Towards an interpretable model of learners in a learning environment based on knowledge graphs

Antonia Ettorre

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Vers un modèle interprétable pour la prédiction de la performance des apprenants dans un environnement d'apprentissage basée sur l'exploitation de graphes de connaissances

Antonia ETTORRE

Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S)
UMR7271 UCA CNRS

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur

Dirigée par : Catherine FARON, Professeure, Université Côte d'Azur
Co-encadrée par : Franck MICHEL, Ingénieur de recherche, CNRS
Soutenue le : 28-11-2022

Devant le jury, composé de :
Andrea TETTAMANZI, Professeur, Université Côte d'Azur
Sandra BRINGAY, Professeure, Université Paul Valéry - Montpellier III
Nathalie PERNELLE, Professeure, Université Sorbonne Paris Nord
Marie-Christine ROUSSET, Professeure, Université de Grenoble-Alpes
Harald SACK, Professeur, FIZ Karlsruhe - Leibniz
Raphaël TRONCY, Maître de conférences, Eurecom

3iA Côte d'Azur
Institut interdisciplinaire d'intelligence artificielle

i3S sophia antipolis

Inria

VERS UN MODÈLE INTERPRÉTABLE POUR LA PRÉDICTION DE LA PERFORMANCE DES APPRENANTS DANS UN ENVIRONNEMENT D'APPRENTISSAGE BASÉE SUR L'EXPLOITATION DE GRAPHES DE CONNAISSANCES

Towards an interpretable model of learners in a learning environment based on Knowledge Graphs

Antonia ETTORRE

Jury :

Président du jury
Andrea TETTAMANZI, Professeur, Université Côte d'Azur

Rapporteurs
Nathalie PERNELLE, Professeure, Université Sorbonne Paris Nord
Harald SACK, Professeur, FIZ Karlsruhe - Leibniz

Examinateurs
Sandra BRINGAY, Professeure, Université Paul Valéry - Montpellier III
Marie-Christine ROUSSET, Professeure, Université de Grenoble-Alpes
Raphaël TRONCY, Maître de conférences, Eurecom

Directeur de thèse
Catherine FARON, Professeure, Université Côte d'Azur

Co-encadrant de thèse
Franck MICHEL, Ingénieur de recherche, CNRS

Université Côte d'Azur

Antonia ETTORRE

Vers un modèle interprétable pour la prédiction de la performance des apprenants dans un environnement d'apprentissage basée sur l'exploitation de graphes de connaissances

xvii+128 p.

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10.

Impression : output.tex – 5/10/2022 – 9:24

Révision pour la classe : these-ISSS.cls,v 2.10 2020/06/24 14:16:37 mpelleau

Vers un modèle interprétable pour la prédiction de la performance des apprenants dans un environnement d'apprentissage basée sur l'exploitation de graphes de connaissances

Résumé

Ces dernières années, la société a manifesté un besoin croissant de ressources éducatives facilement accessibles et qui rendent l'apprentissage plus efficace et complet. Un nombre croissant de personnes à travers le monde accède à l'éducation en ligne et demande des outils plus efficaces pour permettre d'apprendre n'importe quoi, n'importe où et à n'importe quel moment. Cela nécessite le développement de systèmes éducatifs plus intelligents, qui devraient être capables d'améliorer les courbes d'apprentissage des utilisateurs et de les assister efficacement dans leur processus d'acquisition de connaissances, en ne faisant éventuellement appel à aucun, ou très peu, de soutien humain. Une étape importante pour concrétiser cette vision consiste à personnaliser le processus d'apprentissage afin de l'adapter spécifiquement à chaque utilisateur, en tenant compte de ses antécédents, de son style d'apprentissage, de ses besoins personnels et de ses objectifs. La création de tels environnements adaptatifs et personnalisés requiert de suivre l'évolution des connaissances des utilisateurs au fil du temps et d'évaluer s'ils ont la capacité de faire face à un problème, un exercice ou une question spécifique à un moment donné de leur apprentissage. Ce problème, connu dans la communauté de l'éducation sous le nom de *Knowledge Tracing*, a été largement étudié au cours des 50 dernières années et plusieurs approches de résolution ont été proposées. Bien que leurs performances se soient sensiblement améliorées au cours de la dernière décennie, ces approches présentent plusieurs défauts : de la simplicité excessive dans la représentation de l'environnement d'apprentissage, qui ne tient pas compte de scénarios complexes tels que l'acquisition de compétences non techniques ou la résolution de travaux de groupe, à l'impossibilité d'interpréter les prédictions fournies, par exemple d'expliquer pourquoi un étudiant échouera en essayant de répondre à une question donnée. Dans cette thèse, nous contribuons à la résolution de ces problèmes en explorant et en proposant des approches basées sur l'utilisation de l'IA symbolique, et plus précisément la représentation de connaissances et le raisonnement à base de graphes de connaissances. Premièrement, nous proposons une approche de *Knowledge Tracing* qui étend un modèle existant en introduisant, comme caractéristiques d'entrée supplémentaires, des *plongements de graphes de connaissances*. Ensuite, nous étudions l'explicabilité de l'approche proposée en cherchant à interpréter les plongements de graphes employés. Ceci nous conduit à l'implémentation d'un outil pour l'analyse visuelle conjointe des graphes de connaissance et des plongements de graphes et au développement d'une approche pour vérifier l'information capturée par de tels plongements de graphes. Enfin, nous présentons un modèle de *Knowledge Tracing* reposant exclusivement sur la représentation de l'environnement d'apprentissage sous la forme d'un graphe de connaissances, qui ne nécessite aucun modèle externe supplémentaire pour la prédiction.

Mots-clés : Web Sémantique, Graphes de Connaissances, Éducation, Knowledge Tracing

Towards an interpretable model of learners in a learning environment based on Knowledge Graphs

Abstract

In recent years, society has demonstrated increasing need for more effective, comprehensive and easily accessible educational resources. A growing number of people around the world have gained access to online education and demand more efficient tools to enable learning anything, anywhere, at any moment. This requires the development of smarter educational systems, which should be able to improve users' learning curves and effectively assist them in their knowledge acquisition process, possibly relying on no, or very little, human support. A major step to concretize this vision lies in personalizing the learning process to be specifically adapted to every single user, taking into account their background, learning style, personal needs and objectives. To create such adaptive and customized environments, a major requirement is represented by the ability to trace user knowledge over time and assess whether they have the capacity to face a specific problem, exercise or question. This problem, known in the Education community as *Knowledge Tracing*, has been widely investigated in the last 50 years and several resolution approaches have been proposed. Though their performance improved sensibly over the last decade, such approaches present several shortcomings: from the excessive simplicity in the representation of the learning environment, which does not account for complex scenarios such as acquisition of soft skills or solution of group assignments; to the impossibility of interpreting the provided predictions, e.g. explaining why a student will fail while trying to answer a given question. In this thesis, we try to overcome these issues by exploring and proposing approaches based on the use of Symbolic AI approaches focusing on Graph based Knowledge Representation and Reasoning. Firstly, we propose a Knowledge Tracing approach that extends an existing model by introducing, as additional input features, *Knowledge Graph Embeddings*. Secondly, we investigate the explainability of the proposed approach by seeking the interpretation of the employed Graph Embeddings. This leads us to the implementation of a tool for the joint visual analysis of Knowledge Graphs and Graph Embeddings and to the development of an approach to verify the information encoded by such Graph Embeddings. Finally, we present a Knowledge Tracing model exclusively relying on the representation of the learning environment in the form of a Knowledge Graph, which does not require any additional external model for the prediction.

Keywords: Semantic Web, Knowledge Graphs, Education, Knowledge Tracing

Acknowledgments

Honestly, it does sound pretty incredible, but I am actually at the end of this journey. Odyssey, I would even say. Yes, because, I can definitely say that it was more difficult than what I expected... And it would have been impossible without the help and support of the people around me, who deserve my deepest gratitude.

Firstly, I would like to thank my advisors Catherine and Franck for their incredible support, both scientific and mental, in the most challenging times (which, we know, have been pretty hard on me). Thanks for listening and welcoming every research idea or suggestion I had, and for your sincere and critical scientific feedback. Thanks to Catherine for prescribing me long daily walks for my mental wellness during the difficult COVID period, and to Franck for checking on me regularly to be sure that I was OK. This means a lot to me (and to my mum, who was happy to know that someone was looking after me).

Thanks to all the past and present members of the WIMMICS and SPARKS team, who shared with me this incredible journey, especially to the ones who were always ready for a coffee break or a chocolate tasting session. Please, keep enjoying the chocolate in my name! Among them, a special thank goes to Anna, Thibaut, and Amine, who, for more than three years, had the patience to keep listening to my complaints about writing, deadlines, reviews, impostor syndrome, university bureaucracy, job seeking, apartment searching, gym, food... well, basically everything... I do complain a lot, indeed.

Thanks to Sara, who has been the constant of my four years in France, always there to support and advise me in every matter. We started this journey together and we are ending it together. I don't know what I would have done without you. Well, surely not this Ph.D. since it was you to convince me to accept. I am not sure yet if I should thank you for that.

The deepest and most sincere gratitude goes to my family, my parents, and my brothers, who supported me in every moment, even from far away, even though they didn't fully understand what I was doing exactly (which I am not sure I did either). Thanks for encouraging me when I wanted to quit everything, for convincing me to hold on, and for making me see the silver lining of every cloud.

Finally, an immense thank is to Johann, whose presence in my life made most of the difference between failure and success. Thanks for everything I mentioned above and more: the mental and scientific support, the coffee breaks, for listening to my complaints, for convincing me that it was worth going on even when I thought it was over. You have been my harbor and refuge. You have been my home. Thanks for sharing your life with me. And, especially, for washing the dishes, you know how much I hate it :P.

Ah, a last very uncommon thank to the coin I tossed when I was 15 which decided that I would have studied Computer Science. I didn't know, at that time, that this would have triggered the unexpected series of events that would have eventually brought me here.

Contents

List of Abbreviations

AFM	Additive Factor Model
AIEd	Artificial Intelligence in Education
BFS	Breadth First Sampling
BKT	Bayesian Knowledge Tracing
CBOW	Continuous Bag-Of-Words
CMS	Content Management System
CTT	Classical Test Theory
DFS	Depth First Sampling
DKT	Deep Knowledge Tracing
EDM	Educational Data Mining
FM	Factorization Machine
GCN	Graph Convolutional Network
GE	Graph Embedding
GNN	Graph Neural Network
ILE	Interactive Learning Environment
IRT	Item Response Theory
ITS	Intelligent Tutoring System
KC	Knowledge Component
KG	Knowledge Graph
KGE	Knowledge Graph Embedding
KI	Knowledge Item
KT	Knowledge Tracing
LMS	Learning Management System
LP	Link Prediction
LSTM	Long Short Term Memory
MCQ	Multiple Choice Question
mIRT	Multiple Item Response Theory
MOOC	Massive Open Online Course
OWL	Web Ontology Language
PFA	Performance Factors Analysis
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RNN	Recurrent Neural Network
SPARQL	SPARQL Protocol And RDF Query Language

Chapter 1

Introduction

1.1 Context and Motivation

In the last decades, we have witnessed the impressive transition of most humans' everyday activities to the online world. Socializing, doing groceries, organizing holidays, and banking are only some examples of tasks that are increasingly carried out through the Web and, often, enhanced by the employment of Artificial Intelligence, limiting the requirements for human-to-human interaction. As our lives are being reshaped by the growing need for these technologies, industries and services are evolving and adapting to provide us with increasingly smart online environments. Yet, there are sectors for which the transition towards more intelligent and automated systems is very challenging, and, as clearly shown by the events of the last couple of years related to the COVID-19 pandemic, education is one of these.

In recent years, more and more institutions, schools, and training centers around the world started offering online or hybrid support to their students and a growing number of mobile applications and web platforms have been developed to meet the necessities of the thriving users' community to access high-quality educational resources. Nevertheless, in most cases, e.g. MOOCs and universities' CMS, these web resources represent just catalogs of videos, documents, and simple multiple-choice questions (MCQ) tests, which struggle to keep users engaged, improve their learning curves, and effectively assist them in their knowledge acquisition process. In other words, the quality of these services is far from the one of classical face-to-face education and human interaction is still a fundamental component of the learning process. To truly improve the users' knowledge acquisition journey making it more effective while supporting or reducing human interaction, we need to create smarter environments, able to adapt to the knowledge level and learning habits of every single user, to intelligently support and engage them anticipating their needs and providing them with the right educational resources at the right time. Artificial Intelligence represents the key tool for implementing such flexible, personalized, and adaptive systems.

The use of Artificial Intelligence in Education (AIEd) has been the focus of intense research work in the last three decades (Chen et al., 2020, Luckin et al., 2016, Roll and Wylie, 2016), and several AI tools and techniques have been applied to different scenarios in education. AI is used not only for implementing adaptive learning environments and enhancing online education as described beforehand but it is also adopted as a support to traditional in-class education, for example, to automate grading and evaluation of exams

or to build "smart schools" (Chen et al., 2020); and it can be employed by teachers, trainers and pedagogists to gain insights into the learning process itself, for example, to discover what social factors most affect students' learning habits and efficiency (Luckin et al., 2016). Although the scenarios differ for application context (online or in-classroom), and target subjects (teachers, students, or support staff) they all aim to simplify and improve the global learning experience. The effectiveness of Artificial Intelligence in Education has already been proved in several real-world applications where students' learning is supported by computer-based systems, also known as Intelligent Tutoring Systems (ITS) or Interactive Learning Environments (ILE). For example, (VanLehn, 2011) found that ITS can be almost as effective as human tutoring, while (Cen et al., 2007) proved that Educational Data Mining (EDM) can help improve learning efficiency.

Despite the progress made by research in AIEd during the last 20 years, computer-assisted learning still shows limitations. Indeed, while these systems proved to be useful for the acquisition of simple domain-knowledge concepts, they struggle to provide support for the development of soft skills, such as critical thinking, problem-solving and collaboration, which are of the uttermost importance nowadays (Trilling and Fadel, 2009); and they fail to account for students' personal traits, such as background, learning habits and preferences. In summary, the main challenge for modern AIEd is to create flexible, complete and effective learning environments able to adapt to every single user providing personalized learning paths, guiding them through the acquisition of simple domain-knowledge concepts, but also stimulating the development of soft skills through more complex exercises requiring a wider and more diverse set of skills.

One of the first fundamental steps to building these systems lies in the ability to model the human learning process, taking into account all the possible information about students, educational resources, testing material, their organization, and their interactions. Pragmatically, in order to present learners with the most suited pedagogical content, we need to be able to trace the knowledge they acquire over time and assess their ability to face a specific question, problem, or exercise. According to the Goldilocks principle (Koedinger et al., 2013), the optimal solution is to have users confronted with educational resources that have an intermediate level of difficulty for them. Indeed, presenting users repeatedly with questions that are above their level of knowledge will cause frustration, while proposing exercises that are too easy or cover topics that are already mastered by the users will bore them. In both cases, the learning will not be optimal and the risk is that users will abandon the process.

The challenge of predicting students' outcomes when interacting with a given educational resource, also known as *Knowledge Tracing*, has been largely investigated and several approaches, based on diverse mathematical models, have been proposed throughout the years, e.g. *Bayesian Knowledge Tracing (BKT)* (Corbett and Anderson, 1994), *Additive Factors Model (AFM)* (Cen et al., 2006), and *Performance Factors Analysis (PFA)* (Pavlik et al., 2009). A significant boost to the performance of Knowledge Tracing solutions was given by the employment of Deep Learning techniques (LeCun et al., 2015), with the development of the first approach, Deep Knowledge Tracing (DKT) (Piech et al., 2015), which turned out to be a game-changer, substantially outperforming all pre-

vious models. Consequently, most of the following research work focused on the analysis (Gervet et al., 2020, Kevin H. Wilson and Yan Karklin and Bojian Han and Chaitanya Ekanadham, 2016, Xiong et al., 2016) and extensions of DKT (Sonkar et al., 2020, Wang et al., 2019) and on the development of new "deep" models for Knowledge Tracing (Nakagawa et al., 2019, Vie, 2018, Yang, Yang and Shen, Jian and Qu, Yanru and Liu, Yunfei and Wang, Kerong and Zhu, Yaoming and Zhang, Weinan and Yu, Yong, 2020). However, DKT and other deep approaches present several important shortcomings, some of which are common to previous non-deep models, and some others that are related to the deep nature of the underlying models.

First and foremost, most of the KT models developed so far rely on a very limited amount of information, simplifying excessively the description of the learning environment (Liu et al., 2021). In most cases, the prediction of students' outcomes is based only on the knowledge of the past tests taken by each student, with questions and skills identifiers. Notions, such as possible types of questions and exercises, different kinds of skills required for resolution, i.e. soft or hard, collaborative assignments, and partially correct answers are very rarely taken into account and most often completely ignored by such models. Even in the case of models that are explicitly extended to consider this richer information, additional ad-hoc features need to be handcrafted on a single-case basis by domain experts to be injected into the original model. It is, therefore, impossible to seamlessly model and account for the complexity required by a real-world Interactive Learning Environment.

Another shortcoming common to all the KT models based on Deep Learning techniques is the need for a large quantity of training data (Liu et al., 2021). As proven by (Gervet et al., 2020), DKT is susceptible to overfitting and performs poorly in the case of small datasets. The capacity to work with a limited amount of data is fundamental in this context because a qualitative Knowledge Tracing model should be able to provide trustable predictions also for users who are in an early stage of their learning process and, therefore, who have a short history of previous interactions with the learning material, and in learning contexts involving a limited number of users.

Finally, one of the main drawbacks of any system based on Deep Learning models is the lack of explainability and interpretability of the provided solution. While Deep Knowledge Tracing models can be pretty good in forecasting whether a user will answer correctly or not a given question, they are not able to generate an explanation for this prediction (Liu et al., 2021). This represents a major limitation from both a pragmatic and ethical point of view. Knowing that a student will fail on a given exercise can be useful, but, in order to actually help students overcome their difficulties, it would certainly be better to know why they are failing. Moreover, predicting a student's failure or success is a very sensitive task, and it is essential to be certain that no external bias is introduced into the decision process.

All the downsides exposed so far urge the necessity to investigate other research paths for Knowledge Tracing and to develop more comprehensive models able to *(i)* exploit a richer representation of learning environments, *(ii)* work even with a small quantity of

data, *(iii)* easily generalize over different education contexts without need for additional handcrafted features, and *(iv)* produce results that can be more easily interpreted.

The solution for building such a Knowledge Tracing model, able to overcome the inherited limitations of Deep Learning, i.e. data hunger, limited capacity for transfer, lack of transparency, and non-integration with prior knowledge (Marcus, 2018), might lie in the use of hybrid approaches, combining Deep Learning methods with Symbolic AI.

In the last decade, a growing interest has been sparkling around the joint use of Knowledge Representation and Reasoning and Deep Learning, to overcome the main drawbacks of the latter. Knowledge Graphs have been largely used to integrate prior domain-expert knowledge into Deep Learning models, enabling, for example, zero-shot learning (Lee et al., 2018) and transfer learning (Ammanabrolu and Riedl, 2019). At the same time, an increasing number of scientific contributions have been done to analyze the role that Knowledge Graphs could play in AI explainability and interpretability (Díaz-Rodríguez et al., 2022, Futia and Vetrò, 2020, Gaur et al., 2021, Lecue, 2020, Tiddi and Schlobach, 2022). Also, the range and domain of applications that utilize this kind of combined approaches have been growing steadily: object detection (Fang et al., 2017), art analysis (Castellano et al., 2022) and scientific work classification (Hoppe et al., 2021) are only some examples of applications relying both on Knowledge Graphs and Deep Learning models.

Inspired by these recent research trends, the goal of this Ph.D. thesis is to explore if and how Knowledge Representation and Reasoning can be used and useful for providing better solutions to the Knowledge Tracing problem. We aim to propose new models to describe the human learning process. These models should be able to account for more complex and complete information about both students and learning material, opening the way for the development of more comprehensive intelligent learning environments. We also investigate the interpretability of the proposed methods with the final goal of assessing their usefulness in real-world use cases, optimizing them for different learning contexts, and providing insights into how the learning process happens.

1.2 Research Questions

As described in Section 1.1, our goal is to assess whether and how symbolic AI and more specifically Knowledge Representation and Reasoning can be employed for enhancing online education systems and, in particular, to help solve the task of predicting students' performance, also known as Knowledge Tracing (KT). To this end, we will explore the following research questions:

- **RQ1: How can we exploit Knowledge Representation for Knowledge Tracing purposes?** This question involves two main steps. Firstly, we need to model the complex and heterogeneous information normally available in modern computer-supported learning environments in the form of a Knowledge Graph. Subsequently, we need to design a model which would be able to exploit this rich representation of the learning environment for predicting students' performance. In order to do that,

we will rely on the use of Graph Embeddings, a low dimensional representation of the elements in the graph, which is able to capture from the graph and summarize the main features of each node and edge.

- **RQ2: Under what conditions and in which context does our Knowledge Graph based approach to Knowledge Tracing work? Can it be generalized across different learning environments and to other Knowledge Graphs?** To assess the robustness and generalization capabilities of our approach, we need to identify in which conditions and contexts, our Graph Embeddings enhanced model works. To do so, it is essential to study the behavior of the predictive model with different Knowledge Graphs and embedding settings to understand what piece of information contained in the Knowledge Graph is interesting for the students' outcomes prediction and how to optimally capture it through the embeddings. Therefore, we analyze how modifications in the content and structure of the input Knowledge Graph and the parameters of the embedding model affect the quality of our prediction.

- **RQ3: How to interpret and explain the prediction produced by our Knowledge Graph based model?** The ability to understand why a Knowledge Tracing model predicted students' success or failure is of the uttermost importance. Since our model relies on the use of Graph Embeddings that automatically identify and extract the most relevant features from the Knowledge Graph, the explanations to our predictions will be hidden in the embedding process. Therefore, the more general research question we need to answer is **how to unveil the information captured and encoded by Graph Embeddings?**

1.3 Contributions

The main contributions of the present thesis are the following:

Contribution 1 - Proposition of a first method to integrate Knowledge Graphs in state-of-the-art Knowledge Tracing techniques. To answer RQ1, we start exploring current state-of-art Knowledge Tracing models and analyzing if and how they can be extended to take into consideration richer information about a learning system represented in the form of a Knowledge Graph. After a thorough analysis of the state-of-the-art, we select DeepFM (Guo et al., 2017) as a base model to be extended for two reasons: its high performance in the Knowledge Tracing task already demonstrated by (Vie, 2018) and the simplicity to include further input features extracted or computed from the Knowledge Graph. In our contribution, we build several models obtained by extending DeepFM with the addition of one or more input features either handcrafted from information included in the KG or automatically computed through the Graph Embedding process. We test these newly proposed models on the SIDES use case, which is a real-world educational platform

already relying on a Knowledge Graph to store their data[1]. We found out that some of the models including Graph Embeddings among their input features perform better than both the base model (classic DeepFM) and models including hand-crafted features, meaning that Graph Embeddings are able to identify relevant information described in the graph which are usually ignored by standard models. This contribution has been presented in (Ettorre et al., 2020).

Contribution 2 - Development of a tool which enables a first interpretation of Knowledge Graph Embeddings through visual analysis. Answering RQ2 requires us to test our approach with multiple input Knowledge Graphs, different in nature and structure of the represented information, and several Graph Embeddings settings. While trying to assess the behavior of the proposed model in varying conditions, we determined that the quality of the prediction was strongly dependent on the characteristics of the employed Graph Embeddings and, in particular, the information they can capture from the Knowledge Graph. We discovered that Graph Embeddings computed from different Knowledge Graphs representing slightly different information would produce very diverse Knowledge Tracing results, as well as Graph Embeddings computed from the same KG using different embedding methods or even hyper-parameters. Indeed, even though the Knowledge Graph contains potentially relevant information for the Knowledge Tracing task, we are able to exploit it fully only if we rely on the use of an optimal KG structure and specific Graph Embedding methods, that would allow us to capture and encode such information. Therefore, being able to identify which KG modeling choices, Graph Embeddings models and hyper-parameters would produce the most meaningful embeddings for Knowledge Tracing is fundamental. Pragmatically, we need a way to compare several Graph Embeddings computed using different settings, analyze how and why they are different and get some insights into the information they capture and encode from the KG. To do so, we developed Stunning Doodle, a tool for the the joint visualization and analysis of Knowledge Graphs and their Graph Embeddings. This tool proved useful to analyze several Graph Embeddings computed from the OntoSIDES KG, and helped us discover which modeling choices and embedding models were producing more meaningful embeddings for our task and why. Nevertheless, Stunning Doodle is a general-purpose tool, that can be employed by anyone to visualize any RDF Knowledge Graph and study its embeddings. This contribution has been presented in (Ettorre et al., 2022a) and (Ettorre et al., 2022b).

