, L. Vermue, and V. Tresp, "Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank," *CoRR*, vol. abs/2002.06914, 2020.

[11] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proc. IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[12] B. Kotnis and V. Nastase, "Analysis of the impact of negative sampling on link prediction in knowledge graphs," *CoRR*, vol. abs/1708.06816, 2017.

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 413–422.

[14] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.

[16] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, pp. 70:1–70:73, 2020.

[17] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation," *Mach. Learn.*, vol. 94, no. 2, pp. 233–259, 2014.

[18] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI*. AAAI Press, 2011.

[19] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*. AAAI Press, 2014, pp. 1112–1119.

[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*. AAAI Press, 2015, pp. 2181–2187.

[22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL (1)*. The Association for Computer Linguistics, 2015, pp. 687–696.

[23] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *ICLR (Poster)*. OpenReview.net, 2019.

[24] I. Balazevic, C. Allen, and T. M. Hospedales, "Multi-relational poincaré graph embeddings," in *NeurIPS*, 2019, pp. 4465–4475.

[25] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *CIKM*. ACM, 2015, pp. 623–632.

[26] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*. Omnipress, 2011, pp. 809–816.

[27] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 2071–2080.

[28] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," in *NeurIPS*, 2019, pp. 2731–2741.

[29] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," in *NeurIPS*, 2018, pp. 4289–4300.

[30] I. Balazevic, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," in *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5184–5193.

[31] L. R. Tucker *et al.*, "The extension of factor analysis to three-dimensional matrices," *Contributions to mathematical psychology*, vol. 110119, 1964.

[32] B. Shi and T. Weninger, "Proje: Embedding projection for knowledge graph completion," in *AAAI*. AAAI Press, 2017, pp. 1236–1242.

[33] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *AAAI*. AAAI Press, 2016, pp. 1955–1961.

[34] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *KDD*. ACM, 2014, pp. 601–610.

[35] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013, pp. 926–934.

[36] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," *arXiv preprint arXiv:1712.02121*, 2017.

[37] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*. AAAI Press, 2018, pp. 1811–1818.

[38] H. Zhang, Z. Kyaw, S. Chang, and T. Chua, "Visual translation embedding network for visual relation detection," in *CVPR*. IEEE Computer Society, 2017, pp. 3107–3115.

[39] S. Sharifzadeh, M. Berrendorf, and V. Tresp, "Improving visual relation detection using depth maps," *CoRR*, vol. abs/1905.00966, 2019.

[40] T. Lacroix, N. Usunier, and G. Obozinski, "Canonical tensor decomposition for knowledge base completion," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 2869–2878.

[41] N. Fuhr, "Some common mistakes in IR evaluation, and how they can be avoided," *SIGIR Forum*, vol. 51, no. 3, pp. 32–41, 2017.

[42] S. S. Stevens, "On the theory of scales of measurement," *Science*, vol. 103, no. 2684, pp. 677–680, 1946.

[43] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.

[44] W. W. Denham, "The detection of patterns in alyawara nonverbal behavior," Ph.D. dissertation, University of Washington, Seattle., 1973.

[45] R. J. Rummel, *The dimensionality of nations project: attributes of nations and behavior of nations dyads, 1950-1965*. Inter-university Consortium for Political Research, 1976, no. 5409.

[46] A. T. McCray, "An upper-level ontology for the biomedical domain," *International Journal of Genomics*, vol. 4, no. 1, pp. 80–84, 2003.

[47] T. Rebele, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames," in *International Semantic Web Conference (2)*, ser. Lecture Notes in Computer Science, vol. 9982, 2016, pp. 177–185.

[48] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *CIDR*. www.cidrdb.org, 2015.

[49] R. Wang, B. Li, S. Hu, W. Du, and M. Zhang, "Knowledge graph embedding via graph attenuated attention networks," *IEEE Access*, vol. 8, pp. 5212–5224, 2020.

Mehdi Ali Mehdi Ali received his M.Sc. degree in Computer Science with a focus on intelligent systems from the University of Bonn. Currently, he is a Ph.D. candidate at the computer science department of the University of Bonn and a research associate at the Fraunhofer Institute IAIS. In his Ph.D., he focuses on machine learning models for (knowledge) graphs, multi-modal models that combine graph and textual information, and reproducibility in the field of knowledge graph embedding models.

Max Berrendorf Max Berrendorf received his B.Sc and M.Sc. degree in Computer Science with a minor in Mathematics from RWTH Aachen University. Currently he is pursuing a Ph.D. degree at the chair of Database Systems and Data Mining at Ludwig-Maximilians-Universität München. In his research, he focuses on machine learning on graphs, in particular knowledge graphs, graph matching problems, and reproducibility in machine learning.

Charles Tapley Hoyt Dr. Charles Tapley Hoyt completed his Ph.D. in Computational Life Sciences from the University of Bonn in 2019 and is now affiliated with the Laboratory of Systems Pharmacology at Harvard Medical School, Boston, USA. His interests are in the biological applications of knowledge graph embedding models towards proteochemometrics, target prioritization, drug repositioning, predictive toxicology, and precision medicine.

Laurent Vermue Laurent Vermue received his M.Sc. degree in Industrial Engineering and Management at the Technical University of Berlin and MMSc. degree in Management Science and Engineering at the Tongji University. Currently he is a Ph.D. student at the Section for Statistics and Data Analysis and the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, complex network modeling and open research software.

Mikhail Galkin Dr. Mikhail Galkin received his Ph.D. degree in Computer Science from the University of Bonn in 2018 studying knowledge graphs, their creation, integration, and querying. Currently, he is a postdoctoral fellow at Montreal Institute for Learning Algorithms (Mila) and McGill University. His interests include applications of knowledge graphs and graph representation learning to neural reasoning and natural language processing.

Sahand Sharifzadeh Sahand Sharifzadeh received his M.Sc degree from Technical University of Munich majoring in Computer Vision and Artificial Intelligence. Currently, he is a Ph.D. candidate at Ludwig-Maximilians-Universität München. In his research, he focuses on extracting graphs from images and text, as well as knowledge graph modeling. He often collaborates with biologists, physicists and robotic engineers as interdisciplinary machine learning research is one of his interests.

Asja Fischer Asja Fischer is professor for machine learning at Ruhr University Bochum. Her research interests are focus on the development, analysis, and application of deep learning models and methods. Before becoming a professor in Bochum she was assistant professor at Bonn university, and a post-doctoral researcher at the Montreal Institute for Learning Algorithms (MILA). Between 2010 and 2015, she was employed both at the Institute for Neural Computation at the Ruhr University Bochum and the Department of Computer Science at the University of Copenhagen working on her PhD, which she defended in Copenhagen in 2014. Before, she studied Biology, Bioinformatics, Mathematics, and Cognitive Science at the Ruhr-University Bochum, the Universidade de Lisboa, and the University of Osnabrück.

Volker Tresp Volker Tresp received the Diploma degree from the University of Goettingen, Germany, in 1984 and the M.Sc. and Ph.D. degrees from Yale University, New Haven, CT, USA, in 1986 and 1989, respectively. Since 1989, he has been the head of various research teams in machine learning at Siemens, Research and Technology, Munich, Germany. He filed more than 70 patent applications and was inventor of the year of Siemens in 1996. He has published more than 100 scientific articles and administered over 20 Ph.D. dissertations. The company Panoratio is a spin-off out of his team. His research focus in recent years has been machine learning in information networks for modeling knowledge graphs, medical decision processes, and sensor networks. He is the coordinator of one of the first nationally funded big data projects for the realization of precision medicine. In 2011, he became a Honorary Professor at the Ludwig Maximilian University of Munich, Germany, where he teaches an annual course on machine learning.

Jens Lehmann Prof. Dr. Jens Lehmann leads the "Smart Data Analytics" research group at the University of Bonn and Fraunhofer IAIS with 40 researchers. His research interests involve knowledge graphs, machine learning, question answering, distributed computing and knowledge representation. He is particularly excited about the combination of data- and knowledge-driven AI methods. Prof. Lehmann won more than 10 international awards for his research work. He is founder, leader or contributor of several community research projects, including SANSA, DL-Learner, DBpedia and LinkedGeoData. Previously, he completed his PhD with "summa cum laude" at the University of Leipzig with visits to the University of Oxford. He studied Computer Science at the Technical University of Dresden.

TABLE 2
Investigated interaction models [33] and their required number of parameters. k corresponds to the number of neurons in the hidden layer, n_f to the number of convolutional kernels, k_r and k_c to the height and width of the convolutional kernels.

Model	Parameters
ComplEx[a]	$\|\mathcal{E}\|2d + \|\mathcal{R}\|2d$
ConvE[b]	$\|\mathcal{E}\|d + \|\mathcal{R}\|d + d + n_f k_r k_c + 2 + 2n_f + 2d$ $+(h - k_r + 1)(w - k_c + 1)n_f d + \|\mathcal{E}\|$
ConvKB	$\|\mathcal{E}\|d + \|\mathcal{R}\|d + n_f(d + 4) + 1$
DistMult	$\|\mathcal{E}\|d + \|\mathcal{R}\|d$
ER-MLP	$\|\mathcal{E}\|d + \|\mathcal{R}\|d + k(3d + 2) + 1$
HolE	$\|\mathcal{E}\|d + \|\mathcal{R}\|d$
KG2E	$\|\mathcal{E}\|2d + 2\|\mathcal{R}\|d$
MuRE	$\|\mathcal{E}\|(d + 2) + 3\|\mathcal{R}\|d$
NTN	$\|\mathcal{E}\|d + \|\mathcal{R}\|k(d^2 + 2d + 2)$
ProjE	$\|\mathcal{E}\|d + \|\mathcal{R}\|d + 3d + 1$
QuatE[c]	$\|\mathcal{E}\|4d + \|\mathcal{R}\|4d$
RESCAL	$\|\mathcal{E}\|d + \|\mathcal{R}\|d^2$
RotatE[a]	$\|\mathcal{E}\|2d + \|\mathcal{R}\|d$
SE	$\|\mathcal{E}\|d + 2\|\mathcal{R}\|d^2$
SimplE	$\|\mathcal{E}\|2d + 2\|\mathcal{R}\|d$
TransE	$\|\mathcal{E}\|d + \|\mathcal{R}\|d$
TransH	$\|\mathcal{E}\|d + 2\|\mathcal{R}\|d$
TransR	$\|\mathcal{E}\|d_e + \|\mathcal{R}\|d_r + d_e d_r$
UM	$\|\mathcal{E}\|d$
TuckER	$\|\mathcal{E}\|d_e + \|\mathcal{R}\|d_r + d_e^2 d_r + 4d_e$

[a] $2d$, because of complex valued vectors, i.e. imaginary and real part of a number.
[b] w and h correspond to the height and weight of the reshaped input.
[c] $4d$, because of hyper-complex valued (quaternion) vectors, i.e. a real part and three imaginary parts of a quaternion.

TABLE 3
Denotes for each proposed model whether results have been reported for FB15K, WN18, or their alterations. Furthermore, it indicates whether an official implementation exists where P corresponds to a PyTorch based implementation, T to a TensorFlow based implementation, and O to other implementations. A green background indicates that the full experimental setup was available. The models highlighted with * where included in the reproducibility study.

Model	Code	FB15K	FB15K-237	WN18	WN18RR
ComplEx*	O	✓		✓	
ConvE*	P	✓	✓	✓	✓
ConvKB*	T		✓		✓
DistMult*	-	✓		✓	
ER-MLP	-				
HolE*	O	✓		✓	
KG2E*	-	✓		✓	
MuRE*	P		✓		✓
NTN	-				
ProjE	T	✓		✓	
QuatE*[a]	P	✓	✓	✓	✓
UM	-				
RESCAL	O				
RotatE*	P	✓	✓	✓	✓
SE	O				
SimplE*	T, P	✓		✓	
TransD*	-	✓		✓	
TransE*	O	✓		✓	
TransH*	-	✓		✓	
TransR*	O	✓		✓	
TuckER*	P	✓	✓	✓	✓
UM	-				

[a] Code is based on the framework OpenKE https://github.com/thunlp/OpenKE.

TABLE 4
Hyper-Parameter Ranges for Ablation Experiments

	Hyper-Parameter	Range
Shared	Embedding-Dimension	{64,128,256}
	Initialization	{Xavier}
	Optimizers[a]	{Adam, Adadelta}
	Learning Rate (log scale)	[0.001, 0.1)
	Batch Size[b]	{128, 256, 512}
	Model inverse relations	{Yes, No}
	Epochs	1,000
sLCWA	Loss	{BCEL, MRL, NSSAL, SPL}
	Margin for MRL	{0.5, 1.5, ... , 9.5}
	Margin for NSSAL	{1, 3, 5, ... , 29}
	ADVT for NSSAL	{0.1, 0.2, ... , 1.0}
	Number of Negatives[c]	{1, 2, ... , 100}
LCWA	Loss	{BCEL, CEL, SPL}
	Label Smoothing (log scale)	[0.001, 1.0)

[a] For Kinships, we evaluated Adam and Adadelta, and for the remaining datasets we sticked to Adam since it performed almost in every experiment at least equally good as Adadelta and in many experiments significantly better.
[b] For YAGO3-10, the batch-size has been sampled from the set {1024, 2048, 2096, 8192}.
[c] For YAGO3-10, the number of negative triples per each each positive has been sampled from the set {1, 2, ..., 50}.

TABLE 5
Evaluation statistics

Metric	Value
Datasets	4
Interaction Models	21
Training approaches	2
Loss Functions	5
Negative Samplers	1
Optimizers	2
Ablation Studies	1,207
Number of Experiments	73,683
Compute Time (hours)	24,804

TABLE 6
Summary of main insights over all datasets. Each component (i.e., interaction model, loss function, and training approach) is considered to be among the top-ten performing configurations when they occur at least once in the top-ten performing configurations. Note that a single component is part of several configurations, and therefore, can occur multiple times in the top-ten performing configurations.

Interaction Models	
RotatE	Among top-ten-performing interaction models across all datasets.
MuRE	Among top-ten-performing interaction models on WN18RR, FB15K-237, and YAGO3-10.
ConvE	Among top-ten-performing interaction models on Kinships and FB15K-237 (has not been evaluated on YAGO3-10).
ComplEx	Among top-ten-performing interaction models on Kinships and YAGO3-10.
TuckER	Among top-ten-performing interaction models for Kinships, and FB15K-237 (has not been evaluated on YAGO3-10).
DistMult	Among top-ten-performing interaction models on FB15K-237.
QuatE	Among top-ten-performing interaction models on YAGO3-10.
TransE	Among top-ten-performing interaction models on WN18RR.
SE	Among top-ten-performing interaction models on Kinships.
Loss Functions	
BCEL	Among top-ten-performing loss functions across all datasets.
NSSAL	Among top-ten-performing loss functions across all datasets.
SPL	Among top-ten-performing loss functions across all datasets.
CEL	Among top-ten-performing loss functions on Kinships and FB15K-237 (has not been evaluated on YAGO3-10).
MRL	Among top-ten-performing loss functions on Kinships.
Training Approaches	
sLCWA	Among top-ten-performing training approaches across all datasets.
LCWA	Among top-ten-performing training approaches on Kinships, WN18RR and FB15K-237 (has not been evaluated on YAGO3-10).
Explicit Modeling of Inverse Relations	
	Is usually beneficial in combination with the local closed world assumption.
Configurations	
Performance	Appropriate combination of interaction model, training assumption, loss function, choice of explicitly modeling inverse relations is crucial for the performance, e.g., TransE can compete when with several state-of-the-art interaction models on WN18RR when appropriate configuration is selected. There is no single best configuration that works best for all dataset.
Variance	Some interaction models exhibit a high variance across different configurations, e.g., RotatE on YAGO3-10 (Figure 22 on page 18)
Pareto-Optimal Configurations	Tables 13-16 in Appendix A8 describe Pareto-optimal configurations. It can be seen that there are configurations that require fewer parameters while obtaining almost the same performance. In some cases, for the same interaction model, the model can be significantly compressed.
Reproducibility	
Results	For FB15K, four out of 13, for WN18, five out of 13, for FB15K-237, two out of three, and for WN18RR, three out of five experiments can be categorized as soft-reproducible.
Code	For four out of 15 models, no official implementation was available.
Parameters	For six out of 15 papers, source code was available and full experimental setup was precisely described.
General Insights	
SOTA	For WN18RR, we achieve based on a RotatE-configuration (together with Graph Attenuated Attention Networks [49]) state-of-the-art results in terms of hits@10 through our study (60.09% Hits@10). Furthermore, we found a TransE configuration that achieves high performance beating most of the published SOTA results (56.98% Hits@10). Based on our results, we emphasize to further investigate the hyper-parameters space for the most promising configurations for the remaining benchmarking datasets.
Improvements	For ConvE (56.33% compared to 52.00% [37]), MuRE (57.90% compared to 55.50% [24]) and TuckER (56.09% compared to 52.6% [30]), we are beating the reported results in the original papers due selecting appropriate configurations and hyper-parameters on WN18RR.

TABLE 7
Frequency of detected relation patterns across the benchmark datasets.

pattern dataset	anti-symmetry	composition	symmetry
fb15k237	205	147	3
wn18rr	7	1	3
yago310	30	3	2

RELATIONAL PATTERN ANALYSIS

Knowledge graphs exhibit relational patterns such as symmetry (e.g., the relation *marriedTo*), and the performance of KGEMs depend on how well these patterns can be modeled. Four major relational patterns that have been investigated in the literature are *symmetry*, *anti-symmetry*, *inversion*, and *composition* [23], [27], [43]. Here, we provide a large-scale performance analysis of our investigated KGEMs in modeling *symmetry*, *anti-symmetry*, and *composition* patterns for the datasets FB15k-237, WN18RR, and YAGO3-10. First, we provide statistics about the *support* and *confidence* of the symmetry, anti-symmetry, inversion, and composition patterns in the FB15k-237, WN18RR and YAGO3-10 datasets. Next, we describe our experimental setup. Finally, we present the results of our relational pattern analysis.

Relational Patterns and their Detection

Here, we formally define the relational patterns symmetry, anti-symmetry, inversion, and composition patterns according to [23], the measures *support* and *confidence*, and provide an overview of the *support* and *confidence* of the these patterns in the FB15k-237, WN18RR and YAGO3-10 datasets.

Definition 1 (Symmetric Relation). A relation $r \in \mathcal{R}$ is **symmetric**, if $(h, r, t) \in \mathcal{T} \implies (t, r, h) \in \mathcal{T}$

Definition 2 (Anti-Symmetric Relation). A relation $r \in \mathcal{R}$ is **anti-symmetric**, if $(h, r, t) \in \mathcal{T} \implies (t, r, h) \notin \mathcal{T}$

Definition 3 (Inverse Relation). A relation $r \in \mathcal{R}$ is **inverse** to $r_{inv} \in \mathcal{R}$, if $(h, r, t) \in \mathcal{T} \implies (t, r_{inv}, h) \in \mathcal{T}$. If there exists a $r' \in \mathcal{R}$ with $r' \neq r$ and r' is inverse to r, then we call r an inverse relation.

Definition 4 (Composite Relation). A relation $r \in \mathcal{R}$ is a **composition** of two relations $r_1, r_2 \in \mathcal{R}$, if $(a, r_1, b) \in \mathcal{T} \wedge (b, r_2, c) \in \mathcal{T} \implies (a, r, c) \in \mathcal{T}$. We call r a composite relation, if such two relations exist.

Since KGs are known to be incomplete, a false antecedent, i.e., right-hand side of a rule, may not only be caused by the relation not being of the relation type of interest, but also originate from the KG's incompleteness. Thus, we detect relation types using a support and confidence threshold, defined akin to the concepts of association rule mining.

The *support* of one of the aforementioned patterns p for a relation r indicates the number of different assignments of entities such that the precedent, i.e., the left-hand side of a rule, of a rule holds. For most of the simple rules this is equivalent to the relation frequency, but, e.g., for composite relations, we need to consider all pairs of triples with matching the candidate relations r_1, r_2 and being linked by the intermediate entity b.

The *confidence* of a relational pattern is the number of times the right-hand side hold divided by the support. Thus, it can be interpreted as an estimate of the the conditional probability of the antecedent, given the precedent holds.

Relation Patterns in Benchmark Datasets

Table 7 shows the frequency of the detected pattern types for the three studied benchmark datasets. Similar to related work we used a confidence threshold of 97% [43]. Note that we did not detect a single inverse relation, since FB15k-237 and WN18RR have been explicitly preprocessed to remove such.

Experimental Setup

To measure the performance of the investigated KGEMs in modeling symmetry, anti-symmetry, and composition patterns, we slightly adapted the standard link prediction evaluation procedure (Section 4). Instead of computing the metrics based on all test triples, we extracted for each relational pattern all test triples that contain the associated relations, aggregated the single ranks obtained of each triple in the subset, and computed the hits@10 metric for each subset. Therefore, we can express how well a KGEM can model a specific relational pattern.

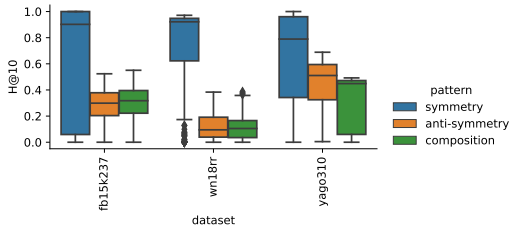


Fig. 26. Performance Distribution of all best models per configuration in H@10.

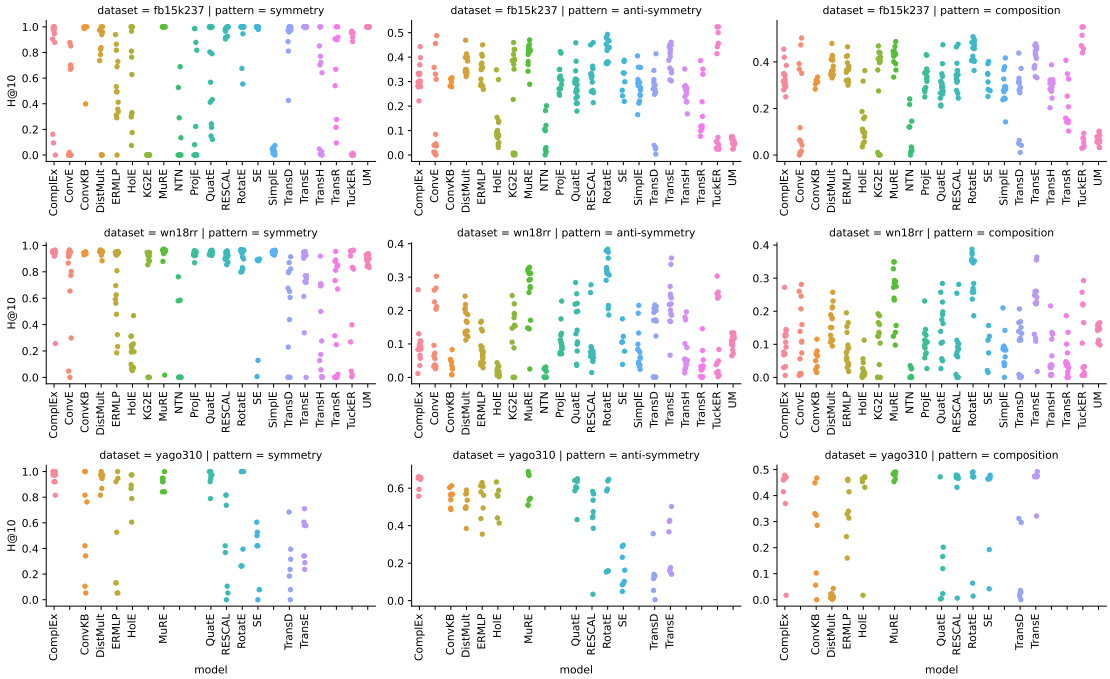


Fig. 27. Performance of best models' for each configuration for each dataset and pattern type, grouped by interaction function.

Results

Figure 26 shows the overall performance on pattern types per dataset. We show the distribution of best models' performance for each configuration in terms of H@10. We generally observe a tendency that symmetric relations are easier to model than anti-symmetric and composite relations, which seem to be equally challenging.

Figure 27 shows the performance of best models' for each configuration for each dataset and pattern type, grouped by interaction function.

For the most simple pattern, symmetry, almost all interaction functions can obtain strong results on WN18RR, with NTN, TransD and SE slightly falling behind. For FB15k237, we observe similar results, except that SimplE and KG2E fail to capture this pattern (while performing still sufficiently good on other patterns). On YAGO3-10, translation-based methods such as TransE or TransD cannot match the performance of, ComplEx, RotatE and DistMult, with ER-MLP's performance in between.

On the more difficult anti-symmetry and composition, the differences are more pronounced. Overall, RotatE and TransE obtain the best results. UM and NTN cannot obtain good results.

ADDITIONAL RESULTS FROM REPRODUCIBILITY STUDY

TABLE 8
Reproduction of Studies on FB15K where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ComplEx	pub (O)	69.20	59.90	75.90		84.00		
	R	21.94 ± 0.71	12.73 ± 0.74	24.18 ± 0.65	30.67 ± 0.60	40.61 ± 0.74	170.56 ± 17.18	2.31 ± 0.23
ConvE	pub (ND)	65.70	55.80	72.30		83.10	51.00	
	R	75.45 ± 0.17	68.26 ± 0.27	80.47 ± 0.08	83.94 ± 0.01	87.68 ± 0.04	43.97 ± 0.60	0.60 ± 0.01
DistMult	pub (U)	35.00				57.70		
	R	28.47 ± 0.23	18.59 ± 0.19	31.77 ± 0.29	38.24 ± 0.38	47.81 ± 0.36	127.16 ± 0.85	1.72 ± 0.01
HolE	pub (ND)	52.40	40.20	61.30		73.90		
	R	39.72 ± 0.32	27.15 ± 0.33	46.13 ± 0.40	54.05 ± 0.36	64.02 ± 0.27	186.22 ± 6.21	2.52 ± 0.08
KG2E	pub (U)					71.50	59.00	
	R	0.63 ± 0.08	0.15 ± 0.04	0.41 ± 0.11	0.66 ± 0.17	1.25 ± 0.21	5784.42 ± 22.26	78.31 ± 0.30
QuatE[1]	pub (O)	77.00	70.00	82.10		87.80	41.00	
	R	22.19 ± 0.17	14.65 ± 0.16	23.76 ± 0.26	29.37 ± 0.40	37.42 ± 0.39	229.99 ± 1.57	3.11 ± 0.02
RotatE	pub (ND)	79.70	74.60	83.00		88.40	40.00	
	R	64.94 ± 0.03	53.05 ± 0.05	73.31 ± 0.06	78.74 ± 0.06	84.85 ± 0.03	35.66 ± 0.06	0.48 ± 0.00
SimplE	pub (O)	72.70	66.00	77.30		83.80		
	R	0.04 ± 0.00	0.01 ± 0.00	0.03 ± 0.01	0.03 ± 0.01	0.05 ± 0.00	7386.02 ± 2.11	99.99 ± 0.03
	O	23.62 ± 12.90	11.67 ± 8.68	24.65 ± 16.33	34.28 ± 20.19	51.91 ± 24.57	148.27 ± 89.28	
	P	0.03 ± 0.00	0.01 ± 0.00	0.03 ± 0.01	0.03 ± 0.01	0.05 ± 0.00	14623.77 ± 91.95	
TransD	pub (U)					77.30	91.00	
	R	37.30 ± 0.05	24.45 ± 0.08	44.22 ± 0.09	51.78 ± 0.09	61.31 ± 0.07	146.55 ± 3.10	1.98 ± 0.04
TransE	pub (U)					47.10	125.00	
	R	29.11 ± 0.20	17.99 ± 0.27	33.53 ± 0.18	40.76 ± 0.21	50.84 ± 0.28	122.01 ± 1.09	1.65 ± 0.01
TransH	pub (U)					64.40	87.00	
	R	2.59 ± 0.27	1.89 ± 0.35	2.87 ± 0.23	3.16 ± 0.11	3.46 ± 0.15	6318.90 ± 18.86	85.54 ± 0.26
TransR	pub (ND)					68.70	77.00	
	R	1.23 ± 0.04	0.38 ± 0.00	1.34 ± 0.10	1.93 ± 0.12	2.79 ± 0.09	6130.41 ± 9.59	82.99 ± 0.13
TuckER	pub (ND)	79.50	74.10	83.30		89.20		
	R	79.02 ± 0.12	73.10 ± 0.11	83.05 ± 0.13	85.93 ± 0.16	89.10 ± 0.10	40.35 ± 0.83	0.55 ± 0.01

TABLE 9
Reproduction of Studies on FB15K-237 where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ConvE	pub (ND)	32.50	23.70	35.60		50.10	244.00	
	R	29.69 ± 0.19	21.13 ± 0.21	32.32 ± 0.19	38.57 ± 0.12	47.19 ± 0.08	245.83 ± 4.97	3.45 ± 0.07
ConvKB	pub (O)	39.60				51.70	257.00	
	R	4.22 ± 0.18	2.75 ± 0.27	3.65 ± 0.19	4.44 ± 0.19	7.18 ± 0.71	4314.45 ± 27.24	60.46 ± 0.38
MuRE	pub (R)	33.60	24.50	37.00		52.10		
	R	25.16 ± 0.20	16.12 ± 0.30	27.67 ± 0.21	34.21 ± 0.32	43.78 ± 0.13	190.61 ± 0.58	2.67 ± 0.01
QuatE[1]	pub (O)	31.10	22.10	34.20		49.50	176.00	
	R	0.26 ± 0.02	0.18 ± 0.03	0.23 ± 0.02	0.25 ± 0.02	0.30 ± 0.01	7119.76 ± 36.06	99.78 ± 0.51
RotatE	pub (ND)	33.80	24.10	37.50		53.30	177.00	
	R	28.79 ± 0.07	19.74 ± 0.08	31.67 ± 0.05	37.89 ± 0.07	47.13 ± 0.07	176.70 ± 0.48	2.48 ± 0.01
TuckER	pub (ND)	35.80	26.60	39.40		54.40		
	R	35.51 ± 0.08	26.20 ± 0.15	39.05 ± 0.10	45.59 ± 0.12	54.11 ± 0.04	152.46 ± 2.32	2.14 ± 0.03