Contribution 3 - Introduction of a systematic approach to help decode the information captured by Knowledge Graph Embeddings Even though Stunning Doodle can help us get an idea of under which settings Graph Embeddings incorporate the most interesting information for our final task, it provides only very high-level insights on what information is possibly encoded by these Graph Embeddings. To be able to interpret the Knowledge Tracing results provided by our model and give an answer to RQ3, we need to decode the information captured by the Graph Embeddings and, consequently, used

[1] A detailed description of the SIDES data and the related Knowledge Graph is given in Chapter 3

to estimate students' success. In this contribution, we present a systematic approach to verify whether a specific piece of information about the elements in the KG is captured and encoded by their Graph Embeddings. We show how this approach can be used to unveil the information encoded by Graph Embeddings computed using different embedding models, hyper-parameters and Knowledge Graphs. With respect to our use case, we rely on the proposed method to improve the GEs employed in our Knowledge Tracing model and verify what information is actually meaningful to enhance the quality of the students' outcome prediction. This contribution has been presented in (Ettorre et al., 2021).

Contribution 4 - Presentation of a Knowledge Tracing model exclusively based on Knowledge Graphs. Thanks to the contributions made in (Ettorre et al., 2021, 2022a), we were able to get a first interpretation of the Graph Embeddings employed to solve Knowledge Tracing task, which highlighted several shortcomings in the model proposed in (Ettorre et al., 2020). To overcome these shortcomings, we proposed a second Knowledge Tracing method entirely and uniquely based on the use of Knowledge Graphs, which does not require any additional handcrafted feature. In this contribution, we reformulate the Knowledge Tracing task as a Link Prediction problem on a Knowledge Graph describing the learning environment and we predict students' outcomes to questions by determining the most probable link between each answer and its correct or wrong realizations. This way, we are able to take into account all the possible information about the learning system without the need for external models or hand-crafted input features. We tested this approach on two different learning contexts: SIDES, which was already employed as use case in the previous experiments; and ASSISTment, a well-known and largely used Knowledge Tracing benchmarking dataset. We proved that, despite its simplicity and the lack of temporal information, this newly proposed approach achieves better results than Deep Knowledge Tracing. This contribution has been presented in (Ettorre et al., 2022c).

1.4 Structure

This thesis is organized as follows:

Chapter 2 introduces the concepts and background knowledge used throughout the thesis. It describes the current state-of-the-art for Knowledge Tracing, as well as the technical and theoretical limitations of Deep Learning. It explores the existing panorama of hybrid AI approaches relying on the use of Knowledge Graphs and Knowledge Graph Embeddings, and it presents the most widely used Graph Embeddings methods in this context.

Chapter 3 describes the input datasets used for the experiments presented in this thesis. It introduces the SIDES project describing the data collected in the context of such project and their representation in the form of a Knowledge Graph, OntoSIDES. Also, the ASSISTment dataset, employed for testing and benchmarking the last proposed model, is illustrated in the same chapter.

Chapter 4 presents the first contribution done during this Ph.D. thesis. It describes a model for Knowledge Tracing, which relies on the use of information extracted from the OntoSIDES Knowledge Graph. It shows the results obtained when introducing Graph Embeddings as input features for the existing DeepFM model.

Chapter 5 details further experiments carried out to study the behavior of the model presented in Chapter 5 with different input data representations and settings for the embedding model. The goal of the experimentation is answering RQ2 by determining in which conditions our predictive model works and to identify the relevant information to be included in a KG representing a learning environment and the best settings for its embedding.

Chapter 6 is about Stunning Doodle, a tool for the joint visualization and analysis of Knowledge Graphs and Graph Embeddings. It describes the tool with its main features which enable to display and navigate any RDF Knowledge Graph and to visualize distances between nodes in the embedding space. These functionalities allow us to compare Graph Embeddings computed through several embedding models or using different settings and to gain first high-level insights on the meaning of such Graph Embeddings. This chapter also illustrates the use of Stunning Doodle in the OntoSIDES use case. We describe how the implemented functionalities are employed to study multiple Graph Embeddings computed from the same KG but with various settings. Through the visual analysis, we can identify the differences in the information encoded when using the same embedding model with various hyper-parameters and, to some extent, explain the varying performances obtained by our Knowledge Tracing model.

Chapter 7 describes a method to precisely and systematically identify the information encoded by Graph Embeddings. The approach, based on a similar technique employed by the NLP community, is explained and a taxonomy of Knowledge Graph properties which can be encoded in the embedding process is presented. We show how the approach can be used to decode information captured by different embedding models from two distinct Knowledge Graphs: OntoSIDES and YAGO3. Moreover, we describe how the approach has been useful to identify the information in the OntoSIDES KG that was mainly exploited by our Knowledge Tracing model to provide its predictions. This analysis helped us discover an important shortcoming in the proposed method.

Chapter 8 presents a new approach to Knowledge Tracing based exclusively on the use of the Knowledge Graph describing the learning system. The newly proposed method overcomes the issues discovered in the previous model. In this chapter, the 4-steps framework which allows us to turn the Knowledge Tracing task into a Link Prediction problem on a Knowledge Graph is described and the results of the application of such method on two different use cases, i.e. SIDES and ASSISTment, are shown.

Chapter 9 summarizes the content of this thesis: the contributions made, the research questions that have been answered and the open challenges for the future. Perspectives for the improvement of the proposed models and developed tools are described, as well as ideas to address the challenges for the future of e-Education.

My publications

Antonia Ettorre. antoniaettorre/stunning_doodle: First version, December 2021. URL https://doi.org/10.5281/zenodo.5769192.

Antonia Ettorre, Oscar Rocha Rodríguez, Catherine Faron, Franck Michel, and Fabien Gandon. A Knowledge Graph Enhanced Learner Model to Predict Outcomes to Questions in the Medical Field. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 237–251. Springer, 2020.

Antonia Ettorre, Anna Bobasheva, Catherine Faron, and Franck Michel. A systematic approach to identify the information captured by Knowledge Graph Embeddings. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2021.

Antonia Ettorre, Anna Bobasheva, Franck Michel, and Catherine Faron. Stunning Doodle: a Tool for Joint Visualization and Analysis of Knowledge Graphs and Graph Embeddings. In *European Semantic Web Conference*, pages 370–386. Springer, 2022a.

Antonia Ettorre, Anna Bobasheva, Franck Michel, and Catherine Faron. Stunning Doodle: un outil pour la visualisation et l'analyse conjointe de graphes de connaissances et leurs plongements. In *IC Journées francophones d'Ingénierie des Connaissances - PFIA 2022*, pages 99–100, 2022b.

Antonia Ettorre, Franck Michel, and Catherine Faron. Prediction of Students' Performance in E-learning Environments Based on Link Prediction in a Knowledge Graph. In Maria Mercedes Rodrigo, Noburu Matsuda, Alexandra I. Cristea, and Vania Dimitrova, editors, *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium*, pages 432–435, Cham, 2022c. Springer International Publishing. ISBN 978-3-031-11647-6.

PART I

Background

CHAPTER 2

State of the Art

This chapter provides the reader with the fundamental concepts used throughout the thesis and it introduces the background knowledge needed for its understanding. Moreover, it describes the current state-of-the-art for Knowledge Tracing, and it exposes the technical and theoretical limitations of Deep Learning approaches. It presents the most important notions related to Knowledge Graphs and their application in real-world scenarios. Finally, it introduces the concept of Knowledge Graph Embeddings and it explores the existing panorama of embedding methods for Knowledge Graphs.

2.1 Knowledge Tracing

As mentioned in Section 1.1, the problem of tracing students' knowledge levels has been largely investigated in the past, mainly because of its importance for the implementation of users' personalized and adaptive learning programs. The research efforts made so far produced a rather wide panorama of distinct approaches whose aim is to model the students learning process and assess their ability to face specific Knowledge Items (KIs), i.e. questions, exercises or problems, at any moment in time.

2.1.1 Classical Knowledge Tracing Models

The first attempt to predict the performance of students was formally made by Gulliksen in his **Theory of Mental Tests** (Gulliksen, 1950). Gulliksen describes a set of equations to compute the test score for each student based on the student's true ability and an estimated error component for the same student. Several subsequent research works further analyzed and formalized the problem (Birnbaum, 1969, Lord et al., 1968, Novick, 1966) leading to the development of the **Item Response Theory (IRT)** (Hambleton et al., 1991), which estimates the success probability of a student on a single Knowledge Item, instead of their score on a complete test.

IRT estimates students' performance by learning a logistic function whose parameters describe the student and the item involved in the interaction. The probability of a student correctly answering an item is a monotonically increasing function of the measured ability of the student and it is characterized by some factors specific to the knowledge item (e.g. difficulty). Throughout the years, several IRT models have been developed with increasing complexity. The first and simplest IRT model, known as Rasch model or 1PL, has been introduced in (Rasch, 1960). It describes the test items in terms of only one parameter: the item difficulty. Therefore, the probability of responding correctly to an item is, as shown in Eq. (2.1), a logistic function defined only by two parameters: the ability of the student (θ_i) and the difficulty of the item (β_j).

$$p_{ij} = \frac{e^{(\theta_i - \beta_j)}}{1 + e^{(\theta_i - \beta_j)}} \tag{2.1}$$

The Rasch model has been subsequently extended by the 2PL model to introduce the *discrimination parameter* which models how well an item can differentiate students. Later, the 3PL model was developed to generalize the 2PL model by adding the *(pseudo)guessing parameter* which expresses the property that even very low ability persons have a positive probability of answering an item correctly, simply by randomly guessing the correct answer. Finally, the last and most complex IRT model is the 4PL model (Barton and Lord, 1981) which adds a fourth parameter modeling the "inattention" of high ability students failing to answer an easy item correctly. The formula to compute the probability according to the 4PL IRT model is the following:

$$p_{ij} = c_j + (d_j - c_j)\frac{e^{a_j(\theta_i - \beta_j)}}{1 + e^{a_j(\theta_i - b_j)}} \tag{2.2}$$

where a_j represents the discrimination parameter, c_j is the guessing parameter and d_j models the "slipping" chance of a prepared student.

These four IRT models were further extended to model richer items parameters. A first important extension to IRT models is represented by *mIRT* (Reckase, 1997), which measures multiple item's latent traits simultaneously, enabling the description of questions requiring the use of multiple skills. Lately, (Wilson et al., 2016) proposed the *Hierarchical IRT (HIRT)* and *Temporal IRT (TIRT)*. HIRT exploits similarities among questions based on the skills they involve. This approach assumes that the difficulty parameters of questions testing similar skills are generated from the same distribution. Intuitively, questions requiring the use of trivial skills will generally be easy to answer, while questions involving complex skills will be more difficult. TIRT, instead, models items' difficulty parameters as time-varying stochastic processes.

Another well-known Knowledge Tracing model based on logistic regression is the **Additive Factors Model (AFM)** (Cen et al., 2006). AFM is built on the assumption that the probability of a student answering correctly a given item depends on four factors: *(i)* the ability of the student (θ_i), *(ii)* the difficulty of each skill (β_k), also called knowledge component or KC, required by the knowledge item (j), *(iii)* the number of times the student interacted with that same skill before (T_{ik}), and *(iv)* the amount of learning gained from each interaction with such a skill (γ_k). The additive combination of these four factors represents the argument of the logistic regression, as shown in Eq. (2.3).

$$p_{ij} = logit(\theta_i + \sum_{k \in K(j)} (\beta_k + \gamma_k T_{ik})) \tag{2.3}$$

AFM has been extended in (Pavlik et al., 2009), which proposes the **Performance Factor Analysis (PFA)**. This new model, also based on logistic regression, modifies AFM by removing the student ability factor and, at the same time, discriminating the students' attempts between successes and failures (see Eq. (2.4)). The rationale behind the PFA model is that correct and incorrect attempts to a same skill do not lead to the same learning.

$$p_{ij} = logit(\sum_{k \in K(j)} (\beta_k + \gamma_k^S T_{ik}^S + \gamma_k^F T_{ik}^F)) \tag{2.4}$$

Another set of Knowledge Tracing solutions is represented by the Bayesian models. The pioneers of the use of such models are Corbett and Anderson, who introduce **Bayesian Knowledge Tracing (BKT)** (Corbett and Anderson, 1994). BKT models the learning process as a Hidden Markov Model based on two types of binary variables: a latent variable representing the mastery of each skill, i.e. *learned* or *unlearned*, and the observed variable indicating the result of the student's attempt to a question, i.e. *correct* or *incorrect*. In this model, every skill is characterized by four parameters: *(i)* the probability of mastering the skill before learning ($p(L_0)$), *(ii)* the probability of transitioning from state *unlearned* to *learned* ($p(T)$), *(iii)* the probability of slipping (i.e. making a mistake even though the skill is learned) ($p(S)$), and *(iv)* the probability of guessing correctly even though the skill is not mastered ($p(G)$). At each time step, the probability that a skill is

mastered is estimated ($p(L_n)$) and, based on that, the probability that a question involving that skill is correctly answered can be computed as the sum between the probability of mastering the skill and not "slipping", and the probability of not mastering the skill but guessing correctly (see Eq. (2.5)).

$$p(L_n) * (1 - p(S)) + (1 - p(L_n)) * p(G) \tag{2.5}$$

BKT presents two main shortcomings which have been addressed by variations of the standard model. Firstly, BKT does not include any parameters representing the different students, meaning that the learning process only depends on the skills to be learned and not on the learner itself. Several extensions of BKT were proposed throughout the years (Khajah et al., 2014, Lee and Brunskill, 2012, Yudelson et al., 2013) moving towards an Individualized Bayesian Knowledge Tracing model, which would take into account the individual characteristics of each student. The second limitation of BKT is that, in the original implementation, it assumes that each knowledge item requires a single skill and skills are mutually independent, but in real-world use-cases questions can be much more complex and involve several skills which, at the same time, can present dependency relations between each other. Dynamic BKT (Käser et al., 2017) has been developed to overcome this issue as it allows to jointly model multiple skills and their dependency.

More recently, Vie and Kashima proposed the **Knowledge Tracing Machines (KTM)** (Vie and Kashima, 2019) a new approach based on a different mathematical model, Factorization Machines (FM) (Rendle, 2010). The main strength of KTM is the ability to consider as input any available information about students, questions, skills or other factors possibly affecting the learning experience. Indeed, an input sample for a KTM is represented by a vector x of dimension N, where N is the number of factors to be considered. Each strictly positive element in such vector ($x_k > 0$) indicates that the corresponding factor ($k \in [1, N]$) is involved in the considered sample. Finally, the probability of a student giving a correct answer is obtained by applying a logistic regression to an additive combination of the elements in the vectors (x_k) with their weights (w_k) and a second term describing the pairwise interactions between input factors, as shown in Eq. (2.6). Given its generic formulation, KTM encompasses several existing models, such as IRT, AFM and PFA, as special cases.

$$p(x) = logit(\sum_{k=1}^{N} w_k x_k + \sum_{1 \leq k < l \leq N} x_k x_l \langle v_k, v_l \rangle + \mu) \tag{2.6}$$

An interesting extension of KTM was presented in (Vie, 2018). The proposed approach relies on Deep Factorization Machines (DeepFM) (Guo et al., 2017), which combines the power of FM for recommendation and Deep Learning for feature learning.

2.1.2 Deep Learning Models for Knowledge Tracing

Recently, approaches based on Deep Learning architectures have gained notable success. The first method to rely on such techniques was **Deep Knowledge Tracing (DKT)** (Piech

et al., 2015) which models students' learning and predicts their outcomes to questions based on their prior activity. This model is trained using the sequence of skills every student faced with the binary result of the interaction, 1 to indicate a correct answer and 0 for an incorrect one. This approach exploits the ability of Recurrent Neural Networks (RNNs) and, more specifically, Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to store a high-dimensional latent state and make predictions based on past information received as input much earlier. Thanks to these characteristics, DKT sensibly outperformed other state-of-the-art models and contributed to creating a new research thread of Knowledge Tracing models based on Deep Learning architectures.

Nevertheless, some subsequent research efforts highlighted several shortcomings in the DKT method and mitigated its claims (Kevin H. Wilson and Yan Karklin and Bojian Han and Chaitanya Ekanadham, 2016, Khajah et al., 2016, Xiong et al., 2016). The authors of (Xiong et al., 2016) identified an important issue in the DKT resolution when questions require multiple skills. In fact, given the formulation of the problem, when a student answers an item tagged with multiple skills, DKT is fed with a sequence of different skills with the same result, therefore, the model learns to predict the same result for all the skills in such a sequence. Let us imagine that a student faces a question $Q1$ requiring the skills $S1$ and $S2$ to be solved and that such question is answered correctly. The corresponding input sequence to DKT would be $S1, S2$ with correctness labels $1, 1$. Given this input, the model will learn that, when skill $S2$ follows skill $S1$, $S2$ will very likely have the same result than the previous $S1$. Xiong et al. demonstrated that, when the duplicated skills sequences are removed from the input data, the performance of DKT significantly decreases and it can be matched by other non-deep Knowledge Tracing models, such as PFA. (Kevin H. Wilson and Yan Karklin and Bojian Han and Chaitanya Ekanadham, 2016), instead, directly compared DKT, IRT and its temporal and hierarchical extensions, i.e. TIRT and HIRT, on three different datasets. The authors found that IRT-based models were able to equal or outperform DKT on all the datasets, while requiring less effort in terms of parameters tuning and minor computational resources. Moreover, (Khajah et al., 2016) carried out a thorough analysis to identify which characteristics of the learning process were exploited by the DKT model that could not be captured by BKT. They determined that input data presented four types of regularities which could not be handled by classical BKT: recency effects, contextualized trial sequences, inter-skill similarity and individual students abilities. To overcome these limitations, they extended BKT to introduce a forgetting behavior, discover interactions between skills and model the individual student abilities. Classical BKT and the implemented variations were then compared against DKT on several datasets. The results of such comparison show that extensions of BKT taking into account regularities in the data matched or even outperformed DKT on some of the tested datasets. Yet, the implementation of such extensions required an important study of the characteristics of the data and of how to exploit them in BKT, while DKT is a general purpose framework able to automatically identify and handle such characteristics. Finally, a few recent research works (Nakagawa et al., 2019, Sonkar et al., 2020, Yang, Yang and Shen, Jian and Qu, Yanru and Liu, Yunfei and Wang, Kerong and Zhu, Yaoming and Zhang, Weinan and Yu, Yong, 2020) pointed out an additional non-negligible limi-

tation of DKT: it models students' performance on questions only based on the previous interactions of such students with the skills involved in the question. This means that several questions involving the same set of skills, but with potentially very different levels of complexity, will be treated in the same way by this model.

Since DKT was proposed in 2015 (Piech et al., 2015), many other deep models for Knowledge Tracing were developed to improve the performance of the former and overcome its limitations. A first set of new deep approaches were presented to improve the temporal behavior of the model and to better represent the real learning process. On one side, models such as *Dynamic Key-Value Memory Network (DKVMN)* (Zhang et al., 2017) and *Sequential Key-Value Memory Network (SKVMN)* (Abdelrahman and Wang, 2019) enable tracing the learning of more complex skills by augmenting LSTM with the addition of external memory units. On the other side, some approaches (Nagatani et al., 2019) were proposed to try to capture students' forgetting behavior to more accurately estimate their current knowledge state.

At the same time, attention mechanisms started to be introduced into KT models to capture the different contributions that each previously answered question could bring to the prediction of the outcome of the current question. *Attentive Knowledge Tracing (AKT)* (Ghosh et al., 2020), *Self-Attentive Knowledge Tracing (SAKT)* (Pandey and Karypis, 2019), and *Relation-Aware Self-Attention for Knowledge Tracing (RKT)* (Pandey and Srivastava, 2020) are only few examples of approaches which learn attention weights associated to every knowledge item. This process allows the model to treat different questions differently even though they are related to the same set of skills.

Another research direction focuses on extending the existing Deep Knowledge Tracing models, merely relying on the sequences of students' interactions with learning material, to take advantage of more complete and comprehensive information about the knowledge items and components. Some of the most recent models (Nakagawa et al., 2019, Tong et al., 2020, Yang, Yang and Shen, Jian and Qu, Yanru and Liu, Yunfei and Wang, Kerong and Zhu, Yaoming and Zhang, Weinan and Yu, Yong, 2020) are able to exploit a graph representation of the mutual interactions between skills and questions. While *Graph-based Knowledge Tracing (GKT)* (Nakagawa et al., 2019) relies on a Graph Neural Network (GNN) taking as input a graph representing the dependency relations between skills, *Graph-based Interaction Knowledge Tracing (GIKT)* (Yang, Yang and Shen, Jian and Qu, Yanru and Liu, Yunfei and Wang, Kerong and Zhu, Yaoming and Zhang, Weinan and Yu, Yong, 2020) exploits a Graph Convolutional Network (GCN) fed with a graph describing questions-skills relations. *Structure-based Knowledge Tracing (SKT)* (Tong et al., 2020), instead, differs from GKT as it models not only the dependency relations between skills but also similarities and prerequisites. Other deep Knowledge Tracing models, like *Exercise-Enhanced Recurrent Neural Network (EERNN)* (Su et al., 2018) are able to exploit the textual content of knowledge items by building a vector representation of questions' text that is after combined with the student interactions as input to a LSTM.

Notwithstanding the progress made by Knowledge Tracing approaches based on Deep Learning and the stunning performance these methods can achieve in specific use-cases,

there are still several open issues that need to be addressed. As reported by (Abdelrahman et al., 2022, Liu et al., 2021), the impossibility of capturing and exploiting the complexity of a real learning environment and the difficulty of interpreting the solutions produced by these models are of major concern. In this Ph.D. thesis, we will try to tackle these issues by relying on well-known models and techniques based on the use of Knowledge Graphs.

2.2 Knowledge Graphs

Knowledge Graphs have their roots in classical Artificial Intelligence and more precisely in the field of Knowledge Representation and Reasoning. Although their origin can be traced back to the 1980s, Knowledge Graphs have raised to fame in 2012, when Google released its *Knowledge Graph*[1], a graph-structured representation of the information that would bring search beyond simple string matching. Nevertheless, before Google Knowledge Graph, there were already several projects aiming at extracting and publishing graph-structured representation of the information on the Web, e.g. DBpedia (Bizer et al., 2009) and Freebase (Bollacker et al., 2008), with the goal of building a landscape of Linked Data (Bizer et al., 2011) and enabling Tim Berners-Lee's vision of the Semantic Web (Berners-Lee et al., 2001). Despite the success and widespread utilization, the concept of Knowledge Graph is still blurry and a unanimously accepted definition has not yet been formulated (Ehrlinger and Wöß, 2016). The most complete and precise definition so far seems to be the following one, given in (Paulheim, 2017):

Definition 1 (Knowledge Graph)**.** A knowledge graph (i) mainly describes real world entities and their interrelations, organized in a graph, (ii) defines possible classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains.

In the context of this thesis, we will be concerned with the use of a specific type of Knowledge Graphs following the RDF standard[2], normally employed by the Semantic Web community, and referred to as RDF Knowledge Graphs.

2.2.1 RDF Knowledge Graphs

The **Resource Description Framework (RDF)** is a framework for expressing information about *resources* (Schreiber et al., 2014). The term "resource" can refer to anything, including documents, people, physical objects, and abstract concepts. Every resource can be described by using *statements*, also called *triples*, which consist of the following elements:

$$(subject, predicate, object). \tag{2.7}$$

[1] https://blog.google/products/search/introducing-knowledge-graph-things-not/

[2] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

Such a statement indicates that the resource *subject* has a relation of type *predicate* with the resource *object*. From a graphical point of view, a triple can be represented as an edge between two nodes, i.e. the resources, in a directed graph. Finally, the same resource, identified through a unique identifier known as **International Resource Identifier (IRI)**, can be referenced by multiple triples. Figure 2.1 shows the Knowledge Graph corresponding to the triples:

$$(Bob, is\,married\,to, Alice).$$
$$(Bob, has\,brother, Charlie).$$

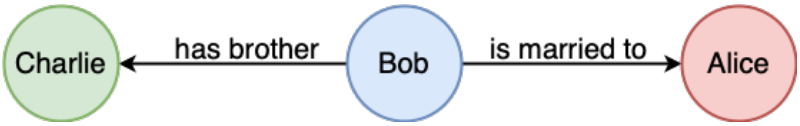


Figure 2.1: Example of a RDF KG representation.

RDF Schema (RDFS)[3] is a general-purpose language to define RDF vocabularies. RDFS provides us with primitives to declare classes, i.e. types of resources, and properties, and to describe their characteristics. For example, with RDFS we can declare a class `human` for Alice, Bob and Charlie and state that property `is married to` relates instances of such a class.

RDFS is a rather simple language which might not be sufficient to define very precise vocabularies covering a specific domain and requiring the formalization of advanced definitions for classes and properties. The **Ontology Web Language (OWL)**[4] was developed to fill this gap. OWL specifies a set of constructs to further describe properties, classes and related constraints. For example, with OWL we can state that classes `Man` and `Woman` are disjoint, as an instance of `Man` cannot be, at the same time, an instance of `Woman`.

Finally, once we have a Knowledge Graph representing our data according to the RDF data model, we need a way to interrogate such Knowledge Graph. Such interrogations are carried out by using **SPARQL**, the query language for RDF.

2.2.2 Knowledge Graphs Applications

Nowadays, KGs are used in a wide range of applications and contexts. The primary goal of Knowledge Graphs is to enable the representation of heterogeneous information in a machine understandable way, which allows machines to reason on the known data and infer new knowledge. Thanks to these features, Knowledge Graphs proved to be extremely efficient in tasks such as search, questions answering and recommendation systems (Ji

[3] https://www.w3.org/TR/2014/REC-rdf-schema-20140225/
[4] https://www.w3.org/TR/owl2-overview/

et al., 2021). For example, search engines can rely on Knowledge Graphs to answer queries like "painter of mona lisa" simply by recognizing the resource "Mona Lisa" and the relation "painted", whose subject is "Leonardo Da Vinci", instead of retrieving all the documents on the Web that would match the terms "painter" and "Mona Lisa". Moreover, KGs are being increasingly employed for the representation and exploitation of domain-specific knowledge (Abu-Salih, 2021): medicine (Huang et al., 2019, Rotmensch et al., 2017), finance (Liu et al., 2019, Ruan et al., 2016), art (Castellano et al., 2022) and tourism (Dadoun et al., 2019) are only some of the domains that start relying on KGs for storing their data without compromising their semantics and exploit them for the implementation of knowledge-aware applications, which integrate symbolic knowledge into external computational models.

A common and fundamental step to be able to implement this kind of applications is represented by the *Knowledge Acquisition* process. Knowledge Acquisition aims to construct Knowledge Graphs from unstructured text and other structured or semistructured sources, complete an existing Knowledge Graph, and discover and recognize entities and relations (Ji et al., 2021). Knowledge acquisition involves mainly three types of tasks:

1. **Knowledge Graph Completion**, which aims to identify missing triples to add to a Knowledge Graph. It includes the subtasks of *Link Prediction*, *Entity Prediction* and *Relation Prediction*.

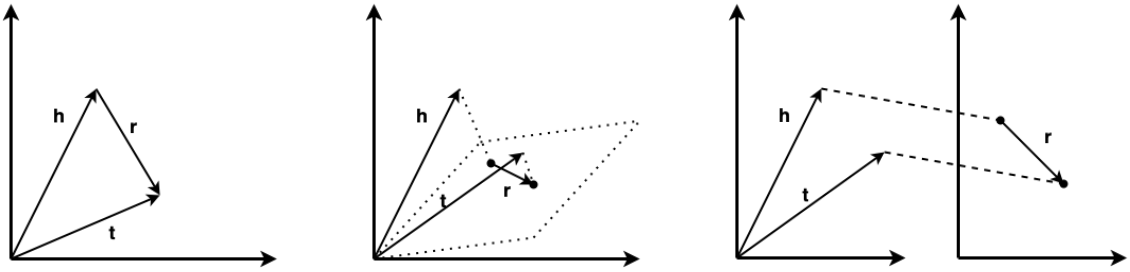
2. **Entity Discovery**, to complete the entity-related knowledge, involving *Entity Recognition*, *Entity Typing*, *Entity Disambiguation* and *Entity Alignment*.

3. **Relation Extraction**, to extract additional relations from external text sources and add them to the KG.

A significant boost to the use and exploitation of Knowledge Graphs was given by the introduction of the **Graph Embedding** concept, that now represents a cornerstone for both *(i)* the development of knowledge-aware applications, allowing the injection of knowledge from the graph into external models; and *(ii)* the solution of tasks related to Knowledge Acquisition.

2.2.3 Knowledge Graph Embeddings

The idea behind Knowledge Graph Embeddings, and more generally Graph Embeddings, is to embed the elements in the graph, i.e. nodes and edges, into a continuous vector space. The result of the embedding process is a low-dimensional vector that represents each element in the graph and captures its most important features. In the last decade, a plethora of graph embedding models have been developed. In the following, we describe some of the most well-known ones, some of which have been employed in the work for this thesis.



(a) TransE (b) TransH (c) TransR

Figure 2.2: Entities and relations embeddings (adapted from (Wang et al., 2017)).

Translational Distance Models. **TransE** (Bordes et al., 2013) is one of the first translation distance embedding models. This means that, in this model, relations are seen as translations in the embedding space. In other words, given a triple $(h, r, t) \in S$ where S is the set of valid triples, h is the head or subject, r is the relation or predicate, and t is the tail or object of the triple, the embedding of t should be closer to the embedding of h plus a vector representing the embedding of r, as shown in Figure 2.2a. For each triple $(h, r, t) \in S$, the embeddings of h, r and t, respectively indicated as $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$, are computed by maximizing the score:

$$- \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_{\ell_1/\ell_2} \tag{2.8}$$

At the same time, the model learns to minimize the score for triples that do not exist in the graph. To do so, a set of corrupted triples S' is generated by replacing, for each existing triple, its head and tail by a random entity, as follows:

$$S'_{(h,r,t)} = \{(h', r, t) | h' \in E\} \cup \{(h, r, t') | t' \in E\} \tag{2.9}$$

where E is the set of entities in the graph.

Despite its simplicity, TransE has proven to be very powerful, nevertheless it presents important shortcomings. (Lin et al., 2015, Wang et al., 2014a) highlighted the difficulties of TransE in dealing with 1-to-N, N-to-1 and N-to-N relations, and proposed extensions to the model to overcome them. Indeed, given the formulation of the problem, all the entities that appear as tails with the same head entity and relation will end up having very close embeddings, even though they might be very different based on other relations. A way for solving this issue would be to have different entity representations for each relation.

TransH (Wang et al., 2014a) implements this idea by embedding each relation r into a different hyperplane and then enforcing the assumption that the projection of the tail on such hyperplane ($\boldsymbol{t}_\perp$) must be close to the projection of the head ($\boldsymbol{h}_\perp$) on the same hyperplane plus the relation vector ($\boldsymbol{r}$). The scoring function becomes the following:

$$- \|\boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp\|_2^2 \tag{2.10}$$

TransR (Lin et al., 2015), instead, embeds entities and relations in two different embedding spaces of possibly different dimensions $\mathbb{R}^d$ and $\mathbb{R}^k$. The scoring function remains

the same as in Eq. (2.10), but, in this case, ($\boldsymbol{t}_\perp$) and ($\boldsymbol{h}_\perp$) are projections of, respectively, t and h in the space $\mathbb{R}^k$, instead of on a hyperplane.

Another shortcoming of TransE is the impossibility of modeling symmetric relations. Indeed, if r is symmetric, both (h, r, t) and (t, r, h) must hold, thus, we have:

$$\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t} \tag{2.11}$$

$$\boldsymbol{t} + \boldsymbol{r} = \boldsymbol{h} \tag{2.12}$$

which leads to $\boldsymbol{h} = \boldsymbol{t}$.

RotatE (Sun et al., 2019) aims to solve this issue by modeling relations as rotations in the complex space instead of simple translations in the real space. Therefore, $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{C}^k$, where the modulus of each element of $\boldsymbol{r}$ is constrained to 1. Finally, the RotatE score function is:

$$-\|\boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t}\| \tag{2.13}$$

where $\circ$ is the element-wise product of two vectors.

Semantic Matching Models. **RESCAL** (Nickel et al., 2011) is an embedding model relying on tensor factorization. For each relation r in the graph, a matrix ($\boldsymbol{M}_r$) is defined to represent pairwise interactions between the latent representations of all entities in the graph. Therefore, the score for a triple (h, r, t) is given by:

$$\boldsymbol{h}^T \boldsymbol{M}_r \boldsymbol{t} \tag{2.14}$$

where $\boldsymbol{h}$ and $\boldsymbol{t}$ are the embeddings of h and t in $\mathbb{R}^d$ and $\boldsymbol{M}_r \in \mathbb{R}^{d \times d}$. This implies that, for each relation, d^2 parameters need to be learned, making the resolution of the problem computationally too heavy.

To overcome this issue, **DistMult** (Yang et al., 2015) was proposed as a simplification of RESCAL. In DistMult, each relation is modeled as a diagonal matrix $\boldsymbol{M}_r$, whose diagonal is represented by a vector $\boldsymbol{r} \in \mathbb{R}^d$. This simplification reduces the parameters per relation from d^2 to d, but, at the same time, it drastically weakens the expressiveness of the embeddings which can only model symmetric relations.

ComplEx (Trouillon et al., 2016) was developed to conjugate the simplicity of DistMult with the original expressive power of RESCAL. In ComplEx, embeddings are represented in the complex space, i.e. $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^d$. In this case, the scoring function for a triple (h, r, t) becomes:

$$Re(\boldsymbol{h}^T diag(\boldsymbol{r}) \bar{\boldsymbol{t}}) \tag{2.15}$$

where $\bar{\boldsymbol{t}}$ is the conjugate of $\boldsymbol{t}$. This way, asymmetric relations can be represented since the score of a triple differs based on the role of the involved entities, i.e. (h, r, t) will have a different score from (t, r, h).

node2vec (Grover and Leskovec, 2016) is a general framework to learn continuous feature representations for nodes in networks. This mean that the node2vec algorithm has not specifically been conceived for Knowledge Graphs, instead it can be applied to any

kind of graph. Nevertheless, there are several works exploiting this model to compute Knowledge Graph Embeddings (Palumbo et al., 2018). The algorithm behind this model is based on the skip-gram architecture (Mikolov et al., 2013) and relies on the concept of random walks to build flexible neighborhoods for each node in the graph.

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges of the graph, the goal is to learn a function $f : V \rightarrow \mathbb{R}^d$, that, for each vertex $v \in V$, computes a feature representation of dimension d. The first step to learn the function f is the computation, for each node $v \in V$, of its neighborhood $N_S(v)$ according to a given strategy S. Once the neighborhood is known, the function f is learned by maximizing the log-probability of observing neighborhood $N_S(v)$ for each node v given its features representation:

$$\max_f \sum_{v \in V} logPr(N_S(v)|f(v)). \tag{2.16}$$

The novelty introduced by node2vec lies in the possibility of personalizing the sampling strategy S, by using parameterized random walks which enable the use of a mixture of Breadth-First Sampling (BFS) and Depth-First Sampling (DFS).

RDF2vec (Ristoski and Paulheim, 2016) is an embedding model developed specifically for RDF graphs. It relies on the use of neural language models to compute embeddings for each vertex in the graph. Normally, language models exploit the word sequences in a text to compute features representation for such words, assuming that closer words in the text should have similar feature representations. To be able to apply this approach to a Knowledge Graph it is necessary to convert the graph into a sequence of tokens. Therefore, given a graph $G = (V, E)$, for each vertex $v \in V$, a set of sequences S_v starting from v needs to be generated. Two strategies are used to compute these sequences: *Graph Walks*, which, given a depth d, lists all the possible walks starting from each vertex; and *Weisfeiler-Lehman Subtree RDF Graph Kernels* (de Vries, 2013), which relies on Weisfeiler-Lehman kernel (Shervashidze et al., 2011) to compute sequences of subtrees for each vertex in the graph. Once the sequence of tokens is generated, the Continuous Bag-Of-Words (CBOW) or Skip-Gram models of Word2vec (Mikolov et al., 2013) can be directly applied to compute the feature representation.

Further embedding models with auxiliary information. Recently, several research efforts are being made to extend embedding models by incorporating additional information that is not normally exploited in a Knowledge Graph. Some works focused on integrating Knowledge Graph Embeddings with textual information by combining graph and text embeddings (Wang et al., 2014b, Xie et al., 2016), while others tried to introduce image representations in the entity embedding space (Xie et al., 2017). Another interesting research direction is the development of time-aware embedding models (Jiang et al., 2016a,b) which are able to represent temporal dependency between triples keeping them temporally consistent in the embedding space.

CHAPTER 3

Datasets

This chapter presents the datasets used for the experiments presented in this thesis. It describes the SIDES project and the related OntoSIDES Knowledge Graph, which has been the first test-bed for the proposed approaches. Additionally, it introduces another well-known and widely used benchmarking dataset for Knowledge Tracing solutions which will be used in the latest experiments: ASSISTments.

3.1 OntoSIDES

Since 2013, teachers of French medical schools have been using a common national platform to create and give local evaluation tests on different devices. The Web-based platform, named *SIDES* (*Intelligent Health Education System*[1]), allows to share these tests among medical schools to build a national database for training, and supports the preparation of medical students for the ECNi (National Computerized Ranking Tests).

The French national project *SIDES 3.0* started at the end of 2017 and aims to develop a new version of the platform meant to offer user-centered intelligent services such as individual monitoring, enriched dashboards, personalized recommendations, augmented corrections for self-assessment, and a standardized digital environment for knowledge sharing. To achieve these goals, the approach taken leverages Semantic Web models and technologies to enrich and integrate these resources in RDF with OWL ontologies. As part of the *SIDES 3.0* project, existing data from the platform, such as annotated questions and students' learning traces, were converted into structured data expressed in RDF using the *OntoSIDES OWL ontology* (Palombi et al., 2019), and stored in the *OntoSIDES* Knowledge Graph. We decided to rely on OntoSIDES as a first dataset to benchmark our Knowledge Tracing solutions because it allows us to have an interesting real-world use-case that is already making use of Knowledge Graph representation.

OntoSIDES (Palombi et al., 2019) is a Knowledge Graph that comprises a domain ontology represented in OWL and a set of statements about the entities on the SIDES platform, linking them to the ontology classes and properties. The *OntoSIDES* Knowledge Graph was automatically generated from the relational database of the *SIDES* platform, in order to populate the developed ontology.

Since its first release, OntoSIDES has been periodically updated to add data about the newest students, questions and interactions, to fix errors in the automatically extracted and converted data, and to refine the underlying ontology. Therefore, there exist several versions of OntoSIDES. In this thesis, we will be using OntoSIDES version 8[2].

The version of the *OntoSIDES ontology* employed in this work contains 79 classes and 45 properties, mainly describing universities, users (students, professors, and administrative staff), tests, questions and answers. The ontology is organized in a hierarchy of classes, whose root is class `sides:sides_entity`. Among its sub-classes, the most relevant for the description of the learning environment are the following ones:

Content : all the educational resources available in SIDES are instances of sub-classes of `sides:content`. Among these, the most interesting class is `sides:question` which is further specialized into 4 sub-classes describing the possible types of questions: `sides:QUA`, for multiple-choice questions with a single correct answer, `sides:QMA` for multiple-choice questions with multiple correct answers, `sides:TCS` for script concordance test and `sides:QSOA` for questions with short open answer. Among the sub-classes of `sides:content`, there is also

[1] Système Intelligent d'Enseignement en Santé. http://side-sante.org
[2] http://ontosides8.ontosides.network/sparql

sides:test to type tests taken by the students, sides:proposal_of_answer to type the possible choices to a multiple-choice question, sides:answer to type the answers given by students, and sides:media, further specialized into sides:picture and sides:video, to type multimedia content possibly present in questions.

Person : the class sides:person gathers all the human resources involved into the learning process. It has 3 sub-classes for students (sides:student), teachers (sides:teacher) and administrative staff sides:administrative_agent.

Referential entity : the class sides:referential_entity is used to type well-known and commonly established entities in medical education, specialized by the sub-class of medical specialties sides:speciality, those of learning objectives and sub-objectives (respectively sides:ECN_learning_objective and sides:ECN_learning_sub_objective), and cross-specialty topics sides:cross_knowledge_entity.

Institute : the class sides:institute is used for all the educational institutes and is specialized in two sub-classes: sides:university and sides:training_institute.

Figures 3.1 to 3.3 depict how instances of these entities are related to each other and show the most relevant properties in the ontology. Figure 3.1 depicts the RDF graph representing an instance of a question (q666472) with multiple possible answers (QMA). A question is linked to the possible options of answers by property has_for_proposal_of_answer, to the related medical specialty (in this case infectious_disease) through is_linked_to_medical_speciality, and to the topics treated by the question (ECN_learning_objective) through is_linked_to_ECN_referential_entity. Other useful nodes further describe questions: has_for_textual_content gives the text of a question, and has_multimedia relates a question to available multimedia content (in this case a picture). It is worth pointing out that questions are normally organized in evaluations that group sets of questions related to similar topics or concerning the same clinical case. Moreover, the question described in Figure 3.1 is only a simplified example, as the complete description of a question can involve more predicates and objects. In particular, questions are very often linked to multiple specialties and learning objectives, they present more than two answer options and can include multiple multimedia contents. Additionally, they are often related to the user and university who created the question.

Figure 3.2 shows the RDF description of a student in OntoSIDES. The representation of a student is much more simple than the one of a question. The known personal information is limited to the previous and current registrations (enrolment) of the student in the French medical academic system. The enrollment description in Figure 3.2 concerns the registration of student stu27880 (property correspond_to_student) to the

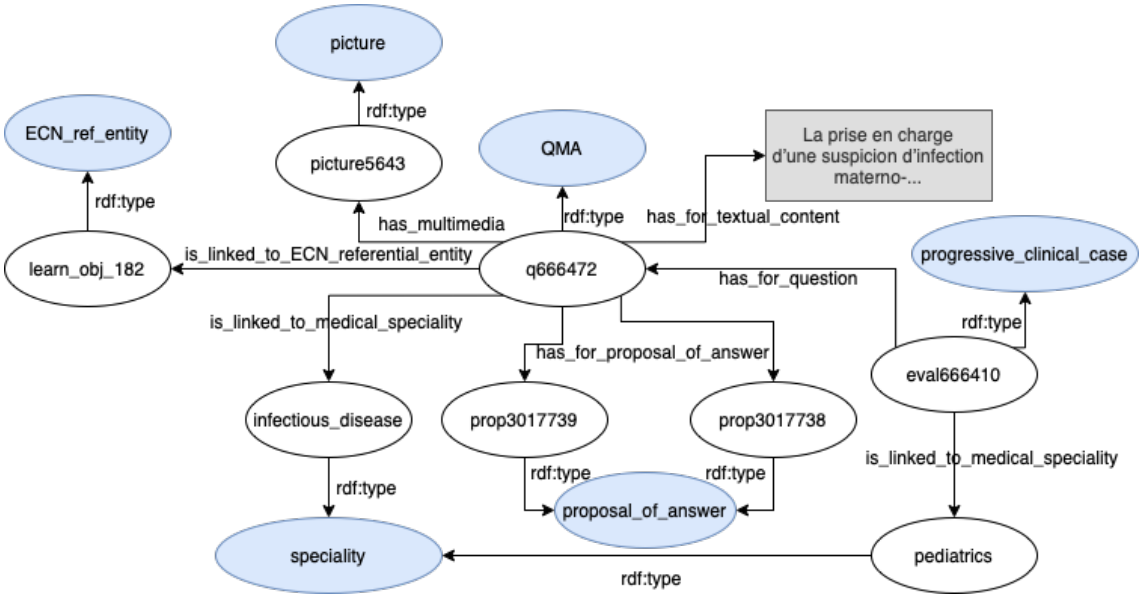


Figure 3.1: RDF graph describing an instance of a question in OntoSIDES. Blue bubbles are classes, white bubbles are class instances and rectangles are literal values.

sixth year (training `med_A6`) of Université de Toulouse (`university_toulouse`). For each registration of any student to any training (or year of study) in any university, a new instance of `enrolment` is created.

Finally, Figure 3.3 depicts the RDF graph for an answer given by a student. For each attempt of a student to a question, an instance of class `answer` is created. This answer is directly linked to the student through property `done_by` and to the question through `correspond_to_question`. An answer is linked to multiple instances of `action_to_answer`, each one representing the action of selecting a single `proposal_of_answer` for the question. The answer described in Figure 3.3 is an attempt of student `stu27880` to answer question `q666472`. While answering, the student has selected (`has_wrongly_ticked`) the wrong option `sides:prop3017739`. The instances of `sides:action_to_answer` are used to compute the number of misticked and non-ticked proposals, indicated respectively as objects of properties `has_for_number_of_wrong_tick` and `has_for_number_of_missed_right_tick`, and then the level of correctness of the given answer, object of property `has_for_result`. Lastly, the information about the timestamp of the answer is given as the object of property `has_for_timestamp`.

OntoSIDES version 8 includes the description of 569,762,878 answers to 1,797,180 questions related to 31 medical specialties and given by 173,533 students. In total the Knowledge Graph contains more than 9.2 billion triples.

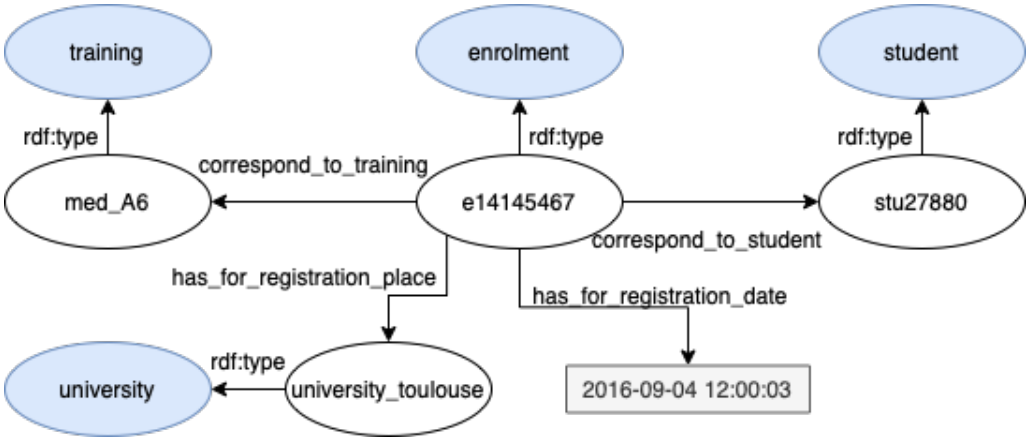


Figure 3.2: RDF graph describing an instance of a student in OntoSIDES. Blue bubbles are classes, white bubbles are class instances and rectangles are literal values.

3.2 ASSISTments

ASSISTments [3] is an online platform that supports math learning in primary and middle schools, by providing feedback assistance to students and assessment data to teachers. The research work carried out to improve the platform is the subject of numerous scientific publications and, therefore, datasets extracted from the learning logs of ASSISTments users are often anonymized and made publicly available[4]. Given the scarce availability of public datasets describing educational data, the ASSISTments data became a fundamental resource for the AIED community and the *de facto* benchmarking datasets for any new Knowledge Tracing approach.