TABLE 10
Reproduction of Studies on WN18 where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ComplEx	pub (O)	94.10	93.60	94.50		94.70		
	R	18.28 ± 2.10	11.65 ± 1.32	19.04 ± 2.44	23.38 ± 3.06	30.70 ± 3.90	442.51 ± 47.32	2.16 ± 0.23
ConvE	pub (ND)	94.30	93.50	94.60		95.60	374.00	
	R	94.23 ± 0.08	93.54 ± 0.16	94.68 ± 0.04	95.03 ± 0.02	95.39 ± 0.09	462.53 ± 32.15	2.26 ± 0.16
DistMult	pub (U)	83.00				94.20		
	R	82.41 ± 0.24	74.74 ± 0.31	89.09 ± 0.19	91.36 ± 0.22	93.44 ± 0.15	454.41 ± 43.08	2.22 ± 0.21
HolE	pub (ND)	93.80	93.00	94.50		94.90		
	R	73.43 ± 0.40	63.22 ± 0.57	81.80 ± 0.40	85.81 ± 0.20	89.30 ± 0.26	786.05 ± 33.16	3.84 ± 0.16
KG2E	pub (U)					92.80	331.00	
	R	3.73 ± 0.22	1.46 ± 0.19	3.27 ± 0.26	4.77 ± 0.32	7.39 ± 0.33	2732.49 ± 57.69	13.35 ± 0.28
	O	3.74 ± 0.22	1.46 ± 0.19	3.27 ± 0.26	4.77 ± 0.32	7.39 ± 0.33	2732.49 ± 57.69	
QuatE[1]	pub (O)	94.90	94.10	95.40		96.00	388.00	
	R	67.28 ± 0.70	58.38 ± 0.88	73.05 ± 0.61	77.86 ± 0.53	83.25 ± 0.38	327.12 ± 12.44	1.60 ± 0.06
RotatE	pub (ND)	94.90	94.40	95.20		95.90	309.00	
	R	93.71 ± 0.03	92.27 ± 0.03	94.87 ± 0.06	95.34 ± 0.04	95.83 ± 0.05	270.22 ± 7.24	1.32 ± 0.04
SimplE	pub (O)	94.20	93.90	94.40		94.70		
	R	0.04 ± 0.02	0.01 ± 0.01	0.03 ± 0.02	0.04 ± 0.03	0.06 ± 0.03	20355.98 ± 19.42	99.48 ± 0.09
	O	32.95 ± 8.10	28.19 ± 6.94	33.94 ± 8.84	37.28 ± 9.69	42.40 ± 10.53	469.49 ± 161.36	
	P	0.03 ± 0.01	0.01 ± 0.01	0.03 ± 0.02	0.04 ± 0.03	0.06 ± 0.03	40242.47 ± 195.66	
TransD	pub (U)					92.20	212.00	
	R	37.33 ± 0.52	4.31 ± 0.42	67.90 ± 0.93	81.01 ± 0.30	87.80 ± 0.33	460.00 ± 7.40	2.25 ± 0.04
TransE	pub (U)					89.20	251.00	
	R	37.04 ± 1.37	9.29 ± 1.83	60.28 ± 1.25	72.02 ± 0.75	81.51 ± 0.52	489.84 ± 42.13	2.39 ± 0.21
TransH	pub (U)					82.30	388.00	
	R	0.17 ± 0.17	0.08 ± 0.12	0.17 ± 0.20	0.21 ± 0.24	0.31 ± 0.29	19551.68 ± 166.54	95.55 ± 0.81
TransR	pub (ND)					92.00	225.00	
	R	0.24 ± 0.03	0.00 ± 0.01	0.22 ± 0.05	0.38 ± 0.05	0.63 ± 0.07	18882.20 ± 240.51	92.27 ± 1.18
TuckER	pub (ND)	95.30	94.90	95.50		95.80		
	R	94.89 ± 0.05	94.52 ± 0.05	95.17 ± 0.07	95.30 ± 0.07	95.50 ± 0.06	532.05 ± 45.91	2.60 ± 0.22

TABLE 11
Reproduction of Studies on WN18RR where **pub** refers to published results, **R** to results based on the realistic ranking, **O** to results based on the optimistic ranking, and **P** to results based on the pessimistic ranking. For published results, there are two additional rank types, **U** for undefined due to missing official implementation and **ND** for non-deterministic. We only show the results of the optimistic and pessimistic ranking in case they differ from the realistic ranking.

model		MRR (%)	Hits@1 (%)	Hits@3 (%)	Hits@5 (%)	Hits@10 (%)	MR	AMR (%)
ConvE	pub (ND)	43.00	40.00	44.00		52.00	4187.00	
	R	45.28 ± 0.13	41.93 ± 0.19	46.64 ± 0.25	49.07 ± 0.22	51.98 ± 0.24	5203.77 ± 129.07	25.67 ± 0.64
ConvKB	pub (O)	24.80				52.50	2554.00	
	R	0.34 ± 0.05	0.11 ± 0.04	0.27 ± 0.02	0.43 ± 0.06	0.63 ± 0.08	13905.99 ± 962.71	68.60 ± 4.75
QuatE[1]	pub (O)	48.10	43.60	50.00		56.40	3472.00	
	R	0.58 ± 0.05	0.38 ± 0.06	0.56 ± 0.08	0.66 ± 0.06	0.88 ± 0.09	20404.47 ± 196.81	100.65 ± 0.97
RotatE	pub (ND)	47.60	42.80	49.20		57.10	3340.00	
	R	49.39 ± 0.06	45.49 ± 0.12	51.03 ± 0.10	53.36 ± 0.15	57.05 ± 0.14	4046.79 ± 89.15	19.96 ± 0.44
TuckER	pub (ND)	47.00	44.30	48.20		52.60		
	R	47.62 ± 0.58	44.91 ± 0.62	48.81 ± 0.59	50.40 ± 0.58	52.80 ± 0.45	5646.84 ± 146.30	27.85 ± 0.72

[a] For MuRE, we obtained non-finite loss values while training on WN18RR with the setting defined in [24]. This might be explained by the fact that the specified learning rate of 50 is comparably large. In our benchmarking study, we show that we can outperform the published results with a different setting (Section 7.2).

TABLE 12
Model sizes in bytes for the best reported configurations studied for the the reproducibility study.

Dataset Model	FB15K	FB15K-237	WN18	WN18RR
ComplEx	26.1 MB	-	49.2 MB	-
ConvE	22.5 MB	20.3 MB	41.2 MB	40.9 MB
ConvKB	-	5.9 MB	-	8.2 MB
DistMult	6.5 MB	-	16.4 MB	-
HolE	9.8 MB	-	24.6 MB	-
KG2E	6.5 MB	-	16.4 MB	-
RotatE	130.4 MB	117.9 MB	163.8 MB	162.3 MB
SimplE	26.1 MB	-	65.5 MB	-
TransD	6.5 MB	-	16.4 MB	-
TransE	3.3 MB	-	3.3 MB	-
TransH	7.1 MB	-	8.2 MB	-
TransR	16.7 MB	-	8.4 MB	-
TuckER	46.1 MB	-	37.6 MB	-

ADDITIONAL RESULTS FROM BENCHMARKING STUDY

TABLE 13
Pareto-optimal models for FB15k237 regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
TuckER	BCEL	LCWA	yes	8.0 MiB	52.857
DistMult	CEL	LCWA	yes	3.7 MiB	47.387
TransE	SPL	LCWA	no	3.6 MiB	45.318
UM	MRL	sLCWA	no	3.5 MiB	3.432
UM	MRL	sLCWA	yes	3.5 MiB	3.305

TABLE 14
Pareto-optimal models for Kinships regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
TuckER	SPL	LCWA	yes	1.0 MiB	98.603
RotatE	MRL	sLCWA	yes	154.0 KiB	98.557
RotatE	MRL	sLCWA	no	129.0 KiB	98.324
SimplE	BCEL	LCWA	yes	77.0 KiB	97.765
ProjE	SPL	sLCWA	yes	39.3 KiB	96.648
ProjE	SPL	sLCWA	no	33.0 KiB	94.600
HolE	CEL	LCWA	no	32.2 KiB	88.873
UM	SPL	LCWA	yes	26.0 KiB	11.313
UM	SPL	sLCWA	yes	26.0 KiB	6.844

TABLE 15
Pareto-optimal models for WN18RR regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
RotatE	BCEL	LCWA	yes	79.3 MiB	60.089
RotatE	SPL	LCWA	yes	19.8 MiB	58.328
TuckER	CEL	LCWA	yes	11.9 MiB	56.088
MuRE	SPL	LCWA	no	10.2 MiB	55.489
TransH	MRL	sLCWA	no	9.9 MiB	48.170
UM	SPL	LCWA	yes	9.9 MiB	44.682
UM	SPL	sLCWA	yes	9.9 MiB	39.022

TABLE 16
Pareto-optimal models for YAGO310 regarding Model Bytes and Hits@10

Model	Loss	Training Approach	Inverse Relations	Model Bytes	Hits@10 (%)
MuRE	SPL	sLCWA	yes	61.1 MiB	66.851
ConvKB	NSSAL	sLCWA	no	30.1 MiB	52.921
DistMult	SPL	sLCWA	yes	30.1 MiB	50.562
TransE	BCEL	sLCWA	no	30.1 MiB	14.663

TABLE 17
Best configuration for each model in FB15k237

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	True	44.838
ConvE	BCEL	LCWA	True	49.212
ConvKB	SPL	sLCWA	False	32.261
DistMult	CEL	LCWA	True	47.387
ERMLP	BCEL	LCWA	True	45.100
HolE	CEL	LCWA	True	42.225
KG2E	SPL	LCWA	True	45.501
MuRE	BCEL	LCWA	True	47.199
NTN	SPL	sLCWA	False	20.342
ProjE	BCEL	LCWA	True	41.616
QuatE	CEL	LCWA	True	46.166
RESCAL	CEL	LCWA	True	46.460
RotatE	NSSAL	sLCWA	False	49.750
SE	NSSAL	sLCWA	True	39.427
SimplE	CEL	LCWA	True	40.307
TransD	MRL	sLCWA	True	41.856
TransE	MRL	sLCWA	False	46.423
TransH	MRL	sLCWA	False	35.295
TransR	CEL	LCWA	True	39.187
TuckER	BCEL	LCWA	True	52.857
UM	CEL	LCWA	False	8.024

TABLE 18
Best configuration for each model in Kinships

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	True	98.371
ConvE	NSSAL	sLCWA	True	98.557
ConvKB	NSSAL	sLCWA	True	97.067
DistMult	CEL	LCWA	True	93.529
ERMLP	SPL	sLCWA	True	97.486
HolE	CEL	LCWA	True	93.715
KG2E	MRL	sLCWA	True	91.853
MuRE	SPL	LCWA	True	95.019
NTN	BCEL	sLCWA	True	93.622
ProjE	SPL	sLCWA	True	96.648
QuatE	CEL	LCWA	True	98.184
RESCAL	SPL	sLCWA	True	97.719
RotatE	NSSAL	sLCWA	False	98.557
SE	NSSAL	sLCWA	True	98.324
SimplE	BCEL	sLCWA	False	98.277
TransD	CEL	LCWA	True	45.205
TransE	CEL	LCWA	True	92.877
TransH	CEL	LCWA	True	52.048
TransR	MRL	sLCWA	False	73.324
TuckER	SPL	LCWA	True	98.603
UM	SPL	LCWA	True	11.313

TABLE 19
Best configuration for each model in WN18RR

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	CEL	LCWA	False	53.745
ConvE	CEL	LCWA	True	56.327
ConvKB	NSSAL	sLCWA	True	42.083
DistMult	CEL	LCWA	True	52.616
ERMLP	SPL	sLCWA	True	47.657
HolE	CEL	LCWA	False	50.017
KG2E	SPL	LCWA	False	52.035
MuRE	SPL	LCWA	True	57.900
NTN	MRL	sLCWA	False	31.857
ProjE	CEL	LCWA	True	51.727
QuatE	CEL	LCWA	False	55.010
RESCAL	CEL	LCWA	False	53.916
RotatE	BCEL	LCWA	True	60.089
SE	SPL	sLCWA	False	45.486
SimplE	CEL	LCWA	True	50.889
TransD	MRL	sLCWA	False	46.546
TransE	SPL	LCWA	False	56.977
TransH	MRL	sLCWA	False	48.170
TransR	MRL	sLCWA	False	42.510
TuckER	CEL	LCWA	True	56.088
UM	SPL	LCWA	False	44.887

TABLE 20
Best configuration for each model in YAGO310

Model	Loss	Training Approach	Inverse Relations	Hits@10 (%)
ComplEx	BCEL	sLCWA	True	62.575
ConvKB	SPL	sLCWA	True	58.149
DistMult	BCEL	sLCWA	False	55.580
ERMLP	BCEL	sLCWA	True	58.531
HolE	BCEL	sLCWA	False	60.177
MuRE	SPL	sLCWA	True	66.851
QuatE	SPL	sLCWA	True	60.709
RESCAL	SPL	sLCWA	True	54.045
RotatE	NSSAL	sLCWA	True	63.077
SE	NSSAL	sLCWA	True	29.757
TransD	MRL	sLCWA	False	35.397
TransE	MRL	sLCWA	True	49.217

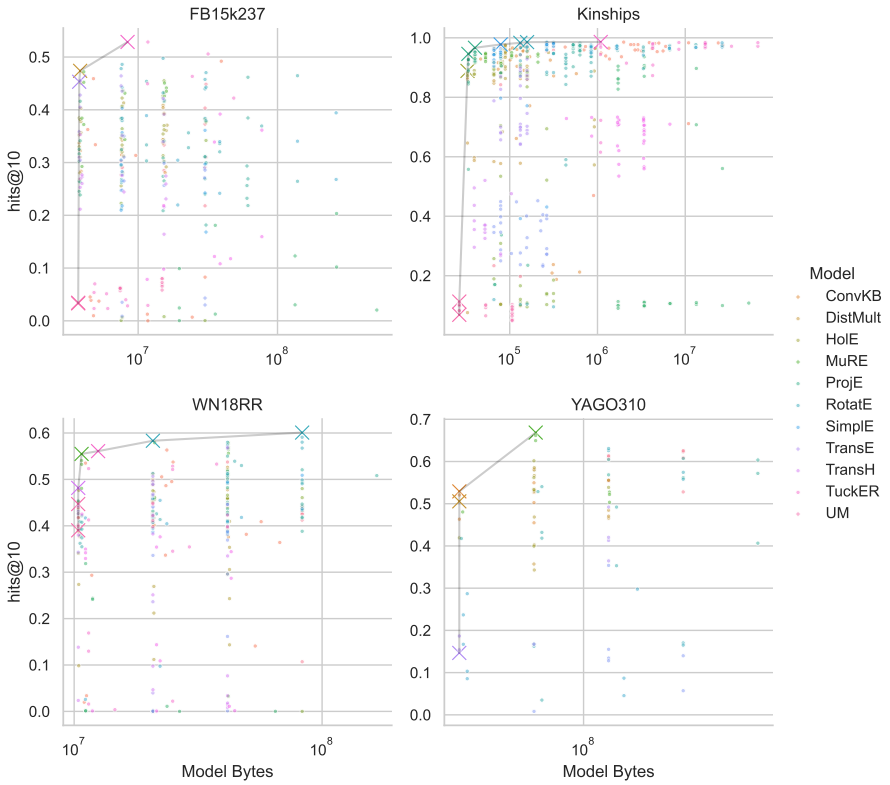


Fig. 28. Scatter plots comparing model size in number of bytes and model performance in terms of Hits@10 for all trained models on each dataset. The color indicates the model type, and the model size is shown on a logarithmic axis. Pareto-optimal models are highlighted by cross symbols. In general we only see a low correlation between model size and performance. A more thorough comparison can be found in Figures 4, 10, 16, and 22.

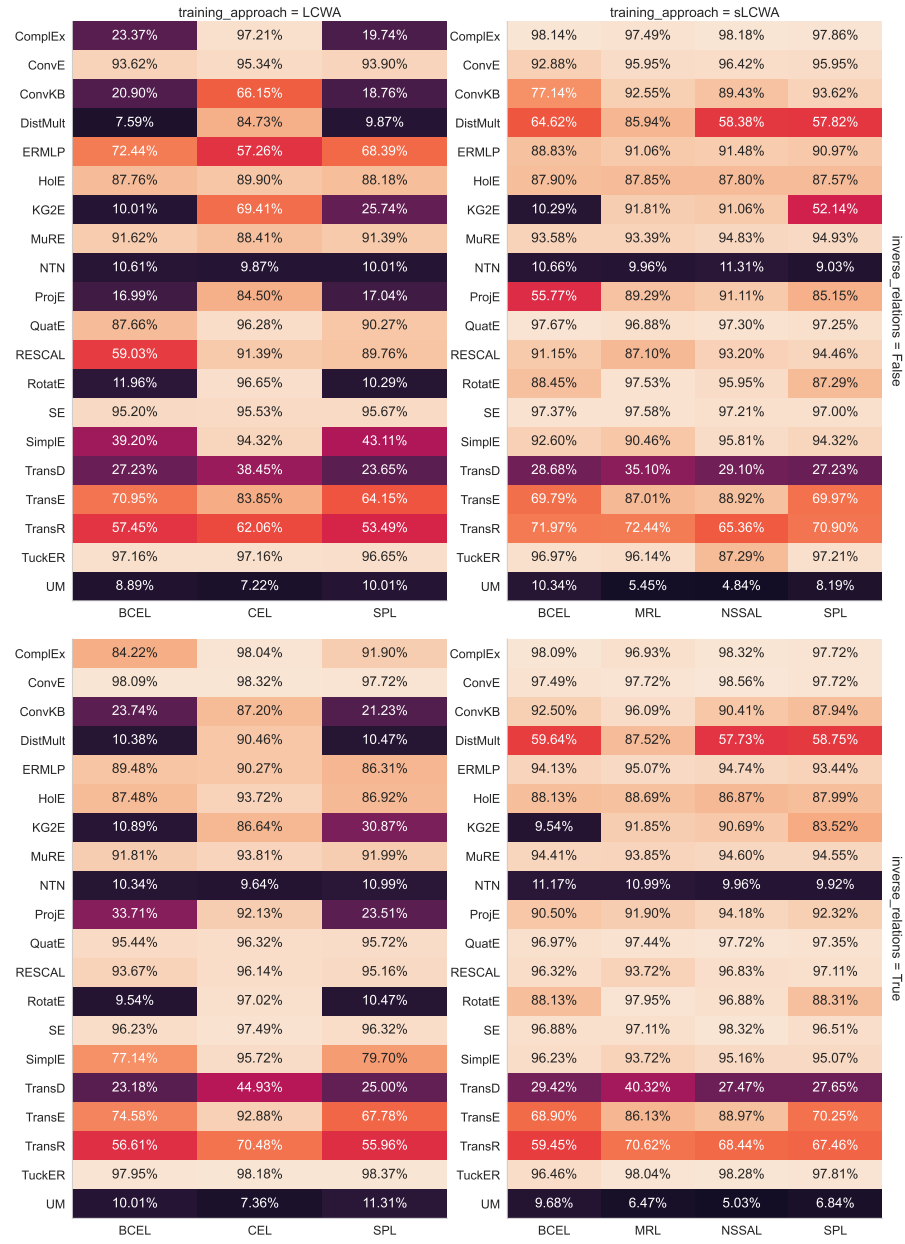


Fig. 29. Results for all configurations on Kinships based on Adadelta. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

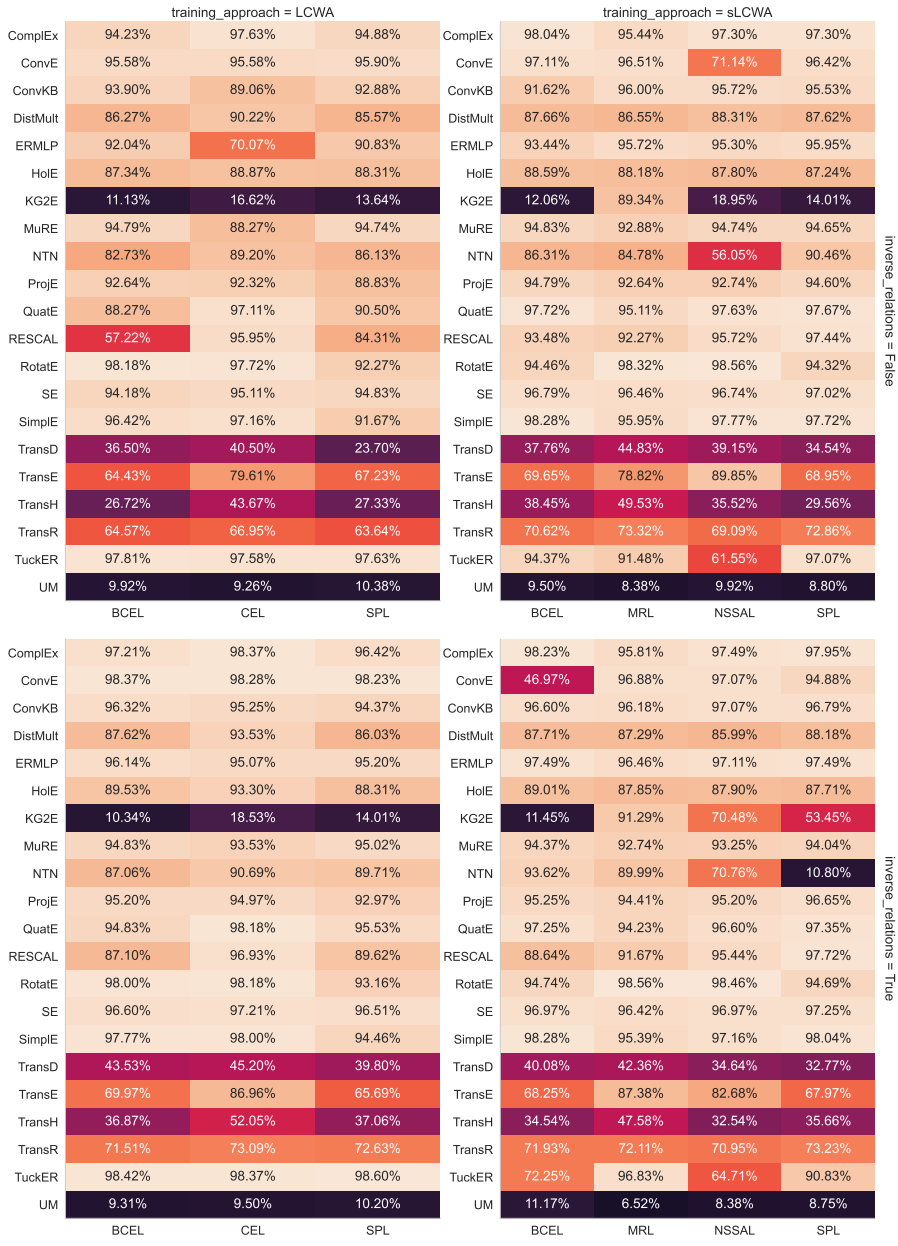


Fig. 30. Results for all configurations on Kinships based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

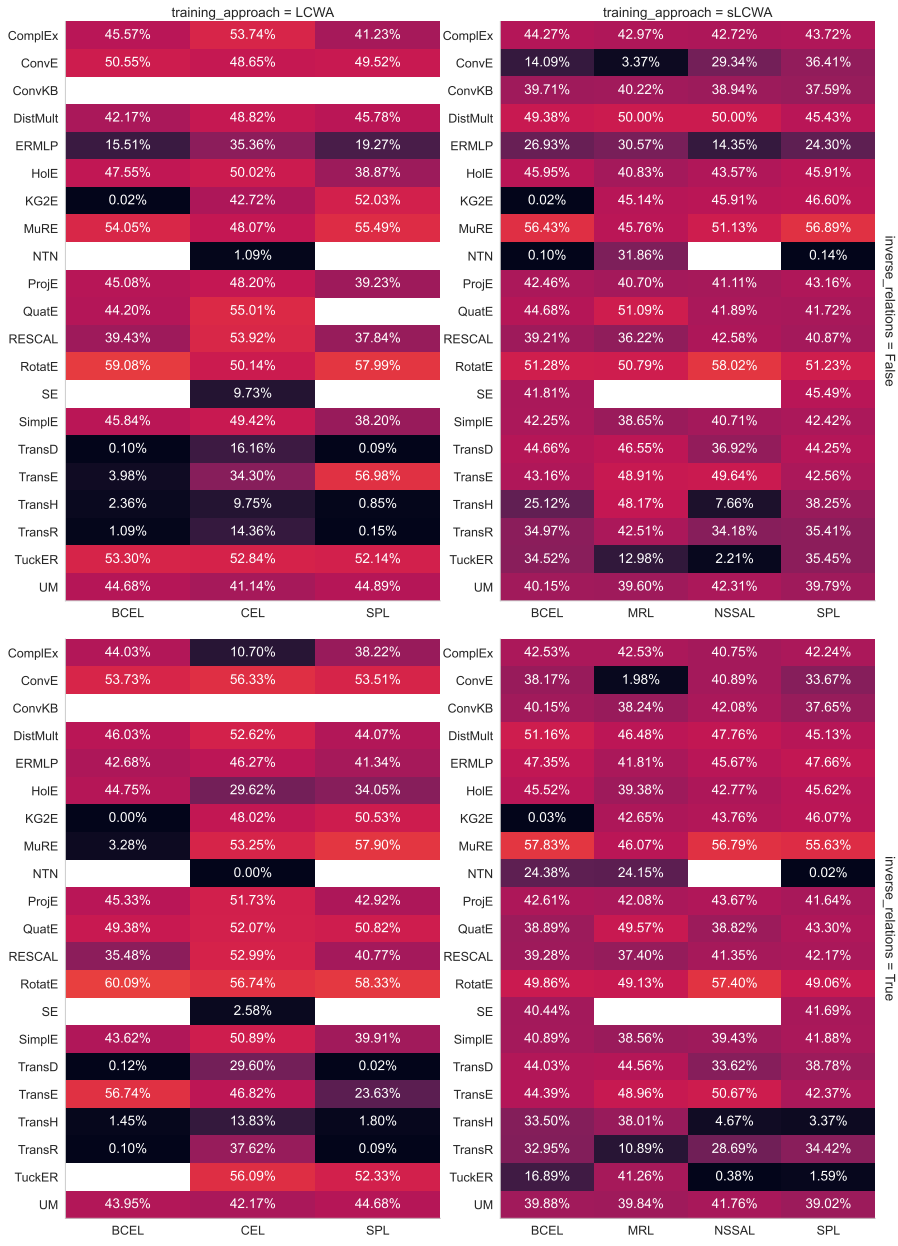


Fig. 31. Results for all configurations on WN18RR based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

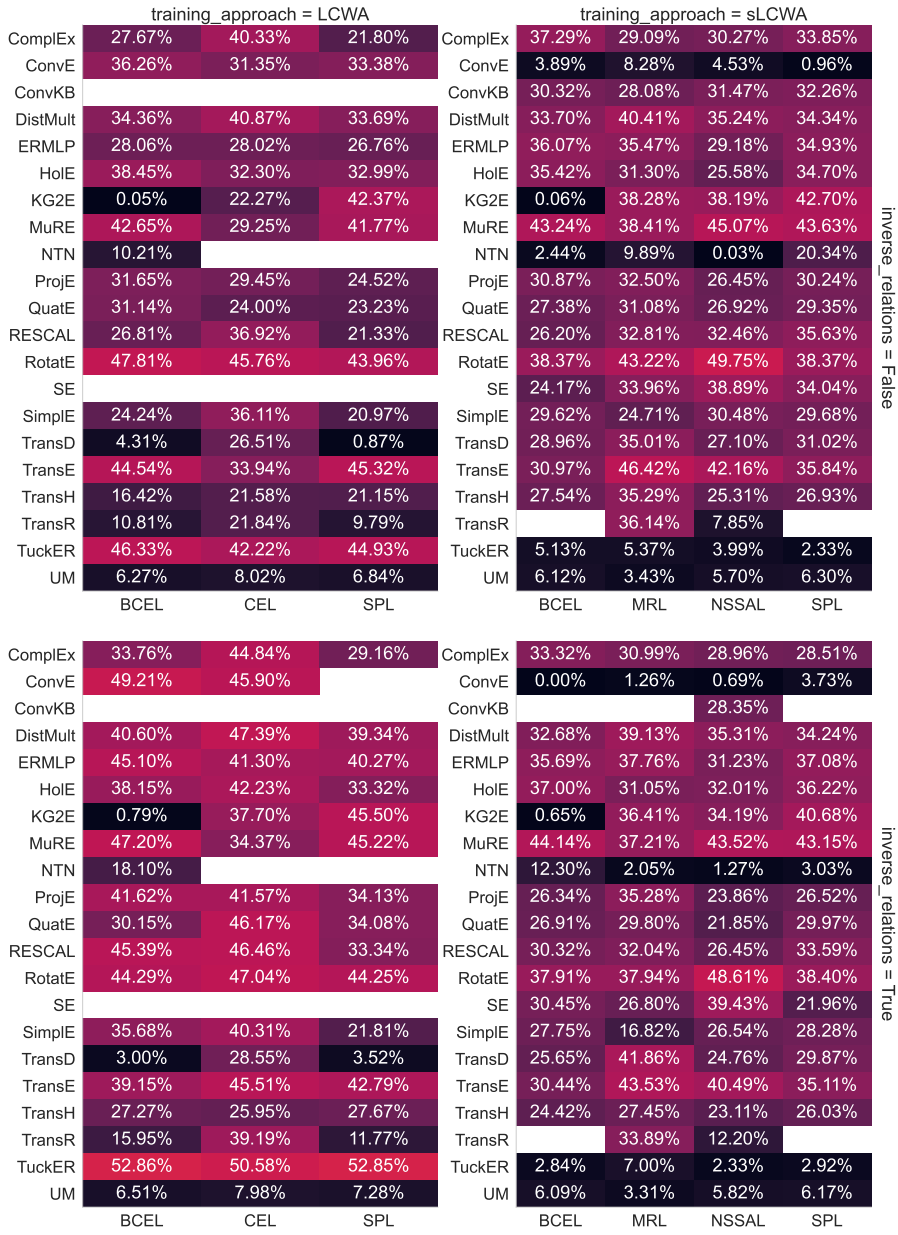


Fig. 32. Results for all configurations on FB15K-237 based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

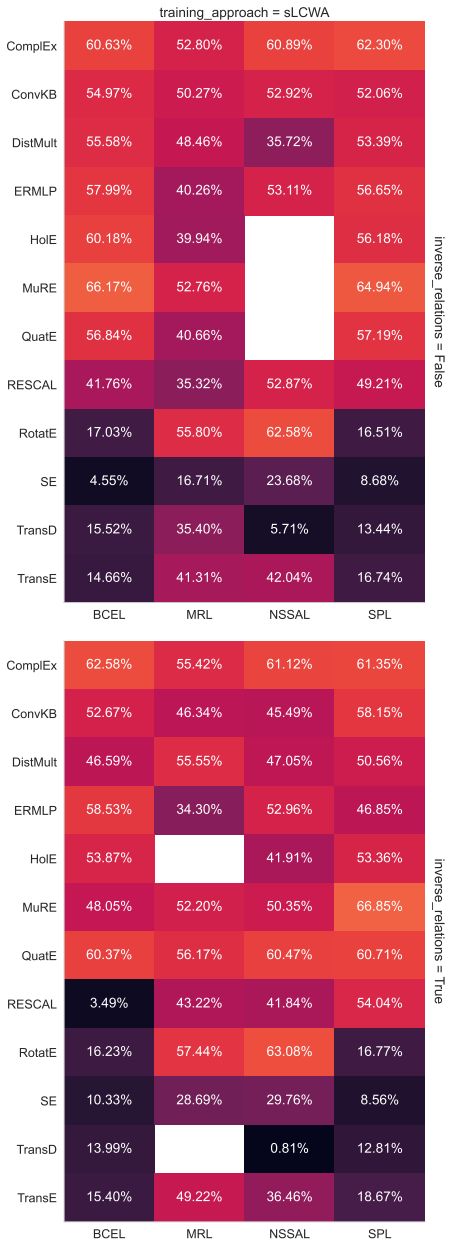


Fig. 33. Results for all configurations on YAGO3-10 based on Adam. **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

Fig. 34. Impact of the training approach on the performance for a fixed interaction model and loss function for the Kinships dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

Fig. 35. Impact of the training approach on the performance for a fixed interaction model and loss function for the WN18RR dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

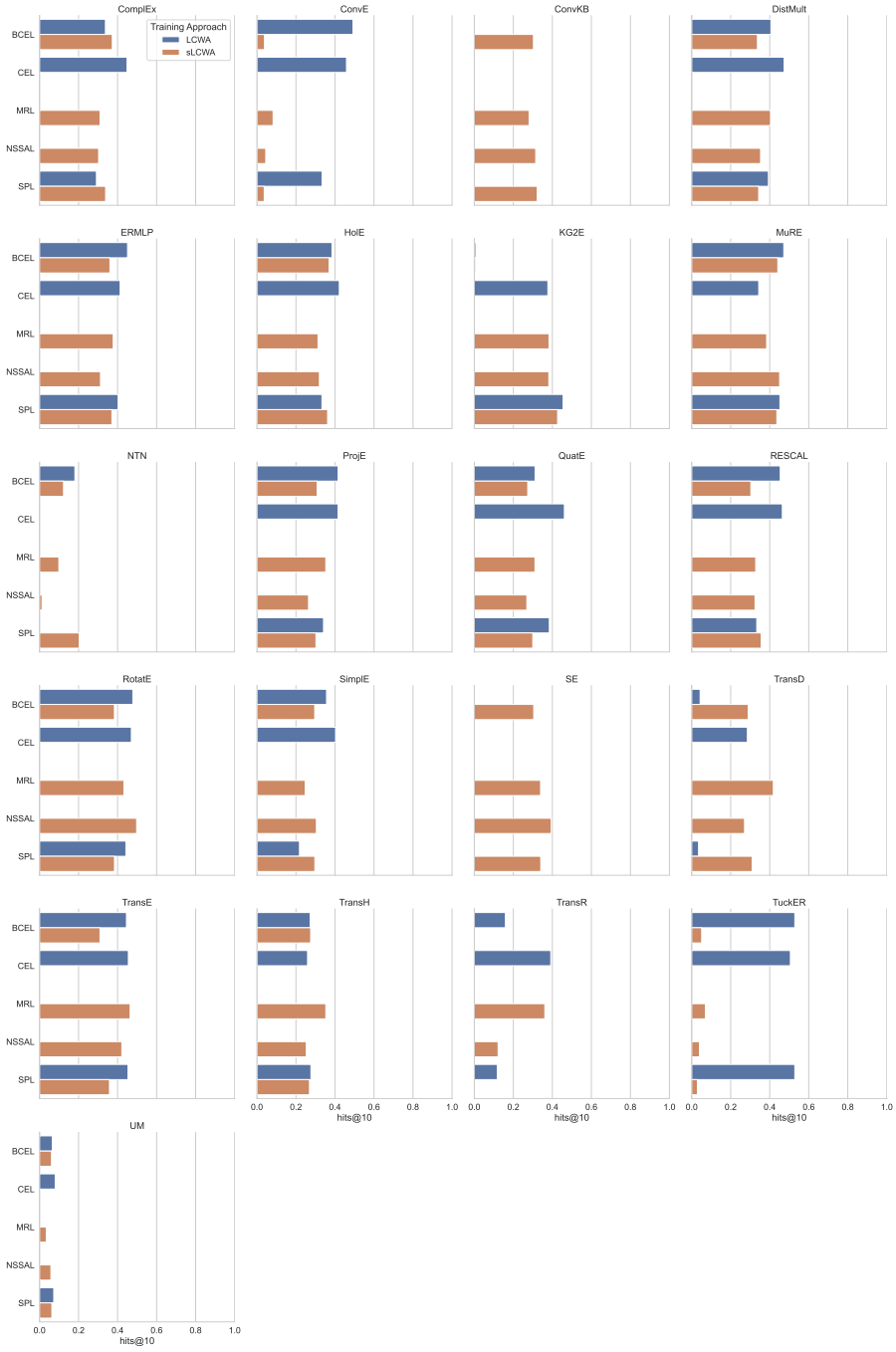


Fig. 36. Impact of the training approach on the performance for a fixed interaction model and loss function for the FB15K-237 dataset (results represent for each setting the best-performing configuration). **BCEL** refers to the binary cross entropy loss, **CEL** to the cross entropy loss, **MRL** to the margin ranking loss, **NSSAL** refers to the negative sampling self-adversarial loss, **SPL** to the softplus loss, **LCWA** to the local closed world assumption training approach and **sLCWA** to the stochastic local closed world assumption training approach.

Appendix B

PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

---

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 Embeddings. *Journal of Machine Learning Research*, 22(82):1–6, 2021. *Equal contribution

Journal of Machine Learning Research 22 (2021) 1-6 Submitted 7/20; Revised 2/21; Published 3/21

PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings

Mehdi Ali* MEHDI.ALI@CS.UNI-BONN.DE
Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

Max Berrendorf* BERRENDORF@DBS.IFI.LMU.DE
Ludwig-Maximilians-Universität München

Charles Tapley Hoyt* CHARLES.HOYT@ENVEDATX.COM
Enveda Biosciences

Laurent Vermue* LAUVE@DTU.DK
Technical University of Denmark

Sahand Sharifzadeh SHARIFZADEH@DBS.IFI.LMU.DE
Ludwig-Maximilians-Universität München

Volker Tresp VOLKER.TRESP@SIEMENS.COM
Ludwig-Maximilians-Universität München & Siemens AG

Jens Lehmann JENS.LEHMANN@CS.UNI-BONN.DE
Smart Data Analytics Group, University of Bonn & Fraunhofer IAIS

Editor: Antti Honkela

Abstract

Recently, knowledge graph embeddings (KGEs) have received significant attention, and several software libraries have been developed for training and evaluation. While each of them addresses specific needs, we report on a community effort to a re-design and re-implementation of PyKEEN, one of the early KGE libraries. PyKEEN 1.0 enables users to compose knowledge graph embedding models based on a wide range of interaction models, training approaches, loss functions, and permits the explicit modeling of inverse relations. It allows users to measure each component's influence individually on the model's performance. Besides, an automatic memory optimization has been realized in order to optimally exploit the provided hardware. Through the integration of Optuna, extensive hyper-parameter optimization (HPO) functionalities are provided.

Keywords: Knowledge Graphs, Knowledge Graph Embeddings, Relational Learning

1. Introduction

Knowledge graphs (KGs) encode knowledge as a set of triples $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where $\mathcal{E}$ denotes the set of entities and $\mathcal{R}$ the set of relations. Knowledge graph embedding models (KGEMs) learn representations for entities and relations of KGs in vector spaces while preserving the graph structure. The learned embeddings can support machine learning tasks such as entity clustering, link prediction, entity disambiguation, as well as downstream tasks such

*Equal contribution.

as question answering and item recommendation (Nickel et al., 2015; Wang et al., 2017; Ruffinelli et al., 2020; Kazemi et al., 2020).

Most publications of KGEMs are accompanied by reference implementations, but they are seldomly written for reusability or maintained. Existing software packages that provide implementations for different KGEMs usually lack composability: model architectures (or interaction models), training approaches, loss functions, and the usage of explicit inverse relations cannot arbitrarily be combined. The full composability of KGEMs is fundamental for assessing their performance because it allows the assessment of individual components and not solely the sum of differences in published approaches (Ruffinelli et al., 2020). In most previous libraries, only limited functionalities are provided, e.g., a small number of KGEMs are supported, or functionalities such as hyper-parameter optimization (HPO) are missing. For instance, in PyKEEN (Ali et al., 2019a,b), one of the early software packages for KGEMs, models can only be trained under the stochastic local closed-world approach, the evaluation procedure was too slow for larger KGs, and it was designed to be mainly used through a command-line interface rather than programmatically, in order to facilitate its usage for non-experts. This motivated the development of a reusable software package comprising several KGEMs and related methodologies that is entirely configurable.

Here, we present PyKEEN (Python KnowlEdge EmbeddiNgs) 1.0, a community effort in which PyKEEN has been re-designed and re-implemented from scratch to overcome the mentioned limitations, to make models entirely configurable, and to extend it with more interaction models and other components.

2. System Description

In PyKEEN 1.0, a KGEM is considered as a composition of four components that can flexibly be combined: an interaction model (or model architecture), a loss function, a training approach, and the usage of inverse relations. PyKEEN 1.0 currently supports 23 interaction models, seven loss functions, four regularizers, two training approaches, HPO, six evaluation metrics, and 21 built-in benchmarking datasets. It can readily import additional datasets that have been pre-stratified into train/test/evaluation and generate appropriate splits for unstratified datasets. Additionally, we implemented an automatic memory optimization that ensures that the available memory is best utilized.

Composable KGEMs To ensure the composability of KGEMs, the interaction models, loss functions, and training approaches are separated from each other and implemented as independent submodules, whereas the modeling of inverse relations is handled by the interaction models. Our modules can be arbitrarily replaced because we ensured through inheritance that all interaction models, loss functions, and training approaches follow unified APIs, which are defined by *pykeen.model.Model*, *pykeen.loss.Loss*, and *pykeen.training.TrainingLoop*. Currently, we provide implementations of 23 interaction models, the most common loss functions used for training KGEMs including the *binary-cross entropy*, *cross entropy*, *mean square error*, *negative-sampling self-adversarial loss*, and the *softplus loss*, as well as the *local closed-world assumption* (also referred as *KvsAll*) and the *stochastic local closed-world assumption* training approach (also refereed as *NegSamp*) (Ruffinelli et al., 2020). In PyKEEN, each interaction model can be trained based on both approaches. To enable users to investigate the effect of explicitly modeling

inverse relations (Lacroix et al., 2018; Kazemi and Poole, 2018) on the model's performance, each model can be trained with explicit inverse relations in PyKEEN 1.0, i.e., for each relation $r \in \mathcal{R}$ an inverse relation r_{inv} is introduced, and the task of predicting the head entity of a (r, t)-pair becomes the task of predicting the tail entity of the corresponding inverse pair (t, r_{inv}).

To facilitate the composition of KGE models for non-experts, we provide the pykeen.pipeline.pipeline() functions, which provides a high-level entry point into the functionalities of PyKEEN. Users define the components to be used, and the pipeline ensures the correct composition of the KGEM and the correct composition of the training and evaluation workflow.

Evaluation KGEMs are usually evaluated on the task of link prediction. Given (h, r) (or (r, t)), all possible entities $\mathcal{E}$ are considered as tail (or head) and ranked according to the KGEMs interaction model. The individual ranks are commonly aggregated to mean rank, mean reciprocal rank, and hits@k. However, these metrics have been realized differently throughout the literature based on different definitions of the rank, leading to difficulties in reproducibility and comparability (Sun et al., 2019). The three most common rank definitions are the *average rank*, *optimistic rank*, and *pessimistic rank*. In PyKEEN 1.0, we explicitly compute the aggregation metrics for all common rank definitions, *average*, *optimistic*, and *pessimistic*, allowing inspection of differences between them. This can help to reveal cases where the model predicts exactly equal scores for many different triples, which is usually an undesired behavior. In addition, we support the recently proposed *adjusted mean rank* (Berrendorf et al., 2020), which allows the comparison of results across differently sized datasets, as well as offering an interface to use all metrics implemented in scikit-learn (Pedregosa et al., 2011), including AUC-PR and AUC-ROC.

Automatic Memory Optimization Allowing high computational throughput, while ensuring that the available hardware memory is not exceeded during training and evaluation, requires the knowledge of the maximum possible training and evaluation batch size for the current model configuration. However, determining the training and evaluation batch sizes is a tedious process, and not feasible when a large set of heterogeneous experiments are run. Therefore, we implemented an automatic memory optimization step that computes the maximum possible training and evaluation batch sizes for the current model configuration and available hardware before the actual experiment starts. If the user-provided batch size is too large for the used hardware, the automatic memory optimization determines the maximum sub-batch size for the training.

Extensibility Because we defined a uniform API for each interaction model, any new model can be integrated by following the API of the existing models (*pykeen.models*). Similarly, the remaining components, e.g., regularizers, and negative samplers follow a unified API, so that new modules can be smoothly integrated.

Community Standards PyKEEN 1.0 relies on several community-oriented tools to ensure it is accessible, reusable, reproducible, and maintainable. It is implemented for Python 3.7+ using the PyTorch package. It comes with a suite of thorough unit tests that are automated with PyTest, Tox, run in a continuous integration setting on GitHub Actions, and are tracked over time using `codecov.io`. Code quality is ensured with flake8 and careful

Library	AMO	Models	HPO	ES	Evaluation Metrics	Set TA	Set Inv. Rels.	Set Loss Fct.	MGS	DTR
AmpliGraph (Costabello et al., 2019)	-	6	✓	✓	3	-	✓	✓	-	-
DGL-KE (Zheng et al., 2020)	-	6	-	-	3	-	-	✓	✓	✓
GraphVite (Zhu et al., 2019)	-	6	-	-	4	-	-	-	✓	-
LibKGE (Broscheit et al., 2020)	-	10	✓	✓	3*	✓	✓	✓	-	-
OpenKE (Han et al., 2018)	-	11	-	-	3	-	-	✓	-	-
PyTorch-BigGraph (Lerer et al., 2019)	-	4	-	-	4	-	-	✓	✓	✓
Pykg2vec (Yu et al., 2019)	-	18	✓	✓	2	-	-	-	-	-
PyKEEN (Ali et al., 2019b)	-	10	✓	-	2	-	-	-	-	-
PyKEEN 1.0	✓	**23**	✓	✓	**6***	✓	✓	✓	-	-

Table 1: An overview of the functionalities (determined July 2020) of PyKEEN 1.0 and similar libraries. **AMO** refers to automatic memory optimization, **ES** to early stopping, ***** indicates that ranking metrics are computed for different definitions of the rank, **Set TA** refers to interchanging the training approach, **Set Inv. Rels.** to the explicit modeling of inverse relations, **MGS** to multi-GPU support, i.e., training a single model across several GPUs, and **DTR** to distributed training.

application of the GitHub Flow development workflow. Documentation is quality checked by doc8, built with Sphinx, and hosted on `ReadTheDocs.org`.

3. Comparison to Related Software

Table 1 depicts the most popular KGE frameworks and their features. It shows that PyKEEN 1.0, in comparison with related software packages, emphasizes on both, full composability of KGEMs and extensive functionalities, i.e., a large number of supported interaction models, and extensive evaluation (several metrics are supported) and HPO functionalities. Concerning the evaluation metrics, PyKEEN and LibKGE are the only libraries that compute the ranking metrics (i.e., *mean rank* and *hits@k*) for different definitions of the rank, which ensures that undesired cases are detected in which the model predicts equal scores for many triples. Finally, PyKEEN 1.0 is the only library that performs an automatic memory optimization that ensures that the memory is not exceeded during training and evaluation. GraphVite, DGL-KE, and PyTorch-BibGraph focus on scalability, i.e., they provide support for multi-GPU/CPU or/and distributed training, but focus less on compositionality and extensibility. For instance, PyTorch-BigGraph supports only a small number of interaction models that follow specific computation blocks.

4. Availability and Maintenance

PyKEEN 1.0 is publicly available under the MIT License at `https://github.com/pykeen/pykeen`, and is distributed through the Python Package Index. It will be maintained by the core developer team that is supported by the Smart Data Analytics research group (University of Bonn), Fraunhofer IAIS, Munich Center for Machine Learning (MCML),

Siemens, and the Technical University of Denmark (section for Cognitive Systems and section for Statistics and Data Analysis). The project is funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and Grant No. 01IS18050D (project MLWin) as well as the Innovation Fund Denmark with the Danish Center for Big Data Analytics driven Innovation (DABAI) which ensures the maintenance of the project in the next years.

References

Mehdi Ali, Charles Tapley Hoyt, Daniel Domingo-Fernández, Jens Lehmann, and Hajira Jabeen. Biokeen: a library for learning and evaluating biological knowledge graph embeddings. *Bioinformatics*, 35(18):3538–3540, 2019a.

Mehdi Ali, Hajira Jabeen, Charles Tapley Hoyt, and Jens Lehmann. The keen universe. In *International Semantic Web Conference*, pages 3–18. Springer, 2019b.

Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'20)*. IEEE, 2020.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. Libkge - A knowledge graph embedding library for reproducible research. In *EMNLP (Demos)*, pages 165–174. Association for Computational Linguistics, 2020.

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`.

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.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295, 2018.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*, 2018.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA, 2019.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*, 2019.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

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.

Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. Dgl-ke: Training knowledge graph embeddings at scale. *arXiv preprint arXiv:2004.08532*, 2020.

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

Appendix C

Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods

---

Max Berrendorf, Evgeniy Faerman, **Laurent Vermue**, and Volker Tresp. Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 371–374, 2020. doi: 10.1109/WIIAT50758.2020.00053

2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) | 978-1-6654-1924-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/WIIAT50758.2020.00053

Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods

Max Berrendorf*, Evgeniy Faerman*, Laurent Vermue† and Volker Tresp*‡
*Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf,faerman}@dbs.ifi.lmu.de
†Technical University of Denmark, Kongens Lyngby, Denmark
lauve@dtu.dk
‡Siemens AG, Munich, Germany
volker.tresp@siemens.com

***Abstract*—In this work, we take a closer look at the evaluation of two families of methods for enriching information from knowledge graphs: Link Prediction and Entity Alignment. In the current experimental setting, multiple different scores are employed to assess different aspects of model performance. We analyze the informativeness of these evaluation measures and identify several shortcomings. In particular, we demonstrate that all existing scores can hardly be used to compare results across different datasets. Therefore, we propose adjustments to the evaluation and demonstrate empirically how this supports a fair, comparable, and interpretable assessment of model performance.**

I. Introduction

Information retrieval systems often require information organized in an easily accessible and interpretable structure. Frequently, Knowledge Graphs (KGs) are used as an information source [9]. Consequently, the successful application of new information retrieval algorithms often depends on the completeness and quality of the information in KGs. *Link Prediction* (LP) [18] and *Entity Alignment* (EA) [4] are two disciplines with the goal to enrich information in KGs. LP makes use of existing information in a single KG by materializing latent links. The goal of EA is to align entities in different KGs, which facilitates the transfer of information between both or a fusion of multiple KGs to a single knowledgebase. Both disciplines work by assigning scores to potential candidates: LP methods compute scores for the facts in question at inference time and EA methods assign scores to candidate alignment pairs. Simple thresholding, or also more advanced assignment methods [15] for EA, can make use of these scores to predict new links or alignments.

During the evaluation, both, LP and EA, evaluate how the "true" entity is *ranked* relative to other candidate entities. Given a *rank* for each test instance, various metrics exist to obtain a single number quantifying the overall performance of an approach. In this paper, we analyze the whole evaluation procedure and make the following contributions:

1) We describe the intuition behind current aggregation scores and argue that they do not always provide a complete picture of the model performance. We show that this is an actual problem in the current evaluation setting, which sometimes may lead to wrong conclusions.

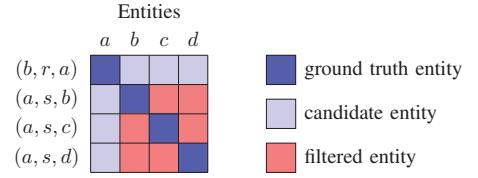


Fig. 1: Visualization of candidate sets for the filtered evaluation setting for link prediction (right side / tail prediction) on a toy example with triples $\{(b, r, a), (a, s, b), (a, s, c), (a, s, d)\}$. Depending on the presence of other triples with shared head-relation pairs, the number of considered candidate entities varies, and consequently the maximum possible rank. In this example, there are three triples starting with (a, s). When, e.g., triple (a, s, b) is evaluated c and d are ignored. Since only two entities remain, the rank cannot be larger than two.

2) We propose a new (adapted) evaluation score overcoming the problems of existing metrics.
3) We empirically demonstrate its usefulness for comparing Link Prediction results across datasets.

The remainder of the paper is structured as follows: In Section II, we discuss the rank definition and aggregation metrics summarizing individual ranks. In Section III, we point out the problems of current evaluation and introduce an adapted aggregation metric, which circumvents the shortcomings of existing aggregations. Afterwards, in Section IV, we discuss related work. Finally, in Section V, we demonstrate empirically the effects of our adaptations and conclude in Section VI.

II. Evaluation Framework

A. Rank for Link Prediction and Entity Alignment

a) Link Prediction: Let a single knowledge graph be represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples. For the task of LP a set of given triples is usually divided in $\mathcal{T}_{train} \subseteq \mathcal{T}$ and $\mathcal{T}_{test} = \mathcal{T} \setminus \mathcal{T}_{train}$, where $\mathcal{T}_{test}$ is used to assess the model performance. A common evaluation protocol is to use every triple $(h, r, t) \in \mathcal{T}_{test}$, and perform left-side

978-1-6654-1924-6/20/$31.00 ©2020 IEEE
DOI 10.1109/WIIAT50758.2020.00053

and right-side prediction. For the right-side prediction, the score for every triple $\{(h, r, e) \mid e \in \mathcal{E}\}$ is computed and the entities e are sorted in decreasing order by the predicted scores. The *rank* of the "true" entity t is computed as the index in the resulting sorted list. The left-side prediction follows analogously. The final rank of the triple is computed as an average over both ranks, left-side and right-side. To account for the possibility of multiple existing links for a given head-relation / relation-tail pair, the filtered evaluation setting was introduced [5]: When scoring tail entities for a triple (h, r, t), all other entities $t \neq t' \in \mathcal{E}$ with triples $(h, r, t') \in (\mathcal{T}_{train} \cup \mathcal{T}_{test})$ are ignored, cf. Figure 1. Therefore, the performance does not decrease when other entities are scored higher than the currently considered one, as long as they are also true. The filtered evaluation protocol is the quasi-standard for link prediction on knowledge graphs, and unfiltered scores are rarely reported.

b) Entity Alignment: For this task, there are two knowledge graphs $\mathcal{G}_L = (\mathcal{E}_L, \mathcal{R}_L, \mathcal{T}_L)$ and $\mathcal{G}_R = (\mathcal{E}_R, \mathcal{R}_R, \mathcal{T}_R)$, and a set of aligned entities $\mathcal{A} \subseteq \mathcal{E}_L \times \mathcal{E}_R$. Analogous to the previous evaluation setting, the set of alignments is divided into $\mathcal{A}_{train} \subseteq \mathcal{A}$ and $\mathcal{A}_{test} = \mathcal{A} \setminus \mathcal{A}_{train}$. The common evaluation scheme [6], [7], [11], [21], [26]–[28], [30], [33], [35], [37], [38] now computes scores for every candidate pair $\{(a_L, e_R) \mid e_R \in \mathcal{E}_R, \exists a'_L \in \mathcal{E}_L : (a'_L, e_R) \in \mathcal{A}_{test}\}$, and determines the rank of the "true" score of (a_L, a_R). The right-side prediction is defined correspondingly. Notice, that only those entities are considered for which there exists an aligned entity from the other graph in the *test* part of the alignment.

B. Overall metrics

Given the set of individual rank scores $\mathcal{I}$, the following scores are commonly used as aggregation.

1) Hits @ k: The *Hits @ k* (H@k) score describes the fraction of hits, or fraction of instances, for which the "true" entity appears under the first k entities in the sorted list:

$$H@k := \frac{|\{r \in \mathcal{I} \mid r \leq k\}|}{|\mathcal{I}|}. \tag{1}$$

In the context of information retrieval, this metric is also known as Precision@k. One of the advantages of this metric is that it is easily interpretable. Since for many applications only the first outputs are taken into account, it can help to directly assess the method's applicability to the use-case. However, this metric does not distinguish the cases, where the rank is larger than k. Thus, the ranks $k + 1$ and $k + d$, where $d \gg 1$, have the same effect on the final score. Therefore, it is less suitable for the comparison of different models.

2) Mean Rank: The *mean rank* (MR) computes the mean over all individual ranks:

$$MR := \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r. \tag{2}$$

The advantage of the MR score is that it is sensitive to any model performance changes. If the rank on the same evaluation set becomes better on average, the improvement is always reflected by the MR score. While the MR is still interpretable, it is necessary to keep the size of the candidate set in mind to assess the model performance and interpret its value: A MR of 10 might indicate strong performance for a candidate set size of 1,000,000, but for a candidate set of only 20 candidates it equal to the expected performance of a model with random scorings.

3) MRR: The *mean reciprocal rank* is still often reported along with other scores. It is defined as

$$MRR := \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} \frac{1}{r}. \tag{3}$$

While the MRR is less sensitive to outliers and has the property to be bounded in the range $(0, 1]$, it was shown that this metric has serious flaws and therefore should not be relied upon [10]. However, especially in LP codebases, the MRR is often used for early stopping. Presumably, the main reason for that is the behavior of the reciprocal function: While the Hits@k score ignores change among high rank values completely, MR values changes uniformly among the full value range. The MRR score, in contrast, is more affected by changes of low rank values than high ones, but it does not completely disregard them. Therefore, it can be considered as soft a version of Hits@k.

III. Our Evaluation Approach

A. Adjusted Mean Rank

While the *H@k* score enables assessments of the model's suitability for a use-case, the *MR* allows a more fine-grained comparison between different models. Both metrics are necessary to get the entire picture of the model performance, e.g. the evaluation in [31] demonstrates, that an excellent *H@k* score does not necessarily coincide with a good *MR*. However, since the *MR* score denotes the absolute position, it is not easily interpretable. Therefore, the comparison between experiments with different sizes of candidate sets is not easily possible with implications for the evaluation of both tasks.

a) Link Prediction: The results on datasets with a different number of entities are not directly comparable. However, comparability of performance on different datasets is important, for example, to assess the task complexity, choose benchmarks, or investigate model generalization. For instance, surprisingly good test scores can be an indication for test leakage, see e.g. [29]. Intuitively, the number of candidates is an important factor directly affecting the task complexity, while it is not the only factor.

b) Entity Alignment: While the comparison of the performance on different datasets is also difficult for EA, there is the additional problem that only those entities are considered as candidates, which occur in at least one *test* alignment. Therefore, the number of candidates depends on the size of the evaluation alignment set. Thus, results on the same dataset are not comparable for different train/test splits or between train and test sets. This can lead to various misinterpretations of results. For instance, in [17], [33], the authors show an experiment where they increase the training size step-wise

and evaluate the model on the rest of the data. Based on the score improvement, they conclude that the model benefits from additional training data. While this claim can still be true, we argue that another evaluation is necessary to support it. The necessary condition for such an evaluation is either independence on candidate set size or the same candidate set for all experiments. One possible solution would be to use all entities in the KG as candidates analogous to LP. However, this still would leave us with the unresolved problem of performance comparison across datasets. Therefore, we propose an adjustment to the *MR* score that assesses the model performance independently of the candidate set size.

c) Adjusted Mean Rank Index: Since we are interested in evaluating model performance, we start with the mean rank as our starting point. To compute a rank in LP and EA evaluation, we are given a list of scores $\mathcal{S} = [\beta_1, \dots, \beta_C]$ for each test instance with $|\mathcal{C}| = C$, where $\mathcal{C}$ is a set of candidates. We denote the score of the "true" entity as α, and its position in the decreasingly sorted list as $rank(S, \alpha)$. If $\alpha, \beta_1, \dots, \beta_C$ are i.i.d and drawn at random, and therefore the element can appear at any position with the same probability, the expected rank is also the middle of the sorted array:

$$\mathbb{E}[rank(\mathcal{S}, \alpha)] = \frac{1}{n}\sum_{i=1}^{|\mathcal{S}|} i = \frac{1}{2}(|\mathcal{S}| + 1) \tag{4}$$

Inspired by the Adjusted Rand Index (ARI) [22], we aim to adjust it for chance. Therefore, we compute the *expected mean rank* following the assumption that the individual ranks are independent:

$$\begin{aligned}\mathbb{E}[MR] &\overset{(2)}{=} \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} rank(\mathcal{S}_i, \alpha)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[rank(\mathcal{S}_i, \alpha)] \\ &\overset{(4)}{=} \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathcal{S}_i| + 1}{2} = \frac{1}{2n}\sum_{i=1}^{n}(|\mathcal{S}_i| + 1)\end{aligned}$$

Now, we define the *adjusted mean rank* as the MR divided by its expected value:

$$AMR = \frac{MR}{\mathbb{E}[MR]} = \frac{2\sum_{i=1}^{n} r_i}{\sum_{i=1}^{n}(|\mathcal{S}_i| + 1)}$$

Finally, to obtain a measure where 1 corresponds to optimal performance, we transform the adjusted mean rank to *adjusted mean rank index* (AMRI) as follows:

$$AMRI = 1 - \frac{MR - 1}{\mathbb{E}[MR - 1]} = \frac{2\sum_{i=1}^{n}(r_i - 1)}{\sum_{i=1}^{n}(|\mathcal{S}_i|)} \tag{5}$$

Since $r_i - 1 \leq |\mathcal{S}| - 1$ the AMR has a bounded value range of $[-1, 1]$. A value of 1 corresponds to optimal performance where each individual rank is 1. A value of 0 indicates model performance similar to a model assigning random scores, or equal score to every candidate. The value is negative if the model performs worse than the constant-score model.

IV. Related Work

A special property of the *ranking* evaluation is that the candidate scores for each test instance are only required to be comparable within a single candidate set. The scores of candidates for another test instance may have a different value range, but since they are not compared with the candidates of other test instances, this does not affect the results. Therefore, ranking evaluation is appropriate for a setting where a human can evaluate model proposals. If, on the other hand, the decision has to be made automatically, e.g. using a fixed threshold, the *classification* setting is more appropriate. In the following, we review related approaches and demonstrate that classification and ranking evaluations are used interchangeably.

A. Triple Classification

In the LP task, we are given a pair of a head/tail entity $e \in \mathcal{E}$ and relation $r \in \mathcal{R}$, and rank a set of possible tail/head entities $e' \in \mathcal{E}$ according to the plausibility of the triple (e, r, e') / (e', r, e). In contrast, for *triple classification*, we aim at classifying whether a triple is true or false irrespective of the plausibility of other triples [12], [16], [25], [32], [34]. Consequently, a global threshold for the score of triples is required for the classification decision. If the threshold is chosen manually, classification metrics such as accuracy or F_1-measure can be used. Otherwise, the area under the precision-recall curve (PR-AUC), or receiver-operator curve (ROC-AUC) are used to summarize the performance over all possible decision thresholds. Link prediction and triple classification are sometimes evaluated alongside to demonstrate the effectiveness of novel knowledge graph embedding models across different tasks [14], [19], [25].