The first version of the ASSISTments was published in 2009 (Feng et al., 2009) and included two separated datasets, skill-builder and non-skill-builder which differ for the set of problems included in the logs. More precisely, the skill-builder dataset deals with problems used for skill-mastering tests in which problems involving a same skill are presented to learners till they achieve to get three correct answers in a row. After that, the involved skill is considered to be mastered and no problem requiring such skill is further presented to the learners. Further versions of ASSISTments data have been published in 2013, 2015 and 2017.

To be able to compare our results with state-of-the-art Knowledge Tracing approaches, we used, for the experiments reported in this thesis, the ASSISTments 2009-2010 skill-builder dataset, as it is the most widely employed in the literature for testing novel Knowledge Tracing models. This dataset contains the log of the users as tabular data including information about students' answers, problems, skills, tests, classes and teachers. Each record in the dataset corresponds to an attempt of a user to a problem and it is characterized by several attributes reporting information about:

[3] https://new.assistments.org
[4] https://sites.google.com/site/assistmentsdata/datasets

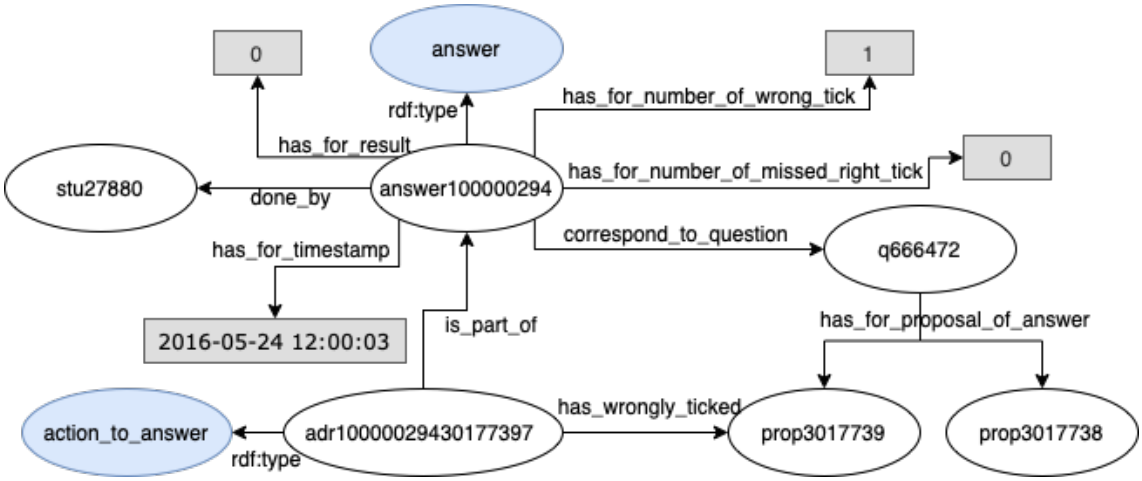


Figure 3.3: RDF graph describing an instance of an answer in OntoSIDES. Blue bubbles are classes, white bubbles are class instances and rectangles are literal values.

students: id of the user who answered the problem;

problems: including the id of the problem, its type, which can be multiple-choice, fill-in or algebra problem, names and identifiers of the skills required for the resolution of the problem;

answers: including the result of the answer, i.e. correct or incorrect, the response time, the number of hints used to solve the problem, and the text of the answer (for algebra and fill-in problems) or the id of the selected option (for multiple-choice problems);

previous students' history: number of previous students' attempts to problems involving the same skill;

assignments: including assignment's id, ids of the teacher who gave the assignment and of the school and the class to which the assignment was given.

This dataset contains, in total, 346.860 attempts to 26.688 problems made by 4.217 students. Additionally, there are 123 different skills required for the resolution of the problems, a total of 3.521 assignments given by 153 teachers to 250 classes in 75 different schools. Table 3.1 shows a comparison of the two datasets in terms of number of involved entities. Please note that, for OntoSIDES, skills correspond to learning objectives and sub-objectives.

To be used in the present work, the ASSISTments tabular data need to be modeled as a RDF Knowledge Graph. The design of an optimal and complete RDF graph describing the whole ASSISTments learning environment requires a deep knowledge of all the elements involved in the platform and a comprehensive understanding of their interactions, which can not be easily gained through the public available resources. Therefore, we will limit our RDF model of the ASSISTments environment to the knowledge useful for solving

Table 3.1: Statistics of the datasets.

	SIDES	**ASSISTments**
# answers	569.762.878	346.860
# students	173.533	4.217
# questions	1.797.180	26.688
# skills	1.283	123

the Knowledge Tracing task, and we will optimize such model to improve the quality of the results with respect to employed Knowledge Tracing resolution approach. A Knowledge Graph partially representing the ASSISTments data is presented and described in Chapter 8.

PART II

Contributions

CHAPTER 4

A Knowledge Graph Enhanced Learner Model to Predict Outcomes to Questions

This chapter presents our fist attempt to conjugate Knowledge Graphs and Deep Learning for solving the Knowledge Tracing task. In this contribution, we focus on the SIDES use-case and we propose a model to predict the students' outcomes to medical questions, by exploiting the structured and semantically annotated knowledge stored by the OntoSIDES Knowledge Graph. We start by extending the well-know deep Knowledge Tracing model DeepFM by injecting into such model knowledge extracted from the OntoSIDES Knowledge Graph. The novelty of the proposed approach lies into the addition, as input features, of Graph Embeddings representing interesting nodes of the OntoSIDES Knowledge Graph. The analysis carried out to identify the most relevant features for this model shows that Graph Embeddings of certain nodes bear meaningful information that can help solve the Knowledge Tracing task. This chapter is based on the work published at the International Conference on Knowledge Engineering and Knowledge Management (EKAW2020) (Ettorre et al., 2020).

4.1 Introduction

As explained in Chapters 1 and 2, the limitations of Deep Learning approaches for Knowledge Tracing call for the necessity to seek new research directions exploiting other forms of Artificial Intelligence. In the context of this thesis, we explore the use of Knowledge Graphs to develop better performing, more comprehensive, reusable and explainable models for Knowledge Tracing. To this end, we start tackling the first research question presented in Section 1.2:

RQ1 : How can we exploit Knowledge Representation for Knowledge Tracing purposes?

Since a Knowledge Graph representing the learning environment is already available in the context of the SIDES project, we decided to focus on this use-case as a first testbed for our hybrid AI Knowledge Tracing models. Therefore, our goal, in this chapter, is to develop a model for the prediction of students' outcomes to questions on the SIDES platform, which exploits the existing OntoSIDES Knowledge Graph.

More precisely, we seek an answer to the following use-case specific research questions:

- How to model students' learning on the *SIDES* platform to predict their outcomes to medical questions?
- Which set of features should be extracted from the *OntoSIDES* Knowledge Graph and considered for learning the student model?
- Can taking into account the Knowledge Graph structure of *OntoSIDES* improve the performance of the prediction of students' answers to questions?

To answer these questions, we present (1) our model to predict the outcome of students' answers to questions, and (2) an evaluation of our model focused on the pediatrics and cardiovascular specialties.

After thoroughly reviewing the state-of-the-art of Knowledge Tracing techniques (see Section 2.1), we decided to rely on a well-known deep learning model for Knowledge Tracing: Deep Knowledge Tracing Machines (Vie, 2018). This choice is motivated by two factors: *(i)* the high performance that this model shows in the resolution of the Knowledge Tracing task and *(ii)* the simplicity of extending the basic model with additional customized input features, which allow us to easily inject and exploit the knowledge described in the *OntoSIDES* Knowledge Graph. Through experimentation and evaluation, we validated a new model that makes the most accurate predictions by considering these features as input of a Deep Knowledge Tracing Machine.

The remainder of this chapter is organized as follows. The features extracted or computed from the *OntoSIDES* Knowledge Graph to model students' learning are detailed in Section 4.2. In Section 4.3, we present the experiments performed in order to define our model, and we analyze the results of these experiments. Finally, conclusions are presented in Section 4.4.

4.2 Features Selected or Computed from OntoSIDES to Learn a Student Model

Based on the *OntoSIDES* Knowledge Graph, whose structure and content have been previously described in Chapter 3, our aim is to predict the outcome of a student to a question, that is, the value related to an instance of class `sides:answer` by property `sides:has_for_result`, which is equal to 1 if the student answered the question correctly, and 0 otherwise. Therefore this amounts to a binary classification.

In this section, we describe the candidate features that we selected or computed from the *OntoSIDES* Knowledge Graph to build a student model. We hypothesize that these features may improve the quality of the binary classification carried out by the algorithm to predict a student's outcome to a question. In section 4.3.2, we draw some conclusions with respect to this hypothesis based on the results of our experiments.

4.2.1 Basic Features

A first set of basic (or raw) features concerns the entities that can be extracted by simply querying the *OntoSIDES* Knowledge Graph, without further processing. These features are as follows:

student: the identifier of a student who answers a question, specifically, the URI of an instance of class `sides:student` related to an instance of class `sides:answer` by property `sides:done_by`.

answer: the identifier of an answer given by a student, that is, the URI of an instance of class `sides:answer`.

question: the identifier of a question answered by a student, that is, the URI of an instance of class `sides:question`.

timestamp: the date and time when a student answered a question, that is, the value related to an instance of `sides:answer` by property `sides:has_for_timestamp`.

4.2.2 Calculated Features Conveying a Temporal Dimension

A set of additional features is computed from the above described raw features. They are meant to provide insight into students' level of knowledge over time, difficulty level of questions and number of prior attempts that a student carried out to answer a question. Together, they convey a temporal dimension to the model that is richer than the raw timestamp. These features are as follows:

wins: given a question and a student, it represents the number of times that this student has previously answered that question correctly.

fails: given a question and a student, it represents the number of times that this student has previously answered that question incorrectly.

attempts: wins + fails.

question_difficulty: for a given question, it is an estimation of its difficulty and assumes values between 0 and 1, 1 being the highest difficulty. It is computed by dividing the number of incorrect answers by the number of answers given to that question by all the students in the OntoSIDES KG.

static_student_ability: a static estimate of the student's overall ability, valued between 0 and 1, 1 being the highest ability. It is computed as the student's total number of correct answers divided by the student's total number of answers.

progressive_student_ability: this feature follows the evolution of the student's ability over time. It draws her learning curve. For each attempt, it is computed as the ratio between the number of correct answers and the number of all the answers given by the student up to that moment. At the beginning of the training, the student's ratio of correct answers is likely to be low to medium. Then, in time, this ratio increases, reflecting the growth of her level of knowledge and expertise.

4.2.3 Text Embeddings of Questions

We hypothesize that the questions' text may provide valuable information to predict the answer of a student to a question. To test this hypothesis, we queried the OntoSIDES Knowledge Graph to extract the text of the questions, i.e. the value of the property `sides:has_for_textual_content`, and we computed their vector representation by using the state-of-the-art word embedding algorithm *fastText* (Bojanowski et al., 2017). We used the *flair framework* (Akbik et al., 2018) implementation which provides embeddings pre-trained with the French chapter of Wikipedia. Applying this approach to the text of each question yields vectors of 300 dimensions. Later on, we refer to this set of vectors as **questions_temb**.

4.2.4 Knowledge Graph Embeddings of Questions, Answers, and Users

Lastly, we hypothesize that the *OntoSIDES* graph topology may convey valuable knowledge to predict the answer of a student to a question. To test this hypothesis, we used the state-of-the-art *node2vec* algorithm (Grover and Leskovec, 2016) to construct vector representations of the Knowledge Graph nodes. To do this, we used the SNAP project implementation (Leskovec and Sosič, 2016)[1] of *node2vec* to extract vector representations of dimension 100 for each of the nodes in our training dataset (described in Section 4.3).

[1] `http://snap.stanford.edu/index.html`

Among these embeddings, we selected and kept only the ones corresponding to nodes of type answer, question and student. In the following, we refer to these vectors as **answers_gemb**, **questions_gemb** and **students_gemb** respectively.

4.3 Empirical Determination of a Learner Model

This section describes a comparative evaluation of several student models, that we carried out to determine which of them produces the best prediction of the students' answers to questions. Each model relies on a specific set of features selected among those described in Section 4.2.

4.3.1 Experimental Settings

4.3.1.1 Training Data.

To evaluate the prediction performance of the proposed models we trained and tested them on two different sub-graphs of the OntoSIDES Knowledge Graph. We decided to apply our approach for modeling separately the training of students in two medical specialties: pediatrics and cardiovascular. We chose to model each specialty independently because it is reasonable to assume that the preparation of a student on a given specialty does not influence their performance in any other specialty, as the involved topics are generally very different. For example, a student can be very good in pediatrics and, at the same time, have very poor performance in cardiology. We selected pediatrics and cardiovascular as medical specialties because they are the ones with the highest number of recorded students' answers, having respectively 10.474.170 and 8.187.346 answers.

Nevertheless, the size of the retrieved subgraphs would be too large to allow the training of the proposed models within a reasonable amount of time. Therefore, we decided to train and test our approach on a subset of answers of each specialty including the most recent attempts made by the most active students, i.e. students who answered at least 100 questions in the selected medical specialty. Finally, we obtained the two datasets described in Table 4.1.

	Pediatrics	**Cardiovascular**
Number of answers	73.080	72.804
Number of questions	669	679
Number of students	1.136	1.034
First answer on	17-03-2019	27-01-2019
Last answer on	23-03-2020	22-03-2020
Percentage of correct answers	57,4%	56,9%
Percentage of incorrect answers	42,6%	43,1%

Table 4.1: Datasets characteristics

4.3.1.2 Extracted Subgraph.

To be able to compute the Graph Embeddings corresponding to the nodes of answers, students and questions, we need to extract from OntoSIDES two subgraphs describing the selected pediatrics and the cardiovascular subsets. To do so, for each answer in the dataset, we extracted its neighborhood, using the SPARQL query shown in Listing 4.1.

```
PREFIX sides: <http://www.side-sante.fr/sides#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT {
   sides:answer1234 ?answerP ?answerO .
   ?answerO ?answerOP ?answerOO .
   ?answerOO ?answerOOP ?answerOOO .
   ?answerOOO ?answerOOOP ?answerOOOO .
   ?s ?p sides:answer1234 .
}
WHERE {
   sides:answer1234 ?answerP ?answerO .
   ?answerO ?answerOP ?answerOO .
   ?answerOO ?answerOOP ?answerOOO .
   ?answerOOO ?answerOOOP ?answerOOOO .
   OPTIONAL {?s ?p sides:answer1234} .
   filter(!isLiteral(?answerOOOO))
}
```

Listing 4.1: Query to retrieve the subgraph describing a single answer in OntoSIDES

Then, we merged all the extracted answer subgraphs into a single Knowledge Graph describing the whole dataset, including nodes representing students, questions, specialties, learning objectives and universities. As it can be seen from Listing 4.1, the retrieved Knowledge Graph does not contain any literal.

4.3.1.3 Candidate Models.

Relying on the benchmarking approach presented in (Vie, 2018), we defined 14 different models by combining the features described in section 4.2, in order to comparatively evaluate them and determine which one allows the classification algorithms to obtain the best prediction scores.

Each model is identified by a label whose letters each denotes a feature: **s**: students identifiers; **q**: questions identifiers; **a**: number of attempts; **w**: number of wins; **f**: number of fails; **d**: question_difficulty; **b**: static_student_ability; **p**: progressive_student_ability; **T**: questions_temb; **Q**: questions_gemb; **S**: students_gemb; **R**: answers_gemb. With this notation, the candidate models are as follows.

The first three models correspond to state-of-the-art models and will serve as a baseline for richer models:

sq is equivalent to the *1PL IRT* model (Rasch, 1960) when used with the Logistic Regression algorithm. Used with the FM algorithm configured with a “number of factors

used for pairwise interactions" greater than 0 ($d>20$), this model is equivalent to the *mIRT* model (Reckase, 1997).

sqa is inspired by the *AFM* model (Cen et al., 2006) as it takes into account the number of previous attempts but not the skills which is not among our features.

sqawf is inspired by the *AFM* and *PFA* (Pavlik et al., 2009) models as it takes into account the number of previous attempts and the distinction between correct and incorrect attempts.

Additionally, we consider the following models that test other possible features combinations, notably involving text and graph embeddings: **sqawfd**, **sqawfb**, **sqawfdbp**, **sqawfdbpT**, **sqawfdbpR**, **sqawfdbpQ**, **sqawfdbpS**, **sqawfdbpRQ**, **sqawfdbpRS**, **sqawfdbpQS**, and **sqawfdbpRQS**.

4.3.1.4 Classification Algorithm.

For the computation of the **DeepFM** classification algorithm, we used the DeepCTR[2] Python library (version 0.6.0). The results reported hereafter are obtained using 2 hidden layers with 256 units each, L2 regularizers at 0.0001, ReLu as activation function for the neural network and Adam optimizer for the learning rate.

4.3.1.5 Hardware Setup.

We used a *Asus ESC8000G4* GPU node equipped with 8 *GTX 1080 Ti* GPU cards, 2 *Xeon SP Gold 5115* CPUs and 256 GB of RAM.

4.3.1.6 Temporality-Aware Cross-Validation.

The performance of each model was evaluated by means of the *student-based* 5-fold cross-validation technique, in order to take into account the temporal dimension of the knowledge in the graph. Specifically, the list of students included in our dataset is split into five folds; four of them are directly used as training data while the remaining one is split again in two parts following the chronological order of answers: the first half is included into the training data while the last half is used as test set. The rationale behind this splitting method is as follows: for each fold, we train the model using the complete learning path of four fifths of the students, thus learning the entire trend of the students' knowledge acquisition, and learning information about all the questions. Then, using as training data the partial learning traces of the remaining students ensures that the training involves all the students. But by testing on their latest answers, we approach the real use case, in which we want to forecast future answers based on the training history of the student.

[2] https://github.com/shenweichen/DeepCTR

4.3.1.7 Evaluation Metrics.

We evaluate the average of the results obtained on each fold in terms of Accuracy (ACC) which measures the percentage of correct predictions out of the total number of predictions, Area Under the ROC Curve (AUC) which measures the probability of correctness of each answer, F1-scores and execution time.

4.3.2 Results and Discussion

Table 4.2 shows the results of the evaluation of each model on the pediatrics sub-graph, using the DeepFM algorithm. Columns "F1-score (pos.)" and "F1-score (neg.)" report the F1-score for the prediction of positive answers and negative answers respectively. The best results were obtained with the *sqawfdbpRQ* model (*students, questions, attempts, wins, fails, question difficulty, student abilities (static and progressive), answers_gemb, and questions_gemb*) (AUC=0.731, ACC=0.739).

Model	ACC	AUC	F1-score (neg.)	F1-score (pos.)	Time (mm:ss)
sq	0.694	0.694	0.663	0.718	00:48
sqa	0.702	0.697	0.658	0.734	00:53
sqawf	0.703	0.696	0.653	0.740	01:04
sqawfd	0.706	0.688	0.623	0.757	01:00
sqawfdb	0.711	0.698	0.641	0.758	00:50
sqawfdbp	0.713	0.704	0.657	0.753	00:58
sqawfdbpT	0.706	0.695	0.646	0.747	04:55
sqawfdbpR	0.726	0.717	0.673	0.763	02:50
sqawfdbpQ	0.712	0.699	0.644	0.757	02:13
sqawfdbpS	0.712	0.700	0.648	0.755	01:52
sqawfdbpRQ	**0.739**	**0.731**	**0.688**	**0.775**	03:05
sqawfdbpRS	0.705	0.690	0.633	0.752	02:57
sqawfdbpQS	0.706	0.690	0.629	0.753	02:51
sqawfdbpRQS	**0.738**	**0.730**	**0.687**	**0.775**	04:52

Table 4.2: Results for the pediatrics sub-graph. Models with the highest AUC are in bold.

Beyond this overall result, comparing the scores obtained with each of the models can help us point out the contribution (positive, neutral or negative) of some of the features to the predictions:

question difficulty, student abilities: When comparing the results obtained with models *sqawf* and *sqawfdbp*, we notice that adding the features *question_difficulty*, *static_student_ability* and *progressive_student_ability*, increases both the ACC and AUC by approximately 1%.

questions_temb: By comparing models *sqawfdbp* and *sqawfdbpT*, we notice that the results are slightly worse when including feature *questions_temb*: ACC and AUC both decrease by almost 1%. This result is somehow counter-intuitive and may be related to the specificity and variety of medical vocabulary. To investigate further, additional experiments shall test the impact on word embeddings of techniques such as negative sampling, sub-sampling of common words or pre-processing to rewrite common medical expressions into single tokens. Nevertheless, the focus of our work is the exploitation of Knowledge Graphs and Graph Embeddings. Therefore, although potentially interesting, we will not further explore this direction.

Let us also underline that including this feature substantially increases the execution time of the classification algorithm.

answers_gemb: Comparing the results obtained with models *sqawfdbp* and *sqawfdbpR* shows that this feature yields an improvement of 1.3% in terms of AUC and ACC. Also, a higher F1-score of 0.673 and 0.763 was obtained for the negative and positive responses respectively. Execution time remained low at 3 minutes approximately.

questions_gemb: When comparing the results obtained with models *sqawfdbp* and *sqawfdbpQ*, we observe that this feature has almost no impact on the prediction results in terms of ACC, AUC, while the F1-score was reduced for the negative class (-1.3%) and slightly increased for the positive one. Again, this overall negligible impact can seem counter-intuitive, in particular when considering that the embeddings of the answers have a significant positive impact.

students_gemb: Similarly, feature *students_gemb* seems to have no effect on the quality of the prediction. The values of ACC, AUC and F1-score are very close to those obtained when using *questions_gemb*.

Although the contribution of feature questions_gemb may seem negligible when it is considered alone, and, for some metrics, even negative, the best performance in terms of ACC, AUC and F1-scores is obtained when this feature is joined with feature answers_gemb. Indeed, the best model *sqawfdbpRQ* presents an ACC and AUC around 73-74%, with a non-negligible increment (about 4.5%) when compared with the basic model (*sq*). This could be explained by the fact that our model captures high-degree interactions between some features, interactions that, in some cases, turn out to be much more meaningful than the single features themselves.

It is also worth pointing out the constant and consistent difference between the positive and the negative F1-score, with the first one being always 6-9% higher than the last one. The reason for this difference can probably be found in the balancing of the dataset with respect to the target variable. As shown in Table 4.1, the pediatrics subset, as well as the cardiovascular one, is slightly unbalanced, including a higher number of correct answers than incorrect ones.

Finally, we observe that, even with this small dataset, the quality of the prediction is fairly good. This suggests that, in the production environment of the SIDES platform, there shall be no need for training the algorithm on a much larger dataset in order to achieve good performance in the prediction task.

In order to validate our model and assess its flexibility with respect to the considered medical specialty, we trained and tested the DeepFM learning algorithm on the sub-graph related to the cardiovascular answers, extracted as described in Section 4.3.1.1.

Model	ACC	AUC	F1-score (neg.)	F1-score (pos.)	Time (mm:ss)
sq	0.694	0.688	0.650	0.723	00:37
sqa	0.708	0.699	0.657	0.741	00:43
sqawf	0.717	0.708	0.666	0.751	00:55
sqawfd	0.720	0.712	0.671	0.754	00:54
sqawfdb	0.722	0.714	0.672	0.756	00:53
sqawfdbp	0.727	0.718	0.677	0.762	00:52
sqawfdbpT	0.721	0.712	0.672	0.754	04:36
sqawfdbpR	0.727	0.720	0.679	0.761	02:22
sqawfdbpQ	0.722	0.715	0.675	0.756	02:05
sqawfdbpS	0.723	0.715	0.671	0.759	01:58
sqawfdbpRQ	**0.770**	**0.762**	**0.800**	**0.770**	04:29
sqawfdbpRS	0.718	0.710	0.669	0.753	02:39
sqawfdbpQS	0.717	0.702	0.651	0.758	02:40
sqawfdbpRQS	0.736	0.725	0.677	0.775	04:49

Table 4.3: Results for the cardiovascular sub-graph. Models with the highest AUC are in bold.