B. Ontology Matching

Ontology matching or *instance matching* is closely related to EA. Here we seek correspondences between instances of different ontologies based on different data properties of the instances. In contrast to EA, the vast majority of ontology matching approaches are unsupervised, i.e. there are no training alignments, but the instance features that are used for matching [3], [13], [20], [24]. The similarity is often fixed, e.g. to TF-IDF, and the methods optimize the matching process by pruning the candidate match space and selecting subsets of properties used for matching. Ontology matching approaches are evaluated in a classification setting with precision/recall/F_1-measure as evaluation score [1].

V. Experiments

In Table I, we compare the results of the *LP* evaluation in the filtered setting on two datasets, for which we used evaluation results from [31], and also computed results for MuRP [2] using their published code[1]. Given the AMRI score we can clearly conclude that all methods perform better than random. We also can compare the performance of the methods across

[1]https://github.com/ibalazevic/multirelational-poincare

TABLE I: Link prediction results

dataset	WN18RR		FB15k-237	
metric	MR	AMRI (%)	MR	AMRI (%)
DistMult [36]	7,000	65.8	500	93.0
ConvE [8]	4,412	78.4	241	96.6
TransE [5]	2,289	88.8	317	95.6
TransH [32]	2,126	89.6	219	97.0
R-GCN [23]	6,254	69.4	540	92.5
MuRP [2]	2,448	88.0	167	97.7

datasets and observe consistently worse performance on the *WN18RR* dataset. This difference is not only due to the larger number of entities in WN18RR ($\approx 45k$) compared to FB15k-237 ($\approx 15k$), but has to be caused by a different mechanism, e.g. the higher sparsity of WN18RR, or the richer relational patterns in FB15k-237. We leave the detailed analysis of dataset complexity for the future work.

VI. Conclusion

In this work, we address problems in the evaluation of LP and EA models for knowledge graphs. We thoroughly analyzed the current evaluation framework and identified several vulnerabilities. We demonstrated their causes and effects and showed how the problems can be mitigated by a simple adjustment for chance.

References

[1] Algergawy, A., Faria, D., Ferrara, A., Fundulaki, I., Harrow, I., Hertling, S., Jiménez-Ruiz, E., Karam, N., Khiat, A., Lambrix, P., Li, H., Montanelli, S., Paulheim, H., Pesquita, C., Saveta, T., Shvaiko, P., Splendiani, A., Thiéblin, É., Trojahn, C., Vatascinová, J., Zamazal, O., Zhou, L.: Results of the ontology alignment evaluation initiative 2019. In: OM@ISWC. CEUR Workshop Proceedings, vol. 2536, pp. 46–85. CEUR-WS.org (2019)

[2] Balazevic, I., Allen, C., Hospedales, T.M.: Multi-relational poincaré graph embeddings. In: NeurIPS. pp. 4465–4475 (2019)

[3] Belhadi, H., Akli-Astouati, K., Djenouri, Y., Lin, J.C.W.: Data mining-based approach for ontology matching problem. Applied Intelligence pp. 1–18 (2020)

[4] Berrendorf, M., Faerman, E., Melnychuk, V., Tresp, V., Seidl, T.: Knowledge graph entity alignment with graph convolutional networks: Lessons learned. arXiv preprint arXiv:1911.08342 (2019)

[5] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)

[6] Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T.: Multi-channel graph neural network for entity alignment. In: ACL (1). pp. 1452–1461. ACL (2019)

[7] Chen, M., Tian, Y., Yang, M., Zaniolo, C.: Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In: IJCAI. pp. 1511–1517 (2017)

[8] Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI. pp. 1811–1818. AAAI Press (2018)

[9] Dietz, L., Xiong, C., Dalton, J., Meij, E.: Special issue on knowledge graphs and semantics in text analysis and retrieval. Inf. Retr. J. **22**(3-4), 229–231 (2019)

[10] Fuhr, N.: Some common mistakes in IR evaluation, and how they can be avoided. SIGIR Forum **51**(3), 32–41 (2017)

[11] Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: ICML. PMLR, vol. 97, pp. 2505–2514. PMLR (2019)

[12] Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L.: Jointly embedding knowledge graphs and logical rules. In: EMNLP. pp. 192–202. The Association for Computational Linguistics (2016)

[13] Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 7031, pp. 273–288. Springer (2011)

[14] Krompaß, D., Nickel, M., Jiang, X., Tresp, V.: Non-negative tensor factorization with rescal. In: Tensor Methods for Machine Learning, ECML workshop. pp. 1–10 (2013)

[15] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

[16] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. pp. 2181–2187. AAAI Press (2015)

[17] Mao, X., Wang, W., Xu, H., Lan, M., Wu, Y.: MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In: WSDM. pp. 420–428. ACM (2020)

[18] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2015)

[19] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Icml. vol. 11, pp. 809–816 (2011)

[20] Niu, X., Rong, S., Wang, H., Yu, Y.: An effective rule miner for instance matching in a web of data. In: CIKM. pp. 1085–1094. ACM (2012)

[21] Pei, S., Yu, L., Hoehndorf, R., Zhang, X.: Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In: WWW. pp. 3130–3136. ACM (2019)

[22] Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association **66**(336), 846–850 (1971)

[23] Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607. Springer (2018)

[24] Shao, C., Hu, L., Li, J., Wang, Z., Chung, T.L., Xia, J.: Rimom-im: A novel iterative framework for instance matching. J. Comput. Sci. Technol. **31**(1), 185–197 (2016)

[25] Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS. pp. 926–934 (2013)

[26] Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: ISWC (1). pp. 628–644. Springer (2017)

[27] Sun, Z., Hu, W., Zhang, Q., Qu, Y.: Bootstrapping entity alignment with knowledge graph embedding. In: IJCAI. pp. 4396–4402 (2018)

[28] Sun, Z., Wang, C., Hu, W., Chen, M., Dai, J., Zhang, W., Qu, Y.: Knowledge graph alignment network with gated multi-hop neighborhood aggregation. arXiv preprint arXiv:1911.08936 (2019)

[29] Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. pp. 57–66 (2015)

[30] Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: AAAI. pp. 297–304. AAAI Press (2019)

[31] Wang, R., Li, B., Hu, S., Du, W., Zhang, M.: Knowledge graph embedding via graph attenuated attention networks. IEEE Access (2019)

[32] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. pp. 1112–1119. AAAI Press (2014)

[33] Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: EMNLP. pp. 349–357. ACL (2018)

[34] Xie, R., Liu, Z., Sun, M.: Representation learning of knowledge graphs with hierarchical types. In: IJCAI. pp. 2965–2971. IJCAI/AAAI Press (2016)

[35] Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., Yu, D.: Cross-lingual knowledge graph alignment via graph matching neural network. In: ACL (1). pp. 3156–3161. ACL (2019)

[36] Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (Poster) (2015)

[37] Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., Qu, Y.: Multi-view knowledge graph embedding for entity alignment. In: IJCAI. pp. 5429–5435 (2019)

[38] Zhu, Q., Zhou, X., Wu, J., Tan, J., Guo, L.: Neighborhood-aware attentional representation for multilingual knowledge graphs. In: IJCAI. pp. 1943–1949 (2019)

Appendix D

The Bayesian Cut

Petr Taborsky, **Laurent Vermue**, Maciej Korzepa, and Morten Morup. The Bayesian Cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

The Bayesian Cut

Petr Taborsky, Laurent Vermue, Maciej Korzepa, and Morten Mørup

Abstract—An important task in the analysis of graphs is separating nodes into densely connected groups with little interaction between each other. Prominent methods here include flow based graph cutting procedures as well as statistical network modeling approaches. However, adequately accounting for the holistic community structure in complex networks remains a major challenge. We present a novel generic Bayesian probabilistic model for graph cutting in which we derive an analytical solution to the marginalization of nuisance parameters under constraints enforcing community structure. As a part of the solution a large scale approximation for integrals involving multiple incomplete gamma functions is derived. Our multiple cluster solution presents a generic tool for Bayesian inference on Poisson weighted graphs across different domains. Applied on three real world social networks as well as three image segmentation problems our approach shows on par or better performance to existing spectral graph cutting and community detection methods, while learning the underlying parameter space. The developed procedure provides a principled statistical framework for graph cutting and the Bayesian Cut source code provided enables easy adoption of the procedure as an alternative to existing graph cutting methods.

Index Terms—normalized cut, ratio cut, graph cut, modularity, degree-corrected stochastic block modeling, Bayesian inference, incomplete gamma function, image segmentation.

1 Introduction

In the analysis of graphs, partitioning nodes into groups that are highly intra-connected with few inter-group connections has become important in disparate scientific fields - from network science for the identification of communities [1], [2], computer vision for image segmentation [3], [4] and the extraction of superpixel representations [5], scene reconstruction from large community photo collections [6], video decomposition [7], to physics for the splitting of materials [8]. In fact, many problems can be rephrased as a graph partitioning problem. This includes clustering problems based on pair-wise similarity in which graph partitioning approaches have found to have merits over traditional k-means and agglomerative hierarchical clustering procedures [9], and semi-supervised learning problems in which a popular solution procedure is to use graph cuts constrained according to the labelled observations [10], [11].

A variety of computational tools have been developed for graph partitioning. As such, methods based on minimizing flow between the separated entities have been devised based on various quality measures of cutting graphs. Two prominent procedures are the ratio cut [12] and normalized cut [3], for a review see also [4], [9]. On the other end, flexible in objective function, are methods minimizing certain classes of submodular energies in pairwise Markov Random Fields with applications in computer vision [13] and extended to certain nonsubmodular functions in [14]. Recently, inference in sparse graphs recovering true partitions using side information was introduced in [15]. While providing general optimisation frameworks these methods face scaling issues. Within network science a prominent procedure to identify communities is based on optimizing the modularity measure proposed in [1], which contrasts intra-group connectivity structure relative to the connectivity structure as would be expected according to the nodes' degree distribution. Within the social sciences identifying subgroups in graphs has been addressed using stochastic block-models (SBM) [16], [17] that identify homogeneous groups with similar connectivity profiles. This framework has been advanced to community detection by constraining parameters specifying intra-connectivity to be higher than inter-connectivity based on an information theoretic compression imposing intra and inter link constraints [18] or through Bayesian modeling constraining the parameters specifying intra and inter group link densities [19]. When partitioning networks a limitation of the SBM is that it is driven by grouping nodes according to their degree distribution. This issue has been alleviated by the degree-corrected stochastic block model (dc-SBM) proposed in [20] and its non-parametric Bayesian counterpart defined in [21]. Recently, it has been proven that modularity is a special case of maximum-likelihood estimation in the dc-SBM [22] assuming a planted l-partition model [23] in which link densities within l groups are specified only by two parameters; a within community η_{in} and between community strength η_{out} and further assuming the network is community structured, i.e. $\eta_{in} > \eta_{out}$. This then corresponds to the generalized modularity quality function proposed in [24] in which modularity is perfectly recovered when $\frac{\eta_{in}-\eta_{out}}{\log(\eta_{in})-\log(\eta_{out})} = 1$ [22].

In this paper, we propose a novel computational framework for cutting graphs into communities or groups that accounts for parameter uncertainty through Bayesian modeling. Our starting point is the dc-SBM in which we explicitly impose community structure requiring the parameters specifying intra-connectivity to be strictly larger than the corresponding inter-connectivity. Although less flexible than the dc-SBM, our model is more realistic than the planted l-partition model as we endow each community separate link-densities. We derive a Bayesian inference procedure and provide an analytical solution to the corresponding constrained integral representation. On three social networks

• *Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby, Denmark. E-mail: {ptab, lauve, mjko, mmor}@dtu.dk*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 1: Summary of the notation used.

Notation	Meaning	Definition
A	Adjacency Matrix	
A_{ij}	Link strength between node i and j	
b_c	Hyperparameter link density gap between the inter clusters expected link density and expected link density in community c	
C	Number of communities/clusters	
G	Undirected Graph	
d_i	Degree of node i	$A_{ii}/2 + \sum_{j \neq i} A_{ij}$
D_c	Sum of node degrees in cluster c.	$\sum_{i:z_i=c} d_i$
n	Total number of nodes in graph G	
n_c	Number of nodes in cluster c	$\sum_{i:z_i=c} 1$
n_{out}	Number of nodes between the clusters	$\sqrt{n^2 - \sum_{c=1}^{C} n_c^2}$
N	Total number of links in the graph G	$\sum_i A_{ii}/2 + \sum_{i<j} A_{ij}$
N_c	Number of links in cluster c	$\sum_{i:z_i=c} A_{ii}/2$ $+\sum_{i:z_i=c, j<i:z_j=c} A_{ij}$
N_{out}	Number of links between the clusters	$N - \sum_{c=1}^{C} N_c$
z_i	Cluster assignment of node i	
$\boldsymbol{z}$	Set of node assignments z_i for all nodes n	$\{z_1, z_2, \ldots, z_n\}$
Z_G	Normalizing constant of the graph G	$\prod_{i<j} A_{ij}! \prod_i \frac{A_{ii}}{2}! 2^{\frac{A_{ii}}{2}}$.
Z_{BC}	Normalizing constant of the constrained distribution	see (5)
α_c	A priori assumed link counts within community c, $\alpha_c \in \mathbb{R}^+$	
α_{out}	A priori assumed link counts between communities, $\alpha_{out} \in \mathbb{R}^+$	
β_c	A priori assumed number of network entries within community, $\beta_c \in \mathbb{R}^+$	
β_{out}	A priori assumed number of network entries between communities, $\beta_{out} \in \mathbb{R}^+$	
$\boldsymbol{\eta}$	Set of all η parameters	$\{\eta_1, \ldots, \eta_C, \eta_{out}\}$
γ	Degree correction hyperparameter	
η_c	Parameter controlling expected density of links within cluster c	
η_{out}	Parameter controlling expected density of links between clusters	
ϕ_i	Weight of node i	
$\boldsymbol{\phi}$	Set of node weights ϕ_i for all nodes n	$\{\phi_1, \phi_2, \ldots, \phi_n\}$
θ_i	Node degree control weight for node i	$n_{z_i} \phi_i$
$B(\boldsymbol{x})$	Multivariate Beta function	$\frac{\prod_k \Gamma(x_k)}{\Gamma(\sum_k x_k)}$

we demonstrate the importance of correctly accounting for community-structure when clustering nodes in graphs and that our Bayesian approach to cutting graphs have merits in contrast to the prominent graph cutting procedures outlined above. This includes better recovery of the true underlying partitioning structure of nodes into groups and more reliable inference. We further highlight the utility of the procedure for image segmentation considering both the Fast Marching Method (FMM) of [25] and the mean color regional adjacency graph (RAG) of [26] where normalized cut is typically applied. Notably, our results are for illustrative purposes demonstrated in the context of social network modeling in which the true partitioning structure is known, and image segmentation in which results can easily be visually inspected. However, we note that the computational framework developed has application beyond social network modeling and computer vision to the many domains in which graph cuts are currently used.

2 METHOD

Let G be an undirected graph with adjacency matrix A (i.e., $A_{ij} = A_{ji}$) whose elements A_{ij} are equal to the number of links between nodes i and j for $i \neq j$ and for computational reasons [20] twice that number for $i = j$. Let further n define the total number of nodes in the graph.

Following the dc-SBM [20] we assume that G is partitioned into a fixed number of C communities and the number of links between nodes i and j follow a Poisson distribution:

$$A_{ij} = \begin{cases} Poisson(\theta_i \theta_j \eta_{z_i z_j}) \text{ for } i \neq j \\ Poisson(\frac{1}{2} \theta_i^2 \eta_{z_i z_i}) \text{ for } i = j \end{cases}, \quad (1)$$

in which the parameter η_{ce} controls the probability of links between communities c and e, θ_i regulates the probability of links connected to the node i based on the degree of that node, and z_i defines the community assignment of node i. The factor of $\frac{1}{2}$ for $i = j$ results from the factor of two in the definition of diagonal elements of the adjacency matrix. In particular in all presented application in this paper self-links A_{ii} are constant. For the social networks presented they are zeros given by data, while in image applications with well defined similarities (following a common sense that node/pixel is similar to itself) they obtain maximal similarity.

As noted in [20] typically in large scale applications (i.e. images) self-links do not play a role as their effect diminish with scale ($\sim 1/n$). If necessary they can be marginalized as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

suggested in [21]. Although it may be undesired to account for self-links they add to generality of the model that makes computations and (approximate) optimisation easier, i.e. [27].

In order to keep analytic tractability of the constrained model that will be introduced later we assume all links between different communities are generated using the same value, i.e. $\eta_{ce} = \eta_{out}$ for $c \neq e$. We will also refer to η_{cc} simply as η_c and $\boldsymbol{\eta}$ as the set of all $\{\eta_1, \ldots, \eta_C, \eta_{out}\}$ parameters. Accordingly, the probability of graph G can be written as:

$$\begin{aligned} P(G|\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{z}) &= \prod_{i<j} \frac{(\theta_i\theta_j\eta_{z_iz_j})^{A_{ij}}}{A_{ij}!} \exp(-\theta_i\theta_j\eta_{z_iz_j}) \\ &\times \prod_i \frac{(\frac{1}{2}\theta_i^2\eta_{z_iz_i})^{A_{ii}/2}}{(A_{ii}/2)!} \exp(-\frac{1}{2}\theta_i^2\eta_{z_iz_i}) \\ &= \frac{1}{Z_G}\eta_{out}^{N_{out}} \exp(-\frac{n_{out}^2}{2}\eta_{out}) \\ &\times \left[\prod_c \eta_c^{N_c} \exp(-\frac{n_c^2}{2}\eta_c)\right] \left[\prod_i \theta_i^{d_i}\right]. \end{aligned} \tag{2}$$

We have here used that $d_i = A_{ii}/2 + \sum_{j\neq i} A_{ij}$ is the degree of node i; $n_c = \sum_{i:z_i=c} 1$, $N_c = \sum_{i:z_i=c} A_{ii}/2 + \sum_{i:z_i=c, j<i:z_j=c} A_{ij}$ are respectively the number of nodes and links in community c; $n_{out}^2 = n^2 - \sum_{c=1}^C n_c^2$ and $N_{out} = N - \sum_{c=1}^C N_c$ with $N = \sum_i A_{ii}/2 + \sum_{i<j} A_{ij}$, whereas $Z_G = \prod_{i<j} A_{ij}! \prod_i \frac{A_{ii}}{2}! 2^{\frac{A_{ii}}{2}}$. Following [21], given partition $\boldsymbol{z}$, we define a constraint $\sum_{i:z_i=c} \theta_i = n_c$ and parametrize $\theta_i = n_{z_i}\phi_i$ such that parameters $(\phi_i)_{z_i=c}$ for each community c lie on a simplex. We endow all parameters with priors thereby accounting for uncertainty using Bayesian modeling. Thus, for given partition $\boldsymbol{z}$, we assign Dirichlet priors for the $(\phi_i)_{z_i=c}$ parameters of each community c. Further we impose Gamma priors for the elements of $\boldsymbol{\eta}$ and we obtain:

$$\begin{aligned} p(\boldsymbol{\phi}|\boldsymbol{z}) &= \prod_c \frac{1}{B(\gamma \mathbf{1}_{n_c})} \prod_{i:z_i=c} \phi_i^{\gamma-1}, \\ p(\boldsymbol{\eta}) &= \frac{\beta_{out}^{\alpha_{out}}}{\Gamma(\alpha_{out})} \eta_{out}^{\alpha_{out}-1} \exp(-\beta_{out}\eta_{out}) \\ &\times \prod_c \frac{\beta_c^{\alpha_{in}}}{\Gamma(\alpha_c)} \eta_c^{\alpha_c-1} \exp(-\beta_c\eta_c), \end{aligned} \tag{3}$$

where $B(\boldsymbol{x}) = \frac{\prod_k \Gamma(x_k)}{\Gamma(\sum_k x_k)}$ denotes the multivariate Beta function, and γ is a hyperparameter that allows to infer the optimal strength of degree correction for a given graph such that if $\gamma \to \infty$, then $\phi_i \to \frac{1}{n_c}$ and $\theta_i \to 1$ and the model reduces to the corresponding SBM [21]. On the other hand, if $\gamma \to 0$, then $\phi_{i^*} \to 1$ and $\theta_{i^*} \to n_c$ for some node i^* in each community c and thus a network generated according to this prior becomes dominated by a few greedy nodes. α_c and α_{out} denotes the a priori assumed number of links within community c and between communities (i.e., the prior shape parameter of the Gamma distribution) whereas β_c and β_{out} denotes the corresponding a priori imposed number of network entries (i.e., the prior rate parameter of the Gamma distribution) within community c and between communities. Assuming further an uniform prior on $\boldsymbol{z}$, $P(\boldsymbol{z}) = C^{-n}$, we obtain:

$$\begin{aligned} P(G, \boldsymbol{z}) &= \int P(G|\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{z}) p(\boldsymbol{\phi}) p(\boldsymbol{\eta}) P(\boldsymbol{z}) d\boldsymbol{\eta} d\boldsymbol{\phi} \\ &= \frac{C^{-n}}{Z_G} \frac{\Gamma(N_{out}+\alpha_{out})\beta_{out}^{\alpha_{out}}}{\left(\frac{n_{out}^2}{2}+\beta_{out}\right)^{N_{out}+\alpha_{out}} \Gamma(\alpha_{out})} \\ &\times \prod_c \frac{\Gamma(N_c+\alpha_c)\beta_c^{\alpha_c}}{\left(\frac{n_c^2}{2}+\beta_c\right)^{N_c+\alpha_c} \Gamma(\alpha_c)} \frac{B(\gamma \mathbf{1}_{n_c} + (d_i)_{i:z_i=c})}{B(\gamma \mathbf{1}_{n_c})} \eta_c^{D_c}, \end{aligned} \tag{4}$$

where $D_c = \sum_{i:z_i=c} d_i$ is the sum of node degrees in community c. The marginalized parameters $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_C, \eta_{out}\}$ can be interpreted as the densities of links within each community and between the communities respectively.

To ensure community structure in the graph, we presently restrict the model such that the within-community densities are larger than the between-community density. This has previously been considered in the context of the SBM [18], [19] but not in the context of the dc-SBM and without fully analytical tractable solutions to the constraints as presently derived. We constrain $\boldsymbol{\eta}$ parameters such that $\eta_c b_c \geq \eta_{out}$ for each community c where each b_c is a hyperparameter within range $[0, 1]$ specifying a density gap between the inter and intra community densities as considered in the context of the standard SBM in [19]. We introduce this constraint by defining the following constrained prior on the $\boldsymbol{\eta}$ parameters

$$\begin{aligned} p_{BC}(\boldsymbol{\eta}) &= \frac{1}{Z_{BC}} \eta_{out}^{\alpha_{out}-1} \exp(-\beta_{out}\eta_{out}) \\ &\times \left(\prod_{c=1}^C \eta_c^{\alpha_c-1} \exp(-\beta_c\eta_c)\right) I(\boldsymbol{\eta}), \end{aligned} \tag{5}$$

where $I(\boldsymbol{\eta}) = \prod_c \chi_{[0;\infty[}(\eta_c - b_c\eta_{out})$ is an indicator function evaluating to 1 if the constraints are satisfied and zero otherwise ($\chi_{[a;b]}(x)$ is the standard step function evaluating to one if $x \in [a; b]$ and 0 otherwise). Z_{BC} is the normalizing constant of this constrained distribution. For a summary of the notation used see table 1.

Combining priors with the likelihood function and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

marginalizing the $\boldsymbol{\phi}$ and $\boldsymbol{\eta}$ parameters gives:

$$\begin{aligned}
p(G,\boldsymbol{z}) &= \int p(G|\boldsymbol{\phi},\boldsymbol{\eta},\boldsymbol{z})p(\boldsymbol{\phi}|\boldsymbol{z})p_{BC}(\boldsymbol{\eta})p(\boldsymbol{z})d\boldsymbol{\eta}d\boldsymbol{\phi} \\
&= \int \eta_{out}^{N_{out}+\alpha_{out}-1}\exp\left(-\eta_{out}(\frac{n_{out}^2}{2}+\beta_{out})\right) \\
&\times \left[\prod_c \eta_c^{N_c+\alpha_c-1}\exp\left(-\eta_c(\frac{n_c^2}{2}+\beta_c)\right) I(\boldsymbol{\eta}) \right. \\
&\left. \times \frac{B\left(\gamma\mathbf{1}_{n_c}+(d_i)_{i:z_i=c}\right)}{B(\gamma\mathbf{1}_{n_c})} n_c^{D_c}\right] d(\boldsymbol{\eta}) \times \frac{C^{-n}}{Z_G Z_{BC}} \\
&= \int_0^\infty e^{-\eta_{out}(\frac{n_{out}^2}{2}+\beta_{out})}\eta_{out}^{N_{out}+\alpha_{out}-1} \\
&\times \prod_{c=1}^{C}\Gamma\left(N_c+\alpha_c, \eta_{out}\times\left(\frac{\frac{n_c^2}{2}+\beta_c}{b_c}\right)\right)d\eta_{out} \\
&\times \left[\prod_{c=1}^{C}\left(\frac{n_c^2}{2}+\beta_c\right)^{-(N_c+\alpha_c)}\right. \\
&\left. \times \frac{B\left(\gamma\mathbf{1}_{n_c}+(d_i)_{i:z_i=c}\right)}{B(\gamma\mathbf{1}_{n_c})} n_c^{D_c}\right] \times \frac{C^{-n}}{Z_G Z_{BC}},
\end{aligned} \tag{6}$$

where in the second step we used change of variables $s = \eta_c(\frac{n_c^2}{2}+\beta_c)$ to obtain each of c integrals in the form of an upper incomplete gamma function (in the following simply referred to as incomplete gamma function) given by [28]:

$$\Gamma(\alpha,x) = \int_x^\infty s^{\alpha-1}e^{-s}ds \tag{7}$$

A major challenge that remains and we presently solve is to analytically marginalize η_{out} in the above expression thereby solving analytically for the constraints specified by $I(\boldsymbol{\eta})$.

2.1 Marginalization of constrained η parameters

According to eq. (6) marginalizing under the constraint imposed by $I(\boldsymbol{\eta})$ requires the solution to an integral of the following form:

$$\int_0^\infty e^{-B_0 x}x^{\mu_0-1}\left(\prod_{c=1}^{C}\Gamma(\mu_c,B_c x)\right)dx, \tag{8}$$

(Marginalizing integral)

Where we have used the following substitutions, $x = \eta_{out}$, $\mu_c := N_c + \alpha_c$, $\mu_0 := N_{out} + \alpha_{out}$, $B_c := \frac{n_c^2}{2} + \beta_c$, $B_0 := \frac{n_{out}^2}{2} + \beta_{out}$, and ignored all terms independent on η_{out}. As a result, the μ_c and B_c elements in (8) relate respectively to scale and rate parameters of the involved incomplete gamma functions.

We outline what is to the best of our knowledge a novel approach solving integrals of the form presented in Eq.(8). We exploit the following known recurrence property of incomplete gamma functions (see Theorem 1 in [29]): $\Gamma(a+1,x) = a\Gamma(a,x) + x^a e^{-x}$ for $a \in \mathbb{R}, a > 0$. This can be considered a generalization of $\Gamma(n+1) = n\Gamma(n)$ to the incomplete Gamma function. By a simple recursion of this property we obtain

$$\Gamma(a,x) = \frac{\Gamma(a+K,x)}{(a)^{\dot{K}}} - x^a e^{-x}\sum_{i=0}^{K-1}\frac{x^i}{(a)^{\dot{i+1}}}$$

(K-recurrence of $\Gamma's$)

where $(a)^{\dot{n}}$ is the Pochhammer symbol (a.k.a. "rising factorial") defined as $(a)^{\dot{n}} = \Gamma(a+n)/\Gamma(a)$. This recursion formalizes idea of "shifting" of shape parameters of gamma distribution as shown in figure 1.

The following theorem presents application of the "shifting" method described above to solve the multidimensional incomplete gamma integral in equation (8) up to an arbitrary precision.

Theorem 2.1. *For every $C \in \mathbb{N}^+$, $\mu_i, B_i \in \mathbb{R}, \mu_i > 0, B_i > 0$ for $i \in \{1,...,C\}$ and $K \in \mathbb{N}^+$ following equality holds:*

$$\begin{aligned}
&\int_0^\infty e^{-xB_0}x^{\mu_0-1}\prod_{c=1}^{C}\Gamma(\mu_c,B_c x)dx \qquad (9)\\
&= \sum_{m_1=1}^{C}\sum_{\substack{m_2=1,\\ m_2\neq m_1}}^{C}\cdots\sum_{\substack{m_C=1,\\ m_C\neq m_1,\ldots,m_{C-1}}}^{C}\sum_{i_1=0}^{K-1}\cdots\sum_{i_C=0}^{K-1} \\
&\prod_{w=1}^{C}\frac{B_{m_w}^{\mu_{m_w}}(B_0+\sum_{j=1}^{w-1}B_{m_j})^{i_w}}{\left(\mu_0+\sum_{j=1}^{w-1}(\mu_{m_j}+i_j)\right)^{\dot{i_w+1}}} \\
&\times \frac{\Gamma\left(\mu_0+\sum_{j=1}^{C}(\mu_{m_j}+i_j)\right)}{(B_0+\sum_{j=1}^{C}B_{m_j})^{(\mu_0+\sum_{j=1}^{C}(\mu_{m_j}+i_j))}} \qquad (10)\\
&+ E(K), \qquad (11)
\end{aligned}$$

where the error term $E(K)$ satisfies $\lim_{K\to\infty}E(K) = 0$.

Proof. Detailed proof altogether with additional two proven lemmas is to be found in appendix. (6.3) □

To evaluate the joint distribution $p(G,\boldsymbol{z})$ the integral (8) is to be evaluated twice. First to compute prior normalization factor of hyperparameters (α's being gamma priors), denoted Z_{BC}, and second to evaluate the integral (6) with shape parameters μ's that are result of α's added together with link counts from the respective clusters.

While the former can be efficiently solved by theorem 2.1 as the prior values are typically small requiring a small value of K, the latter imposes substantial computational challenges especially for large and dense graphs where the use of theorem 2.1 becomes computationally heavy as the required K has to be in orders of magnitudes of the number of links in the largest cluster.

Rather than resorting to analytical integration one could opt for the use of point estimates in the large setting where the posterior distribution can be expected to be peaked and thereby point estimates to provide reasonable accuracy or apply simple normal approximations through the Laplace

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

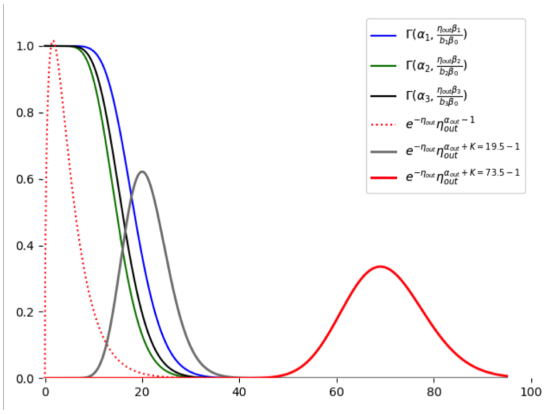


Fig. 1: Decomposition of integrand of Part A in lemma 6.1 into elements and shift of original gamma pdf (dotted red) using (K-recurrence of $\Gamma's$) to the inadequately shifted (gray curve K=19.5) and adequately shifted (red curve K=73.5) with close to zero-mass area of all the considered incomplete gamma functions (blue, green, and black curves) whereby the product in Part A becomes close to zero. As a result, the size of the shift controls the closeness to zero of Part A.

procedure also potentially accounting for the constraints using the result of the work of Hartman at al. [30] from 2017. Notably, a simple point estimate would be the maximum a posteriori of $\boldsymbol{\eta}$ under the required constraint and as the posterior is convex with convex constraints on $\boldsymbol{\eta}$ the MAP estimation of the constrained $\boldsymbol{\eta}$ is convex. Alternatively, η_{out} could be sampled and conditioned on the sampled value of η_{out}, η_c could be analytically marginalized using the incomplete Gamma function. While these approaches are scalable they are approximate and for the large scale setting we therefore opt for the following analytic procedure accounting explicitly for the uncertainty of $\boldsymbol{\eta}$ while keeping complexity at $O(C)$ for evaluating (8) which is the same as can be achieved by use of point estimates.

2.2 Large Scale Settings

Up until now there were no limitations set on values of η and in particular of hyperparameters μ and B, besides being real and positive. In large scale applications however, we are often facing large values of $\mu_{c,c\in\{1,\dots,C\}}$. In such case, it is convenient to consider evaluation of the integral for integer values of the $\mu's$. As we present in the following theorem, for integer $\mu's$ the integral is proportional to the CDF of the Negative Multinomial distribution with easy to evaluate limiting distribution. Notably, it is shown in section 3.1 that resorting to the integer setting imposes no significant constraints for most large scale applications.

Next we present main result of this section: exact evaluation of the integral (8) in case of integer shape parameters of involved gamma densities:

Theorem 2.2. *For $C \in \mathbb{N}^+$, $\mu_i \in \mathbb{N}^+$ and $B_i \in \mathbb{R}^+$, $i \in \{0, ..., C\}$ integral* (8) *is proportional to the cumulative distribution function of the Negative Multinomial (NMn) distribution and the following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0-1} \prod_{i\in\{1,\dots,C\}} \Gamma(\mu_i, B_i x)dx$$
$$= \frac{\prod_{c=0}^{C}\Gamma(\mu_c)}{B_0^{\mu_0}} \times \tag{12}$$
$$\times \sum_{i_1=0}^{\mu_1-1} \cdots \sum_{i_c=0}^{\mu_c-1} \frac{\Gamma(\mu_0+i_1+\ldots+i_c)}{\Gamma(\mu_0)i_1!\ldots,i_c!} \left(\frac{B_0}{B}\right)^{\mu_0} \prod_{c=1}^{C}\left(\frac{B_c}{B}\right)^{i_c}, \tag{13}$$

where $B := \sum_{i=0}^{C} B_i$

Proof. To be found in Appendix (6.4). For Negative Multinomial distribution definition and properties refer to [31]. □

The connection to Negative Multinomial distribution shown in theorem 2.2 also allows for an interpretation of the marginalized posterior (8) probability. If we consider sequence of independent multinomial trials in each of which event E_i occurs with probability $p_{i,i\in\{0,\dots,C\}}, \sum_{i=0}^{C} p_i = 1$ and let X_i be the frequency of $E_{i,i\in\{1,\dots,C\}}$ "successes" before predefined number μ_0 of X_0 "failures" appears, then $(X_0, X_1, ..., X_C)$ follows the Negative Multinomial distribution NMn [31].

Hence integral (8) is proportional to the likelihood of observing μ_i "successes" (links within clusters) before number of "failures" (links between clusters) reaches at most μ_0, given that number of links in graph follows multinomial distribution with probability of links appearing in cluster c being $\frac{B_c}{B}$, which is positively related to the relative size of a cluster (proportion of nodes in cluster) c.

In the following section we make use of favourable asymptotics of the Negative Multinomial distribution to derive a fast evaluation of the integral for the large scale setting.

3 Inference

We presently show how to efficiently evaluate Theorem 2.1 for $C = 2$ clusters. In this case, the formula (omitting the error term) can be written as:

$$B_1^{\mu_1} B_2^{\mu_2} \Gamma(\mu_0) \sum_{m=1}^{2} \sum_{i_1=0}^{K-1} \sum_{i_2=0}^{K-1} \frac{(1+B_m)^{i_2}}{(1+B_1+B_2)^{\mu_T+i_1+i_2}}$$
$$\times \frac{(\mu_0+i_1+1)^{(\mu_m-1)}\Gamma(\mu_T+i_1+i_2)}{\Gamma(\mu_0+\mu_m+i_1+i_2+1)},$$

where $\mu_T = \mu_0+\mu_1+\mu_2$. If we apply substitution $v = i_1+i_2$, we can rewrite the above expression as:

$$B_1^{\mu_1} B_2^{\mu_2} \Gamma(\mu_0) \sum_{m=1}^{2} \sum_{v=0}^{2(K-1)} \frac{(1+B_m)^{v}\Gamma(\mu_T+v)}{(1+B_1+B_2)^{\mu_T+v}}$$
$$\times \sum_{i_1=0}^{min(v,K-1)} \frac{(\mu_0+i_1+1)^{(\mu_m-1)}}{(1+B_m)^{i_1}}.$$

We notice that the sums dependent on v or i_1 can be evaluated independently in $\mathcal{O}(K)$ time which allows for efficient evaluation compared to the original $\mathcal{O}(K^2)$ time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

With regards to control of the approximation error Theorem 2.1 gives for arbitrary error thresholds ϵ the existence of K that evaluates this integral up to ϵ precision. However, the Theorem is not explicit about the choice of a sufficient value of K. One simple approach for finding K to control approximation error we used to produce the results presented in section 2.1 is to set K such that the mode of inter cluster link density $\frac{\mu_0+K-1}{B_0}$ is equal or greater than the q-quantile of all gamma distributions controlling intra clusters link densities. An accuracy is then controlled by setting values of q. Results of this application on karate network are shown in figure 2. There are many alternative choices for K, however, we found this approach to be easy and efficient in practice. For the purpose of error evaluation we compared results of Theorem 2.1 with results of the scipy.integrate.quad function from the scipy 1.2.0 python package. From the figure we can observe how increasing K, corresponding to increasing the q-quantile according to the method described above, controls the absolute error on the evaluation of the integral. For the results obtained in the following we used $q = 0.9999$, given that this guarantees an absolute error close to 10^{-5}, but in most cases will range around 10^{-9}.

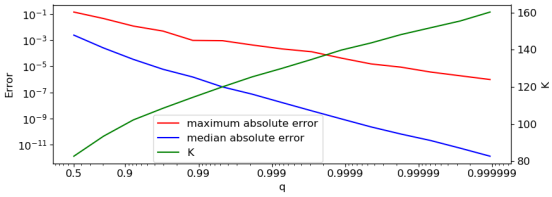


Fig. 2: Maximum and median absolute approximation error and corresponding number of added observations T for karate network based on 100 chains with 100 samples each.

Typically, a graph cut is obtained by optimizing a given cost function. In case of Bayesian Cut, the cost function is defined by the posterior distribution $p(\boldsymbol{z}|G)$ which specifies probability of every possible partition of graph G. While the full posterior would provide lots of insight into different ways of cutting the graph, due to its high complexity, it is not possible to determine it fully. Instead, the most reasonable approach is to search for the maximum of the posterior (MAP) $\boldsymbol{z}_{MAP} = \text{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|G)$. While one could opt for optimization of the posterior distribution of $\boldsymbol{z}$ possibly making use of wide arsenal of approximation methods i.e. [14], [13], [32] or other discrete optimisation methods [33] to this NP hard problem, we advocate using MCMC sampling (for reference see [34], Chapter 11) before performing optimization for a few reasons. First of all, optimization might get stuck in local maxima while sampling given enough time will find the global maximum. In practice, within the sampling budget, the sampler will likely focus on some high density region of the posterior, but it will still explore multiple modes within that region. A comparison of only using optimization compared to using the sampler can be found in the appendix, see section 6.3. Secondly, by using sampling we are able to infer values of specific hyperparameters to create a more plausible model that explains the observed data better and thus learn about the underlying structure of the problem. Finally, sampling produces not only a point estimate but an approximation of the true posterior (more or less accurate depending on its complexity and sampling budget) that can be used to answer more complex questions than what the most probable cut is. To perform MCMC sampling, we use Gibbs sampling and sample each element z_i of $\boldsymbol{z}$ independently:

$$p(z_i|G) = \frac{p(G,\boldsymbol{z})}{p(G,\boldsymbol{z}_{-i})} \propto p(G,\boldsymbol{z}).$$

We treat the hyperparameter γ as a random variable while fixing other parameters to a constant value. We use the non-informative prior $p(\gamma) = \gamma^{-1}$ and after each Gibbs sweep over all nodes in the graph, we perform 20 Metropolis-Hastings (MH) updates using the proposal distribution $\gamma^* = \gamma \exp(\epsilon), \epsilon \sim N(0, \sigma = 0.1)$. Alternatively, if one is not interested in inferring γ, it can be set to 1 which assumes any configuration of node-specific parameters $(\phi_i)_{z_i=c}$ of community c is equally probable (i.e., corresponding to the uniform distribution over the $(n_c$–$1)$-simplex). Furthermore, we fix all α and β parameters to a non-informative value 0.01 and set b to 1 unless specified otherwise. After running out of the sampling budget, we apply deterministic optimization by switching node assignments only when it leads to higher likelihood and we stop when in a full sweep over all nodes we do not observe any further improvement.

3.1 Inference for large graphs

Posterior distribution of η (expected density of links) in BC model has the same form as prior (due to the conjugacy between Poisson and Gamma) where 'shape' μ_c and 'rate' B_c, in general positive real parameters of the involved gamma densities, are by definition priors $\alpha_c \in \mathbb{R}^+$ and $\beta_c \in \mathbb{R}^+$ updated by the added number of links and nodes in cluster c respectively. This often results in large, in general real, values when dealing with large graphs. In order to find a fast evaluation algorithm first let us note that for the cutting of large graphs limiting ourselves to integer shape hyperparameters of both prior and posterior gamma densities while updating real 'rate' impose any relevant constraints in most applications as the prior is overwhelmed by the observed data. Technicaly speaking transformation from $\mu' = \lceil \mu \rceil, B' = \frac{\lceil \mu \rceil}{\mu} B$ keeps mean of posterior gamma distribution unchanged ($\frac{\mu'}{B'}$) while increases its variance (or uncertainty) ($\frac{\mu'}{B'^2}$) by factor diminishing with scale. Therefore and especially with uninformative priors resorting to integer 'shape' should have an insignificant and asymptotically zero effect on posterior for large values of μ and if not the general Theorem (2.1) should be applied.

Secondly, as shown in [31], a limiting distribution of the negative multinomial decomposes into a product of Poisson distributions as $\mu_0 \to \infty$. Making use of this limiting distribution we obtain a large scale (asymptotic) solution of our integral. In the following we make use of the fact that the cummulative density function (cdf) of a Poisson distributed variable $F_{Pois(\lambda)}(\mu_i - 1)$ can be written as $\Gamma(\mu_i - 1; \lambda)$. Let m denote the threshold beyond which the asymptotic is applied. To determine m we analyze in Figure 3 how well the asymptotic approximation of the marginalized integral (8) behaves. The figure shows that absolute error of log integral

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

is close to zero but for the bipartite setting, corresponding to a graph in which all links/similarities are between clusters while there is zero density of link/similarity within clusters. In such setting it is still possible to evaluate the integral exactly using Theorem (2.2) with complexity $O(N)$. However, if the observed graph G has bipartite structure (can be detected prior to application of the method) then the proposed asymptotic becomes expensive. This does not impose any issues for most applications, in particular, for image segmentation where bipartite structures are unlikely. Formally, proposed method to evaluate the marginalized integral (8) in large scale settings depends on sum of weights (in our case number of links) between clusters, μ_0:

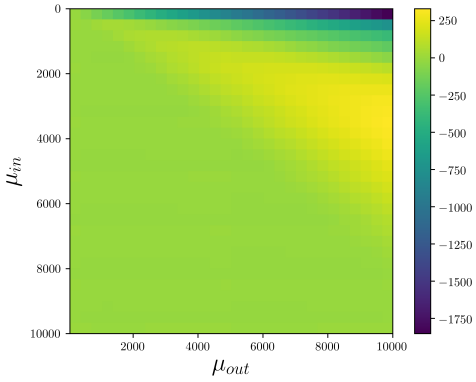


Fig. 3: Error of logarithm of integral (8) evaluated by "shifted" method of theorem (2.1) with bounded error of 10^{-5} and logarithm of same integral evaluated by asymptotic method of section (3.1). For experiments we fixed $B_0 = 60$ and $B_1 = B_2 = 70$ while ranging $\mu_{out} \in (51, 10^3)$ and $\mu_1 = \mu_2 = \mu_{in} \in (0, 10^3)$.

Large $\mu_0 > m$: If μ_0 is sufficiently large, then (41) resolves asymptotically into:

$$\prod_{i=1}^{C} \Gamma(\mu_i - 1; \mu_0 B_i / B_0) \tag{14}$$

Small $\mu_0 \leq m$: In this case there are 2 options:

- In case the smallest of μ_c's, $_{c\in\{1,\dots,C\}}$, is sufficiently large $\min_c(\mu_c) > m$ we apply 'per-partes' on (41) to rotate elements of integral and asymptotic decomposition on each of C summands resulting in:

$$B_0^{-\mu_0} \prod_{i=0}^{C} \Gamma(\mu_i) - \sum_{j=1}^{C} \prod_{\substack{i=0,\\ i\neq j}}^{C} \Gamma(\mu_i - 1; \alpha_j B_i / B_0) \tag{15}$$

- Else, when one or more μ_c's, $_{c\in\{1,\dots,C\}}$, are small $(\min_c(\mu_c) < m)$, asymptotic properties of NMn are of no use. This corresponds to a degenerated case when nodes within one or more clusters are dissimilar or respective clusters contain few nodes. In either case this does not correspond to a preferable cut. Let's note that it is unlikely that the Metropolis - Hastings/ Gibbs MCMC sampler appears to be sampling from an assignment corresponding to this case unless observed graph has aforementioned bipartite structure. In degenerate case sampler would need to accept low probability proposals against the imposed constraints on the link densities. So unless initial assignments of sampler are degenerate or number of clusters C is extremely large compared to nodes in the considered graph, it is unlikely to end up in such case during sampling. Anyway, in such case we evaluate the integral of theorem 2.2 directly at cost of higher complexity $O(N)$ instead of $O(C)$.

In the procedure above m represents a threshold above which asymptotic apply. In our image experiments (non bipartite structure) we applied $m = 50$ given results of fig. 3, striking balance between accuracy and runtimes. More elaborate and/or conservative choices may be better suited, depending on use.

3.2 Reference methods

We contrast the proposed BC to the corresponding dc-SBM without community constraints given by (4) as well as to modularity optimization (Mod), ratio-cut (RC) and normalized cut (NC). The solutions obtained by RC and NC were derived using the spectral clustering procedure described in [9] whereas the modularity objective was optimized using the spectral approach described in [1]. We note that the spectral optimization procedure may be suboptimal to other inference approaches, however, we presently use these solutions for illustrative purposes to characterize the methods and contrast favourable configurations by these approaches to the favorable configurations using the proposed BC procedure.

To evaluate the modularity score of a given partition we use the modularity objective function described in [1], given by

$$\begin{aligned} Q(\boldsymbol{z}) = \frac{1}{4m} \sum_{c=1}^{C} \Bigg[\sum_{i:z_i=c} \Bigg(\sum_{j:z_j=c} (A_{ij} - \frac{k_i k_j}{2m}) \\ - \sum_{j:z_j\neq c} (A_{ij} - \frac{k_i k_j}{2m}) \Bigg) \Bigg], \qquad m = \frac{1}{2} \sum_{i=1}^{n} k_i \end{aligned} \tag{16}$$

To evaluate solutions in the domain of NC and RC we use their respective cost functions as defined in [9]

$$RC(\boldsymbol{z}) = \frac{1}{2} \sum_{c=1}^{C} \frac{1}{n_c} \sum_{i:z_i=c} \sum_{j:z_j\neq c} A_{ij}, \tag{17}$$

$$NC(\boldsymbol{z}) = \frac{1}{2} \sum_{c=1}^{C} \frac{1}{K_c} \sum_{i:z_i=c} \sum_{j:z_j\neq c} A_{ij}. \tag{18}$$

3.3 Visualization technique for solution landscape

To show the solutions supported by each procedure we plot the solution landscapes similar to the method proposed in [35]. These landscapes were created by obtaining a set of V $\boldsymbol{z}$ vectors for the models under scrutiny. This set of vectors is expanded by 50% to cover the in between solution

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

space through pseudo-random vectors, i.e. a new vector is generated by randomly taking two distinct $\boldsymbol{z}$ vectors and combining half of the elements of each vector. The score or likelihood for each vector is subsequently obtained by running the specified model with each unique $\boldsymbol{z}$ vector. As measure of distance between partition vectors we use the Variation of Information [36] between all V vectors. The resulting $V \times V$ dimensional distance matrix is reduced to two dimensions using Multidimensional Scaling [37]. Discrete Sibson Interpolation [38] is subsequently used to obtain a meshgrid of the remaining two dimensions.

4 Results and Discussion

In the following we analyze the properties of the proposed Bayesian Cut (BC) model for community detection in social networks and image segmentation for computer vision.

We first present results on a set of simple synthetic networks (Section 4.1) followed by results for community detection in social networks (Section 4.2) that have an advantage of available "ground truth" as well as unified definition of an adjacency matrix across methods we compare with. Hence presented comparison provides insights on graph cutting performance more clearly than in the subsequent image applications where cuts are used in connection with disparate similarity matrices. In Section 4.3 two often used similarity matrices are presented to demonstrate utility of BC model as a generic tool for graph cuts. We further apply multiple cluster solutions on the images.

The Bayesian Cut source code used for these experiments is provided through a public source code repository, hosted on Github (https://github.com /DTUComputeCognitiveSystems /bayesian_cut), and through the Python Package Index (https://pypi.org/project/bayesian-cut/) to allow a straightforward installation of the package. To ensure accessibility and reproducibility of the results, the repository includes image of "bears" used in experiments (original downloaded from: https://images.app.goo.gl /Mvdra73AwjfRfp629) and the software, that is accompanied by instructions on how to use the package and Jupyter Notebooks that show how the results were obtained.

4.1 Synthetic networks

We test the proposed algorithm on synthetic networks to demonstrate the effect of imposed connectivity constraint on the inference. In this experiment, we fix the total number of nodes to $n = 100$ and links to $N = 1000$. We assume networks are partitioned into two communities having equal number of nodes ($n_1 = n_2 = \frac{n}{2}$) and links ($N_1 = N_2 = N_{in}$). For different values of intra- to inter-community link density ratio ($\eta_{in}/\eta_{out} = \frac{2N_{in}}{N_{out}}$), we generate network to match these predefined properties. We fix γ to 10^6 (to remove effects of degree correction), α_{out} to 10^{-6} (to remove the difference coming from marginalizing the constrained vs. unconstrained prior) while keeping values of the other hyperparameters as specified in Section 3. In Figure 4, we show the posterior densities (up to a constant) of the partition for a wide range of η_{in}/η_{out} density ratios for a constrained (Bayesian Cut) and corresponding unconstrained (dc-SBM) model to demonstrate the effect of the constraint. Condition $\eta_{in}/\eta_{out} < 1$ represents an extent of constraint violation - the closer it is to 0, the stronger the violation. At extreme of 0, the partition represents a bipartite network which is a structure exactly opposite to a community structure. As it can be seen in the figure, unconstrained dc-SBM assigns very high probability to partitions where there is very distinct difference between intra- and inter-community densities, even if inter-community density is higher. On the other hand, the constrained model penalizes partitions that violate the constraint and assigns them even lower probability than to partitions with the density of links uniformly distributed over the whole graph.

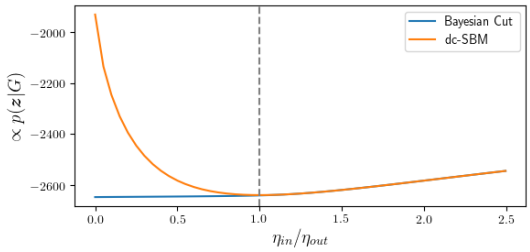


Fig. 4: Experiment on synthetic networks confirms that constrained BC model "Bayesian cut" strongly penalizes partitions that violate graph connectivity constraint, $\eta_{in} \geq \eta_{out}$, compared to unconstrained "dc-SBM" model that assigns very high probability to partitions where there is very distinct difference between intra- and inter-community densities, even if inter-community density is higher.

4.2 Community detection in social networks

For community detection the properties of the proposed Bayesian Cut (BC) model are analyzed based on three real world social networks and contrasted to ratio-cut, normalised cut, modularity and the unconstrained dc-SBM. The networks considered are:

Karate: A social undirected network studied by Zachary [39] of ties in a Karate club that turned out to split in two. The network consists of 34 nodes and 78 edges and was partitioned using modularity in [1].

Polblogs: The political blogosphere (Polblogs) network on US politics assembled by [40]. We consider the largest connected component of the network in the undirected form used in the dc-SBM analysis of [20] which contains 1222 nodes and 16714 edges.

HIV-1: Sexual partnership network extracted from the first study (Colorado Springs Project 90) in HIV Transmission Network Metastudy Project [41]. We consider the largest connected component of the network consisting of 1888 nodes and 2096 edges.

In all analyses we used $C = 2$ corresponding to the ground-truth structure of the split in Karate club and political blogs along party line. Notably, when $C = 2$ there is only one η_{out} parameter in the dc-SBM and our analyses correspond to the dc-SBM parametrization with and with-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

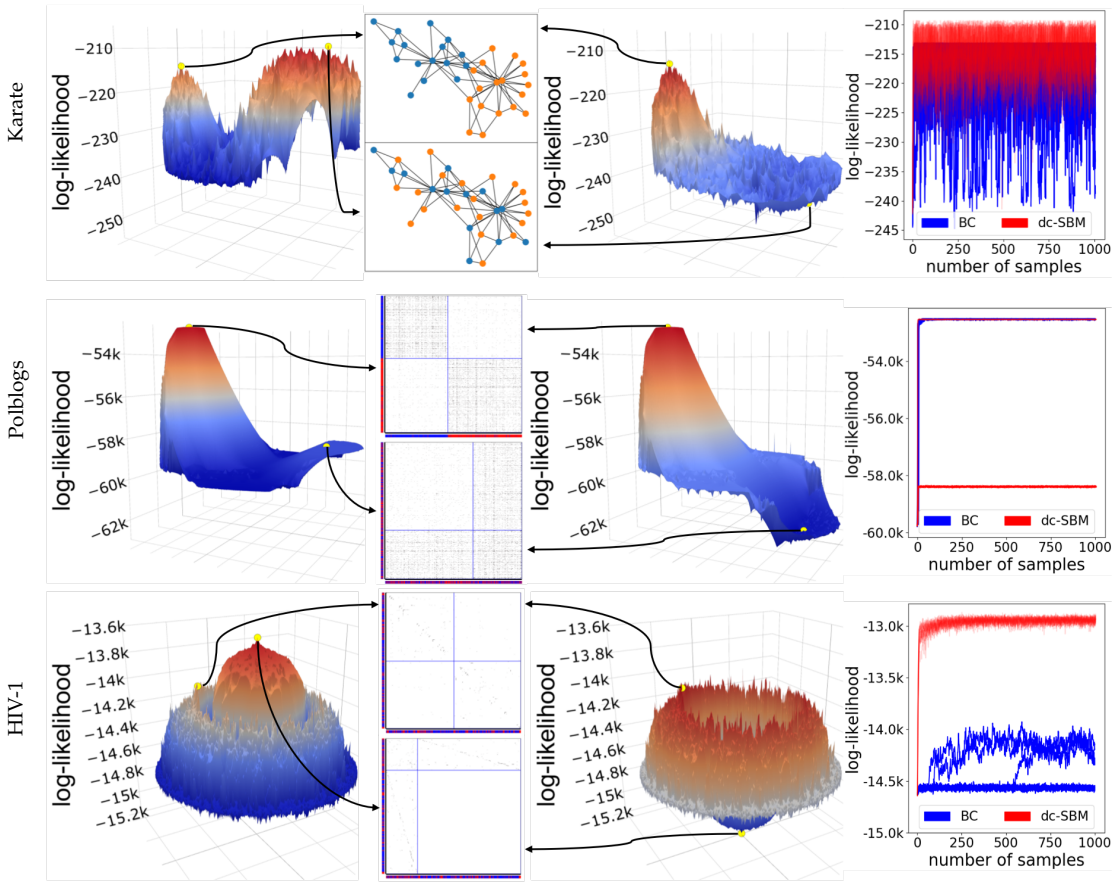


Fig. 5: Comparison of the dc-SBM (left column) and BC (right column) solution landscapes based on $p(\boldsymbol{z}|G)$ as well as the resulting cuts performed on the three networks. The outer right column shows the trace plots obtained running each model with 15 chains and 1000 samples. The dc-SBM model exhibits for all three networks modes and thus resulting cuts that violate the constraint $\eta_{in} \geq \eta_{out}$. In the corresponding adjacency matrices it can be seen that whenever the constraint is violated (lower adjacency matrix/network for each example), the off-diagonal blocks have a higher density than at least one of the diagonal blocks. In contrast, the proposed BC model gives those regions of the solution landscape that violate the constraint lower likelihoods, which leads only to modes and thus resulting cuts that do not violate the constraint.

out the community constraint. For model inference in the dc-SBM and BC we use Gibbs sampling to infer $\boldsymbol{z}$.

4.2.1 Comparison of dc-SBM and BC

Figure 5 shows the results of the unconstrained dc-SBM and our Bayesian Cut (BC) procedure for $b_1 = b_2 = 1$, i.e. imposing the constraint $\eta_1 \geq \eta_{out}$ and $\eta_2 \geq \eta_{out}$. Furthermore, a non-informative prior is used, i.e. $\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$. For the Karate network (top panel) we observe that the conventional Bayesian dc-SBM (given by the likelihood in (4)) creates a substantially different solution from our proposed BC. While our BC peaks around the true split of the Karate network, we observe that the samples of the conventional dc-SBM concentrate around two modes of the distribution in which the other mode represents a configuration that does not comply with the notion of community structure, but has a significantly higher likelihood.

For the larger Polblogs network we again observe that the dc-SBM exhibits one mode that does not comply with the community structure and creates a split leading to one community with high link density and one community with a bipartite structure, while the mode shared with our proposed model corresponds well to a separation along political orientation (i.e., democrat vs. republican). In the bottom panel for the HIV-1 network we observe a substantial difference between the dc-SBM and our proposed BC procedure with no shared modes. Here the unconstrained model identifies a bipartite structure in which one community has very low link density as compared to the inter community link density, whereas the constrained model by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

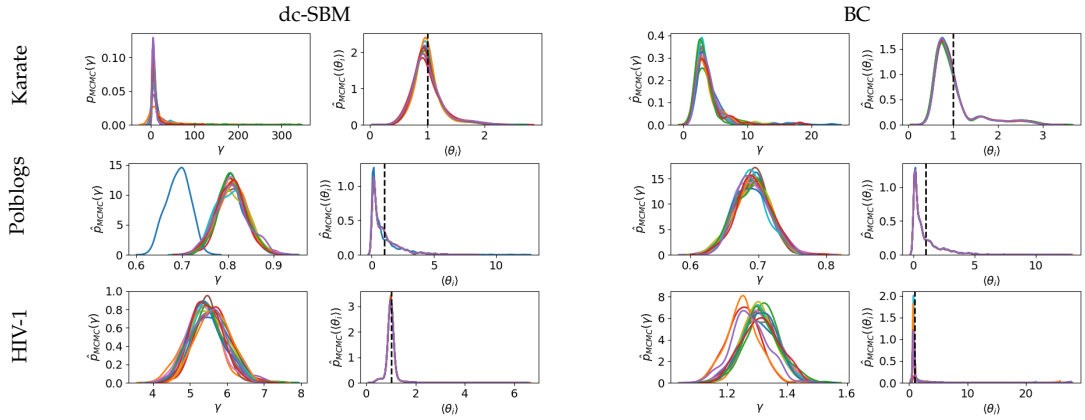


Fig. 6: Gamma inference and resulting node degree correction (theta) of the dc-SBM and BC for all three networks. The dc-SBM and BC models show substantial differences, since the parameter inference resulting from the BC model is more reliable, because it contrary to the dc-SBM does not get stuck in local optima that violate the constraint.

only giving community structure support strives to separate the network according to identifying separate communities.

Overall it can be seen that the BC with its constraints is more in line with the natural splits in the Karate and Polblogs networks and suggests a more sensible split for the HIV-1 network. The sub-optimal congruity of the unconstrained model can be attributed to the local modes of the posterior observed in Figure 5 that are unsupported by the BC procedure.

On the outer right side in Figure 5 the convergence of the dc-SBM and BC is illustrated for 15 chains and 1000 samples. Notably, we observe that for the Karate and political blogosphere networks the unconstrained model explores the mode not complying with community structure. For the political blogosphere the inference for most of the chains is stuck in the local sub-optimal mode of the posterior distribution, incapable of escaping this mode by the Gibbs sampler and recovering the underlying correct structure leading to the lower cut shown in the middle panel of figure 5. In contrast all chains of the BC model converge to the underlying partitioning structure for both networks. When considering the HIV-1 network it can be observed that the solution space supporting community structure is consisting of a vast number of local optima, contrary to the non-community supporting structure, which has a strong global mode. This is causing the community structure inferred to be less reliable and the chains to end in local modes of the community constrained posterior.

The influence of BC and dc-SBM on inferring the parameter controlling for degree (γ) is shown in figure 6. Here the gamma inference as well as the node degree correction distribution of each chain of both the dc-SBM and BC model is shown for the three networks. Focusing on the left column, a substantial difference within the inference of the γ parameter, i.e. controlling the degree correction, is observable. Subsequently, the derived θ parameters differ based on the modes preferred by the models. As previously shown, the dc-SBM model often gets stuck in local modes or exhibits globally preferred modes that do not support the community structure. Accordingly, the parameter inference is biased by the modes in the non-community structure region in those cases. In the above analysis we used non-informative priors on $\boldsymbol{\eta}$, however, we could also impose an informed prior favoring community structure in the dc-SBM. This and role of constraint parameter b is further addressed exemplary on the karate network in the appendix, section 6.4.

TABLE 2: Comparison of Cuts running 100 chains with 1000 samples without and with (in parenthesis) deterministic optimization in terms of their modularity value (Mod.) and correspondence to ground truth partition structure (avaible for Karate and Polblogs) as quantified using normalized mutual information (NMI). $\langle\cdot\rangle$ denotes average value and $\lceil\cdot\rceil$ maximum value.