As it can be seen in Table 4.3, the results for this new specialty are consistent and confirm what we observed earlier for the pediatrics sub-graph. As for the previous experiments, the best model appears to be *sqawfdbpRQ*, including basic and computed features and graph embeddings of answers and questions. It produces the highest values of ACC (0.770), AUC (0.762) and F1-score (0.800 and 0.770 for positive and negative classes respectively). The new results confirm the negative impact of the *questions_temb* feature, as can be seen by comparing the models *sqawfdbp* and *sqawfdbpT*. They also confirm the importance of the interactions between answers and questions graph embeddings. Indeed, in line with the previous case, we observe that, when used alone, the feature *questions_gemb* has no impact on the accuracy and AUC of the model, while when used together with *answers_gemb*, the quality of the prediction is improved.

To sum up, our experiments show that the best student model combines a set of basic features obtained by directly querying the *OntoSIDES* Knowledge Graph– questions, attempts, wins, fails –, a set of additional features computed based on the basic ones – question difficulty, student ability (static and progressive) –, and the vector representations of the answers and questions nodes in the *OntoSIDES* Knowledge Graph.

4.4 Conclusions

In this chapter, we have presented our first step towards developing a hybrid approach for Knowledge Tracing able to exploit the expressive power of Knowledge Graphs. We evaluated and compared several models to predict the outcome of medical students' answers to questions in pediatrics and cardiovascular specialties, on the *SIDES* platform. We have identified as the best model for our task the one relying on a rich set of features including state-of-the-art features such as wins, fails, questions' difficulties and students' abilities; and the structural knowledge provided by the OntoSIDES Knowledge Graph. In particular, we have shown that considering the vector representations of answers and questions nodes had a positive impact on the prediction results: when these two features are used in conjunction, the accuracy and AUC measures of the predictions made by the DeepFM algorithm improved significantly.

CHAPTER 5

Diving into Knowledge Graph Embeddings

In this chapter, we dive deep into the analysis of Graph Embeddings and we study their behavior and their influence on the performance of our predictive model to understand when and why they are useful to improve the quality of the predictions provided by DeepFM. We investigate how changes in the embedding model, its hyper-parameters and in the content of the initial Knowledge Graph affect the final DeepFM results and we identify the best settings for the embedding process and the most important information represented in the Knowledge Graph which is exploited for the prediction.

5.1 Introduction

In the previous chapter, we saw that extending DeepFM with Graph Embeddings of answers and questions' nodes improves the quality of the predictive model, meaning that such vector representations can embed meaningful information from the Knowledge Graph. Nevertheless, to provide comprehensive answers to our initial research questions, we need to go deeper with our analysis to be able to evaluate the real impact of the introduction of Knowledge Graphs for the resolution of the Knowledge Tracing task and discover the reasons behind such an impact. More precisely, to better comprehend the behavior of our model, explain the predictions it outputs, and assess its generalization capabilities, it is essential to understand when Graph Embeddings are useful to improve the performance of our predictive model and why they can bring such an improvement. In other words, we need to identify what meaningful information they are able to capture from the Knowledge Graph and how it is encoded during the embedding process to be further exploited by DeepFM. This requires not only to analyze the content of the original Knowledge Graph to detect the relevant knowledge it contains, but also to dive deeper into the study of the Graph Embeddings to examine how changes in the embedding process affect the performance of the predictive model.

Indeed, the information that embeddings are able to capture is strongly influenced by several factors including the nature and parameters of the employed embedding model. As explained in Section 2.2.3, different embedding algorithms tend to capture different characteristics of the elements in the Knowledge Graph, e.g. TransE cannot model symmetries between relations while RotatE can, as explained in Section 2.2.3. At the same time, many models, such as node2vec, can be parametrized to personalize the learned representations.

In the following, we first use the node2vec model designed for any kind of graph, and explore a large variety of hyper-parameters. Then, we move to other embeddings models more specifically designed for Knowledge Graphs such as TransE, RESCAL, RDF2VEC, etc. The goal is to identify the settings that produce the most meaningful Graph Embeddings. Additionally, we construct and exploit several Knowledge Graphs representing the same subset of the SIDES learning environment, but differing by the information details they include, with the final goal of identifying the relevant knowledge included in the Knowledge Graph. All the experiments in this chapter are executed on the same subset of students' answers in pediatrics introduced in Chapter 4 and, when not otherwise specified, use the same cross-validation technique and hyper-parameters settings.

5.2 Tuning node2vec

In this section, we explore different hyper-parameters settings to identify the ones that generate the best performing embeddings for our predictive model. To do so, we compute embeddings using the node2vec algorithm with varying random walks length, embeddings dimension, and settings for the orientation of the graph, i.e. directed vs undirected.

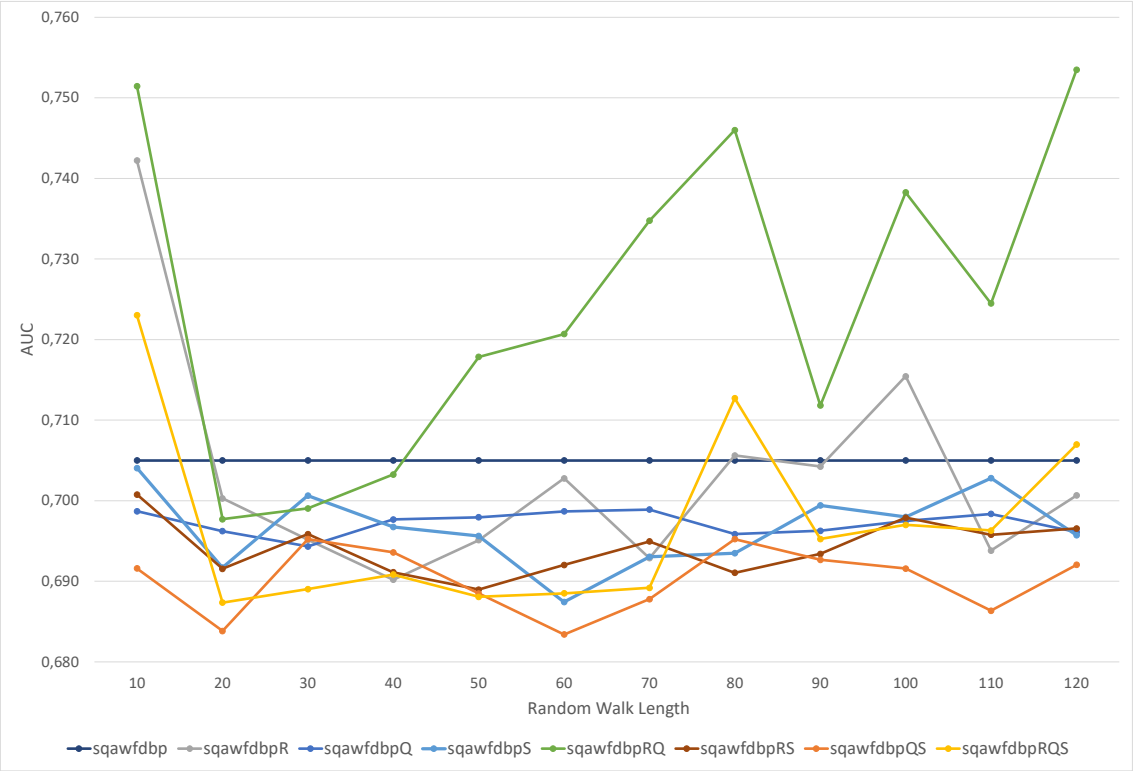


Figure 5.1: AUC of DeepFM models including, as input, Graph Embeddings computed with varying random walk length. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4.

Then, we study how our predictive model behaves with these different Graph Embeddings.

5.2.1 Random Walks Length

As explained in Section 2.2.3, node2vec is an embedding model relying on the use of random walks built through customized sampling strategies. To define the sampling strategy, several hyper-parameters need to be specified, including the length of the random walks starting in each node. We analyze the behavior of DeepFM when Graph Embeddings computed with different random walks lengths are given as input to the model. We test embeddings computed with random walks of length from 10 to 120 hops. The results, in terms of AUC, of the execution of DeepFM using as input embeddings with varying random walks length are illustrated in Fig. 5.1.

The AUC trends depicted in Figure 5.1 show that the best results are achieved by the model including answers' and questions' Graph Embeddings when the random walks length is 10 hops and 120 hops, with AUC around 0.75. In general, all the models including answers' Graph Embeddings, i.e. *sqawfdbpR*, *sqawfdbpRQ*, *sqawfdbpRQS*, present higher AUC values, outperforming the model without embeddings, i.e. *sqawfdbp*. This

confirms that, in general, the most meaningful information is borne by answers' embeddings and that the interaction of answers and questions is relevant. Surprisingly, the performance of these models drops significantly with random walk length of 20; and, while for *sqawfdbpRQ* it grows almost steadily after that to reach the highest AUC when the random walk length is 120 hops, it remains low for the other models including answers' embeddings. Additionally, it is interesting to point out that the other models including students and questions embeddings do not present important variation of AUC with the changing random walk length. Indeed, for the models *sqawfdbpS*, *sqawfdbpQ*, *sqawfdbpRS* and *sqawfdbpQS*, the AUC always fluctuates between 0.690 and 0.700, although the random walks length changes significantly. In conclusion, Figure 5.1 clearly shows that *(i)* the random walk length can have an influence on the performance of the predictive model; *(ii)* the best model is, with any random walks length, the one including questions and answers embeddings, i.e. *sqawfdbpRQ*; and *(iii)* the best performance are achieved with either very short or very long random walks (10 or 120 hops).

5.2.2 Embeddings Dimension

Another fundamental factor that impacts the quality of the embeddings is their dimension: vectors of different dimension will encode knowledge from the graph differently. In this section, we analyze the performance of our DeepFM predictive model when Graph Embeddings with varying dimension are provided in input. We repeated the experiments with embeddings of dimension 25, 50, 75, 100, 125, 150, 175, and 200. The random walks length used in the following experiments is 10. The AUC values of all the DeepFM models relying on embeddings are shown in Figure 5.2.

As for the past experiments, the best DeepFM model, for almost all the values of d, is the one including embeddings of answers and questions nodes, i.e. *sqawfdbpRQ*. As already observed in the previous case, the models including answers Graph Embeddings, i.e. *sqawfdbpR*, *sqawfdbpRQ*, *sqawfdbpRS*, *sqawfdbpRQS*, are the only ones outperforming the basic model *sqawfdbp*. Figure 5.2 shows that the best results, in terms of AUC, are achieved with embedding dimension of 25. Furthermore, it is worth to highlight that, while the AUC of models which do not include answers embeddings remains roughly stable and constant (between 0.69 and 0.70) despite the variation of the vectors dimension, the performance of the models including answers' embeddings tends to decrease with higher d. The only exception to such behavior is given by model *sqawfdbpRQ*, which keeps an AUC oscillating between 0.745 and 0.765 over the range of dimensions. In general, we can confirm that the most meaningful information is encoded in the Graph Embeddings of the answers nodes and that the interaction with the questions' nodes is important. Moreover, the almost constant AUC trends depicted in Figure 5.2 show that using a longer vector representation does not bring any improvement, suggesting that either the **relevant knowledge present in the Knowledge Graph is already captured by low-dimensional embeddings**, or **DeepFM is not able to exploit the further knowledge encoded by longer embeddings**.

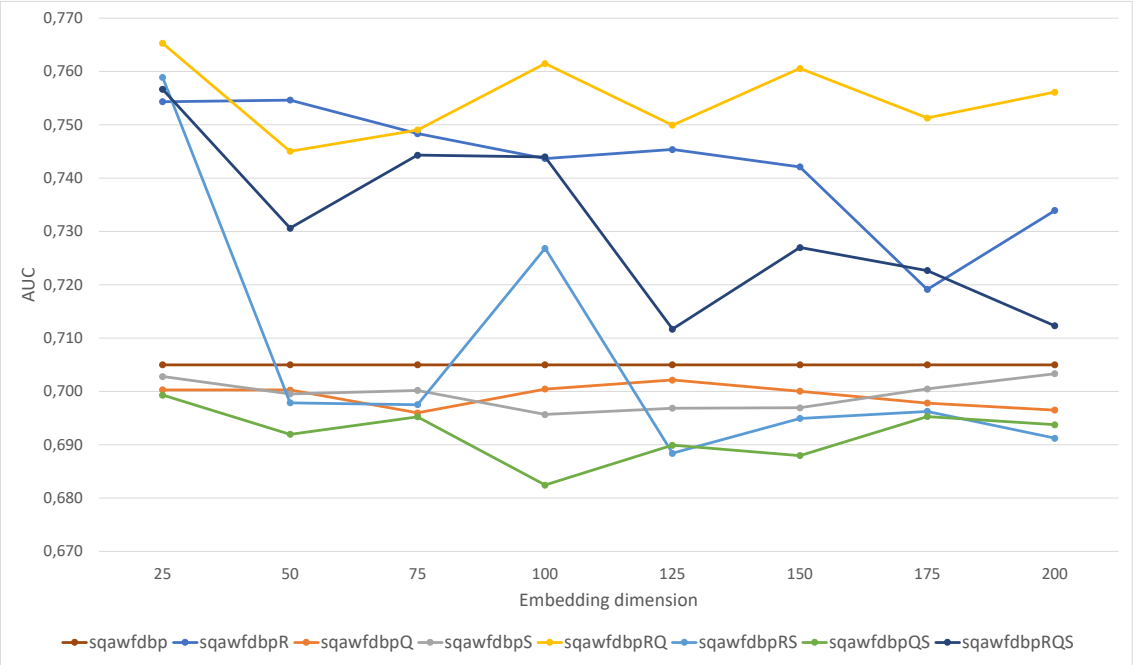


Figure 5.2: AUC of DeepFM models including, as input, Graph Embeddings with varying dimension.Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4.

5.2.3 Undirected vs. Directed Graph

All the experiments illustrated so far have been carried out using Graph Embeddings computed through node2vec considering the input graph as undirected. The reason behind this decision lies in the fact that the direction of the properties in the employed Knowledge Graph are not relevant as their choice is arbitrary, as almost every property could have also been defined in the inverse direction in a meaningful way. For example, the OntoSIDES Knowledge Graph links students to their answers by using property `sides:done_by`, whose subject and object are respectively the answer and the student, but it would have been possible as well to define a property `sides:has_given`, going from students to answers, to state that a student has given an answer.

We want now to study the behavior of our model when Graph Embeddings computed from the directed graph are used as input features. In the following, we try to predict the performance of the students relying on Graph Embeddings computed from the directed graph, where properties (and their directions) are the ones originally defined in the OntoSIDES KG. To compute the embeddings, we use vector dimension of 100 and random walk length of 10. The results are shown in Table 5.2.

Table 5.2 clearly shows that **Graph Embeddings computed from the directed graph are more meaningful for the students' performance prediction than Graph Embeddings computed from the undirected graph**. The comparison between the figures in Table 5.2 and Table 5.1 (which recalls the results in Table 4.2) shows that DeepFM using

Model	ACC	AUC	F1-score (neg.)	F1-score (pos.)
sqawfdbpR	0.726	0.717	0.673	0.763
sqawfdbpQ	0.712	0.699	0.644	0.757
sqawfdbpS	0.712	0.700	0.648	0.755
sqawfdbpRQ	**0.739**	**0.731**	**0.688**	**0.775**
sqawfdbpRS	0.705	0.690	0.633	0.752
sqawfdbpQS	0.706	0.690	0.629	0.753
sqawfdbpRQS	**0.738**	**0.730**	**0.687**	**0.775**

Table 5.1: Results of DeepFM when using Graph Embeddings computed from the undirected graph. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4. The values in bold highlight the best models.

Model	ACC	AUC	F1-score (neg.)	F1-score (pos.)
sqawfdbpR	**0.867**	**0.858**	**0.837**	**0.887**
sqawfdbpQ	0.710	0.701	0.655	0.749
sqawfdbpS	0.710	0.698	0.645	0.754
sqawfdbpRQ	0.864	0.851	0.828	0.887
sqawfdbpRS	0.864	0.855	0.833	0.885
sqawfdbpQS	0.705	0.691	0.635	0.750
sqawfdbpRQS	**0.866**	**0.859**	**0.839**	**0.886**

Table 5.2: Results of DeepFM when using Graph Embeddings computed from the directed graph. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4. The values in bold highlight the best models.

Graph Embeddings of answers' nodes computed from the directed graph largely outperforms all the other models. Indeed, adding the Graph Embeddings of answers' nodes as input features brings an improvement of around 15% in terms of AUC and ACC compared with the model without embeddings. It is also interesting to point out that in the previous experiments (relying on the undirected graph) the best model was always the one combining answers and questions embeddings, showing that the interactions between these features were important; while with the directed graph, the answers embeddings bear all the meaningful information, as the combined models *sqawfdbpRQ* and *sqawfdbpRQS* bring no improvement over *sqawfdbpR*. In conclusion, we discovered that the Graph Embeddings of answers nodes computed from the directed graph bear more meaningful information than the ones computed from the undirected one. Nevertheless, **we are not able yet to identify the reason behind this behavior or to understand how the information encoded in the two cases differs and why**.

5.3 Further Exploring Graph Embedding Models

We found out that the node2vec embedding model used in the directed or undirected setting generates Graph Embeddings which produce very different results once fed into our predictive model. Indeed, the results in Table 5.2 show that Graph Embeddings which exploit properties' directions are better suited as input for predicting students' outcomes to questions through DeepFM. It is reasonable to think that, not only the direction of the properties, but also the semantics stored in the OntoSIDES KG should be taken into account to build more meaningful embeddings. To test this hypothesis, we analyzed the behavior of DeepFM when relying on Graph Embeddings computed through KG-specific embedding models, which are inherently able to account for semantic information. We tested most of the KGE models introduced in Section 2.2.3: RDF2Vec, TransE, TransR, ComplEx, RESCAL and DistMult. For the computation of embedding through RDF2vec we used Python library pyRDF2vec [1], while for all the other KGE models, we relied on the use of the AWS DGL-KGE library (Zheng et al., 2020). Please note that all the employed Graph Embeddings have dimension 100.

Tables 5.3 to 5.5 show the performance, in terms of AUC, of all the DeepFM models relying on embeddings computed through the different models.

	node2vec (und)	**node2vec (dir)**	**RDF2vec**
sqawfdbpR	**0.742**	**0.858**	0.694
sqawfdbpQ	0.699	0.701	0.695
sqawfdbpS	0.704	0.698	0.701
sqawfdbpRQ	**0.751**	**0.851**	**0.720**
sqawfdbpRS	0.701	**0.855**	0.698
sqawfdbpQS	0.692	0.691	0.690
sqawfdbpRQS	**0.723**	**0.859**	0.697

Table 5.3: AUC of DeepFM when using Graph Embeddings computed through GE model based on random walks. The results highlighted in bold are the ones outperforming the basic model without embeddings *sqawfdbp*. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4.

Table 5.3 compares the results obtained through node2vec, both with directed and undirected graph, with the ones of another embedding model based on random walks: RDF2vec. The figures clearly show that **the embeddings computed through RDF2vec do not bear any meaningful information which can be exploited for the prediction task**, as the performance achieved by DeepFM when using these embeddings is constantly lower than the ones obtained with node2vec, and, for all the models but *sqawfdbpRQ*, lower than the basic model *sqawfdbp*, which does not rely on embeddings.

Table 5.4 illustrates, instead, the AUC values achieved by DeepFM on the same prediction task when using RESCAL and its extensions DistMult and ComplEx. We can

[1] https://github.com/IBCNServices/pyRDF2Vec

	RESCAL	DistMult	ComplEx
sqawfdbpR	0.696	0.694	0.689
sqawfdbpQ	0.701	0.702	0.696
sqawfdbpS	0.699	0.695	0.703
sqawfdbpRQ	0.687	0.687	0.688
sqawfdbpRS	0.694	0.694	0.690
sqawfdbpQS	0.697	0.696	0.696
sqawfdbpRQS	0.680	0.690	0.692

Table 5.4: AUC of DeepFM when using Graph Embeddings computed through RESCAL and its extensions. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4.

notice that, as with RDF2vec, none of the tested DeepFM models including embeddings is able to outperform *sqawfdbp*. Therefore, also in this case, **no information exploitable for the prediction is captured from the Knowledge Graph through RESCAL and its extensions**.

	TransE	TransR
sqawfdbpR	**0.828**	0.697
sqawfdbpQ	0.700	0.699
sqawfdbpS	0.701	0.700
sqawfdbpRQ	**0.836**	0.682
sqawfdbpRS	0.693	0.692
sqawfdbpQS	0.692	0.697
sqawfdbpRQS	**0.823**	0.686

Table 5.5: AUC of DeepFM when using Graph Embeddings computed through TransE and its extension TransR. Experiments run on the OntoSIDES pediatrics subgraph introduced in Chapter 4. The results highlighted in bold are the ones outperforming the basic model without embeddings *sqawfdbp*.

Lastly, Table 5.5 depicts the results, in terms of AUC, obtained when using TransE and TransR Graph Embeddings. Surprisingly, we detect that **the "simpler" TransE model is able to encode meaningful knowledge from the Knowledge Graph, which cannot be equally encoded by TransE extension, TransR**. Indeed, Table 5.5 shows that DeepFM using TransE embeddings achieves AUC values comparable with the ones achieved by node2vec on the same directed graph. Also, it confirms that the most relevant information is borne by answers nodes embeddings, as the only models bringing improvement are the ones including those embeddings, i.e. *sqawfdbpR*, *sqawfdbpRQ* and *sqawfdbpRQS*.

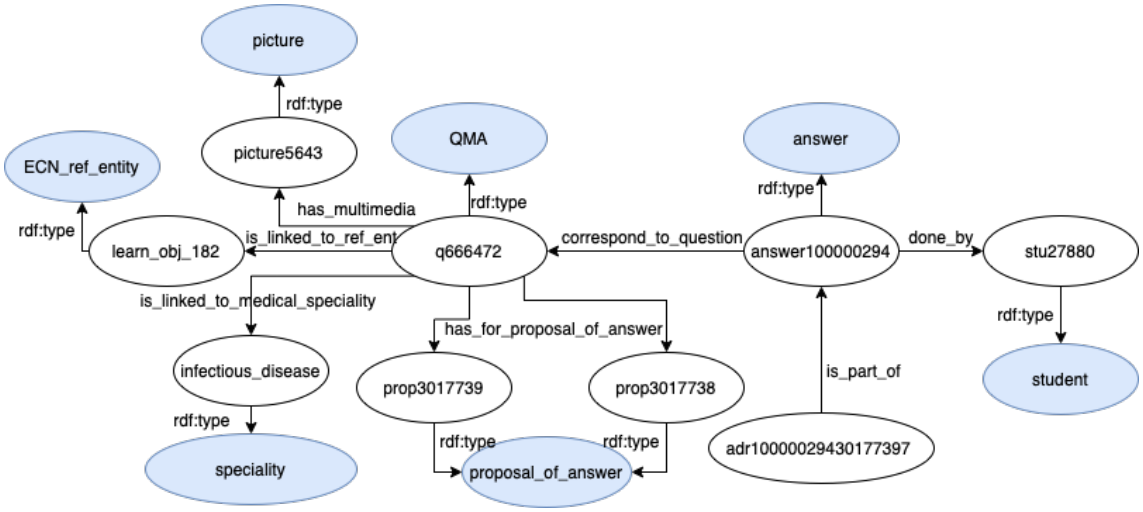


Figure 5.3: Neighborhood of each answer included in the subgraph used for the computation of the Graph Embeddings.

5.4 Identifying Meaningful Information

Until now our experiments helped us understanding which embedding models and parameters settings are the best one to capture meaningful information from the OntoSIDES Knowledge Graph, that could be efficiently exploited by DeepFM for the prediction of students' outcomes to questions. Nevertheless, we do not know yet what this meaningful information is and how to represent it in the Knowledge Graph so that it can be more easily captured by the chosen embedding model. In other words, we aim to answer the questions:

- What are the elements of the graph (nodes and edges) that influence the most the performance of DeepFM when embeddings are injected as input features?
- And how to improve the structure of the Knowledge Graph so that the characteristics of such elements can be more easily encoded in the embeddings?

To answers these questions, we executed several experiments relying on slightly different graph models representing the same pediatrics subset used so far. Starting from the initial subgraph extracted through the query illustrated in Listing 4.1, we add and remove nodes to study the impact that each entity has on the final prediction performance.