	Score	RC	NC	MOD	dc-SBM	BC
Karate	⟨NMI⟩	0.415	0.732	-	0 (0)	**0.837 (0.837)**
	⌈NMI⌉	0.578	0.732	**0.837**	0 (0)	**0.837 (0.837)**
	⟨Mod.⟩	0.236	0.356	-	-0.267 (-0.258)	**0.371 (0.371)**
	⌈Mod.⌉	0.313	0.356	**0.371**	-0.267 (-0.258)	**0.371 (0.371)**
Polblogs	⟨NMI⟩	0.017	0.017	-	0.143 (0.146)	**0.717 (0.718)**
	⌈NMI⌉	0.017	0.017	0.693	0.727 (0.737)	**0.739 (0.739)**
	⟨Mod.⟩	0.001	0.001	-	-0.057 (-0.062)	**0.426 (0.426)**
	⌈Mod.⌉	0.001	0.001	0.424	**0.426 (0.426)**	**0.426 (0.426)**
HIV	⟨Mod.⟩	0.045	0.045	-	-0.363 (-0.365)	**0.185 (0.411)**
	⌈Mod.⌉	0.045	0.045	0.190	-0.357 (-0.359)	**0.385 (0.463)**

4.2.2 *Comparison of dc-SBM and BC to Modularity, NC and RC*

In Table 2 we quantify the correspondence as measured by normalized mutual information (NMI) between the inferred partitions and the partition defined by the underlying split with highest support for each of the considered methods in the Karate network and separation according to party line in Polblogs. Furthermore, we measure the adherence to community structures of each model by calculating the modularity for the inferred partitions using the formula defined in eq. 16. For each calculated metric and network we point out the average and maximum score achieved by that particular method.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

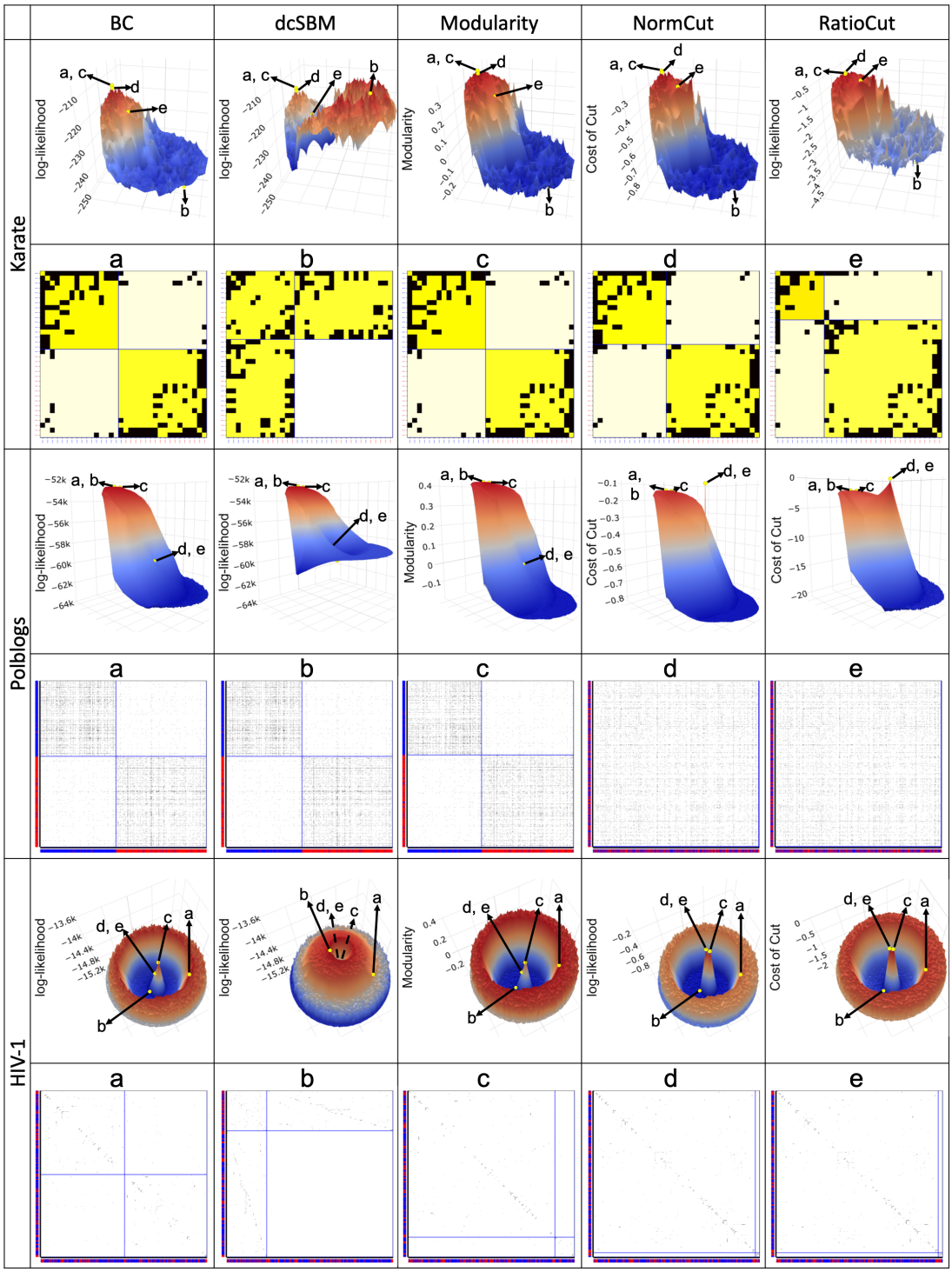


Fig. 7: Solution landscape comparison of BC, dc-SBM, Modularity, NormCut, RatioCut on the three networks. To explore the space, 100 samples from 15 chains were taken for the Bayesian methods, while for the spectral cuts 200 different solutions were generated for each method by randomly alternating 1% of the links within the networks. The costs of Normcut and Ratiocut are inverted to allow for direct landscape comparisons.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

We observe here that the BC achieves superior or on par performance on all three networks. In these results we again observe that the BC differs substantially from the dc-SBM, which is explained by the underlying supported configurations of the model likelihood $P(\boldsymbol{z}|G)$ shown in figure 5. In figure 7 we explore the solution space also of the ratio-cut (RC), normalized cut (NC) and modularity (Q) and how these solutions are supported by their corresponding objective functions.

We notice that the solutions supported (and thus the inference landscape) by the proposed BC is more in agreement with these existing community detection/graph partitioning procedures than the dc-SBM. However, we also observe notable differences of the proposed Bayesian Cut (BC) and these alternative partitioning procedures. In particular, neither RC nor NC provide as balanced solutions as the BC and they provide higher support for solutions further away from the underlying community structure. Here we pay particular attention to the cuts proposed for the Polblogs and HIV-1 network as these appear unsubstantiated due to the fact that they exclude a very small group of persons from the overall population.

For Polblogs both RC and NC exhibit very extreme and local optima in their solution landscape, which lead to a cut that excludes 4 persons from the other 1218 persons in both cases. In the case of RC, defined in eq 17, the dominance of the cost by the flow, i.e. links between two groups, is obvious. Since these two group are only connected by 1 inter-link the cost for performing this is cut is extremely low. In contrast, the true cut along party lines leads to one group with 662 nodes and one with 560, which share 1217 inter-links. To obtain lower costs, no-more than 76 inter-links would be allowed.

One way of alleviating this strong influence of the cut flow is to use NC, defined in eq. 18, which does not divide the cut flow by the number of nodes, but according to the degree of the cluster. However, even though this subtle difference changes the solution landscape in non-community supporting regions as shown in figure 7, the preference for cuts that separate unbalanced groups having very low flow remains. In this case the extreme cut leaves the small group with a degree of 5 and the bigger group with a degree of 16710, which results in very low costs. The above mentioned cut of our model results in a degree of 9464 and 8467 for the group with 662 nodes and 560 nodes respectively. In this case 894 inter-links would already give lower costs for our cut, which shows the improvement over RC, but still is not sufficient.

The highest congruence can be found between the BC and the Modularity method, confirming the community detection support of our proposed BC. Here we observe that both methods exhibit almost identical solution landscapes, which is reflected in the identical or very similar solution landscapes obtained by the methods. Interestingly, for the HIV-1 network the BC obtains a solution with a significantly higher modularity than the spectral modularity method itself identifies. In addition, this solution seems to be more balanced, since it achieves almost equally sized groups, while the proposed solution of the spectral modularity method partitions the network into a small and a large group. This highlights that BC strives for balanced modular structures.

4.3 Image Segmentation

In following we present results of image segmentation suitable for foreground-background or scene recognition. We compare BC model to NC and dc-SBM (with shared density of links out η_{out} in case of more than two segments $C > 2$).

NC implementations are often in practice combined with specific similarity matrices and we make use of the following two widely used procedures:

Mean color RAG: Mean color Regional Adjacency Graph is used to compute similarity matrices on super pixel graphs (RAG) [26] that serves as an input for NC in popular python package for image processing skimage https://scikit-image.org/docs/dev/api/skimage.future.graph.html. To compare with the BC method "cameraman" image and "coffee"available in the skimage package was used.

Fast Marching Method (FMM): This method (a.k.a. geodesical distance) is besides many used in the Graclus software presented in [25]. We used the MATLAB implementation of Jianbo Shi from https://www.cis.upenn.edu/jshi/software/ to generate the FMM similarity matrix. Graclus software optimizes normcut objective in a hierarchical manner [25] with results presented at https://www.cis.upenn.edu/jshi/software/demo2.html. For comparison purposes we use the public image of "baby" from the same site.

These methods produce similarity matrices $\mathbf{S}$ with elements in $[0; 1]$. To convert them into graphs with countable links required by the BC model we follow similar procedure as aforementioned Graclus software [25]. Graclus runs $\mathbf{A} = \lceil 100 * \mathbf{S} \rceil$ while BC implements $\mathbf{A} = \lfloor 100 * \mathbf{S} \rfloor$, both element wise.

Results of the BC model applied on images of "cameraman" and "bears" using RAG can be found in figure (8), "coffee" is presented in Appendix (13) and the results on "baby" using FMM can be found in figure (9). Notably BC model was applied on similarity matrices produced by respective implementations of RAG and FMM described above without further adjustments. In case of RAG and "bears" we adjust sigma for the Gaussian similarity kernel [1] in case of "cameraman" we leave it on default setting. "Cameraman" and "bears" experiments with Mean Color RAG have been ran with no degree correction (corresponds to hyper parameter γ set extremely large 10^7).

In all applications mentioned BC performs on par or superior to the compared methods (not necessarily state of the art though). In two partitions version considered for the "cameraman" the BC method separates objects from sky. In case of the four partition scenario used on "bears" BC recognizes foreground objects (cub and surrounding), background and adult bear while the other methods only partially succeed. For the "coffee cup" in appendix the BC

1. future.graph.rag_mean_color(img, labels1, mode='similarity', sigma=70**2, segmentation.slic(img, compactness=0.3, n_segments=100)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

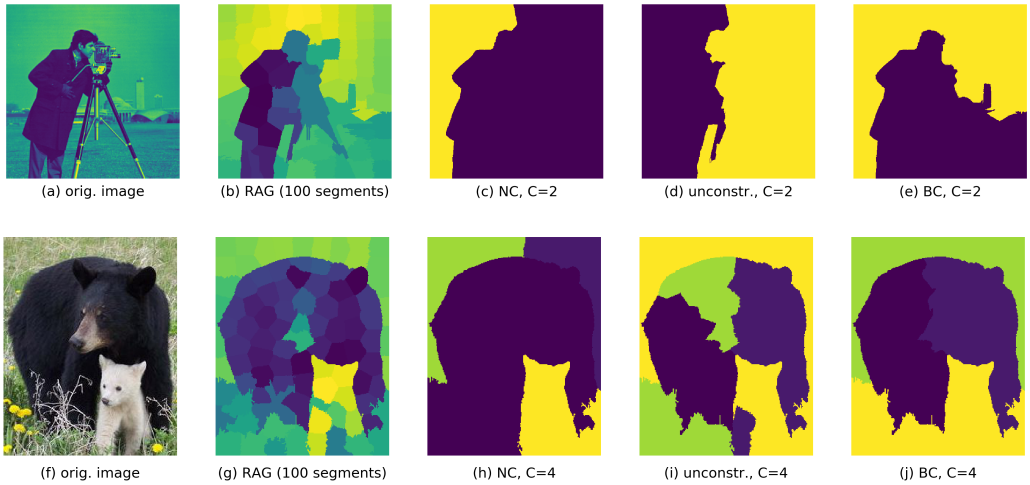


Fig. 8: **Top panel, Cameraman:** Resulting cut of BC model for C=2 (e) segments compared with unconstrained dc-SBM with shared η_{out} as well as spectral Norm cut. Fig (b) shows for reference RAG super pixel graph that is used to compute similarity matrix. Results demonstrate on par or better results of BC against referenced methods. Also it shows effect of constraint included in the model (we emphasized the effect of constraint by setting b=10 corresponding to 10x times higher within segment links density compared to links density among segments): (e) vs (d) a constraint model improves the results. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each.
Bottom panel, Bears: Resulting cut of BC model for C=4 segments (j) compared with unconstrained dc-SBM with shared η_{out} (i) as well as spectral Norm cut (h) applied on mean color RAG similarity matrix. Similar to previous results figures BC demonstrates on par or better results against referenced methods. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each with hyperparameters set on $b = 10^3$ and without degree correction

Fig. 9: Bayesian cut (BC) applied on similarity matrix obtained by fast marching method implemented by Jianbo Shi from https://www.cis.upenn.edu/~jshi/software/. Resulting cut is sampled MAP obtained from 20 MCMC chains, 1000 samples each. (a) original, (b) $C = 2, b = 10$, with degree correction hyper parameter γ being inferred. Maximum of its posterior obtained at: $\gamma_{MAP} = 4.07$, (c) $C = 2, b = 10, \gamma = 0.0001$.

removes more of the background than the unconstrained dc-SBM and captures more of the coffee cup object than NC. In case of the "baby" image see figure 9 the effect of degree correction parameter γ controlling "greediness" of clusters is showed. In the more greedy settings, (c) as opposed to gamma being inferred in option (b), fixing it to "greedy" mode recognizes focal object's boundary more complete yet produces artifacts.

In summary, the presented image segmentation results by BC are on-par or superior to NC and the unconstrained version. However, we noted during experiments that the multiple MCMC runs produced slightly different cuts confirming that the inference is prone to sub optimal solutions and multiple restarts are therefore recommended.

5 Conclusion

We have proposed the Bayesian Cut (BC) advancing the degree-corrected stochastic block-model (dc-SBM) to explicitly account for community structure. In contrast to the dc-SBM only one parameter specified inter-group connectivity strength (η_{out}), however, in contrast to the generalized modularity as conforming to an l-partition model with shared link density across communities the proposed BC include more flexible community specific link-densities. We derived a fully Bayesian procedure and demonstrated that the imposed community constraints are analytically tractable even for large graphs by deriving a novel general solution to integrals involving multiple incomplete gamma functions. We expect the presented small and large scale solutions to the integral will have applications beyond community detection in social networks and image segmentation considered in this paper. For instance, for collapsed inference in the performance analysis of cognitive radio networks [42]. We observed that the constraints had significant impact on the inference providing more reliable results in compliance with ground truth for network exhibiting community structure

and it was also empirically confirmed that the constraint had merits for image segmentation in computer vision. We also observed that strictly enforcing community structure enabled to identify configurations where traditional block-modeling would identify bipartite structure. Notably, our Bayesian Cut provides favorable partitions when compared to traditional graph cutting procedures such as the ratio and normalized cut. In particular, we empirically observed that our BC procedure has meritorious properties balancing the partitions more favorable than these existing graph partitioning procedures. We have also derived fast large scale multiple cluster solution that presents generic tool for Bayesian inference.

We presently considered a uniform prior on the partition $P(\boldsymbol{z}) = C^{-n}$ to highlight the influence of the specification of the likelihood $p(G|\boldsymbol{z})$ in identifying partitions. However, we note that within the Bayesian modeling framework other (non-uniform) priors could be applied including the Pólya-urn (i.e., marginalized Dirichlet-Categorical) representation and its infinite limit given by the non-parametric Chinese restaurant process (CRP) also used in stochastic block-modeling [43].

Overall this work presents generic graph based clustering method that can be applied on wide range of similarity matrices. For illustrative purposes we presently applied our BC approach in the context of identifying communities in social networks and image segmentation, however, the approach extends to the many applications in which graph cuts are used. Flexibility with regards to similarity matrix allows for possible applications in areas such as scene reconstruction from large set of community photos [6], where the image set is partitioned into groups of related images, based on the visual structure represented in the image connectivity graph for the collection. Connectivity graph and corresponding similarity matrix is based on scale invariant feature transform, SIFT [44], that extracts image representative features that are used to find matches and define similarity between each image pair. Another possible area of application is Video summarization and scene detection [7], where similarity used for graph partitioning are based on color similarity and temporal frame distance.

In the outlook, although MCMC sampling are suitable for network structure inference, in order to find optimal cuts, future work should investigate alternatives while keeping the properties of the proposed framework. Further concerning image segmentation, this work made use of two popular similarities, Fast Marching Method and Mean Color, that rather relate pixels based on color intensities as opposed to spatial features. As suggested above we leave as future work to explore possibilities of BC applied on other existing or new similarities as well as extension of hereby presented bayesian generative hierarchical BC model to allow for contextual spatial or other features [34].

REFERENCES

[1] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[2] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[4] B. Peng, L. Zhang, and D. Zhang, "A survey of graph theoretical approaches to image segmentation," *Pattern Recognition*, vol. 46, no. 3, pp. 1020–1038, 2013.

[5] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1356–1363.

[6] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. M. Seitz, "Scene reconstruction and visualization from community photo collections," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1370–1390, 2010.

[7] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Video summarization and scene detection by graph modeling," *IEEE Transactions on circuits and systems for video technology*, vol. 15, no. 2, pp. 296–305, 2005.

[8] M. Witman, S. Ling, P. Boyd, S. Barthel, M. Haranczyk, B. Slater, and B. Smit, "Cutting materials in half: A graph theory approach for generating crystal surfaces and its prediction of 2d zeolites," *ACS central science*, vol. 4, no. 2, pp. 235–245, 2018.

[9] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[10] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, 2001, pp. 19–26.

[11] J. Wang, T. Jebara, and S.-F. Chang, "Semi-supervised learning using greedy max-cut," *Journal of Machine Learning Research*, vol. 14, no. Mar, pp. 771–800, 2013.

[12] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[13] V. Kolmogorov and R. Zabih, "What energy functions can be minimizedvia graph cuts?" *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 2, pp. 147–159, 2004.

[14] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts-a review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 7, pp. 1274–1279, 2007.

[15] D. J. Foster, D. Reichman, and K. Sridharan, "Inference in sparse graphs with pairwise measurements and side information," *arXiv preprint arXiv:1703.02728*, 2017.

[16] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[17] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American statistical association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[18] M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7327–7331, 2007.

[19] M. Mørup and M. N. Schmidt, "Bayesian community detection," *Neural computation*, vol. 24, no. 9, pp. 2434–2456, 2012.

[20] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 016107, 2011.

[21] T. Herlau, M. N. Schmidt, and M. Mørup, "Infinite-degree-corrected stochastic block model," *Physical review E*, vol. 90, no. 3, p. 032819, 2014.

[22] M. E. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Physical Review E*, vol. 94, no. 5, p. 052315, 2016.

[23] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures & Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[24] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, p. 016110, 2006.

[25] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[26] A. Trémeau and P. Colantoni, "Regions adjacency graph applied to color image segmentation," *IEEE Transactions on image processing*, vol. 9, no. 4, pp. 735–744, 2000.

[27] Y. Zhang, Z. Ghahramani, A. J. Storkey, and C. A. Sutton, "Continuous relaxations for discrete hamiltonian monte carlo," in *Advances in Neural Information Processing Systems*, 2012, pp. 3194–3202.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2994396, IEEE Transactions on Pattern Analysis and Machine Intelligence

[28] R. AlAhmad, "Products of incomplete gamma functions," *Analysis*, vol. 36, no. 3, pp. 199–203, 2016.

[29] G. Jameson, "The incomplete gamma functions," *The Mathematical Gazette*, vol. 100, no. 548, pp. 298–306, 2016.

[30] M. Hartmann, "Extending owen's integral table and a new multivariate bernoulli distribution," *arXiv preprint arXiv:1704.04736*, 2017.

[31] M. Sibuya, I. Yoshimura, and R. Shimizu, "Negative multinomial distribution," *Annals of the Institute of Statistical Mathematics*, vol. 16, no. 1, pp. 409–426, Dec 1964. [Online]. Available: https://doi.org/10.1007/BF02868583

[32] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," 1988.

[33] R. G. Parker and R. L. Rardin, *Discrete optimization*. Elsevier, 2014.

[34] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.

[35] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science advances*, vol. 3, no. 5, p. e1602548, 2017.

[36] M. Meilă, "Comparing clusterings by the variation of information," in *Learning theory and kernel machines*. Springer, 2003, pp. 173–187.

[37] I. Borg and P. Groenen, "Modern multidimensional scaling: theory and applications," *Journal of Educational Measurement*, vol. 40, no. 3, pp. 277–280, 2003.

[38] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. H. Hamann, "Discrete sibson interpolation," 2006.

[39] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[40] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.

[41] M. Morris and R. Rothenberg, "Hiv transmission network metastudy project: An archive of data from eight network studies, 1988–2001," 2011.

[42] B. Van Nguyen, H. Jung, D. Har, and K. Kim, "Performance analysis of a cognitive radio network with an energy harvesting secondary transmitter under nakagami-m fading," *IEEE Access*, vol. 6, pp. 4135–4144, 2018.

[43] M. N. Schmidt and M. Morup, "Nonparametric bayesian modeling of complex networks: An introduction," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 110–128, 2013.

[44] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[45] G. Jameson, "A simple proof of stirling's formula for the gamma function," *The Mathematical Gazette*, vol. 99, no. 544, pp. 68–74, 2015.

Petr Taborsky received his M.Sc. degree in Mathematics, Mathematical Statistics and Probability at Charles University, Prague. Currently he's PhD understudy at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. He's also heading Digital Labs_ at Telenor Danmark and his research interests include federated machine learning, neural networks, and complex network modeling.

Laurent Vermue received his M.Sc. degree in Industrial Engineering and Management at the Technical University of Berlin and MMSc. degree in Management Science and Engineering at the Tongji University. Currently he is a Ph.D. student at the Section for Statistics and Data Analysis and the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, complex network modeling and open research software.

Maciej Korzepa received his M.Sc. degree in Digital Media Engineering at the Technical University of Denmark. He is currently a Ph.D. student at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, intelligent interfaces, and complex network modeling.

Morten Mørup received his M.S. and Ph.D. degrees in applied mathematics at the Technical University of Denmark and he is currently Professor at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. He has been associate editor of IEEE Transactions on Signal Processing and his research interests include machine learning, neuroimaging, and complex network modeling.

6 SUPPLEMENTARY MATERIAL/APPENDIX

6.1 Theoretical results and proofs

Due to incomplete gamma recurrence (K-recurrence of $\Gamma's$) it is possible to rewrite the integral (8) according to the following lemma (6.1):

Lemma 6.1. *For $C \in \mathbb{N}$ and all real $\mu_i > 0, B_i > 0, i \in \{0, ..., C\}$ the following equalities hold*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{19}$$

$$= \frac{\Gamma(\mu_0)}{B_0^{\mu_0}} \prod_{c=1}^{C} \Gamma(\mu_c) - \sum_{m=1}^{C}$$

$$\frac{B_m^{\mu_m}}{B_0^{\mu_0}} \int_0^\infty \Gamma(\mu_0, B_0 x) e^{-B_m x} x^{\mu_m - 1} \prod_{\substack{c=1 \\ c \neq m}}^{C} \Gamma(\mu_c, B_c x) dx \tag{20}$$

$$= \underbrace{\int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx}_{\text{Part A}} + \sum_{m=1}^{C} \sum_{i=0}^{K-1} \tag{21}$$

$$\underbrace{\frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \int_0^\infty e^{-x(B_0 + B_m)} x^{\mu_0 + \mu_m + i - 1} \prod_{\substack{c=1 \\ c \neq m}}^{C} \Gamma(\mu_c, B_c x) dx}_{\text{Part B}} \tag{22}$$

Proof. The first equality follows directly by applying integration by parts using $u' = e^{-xB_0} x^{\mu_0 - 1}$ and $v = \prod_{c=1}^{C} \Gamma(\mu_c, B_c x)$ and the second by the use of (K-recurrence of $\Gamma's$). □

Using the above lemma, the aim is to select the number of recurrences K such that Part A is made arbitrarily small as illustrated by the underlying terms given in Figure 1. The following lemma proofs that Part A can indeed be made arbitrarily small.

Lemma 6.2. *For all real $\mu_i, B_i \in \mathbb{R}, \mu_i > 0, B_i > 0, i \in \{0, ..., C\}$ and constant $Q, Q \in \mathbb{R}, Q \geq 0$ following limit exists and holds:*

$$\lim_{K \to \infty} K^Q \times \int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx = 0 \tag{23}$$

Proof. We proof the lemma by showing that for every K there exists q such that integral over (q, ∞) is below threshold $\epsilon/2$ and consequently there exists K large enough such that the integral over $(0, q]$ is also below $\epsilon/2$.

Without loss of generality we assume $B_0 = 1$. (If $B_0 \neq 1$, we apply a transformation of variables $y = B_0 x$ on (23) and take $\epsilon \times B_0^{\mu_0}$ as a new epsilon and $\frac{B_c}{B_0}$ as new $B_c's$).

Denote $f(x, K) = \frac{e^{-x} x^{\mu_0 + K - 1}}{\Gamma(\mu_0 + K)}$, which is a density function of the gamma distribution with shape parameter $\mu_0 + K$ and rate 1 such that $f(0) = 0$. For every $K, K \geq 1$ $f(x, K)$ is positive and increasing on $(0, m(K))$, where $m(K)$ is the mode $\mu_0 + K - 1$, controlled by K which follows well known characteristics of the gamma distribution.

Upper bound on $(0, q]$: Denote the regularized upper incomplete gamma function $\frac{\Gamma(\alpha, bx)}{\Gamma(\alpha)}$ which can be written as 1 - the cumulative distribution function [29] of the gamma distribution and is therefore positive, decreasing on $\mathbb{R}^+$, and bounded from the top by 1.

For every $q \in \mathbb{R}, q > 0$ and for every $K \in \mathbb{N}^+$ large enough to ensure that q is below the mode of the gamma distribution with density $f(x, K)$, that means $K > q - \mu_0 + 1$, the following inequality holds:

$$K^Q \times \int_0^q \frac{e^{-x} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx$$

$$\leq q K^Q f(q, K) \prod_{c=0}^{C} \Gamma(\mu_c) \tag{24}$$

We make use of Stirling's formula [45]:

$$\Gamma(x) \sim (2\pi)^{\frac{1}{2}} x^{x - \frac{1}{2}} e^{-x} \text{ as } x \to \infty, \tag{25}$$

where notation $g(x) \sim h(x)$ means that $\frac{g(x)}{h(x)} \to 1$ as $x \to \infty$. Taking $K \to \infty$ and using Stirling's formula above on $f(q, K)$ gives us existence of the following limit:

$$\lim_{K \to \infty} q K^Q f(q, K) * \prod_{c=0}^{C} \Gamma(\mu_c) = 0 \tag{26}$$

From (ϵ, δ)-limit definition of (26) we get that for every $q \in \mathbb{R}, q > 0$ and every $\epsilon' > 0, \epsilon \in \mathbb{R}^+$ there exists $K' \in \mathbb{N}, K' > 0$ such that for every $K \geq \max(K', q - \mu_0 + 1), K \in \mathbb{N}$ equation (24) $\leq \epsilon'$.

Upper bound on $[q, \infty)$: For every $q > 0$ we have from definition that all upper incomplete gamma functions are smooth and decreasing on $[q, \infty)$ and can be bounded from top by its value at q. We thereby get the following upper bound by taking the integral over region from q to ∞, recognizing the upper incomplete gamma function and using $\frac{\Gamma(a,x)}{\Gamma(a)} \leq 1$ (which trivially follows from the definition of $\Gamma(a, x)$):

$$K^Q \int_q^\infty \frac{e^{-x} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{27}$$

$$\leq K^Q \frac{\Gamma(\mu_0) \prod_{c=1}^{C} \Gamma(\mu_c, B_c q)}{\Gamma(\mu_0 + K))} \int_q^\infty e^{-x} x^{\mu_0 + K - 1} dx$$

$$= K^Q \frac{\Gamma(\mu_0 + K, q) \prod_{c=1}^{C} \Gamma(\mu_c, B_c q)}{\Gamma(\mu_0 + K)} \Gamma(\mu_0)$$

$$\leq K^Q \prod_{c=1}^{C} \Gamma(\mu_c, B_c q) \Gamma(\mu_0) \tag{28}$$

Following limit exists and follows directly from definition of incomplete gamma function (7):

$$\lim_{q \to \infty} \prod_{c=1}^{C} \Gamma(\mu_c, B_c q) \Gamma(\mu_0) = 0 \tag{29}$$

From the above limit we can now select q such that the error is below $\epsilon/2$ and for this q we select K such that (26) is also below $\epsilon/2$.

□

Theorem 6.3. *For every $C \in \mathbb{N}^+$, $\mu_i, B_i \in \mathbb{R}, \mu_i > 0, B_i > 0$ for $i \in \{1, ..., C\}$ and $K \in \mathbb{N}^+$ following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{30}$$

$$= \sum_{m_1=1}^{C} \sum_{\substack{m_2=1,\\ m_2 \neq m_1}}^{C} \cdots \sum_{\substack{m_C=1,\\ m_C \neq m_1, ..., m_{C-1}}}^{C} \sum_{i_1=0}^{K-1} \cdots \sum_{i_C=0}^{K-1}$$

$$\prod_{w=1}^{C} \frac{B_{m_w}^{\mu_{m_w}} (B_0 + \sum_{j=1}^{w-1} B_{m_j})^{i_w}}{\left(\mu_0 + \sum_{j=1}^{w-1} (\mu_{m_j} + i_j)\right)^{i_w + 1}}$$

$$\times \frac{\Gamma\left(\mu_0 + \sum_{j=1}^{C} (\mu_{m_j} + i_j)\right)}{(B_0 + \sum_{j=1}^{C} B_{m_j})^{(\mu_0 + \sum_{j=1}^{C} (\mu_{m_j} + i_j))}} \tag{31}$$

$$+ E(K), \tag{32}$$

where $E(K)$ satisfies $\lim_{K \to \infty} E(K) = 0$.

Proof. We proof this by induction on C.

Case C=1: According to lemma 6.2 (for Q=0) we can for every given $\epsilon > 0$ find K′ such that Part A in lemma 6.1 will be below ϵ for all $K > K'$ whereas the infinite integral in Part B reduces to the Gamma function and we thereby obtain:

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \Gamma(\mu_1, B_1 x) dx$$

$$= \epsilon + \sum_{i=0}^{K-1} \frac{B_0^i B_1^{\mu_1}}{(\mu_0)^{\dot{i+1}}} \frac{\Gamma(\mu_0 + \mu_1 + i)}{(B_0 + B_1)^{(\mu_0 + \mu_1 + i)}}. \tag{33}$$

Induction step from C to $C+1$: We again apply lemma 6.1 (for Q=0) on $C+1$, and thereby obtain:

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C+1} \Gamma(\mu_c, B_c x) dx \tag{34}$$

$$= \underbrace{\int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C+1} \Gamma(\mu_c, B_c x) dx}_{\text{Part C+1}} \tag{35}$$

$$+ \sum_{m=1}^{C+1} \sum_{i=0}^{K-1} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \tag{36}$$

$$\underbrace{\int_0^\infty e^{-x(B_0 + B_m)} x^{\mu_0 + \mu_m + i - 1} \times \prod_{\substack{c=1 \\ c \neq m}}^{C+1} \Gamma(\mu_c, B_c x) dx}_{\text{Part C}} \tag{37}$$

Part C+1 in the above captures the error $E_{C+1}(K)$ introduced reducing the integral involving C+1 to the integral involving C incomplete gamma functions (due to the $c \neq m$ in the sum) given in Part C in the above, and according to the induction we can assume the theorem holds for Part C up to the error term $E_C(K)$:

$$E_{C+1}(K) + \sum_{m=1}^{C+1} \sum_{i=0}^{K-1} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \times E_C(K)$$

$$\leq E_{C+1}(K) + K(C+1) \max_{i \in \{1, ..., K-1\}} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \times E_C(K). \tag{38}$$

Next we show that elements

$$Q(K) := (C+1) \max_{i \in \{1, ..., K-1\}} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \tag{39}$$

are bounded from the top when $K \to \infty$. To see this we note that it can be split into $\max_{i \leq L}(\dots)$ that is maximum over a finite set of integers lower than some $L \in \mathbb{N}$ (that always has finite upper bound) and maximum over $\{i \geq L\}$ such that maximum over this region is reached by the first index L (that follows from limit $\lim_{i \to \infty} (C+1) \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} = 0$ which we get from Stirling formula (25) applied on $\frac{\Gamma(\mu_0}{\Gamma(\mu_0 + i)}$ in Pochhammer symbol in (39)). So for all sufficiently large K such that $K \geq L$ we can bound $Q(K)$ by a constant we denote $Q^{(1)}$.

Notably, steps from (34) to (37) using lemma 6.1 together with formula for sum of geometric finite sum reveal that total error term of (38) including integral $E_{C+1}(K)$ and sums of $E_C(K)$ comprises not more than $\frac{1 - (K(C+1))^{(C+1)}}{1 - K(C+1)} = 1 + K(C+1) + (K(C+1))^2 + \cdots + (K(C+1))^C$ integrals of the same form as (23) multiplied by fractions of the same structure as (39). Using same logic as for $Q^{(1)}$ earlier that maximized fractions from 1 induction step and again leveraging Stirling formula (25), [45], we get that there exists L such that for all sufficiently large $K \geq L$ we can bound all these fractions from the top. We take their maximum, denoted $Q^{(2)}$.

Since $Q^{(1)}$ is now maximized withing $Q^{(2)}$ we can bound the total approximation error of (34) by:

$$\leq Q^{(2)} \times K \left(\frac{1 - (K(C+1))^{(C+1)}}{1 - K(C+1)} \right) \times$$

$$\max_{i \in \{0, ..., I\}} \left(\int_0^\infty \frac{e^{-xB_0'(i)} x^{\mu_0'(i) + K - 1}}{(\mu_0'(i))^{\dot{K}}} \prod_c \Gamma(\mu_c'(i), B_c' x(i)) dx \right)$$

where we omitted exact expression for the sake of simplicity and rather used symbolic notation instead for all combinations of parameters $\mu'(i), B'(i)$ and finite number of indexes (simplified as i).

Application of lemma 6.2 for the choice of Q such that $K(\frac{1 - (K(C+1))^{(C+1)}}{1 - K(C+1)}) \leq K^Q$ with each integral in max and taking $K \to \infty$ brings the limit of this upper bound to 0. That concludes the proof. □

6.2 Large Scale settings

Assuming hyperparameter priors can be chosen to be integer values following theorem presents exact evaluation of the integral (8):

Theorem 6.4. *For* $C \in \mathbb{N}^+$, $\mu_i \in \mathbb{N}^+$ *and* $B_i \in \mathbb{R}^+$, $i \in \{0, ..., C\}$ (8) *is proportional to the cumulative distribution function of the Negative Multinomial (NMn) distribution and the following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{i \in \{1,\dots,C\}} \Gamma(\mu_i, B_i x) dx$$

$$= \frac{\prod_{c=0}^{C} \Gamma(\mu_c)}{B_0^{\mu_0}} \times \tag{40}$$

$$\times \sum_{i_1=0}^{\mu_1-1} \cdots \sum_{i_c=0}^{\mu_c-1} \frac{\Gamma(\mu_0 + i_1 + \ldots + i_c)}{\Gamma(\mu_0) i_1! \ldots, i_c!} \left(\frac{B_0}{B}\right)^{\mu_0} \prod_{c=1}^{C} \left(\frac{B_c}{B}\right)^{i_c}, \tag{41}$$

where $B := \sum_{i=0}^{C} B_i$

Proof. Equality follows from applying (K-recurrence of $\Gamma's$) from section K-recurrence of $\Gamma's$ on $\Gamma(\mu, x) = \Gamma(1 + (\mu - 1), x)$ setting $K = \mu_i - 1$ and $a = 1$ and further simplifying as follows:

$$\frac{\Gamma(1 + (\mu - 1), x)}{\Gamma(\mu)} = \Gamma(1, x) + x e^{-x} \sum_{i=0}^{\mu-2} \frac{x^i}{\Gamma(i+2)}$$

$$= e^{-x} + e^{-x} \sum_{i=1}^{\mu-1} \frac{x^i}{\Gamma(i+1)} = e^{-x} \sum_{i=0}^{\mu-1} \frac{x^i}{\Gamma(i+1)}$$

('Euler and inc. gamma') (42)

where $a^{\dot{n}}$ is the Pochhammer symbol (a.k.a. "rising factorial") defined as $a^{\dot{n}} = \Gamma(a+n)/\Gamma(a)$. We used the fact that $\Gamma(1, x) = e^{-x}$ (trivially from definition of upper incomplete gamma $\Gamma(a, x)$).

Result follows from changing the order of integration and integrating out x. That leads to gamma functions (by definition) and gives formula to be proven. Alternatively one can recognize inner integral over x as a Laplace transform of $x^{\mu_0 + i_1 + \ldots + i_c}$. □

Note: A similar but infinite sum formula of theorem (6.4) also holds for real parameters μ. It can be shown by use of binomial series expansion $(1 + x)^\alpha, \alpha \in \mathcal{R}^+$ and Laplace transform. However, in a result we obtain infinite series.

6.3 Comparison of sampling vs. optimization

To show the advantage of using the inference method described in section 3 over an optimization heuristic as proposed in [20], we compare these two methods on the HIV-1 network (ref. 4.2). Figure 10 shows the Box-Whisker-Plots for the best obtained cut of each run/chain for each method based on the log-likelihood $p(\boldsymbol{z}|G)$. The two left Box-Whisker-Plots show the results of the optimization heuristic, whereas the third Box-Whisker-Plot shows the results for pure sampling without any optimization. In this direct comparison the optimization obtains better results. However, if the sampling method is followed by the optimization heuristic as proposed in this paper, we achieve much better results compared to pure optimization. As argued in Section 3, the optimization heuristic seems to get stuck in local maxima. This can be observed in the two left Box-Whisker-Plots. Running more optimization initiations does not help to improve the results, since each run apparently gets stuck in the same local maxima, which would explain the marginal difference of obtained best cuts in log-likelihood between 100 runs and 1000 runs. In contrary, the sampler will likely focus on some high density region of the posterior, but it will still explore multiple modes within that region, which can be seen on the wide range covered by its Box-Whisker-Plot. When these obtained best samples are subsequently used with the optimization heuristic, the obtained best cuts are significantly better than the cuts obtained using the optimization heuristic only, as to be seen in the fourth Box-Whisker-Plot. Interestingly, when using the optimization heuristic before sampling to obtain a starting point, we observe that sampling with or without a final optimization obtains the same best cuts as the pure optimization. This is most likely due to the optimization finding an extreme local maximum, as mentioned above, which even the sampler cannot escape.

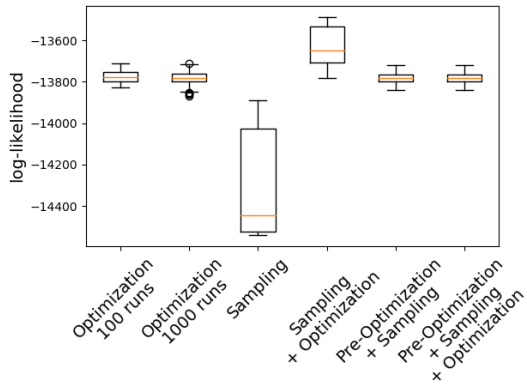
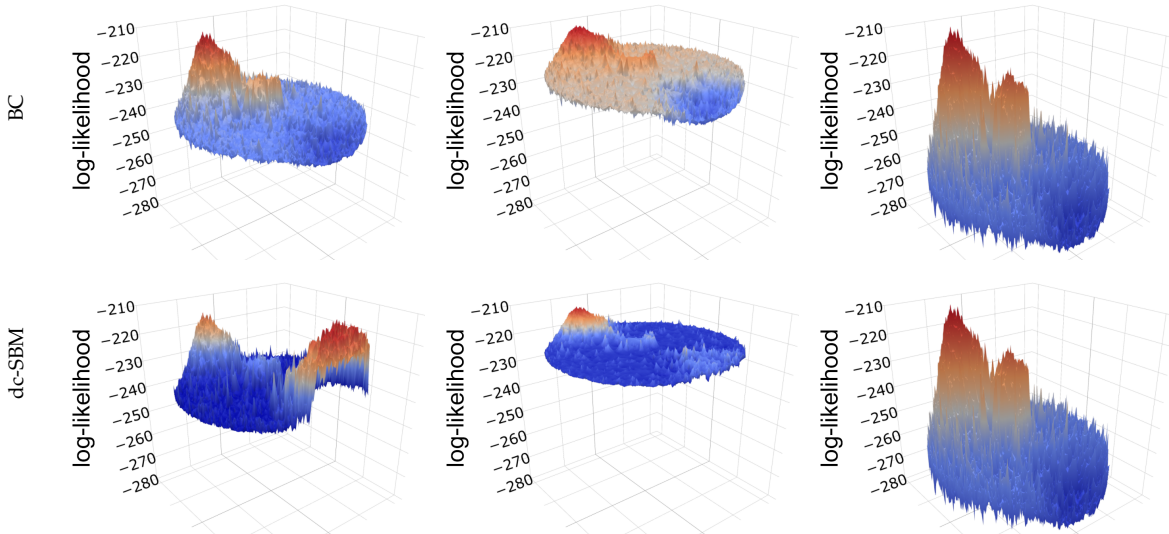


Fig. 10: Comparison of sampling and pure optimization according to [20] on the HIV-1 network through Box-Whisker-Plots of the best obtained cut of each chain/run for each method based on the log-likelihood $p(\boldsymbol{z}|G)$. The sampling results are based on 100 chains and 1000 samples per chain. Sampling followed by optimization shows superior performance compared to only using the optimization or using the optimization before sampling.

6.4 Influence of priors and constraints

In figure 11 the influence of the prior on both models considering also a weak and strongly informative prior on community structure is investigated. The non-informative prior allows the models to find their preferred modes. In this case the dc-SBM clearly exhibits the strongest support in the non-community structure region as discussed in section 4.2.1 . Even though the medium community enforcing prior

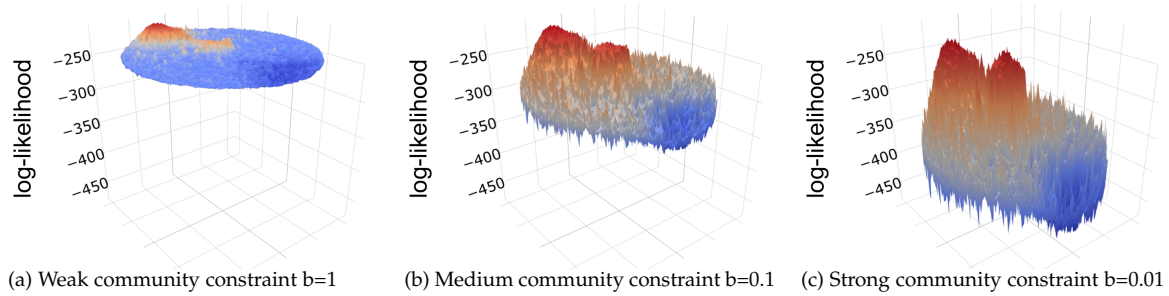


(a) Non-informative prior ($\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$)

(b) Medium community enforcing prior ($\alpha_{in} = 19.5, \alpha_{out} = 39, \beta_{in} = 144.5, \beta_{out} = 289$)

(c) Strong community enforcing prior ($\alpha_{in} = 39, \alpha_{out} = 0.01, \beta_{in} = 144.5, \beta_{out} = 289$)

Fig. 11: Comparison of different priors for α and β on the solution landscape of the Karate network ($b = 1$ for the BC model). The solution landscape was created using all posterior samples generated by running both models with 15 chains and 100 posterior samples for each configuration.



(a) Weak community constraint b=1

(b) Medium community constraint b=0.1

(c) Strong community constraint b=0.01

Fig. 12: Comparison of different constraints (b) on the solution landscape of the Karate network with non-informative prior ($\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$). The solution space is based on the samples generated in figure 11

lowers the support for the non-community structure region of the solution space, it still maintains a mode in the non-desired region. Only the strong community enforcing prior forces the dc-SBM to abandon these regions. However, imputing such a strong prior belief effectively makes the prior the posterior distribution, which is the reason that in this case the solution landscapes for both models look identical. In conclusion, the constraint imposed by BC distinguishes itself by allowing to explore the posterior with a non-informative prior while still enforcing community-structure.

The role of the b parameter, i.e. the strength of the constraint can be seen in figure 12. The lower b, the more the model enforces community structure the more it drops those regions of the solution space not supporting the community structure. Accordingly, b sets the boundaries of the constraints that can also be learned as part of the model inference.

6.5 Additional Experiment

(a) original image (b) norm cut, C=2 (c) unconstrained cut, C=2 (d) bayesian cut (BC), C=2

Fig. 13: **Coffee:** Resulting cut of BC model for C=2 (d) segments compared with unconstrained dc-SBM with shared η_{out} as well as spectral Norm cut. As experiments in the main body also these results demonstrate on par or better results of BC against referenced methods. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each.

Appendix E

Multiblock PLS: Block dependent prediction modeling for Python

---

Andreas Baum and **Laurent Vermue**. Multiblock PLS: Block dependent prediction modeling for Python. *Journal of Open Source Software*, 4(34):1190, 2019

The Journal of Open Source Software

Multiblock PLS: Block dependent prediction modeling for Python

Andreas Baum[1] **and Laurent Vermue**[1]

1 Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads 324, DK-2800 Kgs. Lyngby, Denmark

DOI: 10.21105/joss.01190

Software

- Review
- Repository
- Archive

Submitted: 06 January 2019
Published: 10 February 2019

Introduction

Partial Least Squares (PLS) regression is a statistical method for supervised multivariate analysis. It relates two data blocks $\mathbf{X}$ and $\mathbf{Y}$ to each other with the aim of establishing a prediction model. When deployed in production, this model can be used to predict an outcome $\mathbf{y}$ from a newly measured feature vector $\mathbf{x}$. PLS is popular in chemometrics, process control and other analytic fields, due to its striking advantages, namely the ability to analyze small sample sizes and the ability to handle high-dimensional data with cross-correlated features (where Ordinary Least Squares regression typically fails). In addition, and in contrast to many other machine learning approaches, PLS models can be interpreted using its latent variable structure just like principal components can be interpreted for a PCA analysis.

Multivariate data is often structured in blocks, e.g. $\mathbf{X}_1$, $\mathbf{X}_2$, … , $\mathbf{X}_i$. This could mean that one has obtained data from two different analytic methodologies for a similar set of samples, which may indicate two totally independent feature spaces. In such cases it is often important to understand how each data block contributes to the prediction of $\mathbf{Y}$. Examples for data measured in blocks could be the following.

1. It can be of interest to relate patient clinical records to data obtained through different high-throughput omics measurements. These data could typically be structured in blocks referring to genomics, transciptomics, proteomics, metabolomics etc.
2. Spectroscopic methods are useful to predict and assure food quality parameters. When measuring food samples by several different spectroscopic methods, e.g. by applying near infrared and UV-vis spectroscopy, it is meaningful to combine the data blocks to obtain reliable prediction models.
3. A process utilizing fermentation technology is typically carried out in several sequential production phases, i.e. seed and main fermentation phase. If sensor data is available for all production phases it is meaningful to include these as individual data blocks when establishing prediction models for quality control parameters, such as product yield.

Several Data Fusion approaches were proposed to establish combined prediction models from such multiblock data (Li, Wu, & Ngom, 2016). One of the proposed methods is Multiblock-PLS (MB-PLS) (Westerhuis, Kourti, & MacGregor, 1998). It is closely related to PLS regression, but instead of obtaining an interpretative model for the entire (concatenated) data matrix $\mathbf{X}$ one obtains model parameters for each individual data block $\mathbf{X}_i$. Furthermore, it provides a relative importance measure, i.e. expressing how

much each block $\mathbf{X}_i$ contributes to the prediction of $\mathbf{Y}$. Subsequently, this information can be used to recognize block specific patterns in the data.

At the current stage software packages for MB-PLS exist for Matlab (http://www.models.life.ku.dk/MBToolbox) and R (Bougeard & Dray, 2018). In the following sections we give a brief introduction to the statistical method and its implementation. The package is distributed under the BSD-3-Clause license and made available at https://github.com/DTUComputeStatisticsAndDataAnalysis/MBPLS together with several introductory Jupyter notebook examples. It is also available as `pip` installable Python package (https://pypi.org/project/mbpls/) and comes with a Read-the-Docs documentation (https://mbpls.readthedocs.io).

Methods

The MB-PLS package can be utilized for PLS and MB-PLS regression. The statistical background is briefly introduced in the following. More detailed information is given in the `mbpls` help of the Python package (https://mbpls.readthedocs.io/en/latest/mbpls.html).

PLS

PLS was introduced by S. Wold, Ruhe, Wold, & III (1984) and aims at finding a suitable subspace projection $\mathbf{w}$ which maximizes co-variance between a so called score vector $\mathbf{t}$ and a response vector $\mathbf{y}$ that will yield a least squares solution. The formal PLS criterion for univariate responses $\mathbf{y}$ is given in eq. 1.

$$\underset{\mathbf{w}}{\operatorname{argmax}} \left(cov(\mathbf{t}, \mathbf{y}) \middle| min \left(\sum_{i=1}^{I} \sum_{j=1}^{J} (\mathbf{x}_{ij} - \mathbf{t}_i \mathbf{w}_j)^2 \right) \wedge \|\mathbf{w}\| = 1 \right) \tag{1}$$

This procedure is typically repeated to find K latent variables (LV). In each latent variable step, the score vector $\mathbf{t}_k$ is subsequently projected onto its respective matrix $\mathbf{X}_k$ to find the loading vector $\mathbf{p}_k$ (eq. 2). Once found, $\mathbf{X}_k$ is deflated by the explained variance (eq. 3) and the next latent variable $k+1$ can be calculated using $\mathbf{X}_{k+1}$.

$$\mathbf{p}_k = \mathbf{X}_k \mathbf{t}_k \tag{2}$$

$$\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{t}_k \mathbf{p}_{k^T} \tag{3}$$

Algorithms to perform PLS regression include the **N**onlinear **I**terative **PA**rtial **L**east **S**quares (NIPALS) (S. Wold et al., 1984), **UNI**versal **PA**rtial **L**east **S**quares (UNIPALS) (Dunn III, Scott, & Glen, 1989), Kernel UNIPALS (Lindgren, Geladi, & Wold, 1993; Rännar, Geladi, Lindgren, & Wold, 1995; Rännar, Lindgren, Geladi, & Wold, 1994) and SIMPLS algorithm (de Jong, 1993). While NIPALS represents an iterative approach, the other algorithms are based on Singular Value Decomposition (SVD). All the above mentioned algorithms are implemented in the MB-PLS package. Benchmark results and comparisons to other Software packages are provided below.

MB-PLS

MB-PLS can be understood as an extension of PLS to incorporate several data blocks $\mathbf{X}_1, ..., \mathbf{X}_i$, which all share a common sample dimension. The prediction accuracy does not deviate from normal PLS, if all data blocks were concatenated into a single block, but the

The Journal of Open Source Software

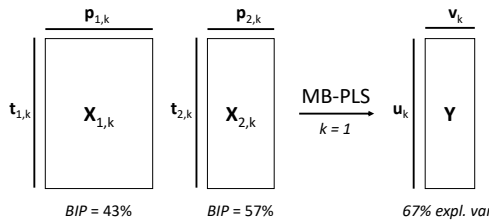


Figure 1: The figure illustrates the extraction of a single LV ($k = 1$). MB-PLS offers extra exploratory features for each block, i.e. block scores, block loadings and block importances (BIP).

advantage of MB-PLS is to gain extra model interpretability concerning the underlying block structure of the data. For each LV one obtains extra block scores, block loadings and block importances (BIP). The extraction of a single LV using MB-PLS is illustrated in figure 1. The results are read in a fashion that 67% variance in $\mathbf{Y}$ are explained by the first LV. The two blocks $\mathbf{X}_1$ and $\mathbf{X}_2$ contribute to the prediction of the 67% with their relative BIPs, 43% and 57%, respectively. More important blocks result in more influential block loadings and contribute stronger to the prediction of $\mathbf{Y}$. Hence, interpretation of patterns among block scores with high importance are recommended.

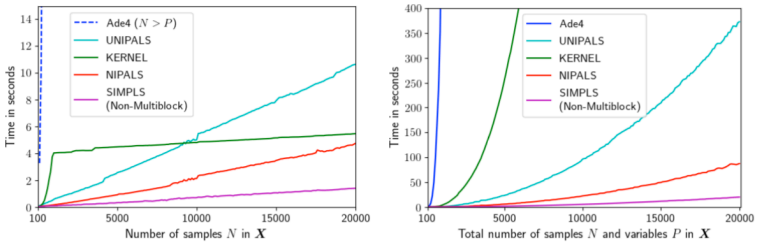
To assert that the BIP is a meaningful indicator it is necessary to standardize the data prior to MB-PLS analysis. When standardization is employed all features in all blocks have a variance of 1. For post-hoc interpretation of the loadings an inverse transformation is carried out to ensure straight forward interpretation of the results.

Software and Implementation

The package is written in pure Python 3. In its core it builds on Numpy and Scipy for efficient data handling and fast mathematical operations of big data-sets. To achieve a fast implementation all algorithms using SVD employ Scipy's partial SVD capability, i.e. by only calculating the first singular value at each PLS iteration. Multiple matrix multiplications use the optimized Numpy multi-array multiplication. In addition, the MB-PLS implementation can handle missing data without prior imputation based on the sparse NIPALS algorithm by H. Martens & Martens (2001). The overall code design follows the structure and philosophy of Scikit-learn (Pedregosa et al., 2011). Therefore, objects are instantiated in a Scikit-learn manner and can be accessed with the same methods, i.e. fit, predict, transform and score. Furthermore, Scikit-learn's base classes and validation methods are incorporated. As a result, all objects are fully compatible to Scikit-learn and, thus, allow the use of model selection functions, e.g. cross validation and grid search, as well as a processing pipeline. For exploratory analysis, each fitted model contains a custom `plot` method that the fitted model attributes in a meaningful manner using Matplotlib, which allows a straight forward evaluation of the MBPLS results without requiring any additional coding.

Benchmark

To compare the four algorithms, the run-times are analyzed for different data-set sizes with two basic shapes, i.e. non-symmetric shapes with more samples than variables ($N > P$) or vice versa ($N < P$) and symmetric shapes ($N = P$). To simulate the multiblock and multivariate behaviour, each data-set is split into two $\mathbf{X}$-blocks with the size $N \times \frac{P}{2}$ and



(a) Non-Symmetric $\mathbf{X}$ matrices with $P = 1000$ (b) Symmetric $\mathbf{X}$ matrices with $N = P$

Figure 2: Comparison of run-times based on different data-set sizes

accompanied by a $\mathbf{Y}$-block of size $N \times 10$. The data is randomly generated for each run, so that the obtained times exhibit worst-case behaviour, since there are no actual latent structures. All algorithms are set to find the first 20 LVs and are run three times for each data-set size on a machine with two Intel ® Xeon ® X5650 @ 2.67 GHz processors and 48 GB RAM.

As to be seen in both both plots of figure 2 all algorithms implemented in the Python mbpls package substantially outperform the above mentioned R-package Ade4-MBPLS by Bougeard & Dray (2018), which was run on the same machine. In general NIPALS is the fastest multiblock algorithm that is only outperformed by the SIMPLS algorithm, which only supports single block PLS. However, figure 2a shows how the KERNEL algorithm performs progressively better in cases where $N >> P$ or $N << P$. As to be seen in this plot, the runtime of this algorithm is a combination of an exponential part given by the right plot and dependent on $min(N, P)$ and a linear part defined by $diff(N, P)$. Due to the exponential part, $min(N, P)$ has to be considered carefully when choosing the KERNEL algorithm over NIPALS.

An important feature of this Python mbpls package is its invariance to shape rotations, i.e. it obtains the same run-times for both $N > P$ and $N < P$ given the same ratio $\frac{N}{P}$ and its respective inverse, which e.g. is not the case for the R-package.

Acknowledgement

The authors gratefully acknowledge the financial support through the BioPro (Innovationsfonden project nr. 10513) and DABAI (Innovationsfonden project nr. 10599 and 10577) project.

References

Bougeard, S., & Dray, S. (2018). Supervised multiblock analysis in r with the ade4 package. *Journal of Statistical Software*, *86*(1), 1–17. doi:10.18637/jss.v086.i01

de Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems*, *18*(3), 251–263. doi:10.1016/0169-7439(93)85002-X

Dunn III, W. J., Scott, D. R., & Glen, W. G. (1989). Principal components analysis and partial least squares regression. *Tetrahedron computer methodology*, *2*(6), 349–376. doi:10.1016/0898-5529(89)90004-3

Li, Y., Wu, F.-X., & Ngom, A. (2016). A review on machine learning principles for multi-view biological data integration. *Briefings in Bioinformatics*, *19*(2), 325–340. doi:10.1093/bib/bbw113

Lindgren, F., Geladi, P., & Wold, S. (1993). The kernel algorithm for pls. *Journal of Chemometrics*, *7*(1), 45–59. doi:10.1002/cem.1180070104

Martens, H., & Martens, M. (2001). *Multivariate analysis of quality: An introduction*. Chichester: Wiley.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Rännar, S., Geladi, P., Lindgren, F., & Wold, S. (1995). A pls kernel algorithm for data sets with many variables and few objects. Part ii: Cross-validation, missing data and examples. *Journal of Chemometrics*, *9*(6), 459–470. doi:10.1002/cem.1180090604

Rännar, S., Lindgren, F., Geladi, P., & Wold, S. (1994). A pls kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm. *Journal of Chemometrics*, *8*(2), 111–125. doi:10.1002/cem.1180080204

Westerhuis, J. A., Kourti, T., & MacGregor, J. F. (1998). Analysis of multiblock and hierarchical pca and pls models. *Journal of Chemometrics*, *12*(5), 301–321. doi:10.1002/(SICI)1099-128X(199809/10)12:5<301::AID-CEM515>3.0.CO;2-S

Wold, S., Ruhe, A., Wold, H., & III, W. J. D. (1984). The collinearity problem in linear regression. The partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, *5*(3), 735–743. doi:10.1137/0905052

Appendix F

Unraveling fermentation data – a Novozymes case study

Andreas Baum, Rasmus Devantier, **Laurent Vermue**, Rayisa Moiseyenko, and Thomas Martini Jørgensen. Unraveling fermentation data - a Novozymes case study. In *Recent Advances in Fermentation Technology (RAFT 2017)*, 2017

Unraveling fermentation process data – a Novozymes case study

Andreas Baum[1], Rasmus Devantier[2], Laurent Vermue[1], Rayisa Moiseyenko[1] and Thomas Martini Jørgensen[1]

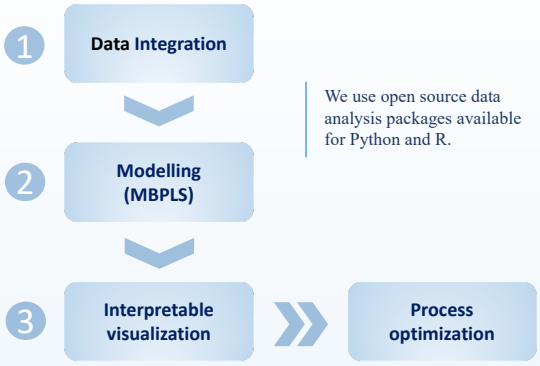
[1] DTU Compute, Technical University of Denmark, 2800 Lyngby, Denmark
[2] Novozymes A/S, Fermentation Pilot Plant, 2880 Bagsvaerd, Denmark

Abstract

Industrial fermentation processes are monitored using a variety of sensors. Typically, measurements are taken through-out the entire production process. Production may be carried out under supervision of different operators (operator variation), on different sites (global variation), in different buildings and/or in different tanks (local variation). However, up to now processes are mainly controlled according to traditional recipes and experience.

The massive amount of available process data combined with multivariate statistics enable new "Big Data" approaches to identify and extract significant patterns in the process control which may lead to considerable improvements with respect to process efficiency and yield maximization.

In our case study we propose a three-step approach to unravel the data.

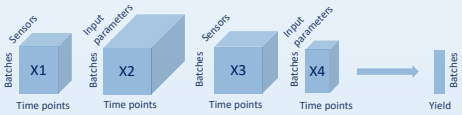


We use open source data analysis packages available for Python and R.

The Data

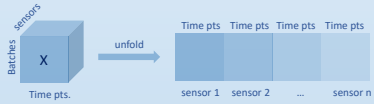
Production fermentations typically undergo several process steps, e.g. seed fermentation and main fermentation phases etc. Each process step is monitored by a number of sensors (performance). In addition, input parameters can be set to control the process phases.

Hence, fermentation process data can be understood as to be organized in distinct blocks.