Figure 5.3 depicts the neighborhood of each answer initially extracted through the query in Listing 4.1. We can see that the available information for each answer to be predicted includes two main elements: the question to be answered and the student answering such question. Moreover, the subgraph contains details about every question, such as answers' proposals, involved topics (learning objectives and sub-objectives), related medical specialties and included multimedia content. Additionally, we can notice the presence of a third type of element in the close neighborhood of

the answer, `adr10000029430177397` in Fig. 5.3, which is an instance of class `sides:action_to_answer`, even though this information is not included in the extracted subgraph. It is worth to remind that a node of type `action_to_answer` is created each time a proposal of answer is selected by students when they answer a question. The presence of this type of entities in the extracted subgraph represents an important issue for the soundness of our model. Indeed, even though the node itself does not bear any direct knowledge about the answers' correctness, it injects in the input data knowledge about the answer which should not be known before its realization. Pragmatically, when predicting the outcome of a new attempt of a student to a question, we know the identity of the involved student and question, represented by the links from the answer to student's and question's nodes, but we have no information about the actions made by the student while answering the question, nodes of type `action_to_answer`.

To evaluate the impact of this type of nodes, our first experiment consists in removing the instances of `sides:action_to_answer` from the Knowledge Graph and verifying whether the performances of DeepFM are still improved when using answers' Graph Embeddings.

	node2vec (und)	**node2vec (dir)**
sqawfdbpR	0.694	0.694
sqawfdbpQ	0.702	0.695
sqawfdbpS	0.700	0.697
sqawfdbpRQ	0.688	0.685
sqawfdbpRS	0.687	0.688
sqawfdbpQS	0.684	0.695
sqawfdbpRQS	0.679	0.686

Table 5.6: AUC of DeepFM when using Graph Embeddings computed from the subgraph not including instances of `sides:action_to_answer`.

Table 5.6 shows the results of DeepFM, in terms of AUC, when using node2vec embeddings computed from the Knowledge Graph without the nodes representing instances of `sides:action_to_answer`. The employed embeddings are computed with dimension 100 and random walks length of 10. Surprisingly, the figures illustrated in Table 5.6 clearly show that no improvement is brought by the Graph Embeddings computed from the newly obtained Knowledge Graph. This result leads us to conclude that **the information exploited by DeepFM when including answers' and questions' Graph Embeddings as input features concerns some characteristics of the nodes of type `action_to_answer`**. Nevertheless, as explained beforehand, these nodes bear no direct information about the correctness of the answer to be predicted. As we can see from Fig. 5.3, the employed subgraph does not include information about which proposal of answer was selected during the student's action or whether such selection is correct or not. Nevertheless, there is a meaningful piece of information that is introduced in the Knowledge Graph when instances of *action_to_answer* are included: the number of pro-

posal of answers selected by the student when answering the question which corresponds to the number of instances of `action_to_answer` linked to an answer. Therefore, we hypothesize that **introducing Graph Embeddings, and in particular answers' Graph Embeddings, as input of DeepFM improves the quality of the prediction because such embeddings, when computed through node2vec and TransE models, are able to capture from the Knowledge Graph and encode the number of selected proposals for each answer**. This surely represents a meaningful information for the prediction as it allows the classification model to easily identify most of wrong answers, i.e. the ones for which the number of selected proposals is too low or too high.

Further experiments were run using different graph models without instances of `action_to_answer` but including additional information, e.g. students' year of study and university or organization of questions in tests, or modifying the structure of the Knowledge Graph, e.g. adding or removing classes and properties, but in none of them Graph Embeddings resulted to be useful for improving the performance of the predictive model. Surprisingly, we found out that not even the addition or removal of the links between questions and their related topics had any relevant impact on the quality of the final predictive model.

Yet, it is worth pointing out that, through these experiments, we discovered that Graph Embeddings are able to encode meaningful information from the Knowledge Graph, which, in the case of OntoSIDES, is represented by the instances of `action_to_answer`. Nevertheless, this result is valid only for the OntoSIDES use case, and it is possible that, in other educational contexts and with different Knowledge Graphs, relevant knowledge could be borne by diverse elements in the graph and, therefore, it could be captured through embeddings to be possibly exploited by the DeepFM model.

5.5 Conclusions

In this chapter, we went deeper into the analysis of the behavior of our predictive model when relying on different Graph Embeddings. We studied the influence of different node2vec embedding parameters, tested different embedding models and verified the behavior of Graph Embeddings computed from different Knowledge Graphs to try to identify which factors affected the most the predictions made by our DeepFM model through the embeddings and how. Our ultimate goal was to find an explanation for the behavior of our predictive model when using Graph Embeddings and to provide an answer to the following question: **under what conditions and in which context are Graph Embeddings useful to predict students' outcomes?**

Through our experimentation, we found out that the most meaningful embeddings for the prediction of students' outcomes are the ones of the answers' nodes, which have a more important effect when combined with the questions' Graph Embeddings. Afterwards, **we identified as best embeddings for our task the ones computed through the node2vec model considering the input graph as directed**. We discovered that, when the

input graph is undirected, the best embeddings are obtained with either very short or very long random walks, and that even short Graph Embeddings, with $d = 25$, are able to encode meaningful information from the Knowledge Graph. Additionally, we benchmarked several Knowledge Graph Embedding models and determined that **the only one able to capture exploitable knowledge for the students' outcomes prediction is TransE**.

Finally, we found out that **the main factor of improvement in prediction quality was the presence of the nodes representing instances of `action_to_answer`**. We hypothesized that this is mainly due to the unexpected available knowledge about the number of selected proposals for each answer which can be inferred through the answers' Graph Embeddings. Yet, this hypothesis needs to be confirmed through a more thorough analysis of the answers' Graph Embeddings.

This analysis surely shed a light on the important divergences among Graph Embeddings computed through different models and with diverse parameters and it exposed a significant issue with the input data fed to the DeepFM predictive model. At the same time, it highlighted several open issues which need to be tackled to be able to create a highly-performant model for Knowledge Tracing based on Knowledge Graphs, whose predictions can be easily interpreted.

Firstly, even though we are now aware of the importance of the embedding model and its parameters for capturing meaningful information from the Knowledge Graph, we do not know yet how these embeddings models differ and what information they are able to encode. In other words, why Graph Embeddings computed from the directed graph perform better than the ones computed from the undirected graph? Or why node2vec embeddings outperform all the other Knowledge Graph Embeddings? To answer these questions, **we need a method to decode the knowledge captured by Graph Embeddings or, at least, to get some insights into it.** Such a technique could also allow us to verify the hypothesis on the encoded number of selected options by answers' embeddings.

Nevertheless, being able to interpret the information encoded by the Graph Embeddings could not be enough to get meaningful explanations for our predictions. Indeed, even if we were able to perfectly decode the content of the Graph Embeddings, to fully explain our predictions we would need to understand whether this information is understood and how it is exploited by our deep predictive model, DeepFM. Moreover, the approach proposed in Chapter 4 makes use of only a part of the knowledge available in the OntoSIDES Knowledge Graph, as exclusively answers, questions and students Graph Embeddings are employed as input features. Yet, through our experiments we saw that interactions between different embeddings can be very meaningful, e.g. answers and questions' nodes; and that other types of nodes can bear important information, e.g. instances of `action_to_answer`. **A research direction to solve these issues could be to develop an approach which relies solely and entirely on the Knowledge Graph and its embeddings to provide more accurate predictions and explanations.**

CHAPTER 6

Stunning Doodle: a Tool for Joint Visualization and Analysis of Knowledge Graphs and Graph Embeddings

In this chapter, we introduce a method to gain insights into the meaning of Knowledge Graph Embeddings, which exploits the joint visual analysis of Knowledge Graphs and their Graph Embeddings. We present Stunning Doodle*, a tool that enriches the classical visualization of Knowledge Graphs with additional information meant to enable the visual analysis and comprehension of Graph Embeddings. The idea is to help the user figure out the logical connection between (1) the information captured by the Graph Embeddings and (2) the structure and semantics of the Knowledge Graph from which they are generated. We describe the main functionalities of* Stunning Doodle *and we detail how they have been used in our scenario to interpret the information captured by the Graph Embeddings used for the prediction of students' outcomes and, to some extent, explain their different behavior and performance. This chapter is based on the work published at the Extended Semantic Web Conference (ESWC2022) (Ettorre et al., 2022a) and at the French conference on Knowledge Engineering, Ingénierie des Connaissances (IC 2022) (Ettorre et al., 2022b).*

6.1 Introduction

The previous chapter highlighted the potential impact that Graph Embeddings can have in the prediction of the students' performance, but it also underlined the importance of using the right embedding model and tuning it correctly. We saw how different embedding models and settings can generate very diverse low dimensional representations for the same elements in the graph. Nevertheless, we are not aware yet of how and why these vectors deeply differ from each other and which kind of information they represent.

Understanding the knowledge captured and encoded by Knowledge Graph Embeddings is not only essential in our use case for the development of an explainable Knowledge Graph based model for Knowledge Tracing, but it is also one of the most important open challenges for the whole AI community. As explained in Chapter 1, the adoption of Knowledge Graphs in multiple domains has increased steadily during the last decade and one of the reasons for this success is the possibility to easily employ them as input features for several Machine Learning methods thanks to the graph embedding process. In this context, the need arises for users to be able to analyze and interpret the knowledge encoded in the Graph Embeddings.

Nevertheless, the comprehension of the information captured by GEs is a very challenging process. Indeed, GEs are computed using "black-box" Machine Learning (ML) techniques that translate each element in the graph into a low-dimensional vector. Even though the algorithmic process to compute embeddings is well understood, a relation between the characteristics and role of an element in the graph and its vector representation in the embedding space cannot be established with certainty. In other words, multiple questions cannot be answered easily, such as:

- What do my embeddings represent?
- How are they related to the structure and semantics of my KG?
- How can I improve my embeddings to be better suited to my downstream application?

Recently, several research efforts have been made in this direction to start making sense of the information captured by GEs. Some approaches propose explainable models for computing GEs (Kazemi and Poole, 2018) while others implement explanation strategies for specific embedding models (Ying et al., 2019).

In this chapter, we aim to tackle this issue from a different point of view. We think that the information borne by GEs could be explored and unveiled through the use of visualization techniques that would favor the discovery of the logical connection between the graph and its embeddings. Our goal is therefore to provide a visualization tool supporting the analysis and decoding of the information captured by KGEs by unveiling the relationship between, on one hand the structure and semantics of the KG, and on the other hand the KGEs generated from it.

To achieve this goal, we present *Stunning Doodle*, a tool designed for the visualization of RDF-based KGs and GEs. *Stunning Doodle* first provides a visualization of the graph to

be analyzed, offering a rich overview of both the structure and the semantics of the data. We believe that this visualization should allow users to gain a general and immediate understanding of the displayed KG while presenting a complete and detailed view of each of its components. More interestingly, this visualization is enriched by connecting the nodes in the graph with the corresponding GEs to be analyzed. It enables to select a node in a KG and visualize its neighborhood in the embedding space, and conversely to pick any of its neighbors in the embedding space and visualize their links in the KG. We argue that the joint visualization of GEs and the KG from which they are generated, with its structure and semantics, will help users, may they be non-expert users, RDF experts or Machine Learning experts, to gain interesting insights into the information captured during the embedding process.

The remainder of this chapter is organized as follows. In Section 6.2 we present our KGEs analysis tool, while in Section 6.3 we demonstrate how our tool has been used and useful to get insights into the information encoded by Graph Embeddings in the OntoSIDES use case. In Section 6.4 we review related work. Finally, conclusions and future work are discussed in Section 6.5.

6.2 Stunning Doodle

Stunning Doodle has been developed to fill the gap in the field of visual analysis of KGEs. Its main goal is to provide users with an advanced visualization of RDF graphs, enriched with information extracted from the GEs generated from those graphs. To achieve this goal, *Stunning Doodle* offers two main functionalities: (1) the visualization and navigation of RDF graphs and, (2) for each node, the enrichment of such visualization with the addition of its neighborhood in the embedding space, w.r.t. a chosen similarity measure. Additionally, *Stunning Doodle* presents semantic-based filtering and customization functionalities for nodes and links which make the visualization clearly understandable even for users with no familiarity with RDF and SPARQL concepts. To improve the readability and comprehension of the displayed KG, *Stunning Doodle* implements a partial visualization of the graph and allows users to navigate and explore it incrementally.

6.2.1 Knowledge Graphs Visualization

The first function of *Stunning Doodle* is the dynamic visualization of a KG using a regular graph layout. The process starts with the upload of the RDF graph file in Turtle syntax. Once the file is uploaded, the graph will be displayed as shown in Fig. 6.1. On the left side of the page, under the “Upload Graph” menu, three menus provide relevant information about the graph: the list of the declared namespaces with their prefixes (1), and the legend, and advanced customization options for nodes (2) and links (3) that will be detailed later in this section. The visualization is interactive: nodes can be moved and zoom and spanning functionalities are available. Nodes can be selected by clicking on them, and the triples associated with the currently selected node are listed in the component

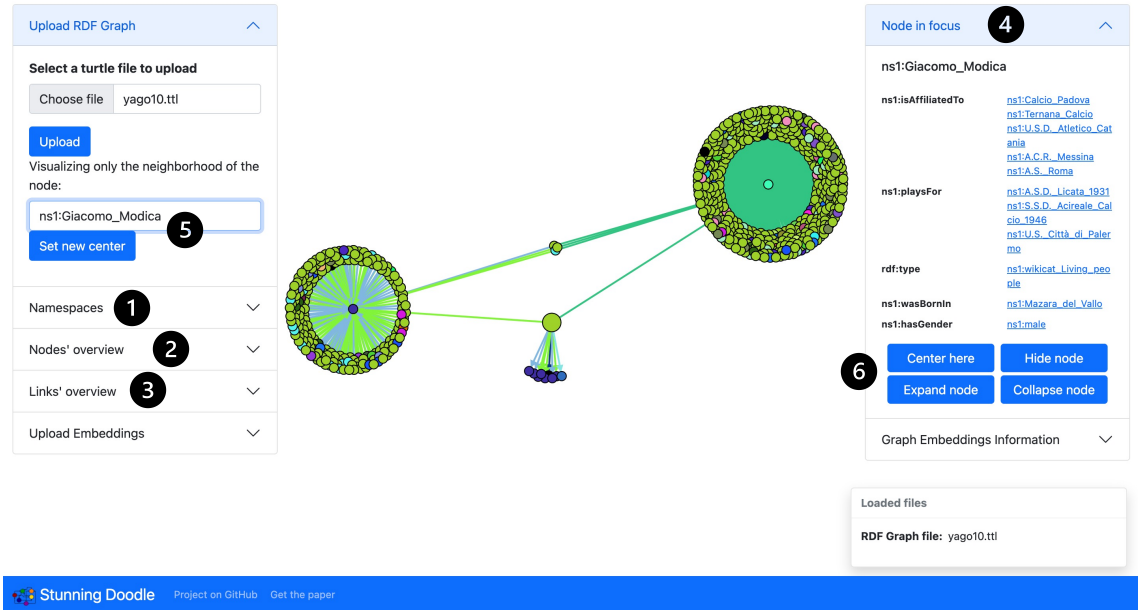


Figure 6.1: Screenshot of the partial visualization of YAGO3 illustrating *Stunning Doodle*' exploration capabilities.

"Node in focus" (4) on the right. In the example of Figure 6.1, which shows an extract of YAGO3 (Mahdisoltani et al., 2014), the selected node is the largest one at the bottom of the graph (`ns1:Giacomo_Modica`) and the triples for which this node is a subject are listed on the right. Each object in this list can be clicked to select the corresponding node in the graph.

6.2.1.1 Graph Exploration Interface.

One of the main characteristics of *Stunning Doodle* is the ability to display nodes incrementally, thanks to the graph exploration system.

To deal with the large number of nodes possibly described in a KG, existing visualization tools employ different strategies: *(i)* relying on clusterized views which group similar (or close) nodes together (Po and Malvezzi, 2018), *(ii)* visualizing nodes incrementally based on the displayed area (Bikakis et al., 2015, 2016), or *(iii)* showing only the nodes that are considered the most relevant based on diverse statistical metrics (Santana-Pérez, 2018). In *Stunning Doodle* we opted for a different approach: a user-guided incremental exploration. Indeed, through our tool, the user decides which nodes to visualize according to their interests and needs. This choice is motivated by the fact that different users can have different, possibly very specific requirements, e.g. analyzing only the facts related to individuals of a given type while disregarding all the other possible predicates; and these requirements do not always correspond to the standard clusterization criteria or relevance metrics. As Figure 6.1 shows it, after the upload of an RDF graph, only one node, randomly selected among all the nodes in the graph, is displayed alongside its close

Figure 6.2: *Stunning Doodle* interface with the available customization options.

neighborhood, i.e. nodes at one-hop distance from the main node. Users can change the displayed node by typing the URI of the node they want to visualize in the component "Upload Graph" and setting it as a new center (5). Users can also navigate the graph starting from a node of interest by using the buttons in the "Node in focus" component (6). Indeed, once a node is selected, users can decide to either *(a)* center the graph on that node, causing only the selected node and its neighborhood to be visualized, while hiding the other nodes and links; *(b)* hide the node, producing the removal of that node together with the nodes that are connected only to it; *(c)* expand the node, i.e. adding the direct neighbors of the node to the visualization; and *(d)* collapse the node, i.e. removing all the nodes that are only connected to that node while keeping the node itself. For example, the visualization in Figure 6.1 has been obtained by centering the graph on node `ns1:male` at first, and then expanding the selected node (`ns1:Giacomo_Modica`) to display its neighborhood.

6.2.1.2 Filtering and Customizing Nodes and Edges.

Another helpful feature of *Stunning Doodle* is the possibility of filtering nodes based on their types (values of property `rdf:type`) and customizing nodes and links colors. These features help the user to easily recognize the nodes and links they want to analyze and to focus only on them while removing information that is irrelevant for their exploration needs.

In Figure 6.2 we show an example customized visualization of an RDF graph and the settings of the visualization modes. These are optional and can be activated by selecting the corresponding options in the "Nodes' overview" and "Links' overview" menus. In

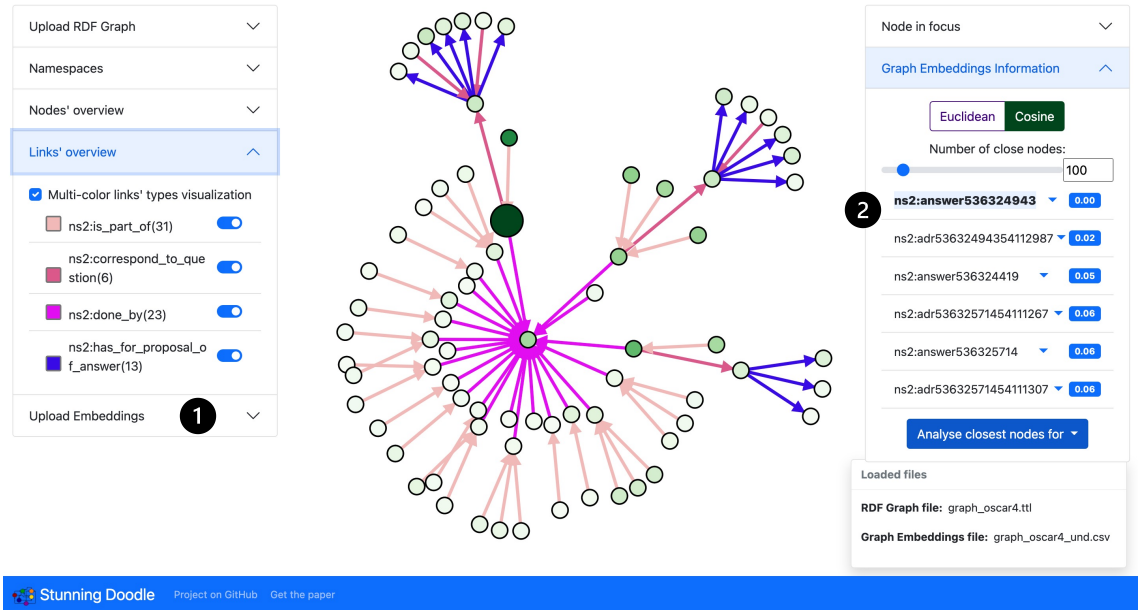


Figure 6.3: Screenshot of *Stunning Doodle* showing closest nodes in the embedding space. Closest nodes to selected one are shown with a gradient of color representing the distance: darker nodes are closer in the embedding space, while lighter nodes are more distant.

the "Links' overview" menu (1), all the possible predicates are listed, each one with the number of edges of that type currently displayed on the graph and the corresponding colors which can be customized by the user (2). Similarly, in the "Nodes' overview" menu (3), the types of all the displayed nodes are listed with the count of occurrences of every type (4). For each type, a filter is added to hide/show the corresponding nodes (5). In the same component, we find two additional options that allow the user to enable and disable labels and literals visualization (6). Once activated, the former will display labels, i.e. values of properties `rdfs:label` and `skos:prefLabel`, next to the nodes subject of these properties; while the latter will add literals as leaves in the graph.

6.2.2 Graph Embeddings Visual Analysis

In addition to the dynamic visualization of a KG, the key functionality provided by *Stunning Doodle* which represents a major step forward when compared with the state-of-the-art tools described in Section 6.4 is the possibility of having a first, simple visual analysis of the GEs computed from the visualized KG.

As explained in Chapter 1, the recent success of KGs is mainly due to the growing number of AI applications relying on KGs through the use of KGEs. Unfortunately, the main stumbling block to extensive utilization of such representation is its difficult interpretability.

With *Stunning Doodle*, we take a first step in the understanding of KGEs through the joint visualization of both KGEs and the KG from which they are generated. More

precisely, *Stunning Doodle* enables to visualize, for each node, its closest nodes in the embedding space, according to either Euclidean distance or cosine similarity. Figure 6.3 shows an example of KGEs analysis. The "Upload Embeddings" menu allows the user to upload a CSV file containing the GEs computed from the visualized KG (1). Then, the user can select a node of interest and visualize its closest neighbors in the embedding space. The closest nodes are displayed with a gradient of color that represents their distance from the node whose embedding is analyzed, i.e. darker nodes are closer (in the embedding space) to the selected node while lighter nodes are more distant. If a relation between any couple of visualized nodes exists in the KG, then the corresponding link is directly displayed in the graph according to the selected customization settings.

The list of the closest nodes with their distance is shown in the "Graph Embeddings Information" menu on the right side of the page (2). Together with this list, additional options allow the user to choose the desired distance metric (Euclidean or cosine in the current implementation), and to customize the number of closest nodes to be displayed. The example in Figure 6.3 shows the 100 closest nodes in the embedding space to the node `ns1:answer536324943`, according to cosine distance. The node currently selected (biggest node) is the node that we are analyzing and from which the distances are computed. This is obvious from the fact that its URI is the first item in the list of nodes and it is also the darkest node in the graph, as the distance from itself is 0. The different shades of green of each node clearly highlight which nodes among the 100 visualized are closer, while the links show how they are connected in the KG.

While visualizing a node's neighbors in the embedding space, it is still possible to access the functionalities for navigating in the KG through the buttons in "Node in focus". Therefore, any displayed node can be expanded to show additional nodes linked to it in the KG even if they are not close in the embedding space. Naturally, any new node can be further expanded to visualize the desired portion of the KG. All the links and the displayed nodes whose embeddings are not close to the initial node will be visualized according to the selected customization options in "Nodes' overview" and "Links' overview". To switch back to the simple view of the KG the user simply has to recenter the graph on any of the displayed nodes.

To sum up, *Stunning Doodle* enables the user to understand in a glance which nodes of a KG are considered to be similar in the embedding space, while keeping track of their connections in the KG. This permits to gain immediate insights on the information captured by KGEs, e.g. which predicates have the highest impact or what connectivity patterns are more taken into account during the embedding process.