Data blocks may vary in used sensor types and time points recorded, but share the dimension of production batches

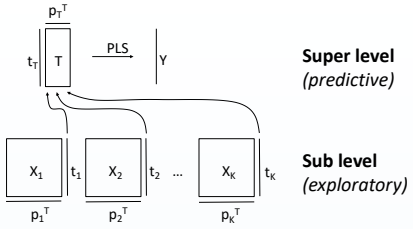
X1 main fermentation – performance
X2 main fermentation – input
X3 seed fermentation – performance
X4 seed fermentation – input



Each data block can be understood as a three-way array. For bilinear modeling purposes the data "cube" can be unfolded across its sensor dimension

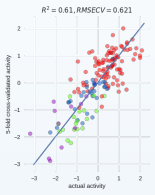
The Model

To enhance model interpretability we propose the use of Multiblock Partial Least Squares regression (MBPLS). Although no improvement of prediction is expected (when comparing to PLS), the model can be interpreted in a more intuitive fashion.



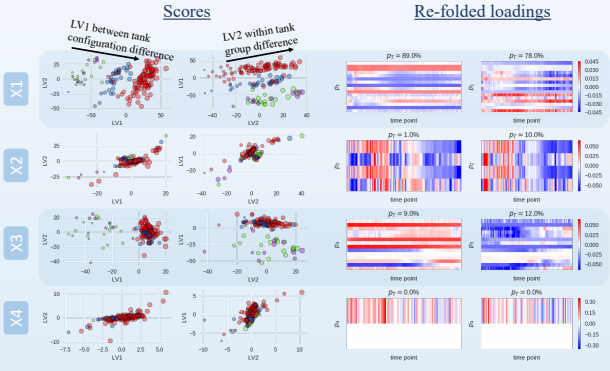
The super level is used to find predictive importance of the data blocks. The Sub level is useful for interpretation of the important variable patterns in the individual blocks

Results



MBPLS activity prediction model; 5-fold Cross Validation indicated 2 LVs

Red – tank group 1
Blue – tank group 2
Tank configuration 1
Green – tank group 3
Purple – tank group 4
Tank configuration 2



LV1 60% explained var. in y
LV2 14% explained var. in y
LV1 60% explained var. in y
LV2 14% explained var. in y

Taking the score and loading plots into account the following major findings could be concluded:

1. The major activity difference (60% variance) could be described as difference between tank configurations 1 and 2.
2. The most important blocks to describe this discrepancy were X1 (89%) and X3 (9%)
3. Within tank group variations of the final activity could be explained by loading plots from LV2

Conclusion

PLS offers a straight forward strategy to predict process yield, i.e. enzyme activity. Using Multiblock PLS the interpretability of the latent variable structure enables potential process optimization on a single parameter level.

Acknowledgement

We gratefully acknowledge financial support from BioPro (biopro.nu)

Bibliography

[1] Emmanuel Abbe. Community Detection and Stochastic Block Models: Recent Developments. Technical report, 2018. URL https://jmlr.csail.mit.edu/papers/volume18/16-480/16-480.pdf.

[2] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *The VLDB Journal*, 24(4):557–581, 8 2015. ISSN 1066-8888. doi: 10.1007/s00778-015-0389-y. URL http://link.springer.com/10.1007/s00778-015-0389-y.

[3] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. Detecting data errors. *Proceedings of the VLDB Endowment*, 9 (12):993–1004, 8 2016. ISSN 21508097. doi: 10.14778/2994509.2994518. URL http://dl.acm.org/citation.cfm?doid=2994509.2994518.

[4] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Data Profiling. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1747–1751, New York, NY, USA, 5 2017. ACM. ISBN 9781450341974. doi: 10.1145/3035918.3054772. URL https://dl.acm.org/doi/10.1145/3035918.3054772.

[5] Farahnaz Akrami, Lingbing Guo, Wei Hu, and Chengkai Li. Re-evaluating Embedding-Based Knowledge Graph Completion Methods. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1779–1782, New York, NY, USA, 10 2018. ACM. ISBN 9781450360142. doi: 10.1145/3269206.3269266. URL https://dl.acm.org/doi/10.1145/3269206.3269266.

[6] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens

Lehmann. Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models under a Unified Framework. *Under review at IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[7] Doan AnHai, Halevy Alon, and Ives Zachary. *Principles of Data Integration*. Elsevier Inc., 2012. ISBN 9780124160446. doi: 10.1016/C2011-0-06130-6.

[8] R.C. Atkinson and R.M. Shiffrin. Human Memory: A Proposed System and its Control Processes. *Psychology of Learning and Motivation*, 2:89–195, 1 1968. ISSN 0079-7421. doi: 10.1016/S0079-7421(08)60422-3. URL `https://www.sciencedirect.com/science/article/pii/S0079742108604223`.

[9] Andreas Baum and Laurent Vermue. Multiblock PLS: Block dependent prediction modeling for Python. *Journal of Open Source Software*, 4(34):1190, 2019.

[10] Andreas Baum, Rasmus Devantier, Laurent Vermue, Rayisa Moiseyenko, and Thomas Martini Jørgensen. Unraveling fermentation data-a Novozymes case study. In *Recent Advances in Fermentation Technology (RAFT 2017)*, 2017.

[11] Andreas Baum, Malgorzata Dominiak, Silvia Vidal-Melgosa, William G T Willats, Karen M Søndergaard, Per W Hansen, Anne S Meyer, and Jørn D Mikkelsen. Prediction of pectin yield and quality by FTIR and carbohydrate microarray analysis. *Food and Bioprocess Technology*, 10(1):143–154, 2017.

[12] Max Berrendorf, Evgeniy Faerman, Valentyn Melnychuk, Volker Tresp, and Thomas Seidl. Knowledge Graph Entity Alignment with Graph Convolutional Networks: Lessons Learned. pages 3–11. Springer, Cham, 4 2020. doi: 10.1007/978-3-030-45442-5_1. URL `http://link.springer.com/10.1007/978-3-030-45442-5_1`.

[13] Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. Interpretable and Fair Comparison of Link Prediction or Entity Alignment Methods. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 371–374, 2020. doi: 10.1109/WIIAT50758.2020.00053.

[14] Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. On the Ambiguity of Rank-Based Evaluation of Entity Alignment or Link Prediction Methods. *arXiv preprint arXiv:2002.06914v3*, 2021.

[15] J Bleiholder and F Naumann. Data Fusion. *Data fusion. ACM Comput. Surv*, 41(1):41, 2008. doi: 10.1145/1456650.1456651. URL `http://doi.acm.org/10.1145/1456650.1456651`.

[16] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[17] Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, and William L Hamilton. Understanding the Performance of Knowledge Graph Embeddings in Drug Discovery. *arXiv preprint arXiv:2105.10488*, 2021.

[18] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data, 2013. URL `http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-rela`.

[19] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[20] Stéphanie Bougeard and Stéphane Dray. Supervised multiblock analysis in R with the ade4 package. *Journal of statistical software*, 86(1):1–17, 2018.

[21] Mokrane Bouzeghoub and Maurizio Lenzerini. Introduction to: data extraction, cleaning, and reconciliation a special issue of Information Systems, An International Journal. *Information Systems*, 26:535–536, 2001. URL `http://production.datastore.cvt.dk/filestore?oid=539ff0296df0df9c4c05cd90&targetid=539ff0296df0df9c4c05cd92`.

[22] Agustina Buccella, Alejandra Cechich, and Nieves R Brisaboa. Applying an ontology on data integration. In *V Workshop de Investigadores en Ciencias de la Computación*, 2003.

[23] Pau Cabaneros Lopez, Isuru A. Udugama, Sune T. Thomsen, Christian Roslander, Helena Junicke, Miguel M. Iglesias, and Krist V. Gernaey. Transforming data to information: A parallel hybrid model for real-time state estimation in lignocellulosic ethanol fermentation. *Biotechnology and Bioengineering*, 118(2):579–591, 2 2021. ISSN 0006-3592. doi: 10.1002/bit.27586. URL `https://onlinelibrary.wiley.com/doi/10.1002/bit.27586`.

[24] Federico Castanedo. A Review of Data Fusion Techniques. *The Scientific World Journal*, 2013:704504, 2013. ISSN 2356-6140. doi: 10.1155/2013/704504. URL `https://doi.org/10.1155/2013/704504`.

[25] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A Data Discovery System. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE, 4 2018. ISBN 978-1-5386-5520-7. doi: 10.1109/ICDE.2018.00094. URL https://ieeexplore.ieee.org/document/8509315/.

[26] Michelle Cheatham and Catia Pesquita. Semantic Data Integration. In *Handbook of Big Data Technologies*, pages 263–305. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-49340-4_8. URL http://link.springer.com/10.1007/978-3-319-49340-4_8.

[27] Jingliang Chen, Dmytro Dosyn, Vasyl Lytvyn, and Anatoliy Sachenko. Smart Data Integration by Goal Driven Ontology Learning. pages 283–292. Springer, Cham, 2017. doi: 10.1007/978-3-319-47898-2_29. URL http://link.springer.com/10.1007/978-3-319-47898-2_29.

[28] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *Proceedings - International Conference on Data Engineering*, 2013. ISBN 9781467349086. doi: 10.1109/ICDE.2013.6544847.

[29] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1247–1261, 2015.

[30] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. Data Cleaning. In *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*, pages 2201–2206, New York, New York, USA, 2016. ACM Press. ISBN 9781450335317. doi: 10.1145/2882903.2912574. URL http://dl.acm.org/citation.cfm?doid=2882903.2912574.

[31] Manuel Ciosici, Tobias Sommer, and Ira Assent. Unsupervised Abbreviation Disambiguation Contextual disambiguation using word embeddings. 4 2019. URL http://arxiv.org/abs/1904.00929.

[32] William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data - SIGMOD '98*, volume 27, pages 201–212, New York, New York, USA, 1998. ACM Press. ISBN 0897919955. doi: 10.1145/276304.276323. URL http://portal.acm.org/citation.cfm?doid=276304.276323.

[33] Diego Collarana, Mikhail Galkin, Ignacio Traverso-Ribon, Christoph Lange, Maria-Esther Vidal, and Soren Auer. Semantic Data Integration

for Knowledge Graph Construction at Query Time. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pages 109–116. IEEE, 2017. ISBN 978-1-5090-4284-5. doi: 10.1109/ICSC.2017.85. URL https://ieeexplore.ieee.org/document/7889517/.

[34] Nelson Cowan. Chapter 20 What are the differences between long-term, short-term, and working memory? *Progress in Brain Research*, 169:323–338, 1 2008. ISSN 0079-6123. doi: 10.1016/S0079-6123(07)00020-9. URL https://www.sciencedirect.com/science/article/pii/S0079612307000209.

[35] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*, pages 1277–1286, 2012.

[36] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. NADEEF: A Commodity Data Cleaning System. In *Proceedings of the 2013 international conference on Management of data - SIGMOD '13*, page 541, New York, New York, USA, 2013. ACM Press. ISBN 9781450320375. doi: 10.1145/2463676.2465327. URL http://dl.acm.org/citation.cfm?doid=2463676.2465327.

[37] Sijmen de Jong. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, 3 1993. ISSN 01697439. doi: 10.1016/0169-7439(93)85002-X. URL http://linkinghub.elsevier.com/retrieve/pii/016974399385002X.

[38] Luc De Raedt. *Logical and relational learning*. Springer Science + Business Media, 2008. ISBN 978-3-540-20040-6.

[39] Woodrow W Denham. *The detection of patterns in Alyawara nonverbal behavior*. PhD thesis, University of Washington., 1973.

[40] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. *Thirty-Second AAAI Conference on Artificial Intelligence*, 4 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/17366.

[41] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion, 2014. URL https://ai.google/research/pubs/pub45634.

[42] Lucas Drumond, Steffen Rendle, and Lars Schmidt-Thieme. Predicting RDF triples in incomplete knowledge bases with tensor factorization. In

Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12, page 326, New York, New York, USA, 2012. ACM Press. ISBN 9781450308571. doi: 10.1145/2245276.2245341. URL http://dl.acm.org/citation.cfm?doid=2245276.2245341.

[43] Kasper A. Einarson, Andreas Baum, Terkel B. Olsen, Jan Larsen, Ibrahim Armagan, Paloma A. Santacoloma, and Line K. H. Clemmensen. Predicting pectin performance strength using near-infrared spectroscopic data: A comparative evaluation of 1-D convolutional neural network, partial least squares, and ridge regression modeling. *Journal of Chemometrics*, page e3348, 5 2021. ISSN 0886-9383. doi: 10.1002/cem.3348. URL https://onlinelibrary.wiley.com/doi/10.1002/cem.3348.

[44] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[45] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545, 2011.

[46] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Ramasamy Uthurusamy. Summary from the KDD-03 panel. *ACM SIGKDD Explorations Newsletter*, 5(2):191, 12 2003. ISSN 19310145. doi: 10.1145/980972.981004. URL http://portal.acm.org/citation.cfm?doid=980972.981004.

[47] Samuel Fosso Wamba, Shahriar Akter, Andrew Edwards, Geoffrey Chopin, and Denis Gnanzou. How 'big data' can make big impact: Findings from a systematic review and a longitudinal case study. *International Journal of Production Economics*, 165:234–246, 2015. ISSN 09255273. doi: 10.1016/j.ijpe.2014.12.031. URL http://www.sciencedirect.com/science/article/pii/S0925527314004253.

[48] Thorben Funke and Till Becker. Stochastic block models: A comparison of variants and inference methods. *PloS one*, 14(4):e0215296, 2019.

[49] John Gantz and David Reinsel. THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. *Idc*, (December 2012):1–16, 2012. URL https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.

[50] Ian Gemp, Georgios Theocharous, and Mohammad Ghavamzadeh. Automated Data Cleansing through Meta-learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4760–4761, 2017.

[51] W Graham Glen, M Sarker, W J Dunn III, and D R Scott. UNIPALS: Software for principal components analysis and partial least squares regression. *Tetrahedron Computer Methodology*, 2(6):377–396, 1989.

[52] Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. A survey of statistical network models. *Foundations and Trends®in Machine Learning*, 2(2):129–233, 2010. ISSN 1935-8237. doi: 10.1561/2200000005. URL http://dx.doi.org/10.1561/2200000005.

[53] Bryce Goodman and Seth Flaxman. EU regulations on algorithmic decision-making and a "right to explanation". *2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, (Whi):26–30, 2016. URL http://arxiv.org/abs/1606.08813.

[54] Irwin R Goodman, Ronald P Mahler, and Hung T Nguyen. *Mathematics of data fusion*, volume 37. Springer Science + Business Media, 2013. ISBN 978-0-7923-4674-6.

[55] David Hall and James Llinas, editors. *Multisensor Data Fusion*. CRC Press, 6 2001. ISBN 9780429127472. doi: 10.1201/9781420038545. URL https://www.taylorfrancis.com/books/9781420038545.

[56] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[57] Tue Herlau, Mikkel N Schmidt, and Morten Mørup. Infinite-degree-corrected stochastic block model. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 90(3), 2014. ISSN 15502376. doi: 10.1103/PhysRevE.90.032819. URL https://journals.aps.org/pre/pdf/10.1103/PhysRevE.90.032819.

[58] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5 (2):109–137, 6 1983. ISSN 0378-8733. doi: 10.1016/0378-8733(83)90021-7. URL https://www.sciencedirect.com/science/article/abs/pii/0378873383900217.

[59] Takoua Jendoubi. Approaches to Integrating Metabolomics and Multi-Omics Data: A Primer. *Metabolites*, 11(3):184, 3 2021. ISSN 2218-1989. doi: 10.3390/metabo11030184. URL https://www.mdpi.com/2218-1989/11/3/184.

[60] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A Survey on Knowledge Graphs: Representation, Acquisition, and

Applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2021. ISSN 2162-237X. doi: 10.1109/TNNLS.2021.3070843. URL https://ieeexplore.ieee.org/document/9416312/.

[61] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge Base Completion: Baselines Strike Back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, 2017.

[62] Brian Karrer and M. E.J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 83(1), 2011. ISSN 15393755. doi: 10.1103/PhysRevE.83.016107.

[63] Seyed Mehran Kazemi and David Poole. SimplE embedding for link prediction in knowledge graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4289–4300, 2018.

[64] Zuhair Khayyat, Ihab F Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. BigDansing: A System for Big Data Cleansing. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015. ISSN 07308078. doi: 10.1145/2723372.2747646.

[65] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery*, 7(1):81–99, 2003. ISSN 13845810. doi: 10.1023/A:1021564703268. URL http://link.springer.com/10.1023/A:1021564703268.

[66] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, {ICLR} 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[67] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, Chris Meek, Jennifer Neville, and others. *Introduction to statistical relational learning*. MIT press, 2007. ISBN 9780262072885.

[68] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, Tim Kraska, Tova Milo, and Eugene Wu. SampleClean: Fast and Reliable Analytics on Dirty Data. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 59–75, 2015. URL http://sites.computer.org/debull/A15sept/p59.pdf.

[69] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating Literals into Knowledge

Graph Embeddings. pages 347–363. Springer, Cham, 10 2019. doi: 10.1007/978-3-030-30793-6_20. URL http://link.springer.com/10.1007/978-3-030-30793-6_20.

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[71] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, pages 2863–2872. PMLR, 2018.

[72] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 11 2017. ISSN 0140-525X. doi: 10.1017/S0140525X16001837. URL https://www.cambridge.org/core/product/identifier/S0140525X16001837/type/journal_article.

[73] Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):122, 12 2019. ISSN 2364-8228. doi: 10.1007/s41109-019-0232-2. URL https://appliednetsci.springeropen.com/articles/10.1007/s41109-019-0232-2.

[74] Jens Lehmann and Johanna Voelker. An introduction to ontology learning. *Perspectives on Ontology Learning. IOS Press, Amsterdam, The Netherlands*, 2014.

[75] Jens Lehmann and Johanna Völker. *Perspectives on ontology learning*, volume 18. IOS Press, 4 2014. ISBN 9781614993797.

[76] Maurizio Lenzerini. Data integration. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*, page 233, New York, New York, USA, 2002. ACM Press. ISBN 1581135076. doi: 10.1145/543613.543644. URL http://portal.acm.org/citation.cfm?doid=543613.543644.

[77] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large scale graph embedding system. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 120–131, 2019. URL https://proceedings.mlsys.org/paper/2019/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf.

[78] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks. In *2021 IEEE 37th International Conference on Data*

Engineering (ICDE), pages 13–24. IEEE, 4 2021. ISBN 978-1-7281-9184-3. doi: 10.1109/ICDE51399.2021.00009. URL https://ieeexplore.ieee.org/document/9458702/.

[79] Fredrik Lindgren, Paul Geladi, and Svante Wold. The kernel algorithm for PLS. *Journal of Chemometrics*, 7(1):45–59, 1 1993. ISSN 0886-9383. doi: 10.1002/cem.1180070104. URL http://doi.wiley.com/10.1002/cem.1180070104.

[80] Pau Cabaneros Lopez, Isuru Abeykoon Udugama, Sune Tjalfe Thomsen, Christoph Bayer, Helena Junicke, and Krist V. Gernaey. Promoting the co-utilisation of glucose and xylose in lignocellulosic ethanol fermentations using a data-driven feed-back controller. *Biotechnology for Biofuels*, 13(1):190, 12 2020. ISSN 1754-6834. doi: 10.1186/s13068-020-01829-2. URL https://biotechnologyforbiofuels.biomedcentral.com/articles/10.1186/s13068-020-01829-2.

[81] Pau Cabaneros Lopez, Isuru A. Udugama, Sune T. Thomsen, Christian Roslander, Helena Junicke, Miguel Mauricio-Iglesias, and Krist V. Gernaey. Towards a digital twin: a hybrid data-driven and mechanistic digital shadow to forecast the evolution of lignocellulosic fermentation. *Biofuels, Bioproducts and Biorefining*, 14(5):1046–1060, 9 2020. ISSN 1932-104X. doi: 10.1002/bbb.2108. URL https://onlinelibrary.wiley.com/doi/10.1002/bbb.2108.

[82] Stuart Madnick and Hongwei Zhu. Improving data quality through effective use of data semantics. *Data & Knowledge Engineering*, 59(2):460–475, 2006. ISSN 0169-023X. doi: http://doi.org/10.1016/j.datak.2005.10.001. URL http://www.sciencedirect.com/science/article/pii/S0169023X05001497.

[83] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Raha: A Configuration-Free Error Detection System. In *Proceedings of the 2019 International Conference on Management of Data*, pages 865–882, New York, NY, USA, 6 2019. ACM. ISBN 9781450356435. doi: 10.1145/3299869.3324956. URL https://dl.acm.org/doi/10.1145/3299869.3324956.

[84] Mohammad Mahdavi, Felix Neutatz, Larysa Visengeriyeva, and Ziawasch Abedjan. Towards automated data cleaning workflows. *Machine Learning*, 15:16, 2019.

[85] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[86] Jonathan I. Maletic and Andrian Marcus. Data Cleansing: A Prelude to Knowledge Discovery. In *Data Mining and Knowledge Discovery Handbook*, pages 19–32. Springer US, Boston, MA, 2009. doi: 10.1007/978-0-387-09823-4_2. URL http://link.springer.com/10.1007/978-0-387-09823-4_2.

[87] Harald Martens and Magni Martens. *Multivariate analysis of quality: an introduction*. John Wiley & Sons, 2001.

[88] Nigel McKelvey, Kevin Curran, and Luke Toland. The Challenges of Data Cleansing with Data Warehouses. In Manoj Kumar Singh and Dileep Kumar, editors, *Effective Big Data Management and Opportunities for Implementation*, chapter 5, pages 77–82. Information Science Reference (an imprint of IGI Global), Hershey PA, USA 17033, 2016. doi: 10.4018/978-1-5225-0182-4.ch005. URL http://www.igi-global.com.

[89] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[90] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[91] Sameh K Mohamed, Vít Novácek, Pierre-Yves Vandenbussche, and Emir Muñoz. Loss Functions in Knowledge Graph Embedding Models. *DL4KG@ ESWC*, 2377:1–10, 2019.

[92] Morten Mørup and Mikkel N. Schmidt. Bayesian Community Detection. *Neural Computation*, 24(9):2434–2456, 9 2012. ISSN 0899-7667. doi: 10.1162/NECO_a_00314. URL http://www.mitpressjournals.org/doi/10.1162/NECO_a_00314.

[93] Morten Mørup, Kristoffer Madsen, Anne-marie Dogonowski, Hartwig Siebner, and Lars K Hansen. Infinite relational modeling of functional connectivity in resting state fMRI. *Advances in neural information processing systems*, 23:1750–1758, 2010.

[94] Heiko Müller and Johann-Christoph Freytag. *Problems, Methods, and Challenges in Comprehensive Data Cleansing*. Professoren des Inst. Für Informatik, 2005. URL http://www.dbis.informatik.hu-berlin.de/fileadmin/research/papers/techreports/2003-hub_ib_164-mueller.pdf.

[95] M E J Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences, U S A*, 103(23):8577–8582, 6 2006. doi: 10.1073/pnas.0601602103. URL http://www.pnas.org/content/103/23/8577.abstract.

[96] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 327–333, Stroudsburg, PA, USA, 2018. Association for Computational Linguistics. ISBN 9781948087292. doi: 10.18653/v1/n18-2053. URL `http://aclweb.org/anthology/N18-2053`.

[97] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189, Stroudsburg, PA, USA, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1226. URL `http://aclweb.org/anthology/N19-1226`.

[98] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[99] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(3):42–73, 2012.

[100] Paulo Oliveira, Fátima Rodrigues, Pedro Henriques, and Helena Galhardas. A taxonomy of data quality problems. In *2nd Int. Workshop on Data and Information Quality*, pages 219–233. Citeseer, 2005.

[101] Aris M Ouksel and Amit Sheth. Semantic interoperability in global information systems. *ACM Sigmod Record*, 28(1):5–12, 1999. ISSN 0163-5808.

[102] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep Learning vs. Traditional Computer Vision. In *Advances in Intelligent Systems and Computing*, volume 943, pages 128–144. Springer, Cham, 4 2020. ISBN 9783030177942. doi: 10.1007/978-3-030-17795-9_10. URL `http://link.springer.com/10.1007/978-3-030-17795-9_10`.

[103] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[104] Richard Qian. Understand Your World with Bing | Bing Search Blog, 2013. URL https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/. Accessed on 05.07.2021.

[105] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66 (336):846–850, 12 1971. ISSN 0162-1459. doi: 10.1080/01621459.1971.10482356. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[106] Stefan Rännar, Fredrik Lindgren, Paul Geladi, and Svante Wold. A PLS kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm. *Journal of Chemometrics*, 8(2):111–125, 3 1994. ISSN 0886-9383. doi: 10.1002/cem.1180080204. URL http://doi.wiley.com/10.1002/cem.1180080204.

[107] Stefan Rännar, Paul Geladi, Fredrik Lindgren, and Svante Wold. A PLS kernel algorithm for data sets with many variables and few objects. Part II: Cross-validation, missing data and examples. *Journal of Chemometrics*, 9(6):459–470, 1995. ISSN 1099128X. doi: 10.1002/cem.1180090604.

[108] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge Graph Embedding for Link Prediction. *ACM Transactions on Knowledge Discovery from Data*, 15(2): 1–49, 4 2021. ISSN 1556-4681. doi: 10.1145/3424672. URL https://dl.acm.org/doi/10.1145/3424672.

[109] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2019.

[110] Mikkel N. Schmidt and Morten Mørup. Nonparametric Bayesian Modeling of Complex Networks. *IEEE Signal Processing Magazine*, 30: 110–128, 12 2013. ISSN 1053-5888. doi: 10.1109/MSP.2012.2235191. URL http://arxiv.org/abs/1312.5889http://dx.doi.org/10.1109/MSP.2012.2235191.

[111] Elena Simperl, Tobias Burger, Simon Hangl, Stephan WWrgl, and Igor Popov. ONTOCOM: A Reliable Cost Estimation Method for Ontology Development Projects. *SSRN Electronic Journal*, 2 2012. ISSN 1556-5068. doi: 10.2139/ssrn.3198977. URL https://www.ssrn.com/abstract=3198977.

[112] Amit Singhal. Introducing the Knowledge Graph: things, not strings, 2012. URL https://blog.google/products/search/introducing-knowledge-graph-things-not/. Accessed on 05.07.2021.

[113] Michael Stonebraker and Ugur Cetintemel. " One size fits all": an idea whose time has come and gone. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 2–11. IEEE, 2005. ISBN 0769522858.

[114] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[115] Bongwon Suh, Gregorio Convertino, Ed H Chi, and Peter Pirolli. The singularity is not near: slowing growth of Wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, pages 1–10, 2009.

[116] 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://iclr.cc/Conferences/2019/Schedule?showEvent=870`.

[117] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A Re-evaluation of Knowledge Graph Completion Methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5516–5522, Stroudsburg, PA, USA, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.489. URL `https://www.aclweb.org/anthology/2020.acl-main.489`.

[118] Petr Taborsky, Laurent Vermue, Maciej Korzepa, and Morten Morup. The Bayesian Cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. doi: 10.1109/TPAMI.2020.2994396.

[119] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020.

[120] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[121] Toni Vallès-Català, Francesco A. Massucci, Roger Guimerà, and Marta Sales-Pardo. Multilayer Stochastic Block Models Reveal the Multilayer Structure of Complex Networks. *Physical Review X*, 6(1):011036, 3 2016. ISSN 2160-3308. doi: 10.1103/PhysRevX.6.011036. URL `https://link.aps.org/doi/10.1103/PhysRevX.6.011036`.

[122] Larysa Visengeriyeva and Ziawasch Abedjan. Metadata-driven error detection. In *Proceedings of the 30th International Conference on Scientific*

and Statistical Database Management, pages 1–12, New York, NY, USA, 7 2018. ACM. ISBN 9781450365055. doi: 10.1145/3221269.3223028. URL https://dl.acm.org/doi/10.1145/3221269.3223028.

[123] Larysa Visengeriyeva and Ziawasch Abedjan. Anatomy of Metadata for Data Curation. *Journal of Data and Information Quality*, 12(3):1–30, 7 2020. ISSN 1936-1955. doi: 10.1145/3371925. URL https://dl.acm.org/doi/10.1145/3371925.

[124] Denny Vrandečić and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85, 9 2014. ISSN 0001-0782. doi: 10.1145/2629489. URL https://dl.acm.org/doi/10.1145/2629489.

[125] W3C. RDF Semantics, 2004. URL https://www.w3.org/TR/rdf-mt/. Accessed on 05.07.2021.

[126] W3C. Resource Description Framework (RDF): Concepts and Abstract Syntax, 2004. URL https://www.w3.org/TR/rdf-concepts/. Accessed on 05.07.2021.

[127] Hao Wang, Xu Zhuge, Girts Strazdins, Zheng Wei, Guoyuan Li, and Houxiang Zhang. Data integration and visualisation for demanding marine operations. In *OCEANS 2016 - Shanghai*, pages 1–7. IEEE, 4 2016. ISBN 978-1-4673-9724-7. doi: 10.1109/OCEANSAP.2016.7485617. URL http://ieeexplore.ieee.org/document/7485617/.

[128] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 2018. doi: 10.1016/j.jmsy.2018.01.003. URL https://doi.org/10.1016/j.jmsy.2018.01.003.

[129] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017. doi: 10.1109/TKDE.2017.2754499.

[130] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[131] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 65–76, 2010.

[132] Johan a Westerhuis, Theodora Kourti, and John F MacGregor. Analysis of multiblock and hierarchical PCA and PLS models. *J. Chemometrics*, 12(5):301–321, 1998. ISSN 1099-128X. doi: 10.1002/(SICI)1099-128X(199809/10)12:5<301::AID-CEM515>3.0.CO;2-S. URL http://dx.doi.org/10.1002/(SICI)1099-128X(199809/10)12:5%3C301::AID-CEM515%3E3.0.CO;2-S.

[133] Wikimedia Foundation. Wikimedia Statistics - English Wikipedia - Pages to date. URL https://stats.wikimedia.org/#/en.wikipedia.org/content/pages-to-date/normal%7ctable%7call%7cpage_type~content%7cmonthly. Accessed on 05.07.2021.

[134] Herman Wold. Soft Modelling by Latent Variables: The Non-Linear Iterative Partial Least Squares (NIPALS) Approach. *Journal of Applied Probability*, 12(S1):117–142, 9 1975. ISSN 0021-9002. doi: 10.1017/S0021900200047604. URL https://www.cambridge.org/core/product/identifier/S0021900200047604/type/journal_article.

[135] S. Wold, A. Ruhe, H. Wold, and W. J. Dunn, III. The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 9 1984. ISSN 0196-5204. doi: 10.1137/0905052. URL http://epubs.siam.org/doi/10.1137/0905052.

[136] Alex Woodie. Why Gartner Dropped Big Data Off the Hype Curve, 2015. URL https://www.datanami.com/2015/08/26/why-gartner-dropped-big-data-off-the-hype-curve/. Accessed on 05.07.2021.

[137] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations*, 12 2015. URL http://arxiv.org/abs/1412.6575.

[138] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 96–104, New York, NY, USA, 1 2019. ACM. ISBN 9781450359405. doi: 10.1145/3289600.3291014. URL https://dl.acm.org/doi/10.1145/3289600.3291014.