6.2.3 Software Design and Limitations

For the sake of simplicity and flexibility of use, *Stunning Doodle* has been implemented as a lightweight web application relying on Python and Javascript, respectively for back-end and front-end. Its setup is fairly simple as it requires only to create a Python virtual environment and it allows the easy deployment on a server to be accessed remotely by multiple users through a web browser. *Stunning Doodle* uses as input for the graph visu-

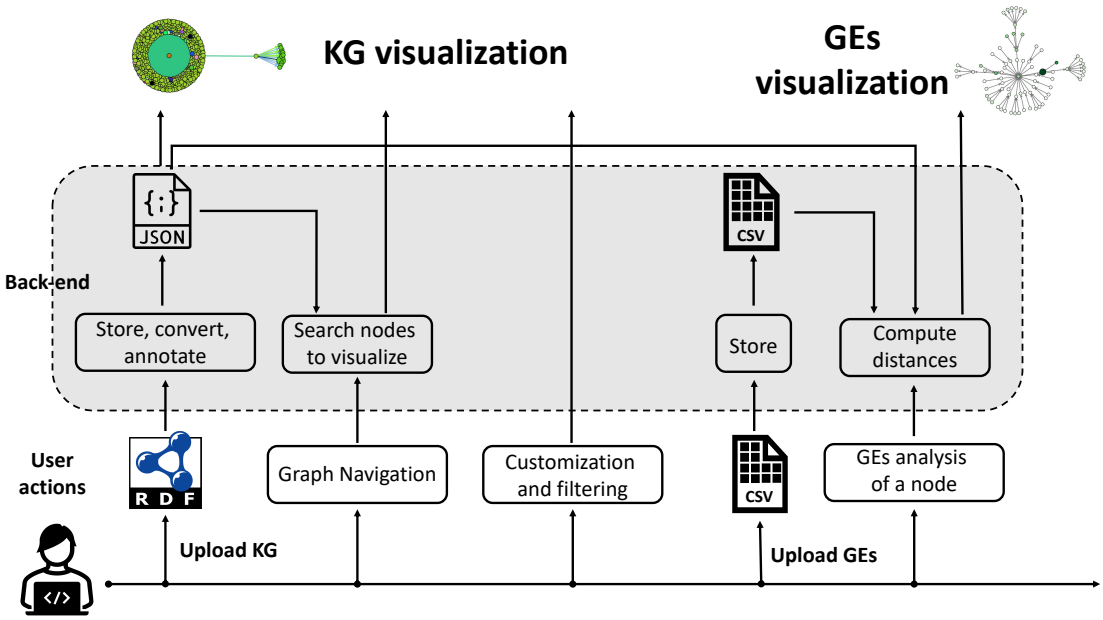


Figure 6.4: Schema of the processing pipeline implemented in *Stunning Doodle*.

alization an RDF Knowledge Graph stored in a Turtle file. For the analysis of the GEs, a CSV file containing the nodes' URIs associated with the corresponding embeddings is required. Figure 6.4 illustrates the processing pipeline of *Stunning Doodle* and shows how the interactions with the user are handled. Firstly, the user needs to upload the RDF graph she wants to visualize. This graph is parsed and converted into JSON, enriching each node description with information about its types and labels to enable on-the-fly customization of the visualization based on this information. When the user navigates the graph by re-centering it or expanding new nodes, a remote request to the back-end is performed to compute and retrieve the new list of nodes to visualize. Since the actions of customization and filtering concern only the nodes and edges currently visualized, they do not require any remote request to the server. To enable the functionalities of analysis of GEs, a CSV file storing such embeddings must be uploaded by the user and stored on the server. At this point, the user can analyze the embeddings of any displayed node, by visualizing its closest nodes in the embedding space. When this action is executed, the back-end is queried to compute the list of the closest nodes with their distances. After mapping the closest embeddings with their corresponding nodes and computing the edges connecting them, the results are returned to the front-end that displays the obtained subgraph.

The main limitation of *Stunning Doodle* lies in the size of the KG to be analyzed. Indeed, the larger the KG and, consequently, the embeddings file are, the longer is the time needed to upload such files, to search for the nodes to be displayed and to compute the distances between embeddings. Table 6.1 reports the time needed to execute the main actions provided by *Stunning Doodle* on the graph shown in Figure 6.1 and Figure 6.2, which contains 920.435 triples describing 124.982 nodes. These timings were obtained

Table 6.1: Execution time with number of displayed nodes when performing actions for the analysis of a subset of YAGO3.

	Load graph	**Set center** (`ns1:male`)	**Expand node** (`ns1:AC_Sparta_Prague`)	**Closest nodes**
Time	120s	21,7s	12,03s	5,4s
# nodes	-	502	271	100

running *Stunning Doodle* on a MacBook Pro with 2,8 GHz Quad-Core Intel Core i7 and 16GB RAM.

6.2.4 Software Availability and Reusability

Stunning Doodle is made available as an open-source software under the MIT License. The tool is identified by means of a DOI provided through Zenodo (Ettorre, 2021) to improve its accessibility and citability. Moreover, *Stunning Doodle* source code is publicly accessible on the GitHub repository[1], which also includes extensive documentation and examples to guide the users. We plan on keeping improving *Stunning Doodle* with new functionalities and offering best-effort support to future users in case of bugs or issues of any kind. Furthermore, we welcome any feedback, idea or contribution by the community.

6.3 Our Use Case: the OntoSIDES Scenario

We describe hereafter the main use case that motivated the development of *Stunning Doodle* and how this tool, with the functionalities it implements, has been useful for helping us understand the different Knowledge Graph Embeddings generated from OntoSIDES and the knowledge they encode. Before demonstrating the usefulness of the Graph Embeddings analysis provided by *Stunning Doodle*, we show how this tool can be used for visualizing and understanding the content and the structure of the original OntoSIDES subgraph from which the analyzed Graph Embeddings are computed. The subgraph described in the following sections slightly differs from the one used for the experimentation in Chapter 5, as it includes further information about the students, i.e. their university, and, in general, a higher number of triples (around 650.000).

6.3.1 Understanding a Knowledge Graph

In this section we explain how *Stunning Doodle* can be used for the visualization and comprehension of the KG in the SIDES scenario. In this use-case, we assume users are only aware of the high-level concepts that are supposedly defined in OntoSIDES and they need to understand what the described entities are, how they are modeled in the KG

[1] https://github.com/antoniaettorre/stunning_doodle

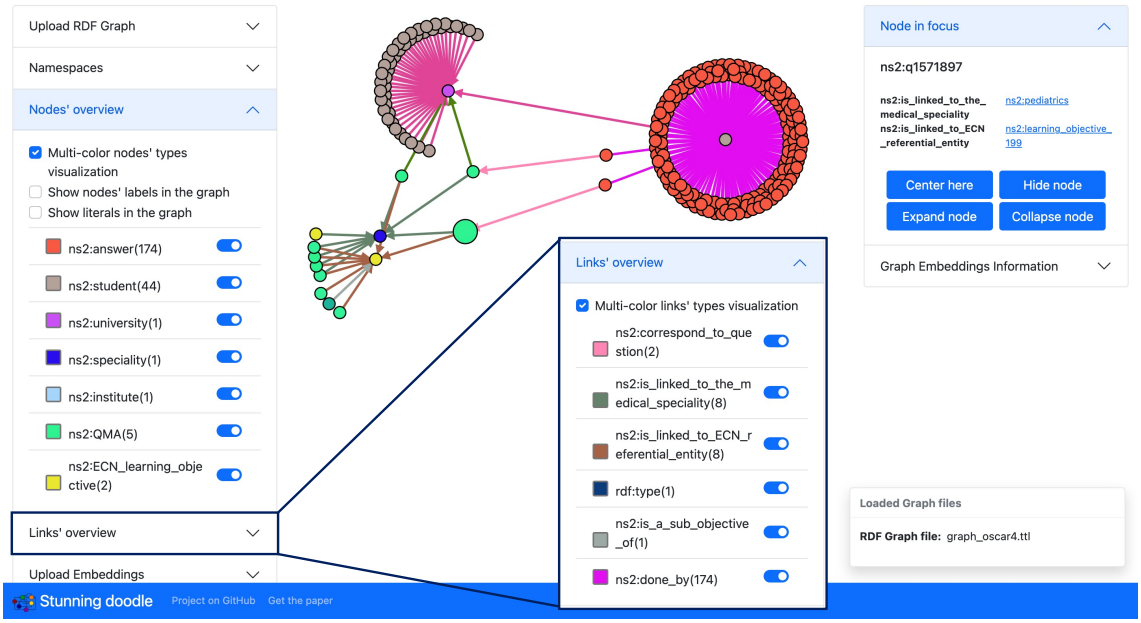


Figure 6.5: Screenshot of *Stunning Doodle* showing the basic entities and relations in the OntoSIDES graph.

and how they are linked to each other through predicates. Expert users are normally able to gain this knowledge by running several prototypical SPARQL queries, but the use of *Stunning Doodle* facilitates this task by retrieving and displaying the same information only through a few clicks, and, at the same time, makes the KG accessible also to users with no expertise in SPARQL.

Once the graph file is uploaded, users can choose a node from which to start the graph exploration. This allows users to expand only the nodes that are relevant for them, displaying only the needed information. After expanding a few nodes, users should be able to visualize a small graph containing all the main elements of interest, i.e questions, students, answers, institutes, and their links. Thanks to the "Nodes' overview" menu, users are able to distinguish the main entities in a glance, based on their colors. Considering the visualization (Figure 6.5), it is evident that instances of *answers* (orange nodes) are directly linked to *questions* (spring green) and *students* (in gray). Moreover, it is possible to see that questions are associated with *specialties* (navy blue nodes) and *learning_objectives* (yellow). Taking a look at the "Links' overview" menu, predicates can also be easily discriminated in the graph thanks to their colors. Moreover, connections between distant nodes can be quickly identified to be possibly used as property paths in SPARQL queries. For example, to analyze which topics a student worked on, the user only needs to follow the path going from a gray node (student) to a yellow node (*learning_objectives*), therefore, chaining the properties *done_by* (fuchsia line), *correspond_to_question* (pink), *is_linked_to_ECN_referential_entity* (brown). Additionally, few statistical observations on the number and types of entities can be done from the menus "Nodes' overview" and "Links' overview". In the example in Figure 6.5, the "expanded" student gave 174 an-

swers, the displayed university has 44 students, all the questions are linked to the same medical specialty. Finally, each node can be analyzed in detail, by selecting the node and looking at the menu "Node in focus" which displays the triples associated with that node.

Thanks to its characteristics, *Stunning Doodle* proved that it can be useful to gain a first, global understanding of the content and organization of the OntoSIDES KG. Its navigation systems, filtering functionalities, and customization capabilities allow the user to explore the graph step by step, focusing at each moment only on the pieces of information of interest for them. We claim that those features are very helpful for the understanding of Knowledge Graphs in any context.

6.3.2 Analyzing and Comparing Knowledge Graph Embeddings

The GEs analysis provided by *Stunning Doodle* allows to gain insights into the information encoded in the GEs and, therefore, to assess their meaningfulness for their final use. In particular, *Stunning Doodle* can be used for comparing GEs computed through different embedding models with several hyper-parameters settings, with the final goal of identifying and tuning the model generating the most meaningful embeddings. In our use case, (1) interpreting the information captured by GEs, (2) identifying the best model for their computation, and (3) tuning the hyper-parameters to obtain a meaningful representation of the nodes are crucial tasks.

In the following, we show how *Stunning Doodle* has been used to carry out these tasks, by analyzing the GEs generated from the OntoSIDES subgraph described in Section 6.3.1. We inspect and compare two sets of GEs, both computed using the *node2vec* model (Grover and Leskovec, 2016). In the first case, the embeddings have been computed considering the graph as directed (i.e. it can be traversed only going from subjects of triples to objects), while in the second case embeddings are obtained considering the graph as undirected (i.e. links are bidirectional, it is possible to go from objects to subjects). In both cases, we study the embeddings of a student node (`stu81235`) to figure out which nodes are considered to be similar from the embeddings point of view and, therefore, what information embeddings can capture from the KG. After uploading the graph and embeddings' files, we select the node corresponding to `ns2:stu81235` and we choose to visualize its closest nodes in the embedding space.

GEs from a Directed Graph. Figure 6.6 shows the result of visualizing the 100 nodes with the shortest Euclidean distance (in the embedding space) from node `ns2:stu81235`. The darkest node is node `ns2:stu81235` itself since its distance from itself is 0. Immediately, it occurs from the visualization that its closest nodes are not directly linked between each other and, based on the gradient color, some of them are much closer than others. Looking at the nodes' list in the menu "Graph Embeddings Information", we can see that all the closest nodes are other instances of the student class. Therefore, we can assume that during the embedding process, nodes with the same type are recognized as similar, ending up close in the embedding space.

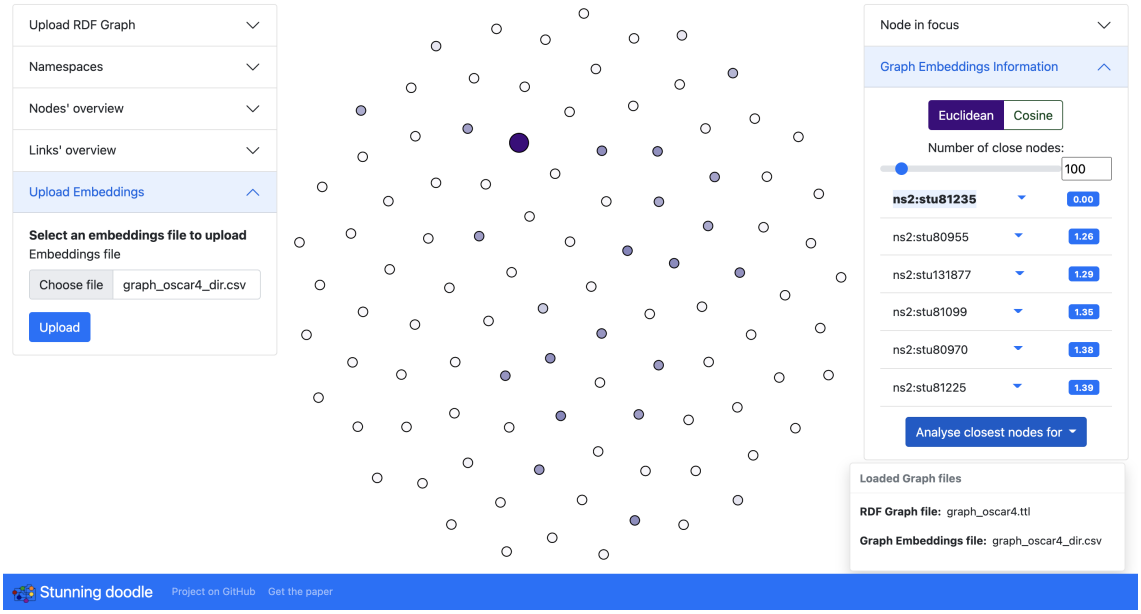


Figure 6.6: Closest nodes in the embedding space to the node `ns2:stu81235` with GEs computed from a directed graph.

To investigate why some students are closer to `ns2:stu81235` than others, we can "expand" a few nodes to find connections among them. Figure 6.7 shows that the closest nodes (darkest color) are connected to the same university (fuchsia node) as `ns2:stu81235`. This highlights the fact that the link with the university has a high impact during the computation of the embeddings of a student's node. Nevertheless, we can notice that not all the students registered at the same university as `ns2:stu81235` are among its closest nodes (gray nodes in Figure 6.7), meaning that there are other factors affecting the similarity between embeddings. Repeating the same analysis for other nodes of type `ns2:student` led to the discovery of similar patterns, i.e. all the analyzed nodes resulted to be similar to other students attending the same university. Thanks to these observations, we can conclude that, in this case, GEs are able to encode the information about the node type and the university for students' nodes.

GEs from an Undirected Graph. Figure 6.8 shows the closest nodes to `ns2:stu81235` for the embeddings computed from the undirected graph. The difference with the Graph Embeddings computed in the previous case is immediately visible. Firstly, by looking at the list of the closest nodes we discover that for undirected embeddings the type of the node does not play an important role in the embedding process. Indeed, the closest nodes are of various types, including `ns2:answer` and `ns2:action_to_answer`. From the visualization, it is obvious that the connections between nodes assume a much more important role, as the closest nodes in the embedding space result to be the ones that are close in the Knowledge Graph as well (at 1-hop or 2-hops distance). On the other side, we notice that the opposite implication does not

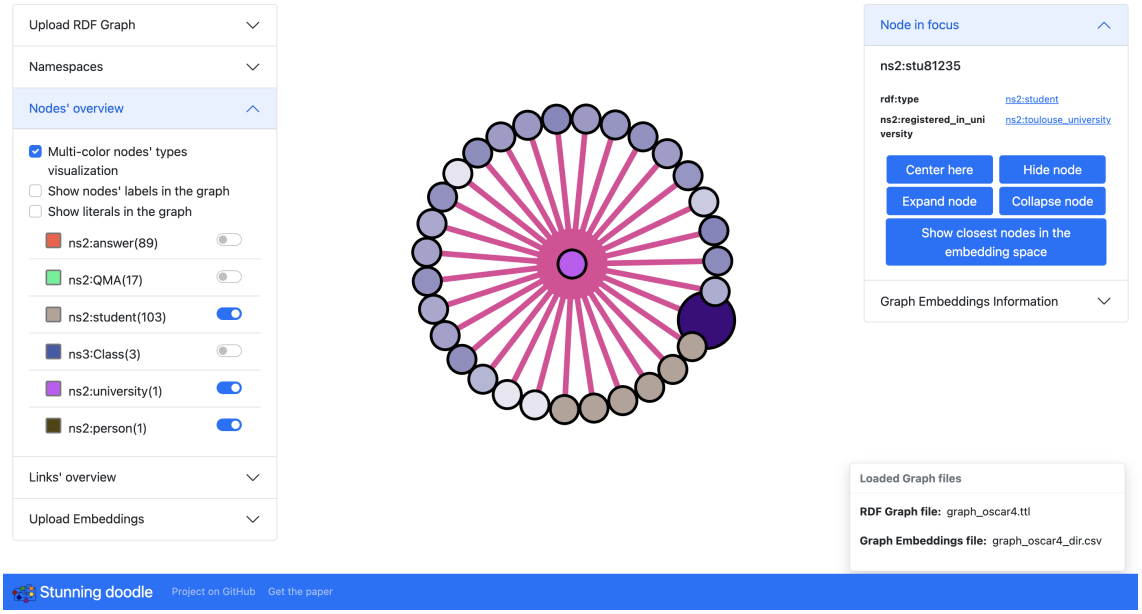


Figure 6.7: Links between `ns2:stu81235` and its closest nodes in the embedding space.

hold. Indeed, when we expand node `stu81235` to display all its direct neighbors in the Knowledge Graph as done in Figure 6.9, we can see that there are additional nodes which were not displayed before as they are not close in the embedding space (answers in orange). Therefore, we can assume that the distance in the Knowledge Graph, though important, is not the only parameter taken into account during the embedding process. Same observations can be done for other nodes of type `ns2:student`.

Finally, uniquely through the joint visualization of the distances in the embedding space and the Knowledge Graph structure, we can draw interesting conclusions on the meaning of Graph Embeddings and, thus, identify the best settings to be used for the embedding computation. In our case, we discovered that Graph Embeddings computed from directed graphs can capture semantic information from the Knowledge Graph, such as the nodes' type and the university associated with each student; while Graph Embeddings obtained from undirected graphs tend to summarize the graph structure. As a consequence, we were able to confirm and partially explain the results obtained through the experimentation in Section 5.2.3. Indeed, we confirm that the most meaningful embeddings are the ones computed from the directed Knowledge Graph as they are able to better encode semantics, which is more relevant for the prediction of students' performance, such as the university attended by the students.

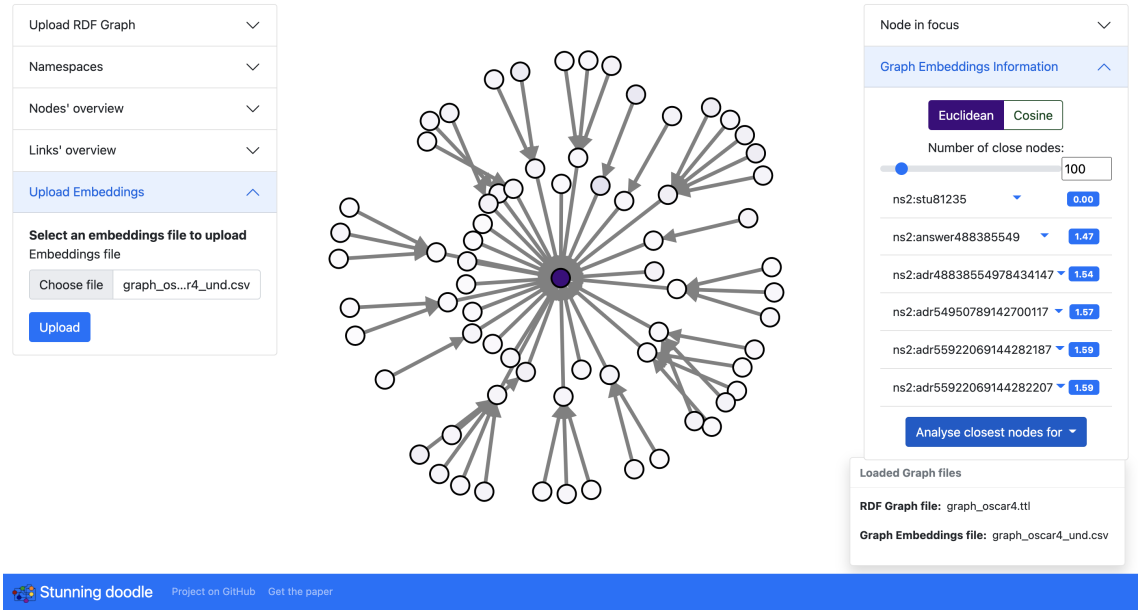


Figure 6.8: Closest nodes in the embedding space to the node `ns2:stu81235` with GEs computed from an undirected graph.

6.4 Related Work

Despite the growing attention dedicated to the issues of interpretability and understandability of Knowledge Graph Embeddings, to the best of our knowledge, no work has been done towards the visual analysis and comprehension of such representations. Yet, some solutions have been developed for the similar task of sentences and words embeddings visualization. *TensorFlow Projector*[2] allows users to upload and visualize their embeddings to help the comprehension of the information they summarize. Through this tool, it is possible to visualize the uploaded embeddings in a 3D space by using dimensionality reduction techniques. Moreover, the tool enables, for each element, to list its closest neighbors in the embedding space with the final goal of providing insights on the meaning of the computed embeddings. Although *TensorFlow Projector* is general enough to visualize any kind of embedding, i.e. word, sentence or graph, its usefulness in understanding Knowledge Graph Embeddings is limited by the absence of KG-specific functionalities, such as displaying the graph structure or considering semantics.

Concerning the visual analysis of Knowledge Graphs, multiple tools have been developed throughout the years. Recent research efforts (Antoniazzi and Viola, 2018, Desimoni and Po, 2020) focused on listing, analyzing, and comparing existing Linked Data (LD) visualization tools able to deal with both KG schema and data. The majority of the available tools rely on a graph-based visualization and present several commonalities with respect to the provided functionalities and input mode. Many of these applications enable

[2] https://projector.tensorflow.org

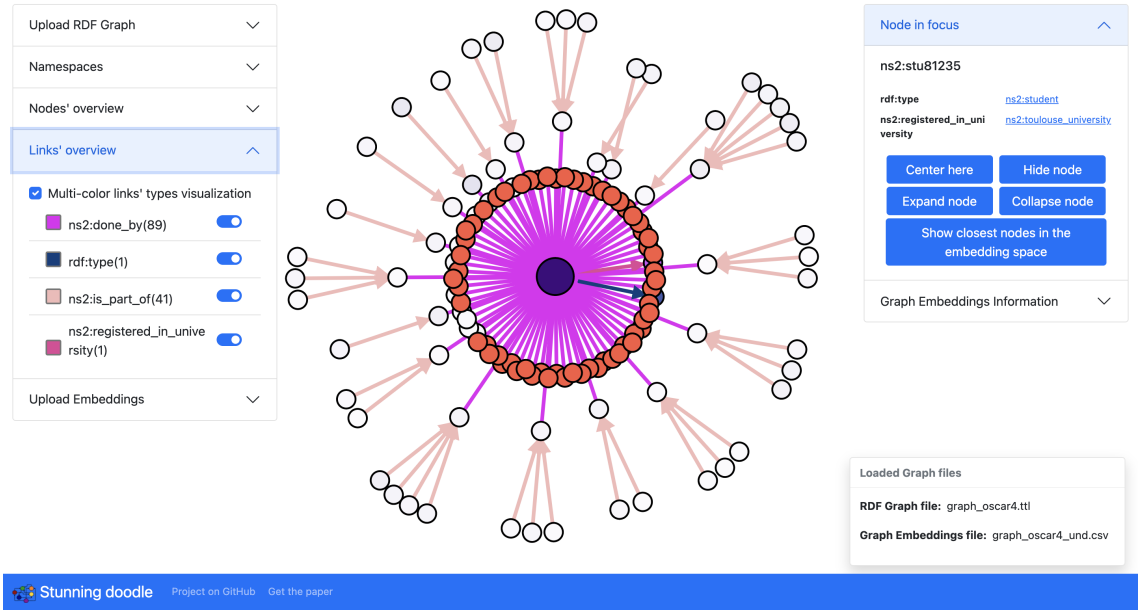


Figure 6.9: `ns2:stu81235` expanded to show its neighbors in the KG that are not close in the embedding space.

to visualize resources accessible through a public SPARQL endpoint (Nuzzolese et al., 2017, Po and Malvezzi, 2018), while only a few of them allow users to upload local RDF graphs (Troullinou et al., 2018). Some of the tools complement graph visualization with advanced features, such as augmenting graph content with external information (Nuzzolese et al., 2017) and collecting statistics about the Knowledge Graph (Santana-Pérez, 2018). Different strategies are used to deal with large-scale Knowledge Graphs, e.g. (Po and Malvezzi, 2018) aggregates nodes in clusters, (Asprino et al., 2021, Troullinou et al., 2018) try to identify and show firstly the most important concepts, while others, such as (Camarda et al., 2012, Micsik et al., 2014), rely on incremental exploration by the user. The conclusion drawn by (Desimoni and Po, 2020) highlights the impossibility of identifying the best tool in the absolute and states that the research in this field is far from conclusion. Moreover, most of the existing applications are developed as proofs of concept or research tools, therefore they are often conceived for a very specific task and dataset and they can hardly be generalized, or they are rather cumbersome to set up. These limitations make them not suitable for widespread use by non-expert users.

Inspired by the Graph Embeddings visualization implemented by Tensorflow, *Stunning Doodle* builds upon the functionalities offered by recent Knowledge Graphs visualization tools and extends them to enable a simple and straightforward visual analysis of the Knowledge Graph Embeddings.

6.5 Conclusions and Future Work

Stunning Doodle is a first step to fill the gap in the field of visual analysis of Knowledge Graph Embeddings. This visualization tool enables to build a link between the content and structure of any Knowledge Graph and its corresponding embeddings. We implemented a set of functionalities to facilitate the exploration and understanding of any Knowledge Graph and to analyze Knowledge Graph Embeddings, connecting the two, and making sense of the information captured by the Knowledge Graph Embeddings. We used *Stunning Doodle* to answer the questions exposed at the beginning of the chapter for the prediction of students' performance through the OntoSIDES Knowledge Graph Embeddings. We discovered that the Graph Embeddings computed from the undirected graph encode structural information, while the ones computed from the directed graph better capture semantics; and, therefore, that the latter are better suited for our final prediction task.

Stunning Doodle proved to be useful for gathering first insights into the information captured through embeddings and it can surely be improved for several aspects. As future work, we plan to implement new functionalities particularly useful for more expert users, such as the visualization of the result of SPARQL queries and the direct access to well-known SPARQL endpoints. Moreover, we aim to provide a deeper analysis of the uploaded Graph Embeddings including advanced statistics on the closest nodes and additional similarity metrics. We also want to optimize the pre-processing pipeline to be able to display and analyze larger Knowledge Graphs. Our hope is that *Stunning Doodle* could build a large community of users and keep improving and growing throughout the years to satisfy their needs.

CHAPTER 7

A Methodology to Identify the Information Captured by Knowledge Graph Embeddings

In this chapter, we aim to tackle the issue of Graph Embeddings interpretability by providing a systematic approach to decode and make sense of the knowledge captured by Graph Embeddings. We propose a technique for verifying whether Graph Embeddings are able to encode certain properties of the graph elements and we present a categorization for such properties. We test our approach by evaluating the embeddings computed from the same Knowledge Graph through several embedding techniques. We analyze the results on the level of encoding of each property by all the benchmarked algorithms with the final goal of providing insights into the choice of the most suitable technique for each context and encouraging a more conscious use of such approaches. This chapter is based on the work published at the International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT2021) (Ettorre et al., 2021).

7.1 Introduction

In the previous chapter, we saw how Stunning Doodle can help us get an idea of under which settings Graph Embeddings incorporate the most interesting information for our final task. Nevertheless, this tool provides only very high-level insights on what information is possibly encoded by these Graph Embeddings. To be able to interpret the Knowledge Tracing results provided by our KG-based model, we need a more analytic method to precisely identify the information captured by the Graph Embeddings and, therefore, used for the prediction of the student's success.

As explained in Chapter 6, unveiling the information encoded by Knowledge Graph Embeddings is a task of general interest for the whole AI community. Thus, several research efforts have been made in the last years to move towards more explainable GEs. These include proposing interpretable embedding methods (Kazemi and Poole, 2018), studying the impact of graph modifications on link prediction (Pezeshkpour et al., 2019) and offering explainer approaches for specific embedding models (Ying et al., 2019). Nevertheless, a methodology to effectively explain the predictions enabled by KGEs is still missing (Palmonari and Minervini, 2020). Moreover, although several evaluations of the performance of different GE techniques have been conducted with respect to their effectiveness in terms of quality of link prediction (Dai et al., 2020, Rossi et al., 2021), to the best of our knowledge no work has been made to systematically analyze the meaningfulness of the information they encode.

A similar challenge has been addressed in the context of text embeddings, for which the final goal is to understand which characteristics of the language are actually encoded in the embeddings of words and sentences. A first attempt of investigating this research question is based on the use of "probing tasks", presented in (Adi et al., 2016). A probing task is an auxiliary classification task that, taking as input the embedding of an element, i.e word, sentence, or node in our case, is trained to predict the value of a given property for this element. For example, (Adi et al., 2016) tested sentences embeddings for properties such as sentences length and word ordering. The general idea behind this approach is that, if some information about a given element is encoded through the embedding process in its vector representation, it should be possible to recover such information from the embedding alone.

Inspired by (Adi et al., 2016), we propose to make use of such auxiliary tasks to verify the hypothesis according to which GEs encode certain properties. We provide a methodology independent from the type, context and final use of the GEs, that allows to systematically analyze the information captured from the KG. We do so by establishing a catalog of probing tasks that can be easily generalized and reused. We focus specifically on RDF Knowledge Graphs, however the presented approach can be applied to other kinds of graphs, such as property graphs, by defining different probing tasks. We rely on this catalog to evaluate the information encoded by several GE algorithms, both general-purpose (node2vec) and specifically designed for KGs (TransE, ComplEx, etc.), in two different contexts: a general-purpose KG describing a large variety of real-world facts and a subset of OntoSIDES KG. Conducting this evaluation on two real-world KGs that describe

very different kinds of data allows us to avoid the bias introduced by inverse triples, data redundancy, and cartesian product relationships, normally present in the benchmarking datasets frequently used to evaluate GE algorithms, i.e. FB15k and WN18 (Akrami et al., 2020). We compare the results achieved by all the algorithms on each task to determine what information is mainly captured by every one of them.

The remainder of this chapter is organized as follows. In Section 7.2 we review existing related works. In Section 7.3 we categorize the information that can be encoded by GEs and we propose a corollary organization of probing tasks, while in Section 7.4 we compare and discuss the results of state-of-the-art GE algorithms on the identified probing tasks. Additionally, in Section 7.5 we show how the probing task approach has been used to verify the hypothesis, exposed in Chapter 5, about the encoding of the number of selected proposals of answers by the Graph Embeddings of answers' nodes. Finally, conclusions are presented in Section 7.6.

7.2 Related Work

Lately, several efforts have been made to investigate the interpretability of GE techniques. (Ying et al., 2019) present GNNExplainer, a tool to provide possible explanations for the results of link prediction through GEs. GNNExplainer is able to identify the subgraph and the node features that are the most relevant for any link prediction made by GNN models. Such identification is achieved by solving an optimization problem to maximize the mutual information between a GNN's prediction and distribution of possible subgraph structures. Other approaches towards explainable link prediction have been proposed in (Bhowmik and de Melo, 2020, Kang et al., 2019). Also, the work proposed in (Pezeshkpour et al., 2019) focuses on the interpretability (and robustness) of link prediction by trying to identify the fact whose addition or removal from the KG causes changes in the prediction for a targeted fact. While these works aim for the explanation of link predictions through embeddings, few attempts have been made to understand the content of the embedded representation independently of the downstream application. In (Sharma et al., 2018) the authors study the geometric characteristics of the computed vectors with respect to the type of embedding model, i.e. additive or multiplicative, and its hyperparameters. (Sharma et al., 2018) show that the nature and the settings of the embedding method can affect the distribution of the vectors in the embedded space.

(Jain et al., 2021) raised doubts on the effective capacity of KGEs of capturing the semantics borne by the KG. The authors define a classification and clustering task to predict entity type and compare several well-known KGE algorithms on these tasks, i.e. TransE(Bordes et al., 2013), RESCAL(Nickel et al., 2011), ComplEx(Trouillon et al., 2016), DistMult(Yang et al., 2015) and ConvE(Dettmers et al., 2018). This work concludes that none of these approaches is able to properly encode semantics, as they can only identify macro classes and not fine-grain ones, e.g. they can distinguish a *person* from a *body_of_water* but not a *scientist* from an *artist*. This approach presents some similarities with the technique used to decode information captured by text embedding

introduced by (Adi et al., 2016), which defines some "probing tasks" to evaluate the information captured by word and sentence embeddings. They provide a set of structural, low-level information properties that can be encoded by the vector representation, i.e. sentence length, word content, and word order. For each one of these properties, they define a classification task that takes as input the word (or sentence) embedding and outputs the value of the property. Through this approach, they evaluate the information encoded by LSTM auto-encoder and CBOW techniques. They also study the changes in the encoded information with the variation of the embeddings' size. In (Conneau et al., 2018), the authors improve the approach proposed by (Adi et al., 2016), providing a proper classification of 10 probing tasks for the properties possibly encoded by text embeddings, hierarchically organized into surface, syntactic and semantic tasks. Moreover, they redefine the probing tasks to be more general, interpretable, and easily re-usable. To this end, they use as input a single sentence embedding, while (Adi et al., 2016) used multiple embeddings of both words and sentences. Finally, they benchmark a much wider selection of text embedding algorithms. The categorization defined by (Conneau et al., 2018) has also been used to uncover the information encoded by BERT, with the specific goal of understanding to which extent this model can capture the structure of the language (Jawahar et al., 2019).

Relying on the idea of probing tasks introduced by (Adi et al., 2016), we present a methodology that allows testing the capacity of KGEs to encode not only entities' type like in (Jain et al., 2021) but also a vast range of properties and characteristics of graph elements that could possibly be encoded in their vector representations. While the work presented in (Bhowmik and de Melo, 2020, Kang et al., 2019, Pezeshkpour et al., 2019, Ying et al., 2019) focuses uniquely on link prediction, the approach we propose can be used to evaluate the meaning of GEs independently of the downstream application.

7.3 Analysing the Information Encoded by Graph Embeddings

Following the footsteps of the previously presented works, we introduce a list of auxiliary tasks that could be used to decode the information captured by GEs. An auxiliary task, or probing task, is a simple classification task that takes as input the embedding of an element and outputs the value of a given property for that same element. If it is possible to train a classifier to solve such a task, we can conclude that the set of input features, i.e. the embeddings of the elements, encode the initial property or characteristic. Throughout this process, one would be able to determine whether GEs capture a given characteristic of the elements in the KG, e.g. entities' types or predicates' constraints. Although this is not a *direct translation process* from the embeddings to the set of encoded properties, it would nevertheless represent a first important step towards GEs explainability if we could define a reasonable set of common properties against which to evaluate our GEs.

To do so, we need to identify which kind of information is more commonly contained in a KG and could possibly be captured by the vector representation. Keeping in mind that

GE techniques can create a vector for each element in the graph, we need to determine which are the properties of nodes or edges that could be transposed in the vector space. To keep our approach and properties' categories as general as possible and, therefore, adaptable to the large variety of embedding algorithms, our analysis will be limited to the properties of nodes, i.e. *Classes* and *Individuals* in the case of RDF graphs, and we will not take into consideration *Predicates* and *Literals*. Nevertheless, we are aware that the vector representation of a node is computed based on the interactions between such node and all the others. Therefore, we will also introduce some properties that define these interactions and that can be captured by GEs, e.g. presence of a direct link, distance in number of hops, and type of link between two nodes. To evaluate the ability of GE models to catch such information, we will define probing tasks that use as input the embeddings of two nodes, instead of one. We describe hereafter the list of probing tasks, defined on three levels of increasing abstraction, as depicted in Figure 7.1.

Structural properties. With the awareness that a Knowledge Graph is, first of all, a data structure that organizes facts in the form of a graph, we define a first, lowest and most general level of our probing tasks' pyramid: the **structural layer**. We argue that the simplest and most straightforward information that GEs can learn from the graph is its structure. This is a direct consequence of the fact that embedding techniques rely on the presence of direct relations between nodes and, for the approaches based on random walks, on the paths built through those relations. The properties in this category are the ones that every node has, independently of its type or role, and whose definition does not require any additional information other than the list of edges. Examples of properties in this category are *in-degree* (i.e. number of incoming connections) and *out-degree* (number of outgoing connections) for each node. These properties are inherently independent of the context and can be reapplied to any graph.

Semantic properties. At a higher level of abstraction, we find properties whose definition makes use of additional information other than the graph structure itself as they require capturing the heterogeneity of the content of the graph. Probing tasks in this class will allow us to test the capacity of GEs to encode **semantic information**, such as *type of nodes and edges*. Here again, these properties are inherently independent of the KG and can be reapplied to any KG. Techniques specifically designed for computing GEs from Knowledge Graphs, such as TransE, ComplEx, DistMult, etc. claim to be able to encapsulate this kind of information.

Context-specific properties. This group includes properties that present the highest level of abstraction as they disregard any information about the structure and organization of the data and are uniquely **based on the human understanding of the context**. In other words, those are the characteristics of the elements described in the graph that any individual would intuitively consider relevant for a given task and that would be meaningful for the final use of the embedded representation. For example, let us suppose that we want

to rely on GEs to develop a recommender system to suggest restaurants to users based on their tastes. In this case, we would imagine the embeddings of restaurants' nodes to encode the information about the "cuisine" prepared by each restaurant (e.g. Italian, French, etc.), since, based on our personal experience, this is relevant information when choosing where to eat. While the two previous categories of tasks are generic and easily reusable to analyze GEs over KGs in different contexts, the properties in this last class need to be defined for each specific use case, based on the knowledge represented in the graph and on the final goal to be achieved through the use of embeddings.

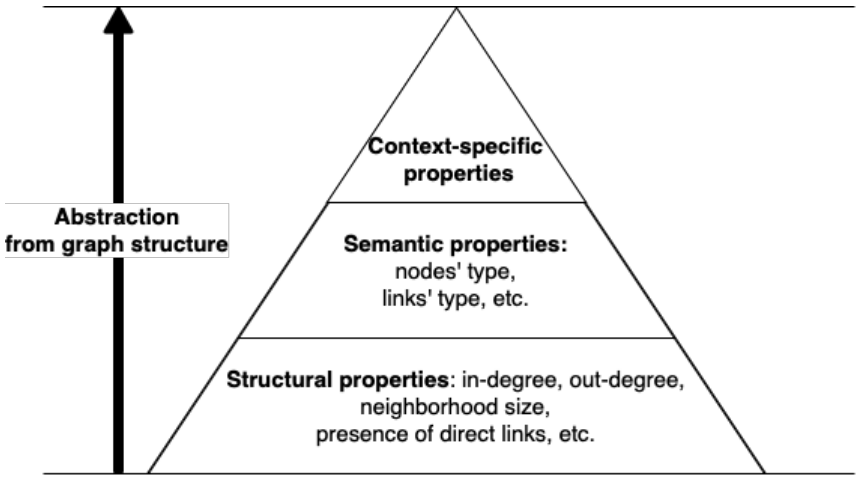


Figure 7.1: Categories of probing tasks.

7.4 Evaluating Graph Embedding Algorithms with Probing Tasks

In this section, we present the results of the application of probing tasks to two KGs described below. We rely on the presented probing tasks' categories to evaluate the information captured from the two graphs by several GE models. We compare them and suggest possible explanations for their behavior with the final goal of gathering meaningful insights to help a more informed use of such techniques in different contexts.[1]

7.4.1 Knowledge Graphs

We demonstrate the use and utility of the previously defined categorization of probing tasks in decoding the information captured from GEs in two very different contexts. Firstly, we assess the information encoded by several GE models on a general-use KG

[1]The code and data for reproducing the experiments can be found at https://github.com/antoniaettorre/probing_tasks_yago.

frequently employed for benchmarking purposes, **YAGO3** (Mahdisoltani et al., 2014). Additionally, we test the same GE techniques on the domain-specific OntoSIDES KG.

7.4.1.1 YAGO3

YAGO3 is a multilingual KG containing almost 150 million facts automatically extracted from Wikipedia and combined with WordNet. It describes about 17 million entities, from *person* to *location* and *organization*, making it an important and heterogeneous source of structured information. Considering the size of YAGO3, we decided to limit our experiments to the subgraph containing only the entities having at least 10 relations. Our subgraph consists of 123.189 entities and 37 relations[2]. To simplify the analysis of the encoded information and to keep our dataset of a reasonable size, we consider for every entity only the most common (frequently used) type. Given the generic nature of the knowledge represented in this graph and the lack of a precise final goal to be achieved through the use of GEs (other than the evaluation of the embeddings themselves), we did not instantiate any context-specific property to be probed. Therefore, for this dataset, we limit our analysis of the information captured by GEs to the first two levels of our hierarchy. More precisely, we evaluate the GEs on the following probing tasks:

- **Structural properties:** *in-degree* and *out-degree* for each node and *presence of direct link* between two nodes;
- **Semantic properties:** `rdf:type` for each entity and *relation* linking two entities.

7.4.1.2 OntoSIDES

In the following experiments, we aim to assess what knowledge in OntoSIDES is actually captured and summarized by GEs. Therefore, we rely on the defined probing tasks to analyze the GEs computed on OntoSIDES. To be able to compare the results with the ones obtained for YAGO3, we test the embeddings for the same properties as those used in the previous case, i.e. *in-degree*, *out-degree* and *presence of direct link* as structural properties, and `rdf:type` and *relation* for semantic properties. Additionally, since OntoSIDES represents very specialized knowledge, we extend the experiments to the following context-specific probing tasks:

- **student's university:** given the GE of a student node, identify the university the student attends;
- **question's topic:** given the GE of a question node, identify the topic (i.e. *learning_objective*) related to that question.

We tested these newly defined probing tasks on a subgraph of OntoSIDES describing the students' learning activity in the pediatrics medical specialty. We picked this specialty

[2]extracted from the YagoFacts theme and extended, for each entity, with the triples stating the `rdf:type` from the the YagoTypes theme.

Table 7.1: List of the probing tasks defined for YAGO3 and OntoSIDES.

KG	Structural properties			Semantic properties		Context-specific properties	
	in-degree	out-degree	direct link	entity type	relation	student's university	question's topic
YAGO3	✓	✓	✓	✓	✓		
OntoSIDES	✓	✓	✓	✓	✓	✓	✓

because it is the one presenting the largest number of attempts by students to answer questions. This choice allows us to obtain a dataset large and heterogeneous enough to train and test all the needed classifiers. Indeed, the size and distribution of the training set are essential to building a high-quality classifier. For example, to be able to associate students to their university, we need a reasonable number of students approximately uniformly distributed among the different universities. Similar considerations hold for all the properties to be probed. Of course, given the nature of the represented information, these constraints are very difficult to meet. It is evident, for instance, that the nodes ot type (`rdf:type`) `sides:answer` will be more numerous than the nodes of type `sides:student`. Keeping in mind these limitations, we extracted a subgraph containing 568.792 entities and 16 types of relations.
The probing tasks defined for both KGs are summarized in Table 7.1.

7.4.2 Classification Model

If our hypothesis is valid and the information we aim to retrieve is indeed available in the input features, even a very simple classifier will likely be able to decode it. For this reason, we decided to rely on *Logistic Regression* as a classification model. Hence, we trained a logistic regression model for each one of the probing tasks to be tested. One of the main issues encountered during the training of such classifiers is the presence of strongly unbalanced classes. As described in Section 7.4.1, extracting a subset of data that would be balanced w.r.t. all the properties to be tested is not straightforward. Therefore we had to consider, for each property, only the subset of its possible values having a number of occurrences above a threshold. At the same time, we needed to undersample the most frequent classes to avoid a highly skewed distribution.

All the classifiers have been trained with a 5-folds cross-validation on 80% of the data and tested on the remaining 20%.

7.4.3 Graph Embeddings Models

We tested and compared some of the most widely used GE algorithms to analyze the differences in their capacity of capturing information from the graph. We evaluated one of the first GE algorithms conceived for generic graphs, i.e. node2vec; and several algorithms specifically designed for KGs: TransE, ComplEx, DistMult, RESCAL, and RotatE. For the computation of the node2vec embeddings we used the implementation provided by

Table 7.2: Weighted F1-scores on the probing tasks with YAGO3 KGEs.

GE algorithm	Structural properties			Semantic properties	
	in-degree	**out-degree**	**direct link**	**entity type**	**relation**
node2vec	**0.12**	0.17	**0.87**	0.79	0.65
ComplEx	0.08	0.15	**0.86**	0.94	0.75
TransE	0.11	**0.18**	**0.87**	0.96	**0.90**
DistMult	0.11	0.16	0.85	**1.00**	0.82
RESCAL	0.09	0.16	0.84	0.94	0.80
RotatE	0.09	0.14	0.76	0.98	0.82

the SNAP framework[3] (Leskovec and Sosič, 2016), while for the other KGEs, we relied on the AWS DGL-KGE library[4] (Zheng et al., 2020). For all the models the dimension of the computed vectors was 100.

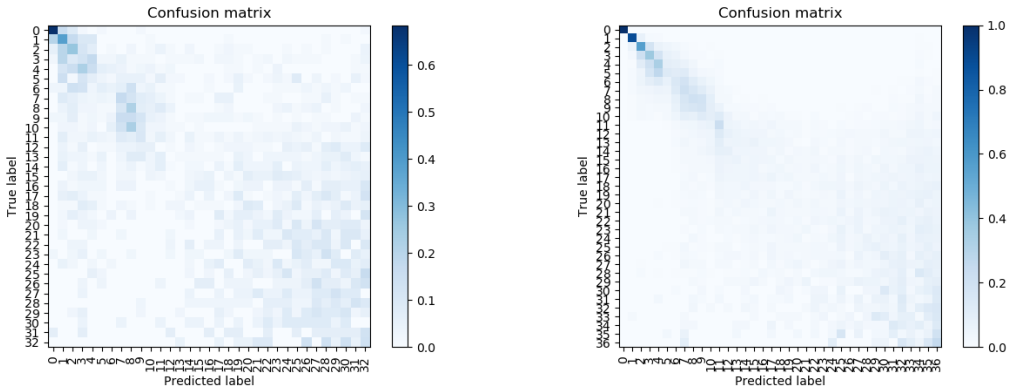
7.4.4 Results and Discussion

7.4.4.1 Evaluation on YAGO3

Table 7.2 reports the results in terms of weighted F1-score obtained for the GEs computed from YAGO3 on all the probed properties described in Section 7.4.1. The GE models with the highest F1-score are highlighted in bold. Overall, Table 7.2 shows that all the GE models present similar results w.r.t the captured structural information, with a maximum difference of 9% among them, while major differences can be noticed when analyzing the semantic properties, e.g. with values ranging from 0.6 to 0.9 for the relation encoding. Moreover, contrary to what could be expected, all the models seem to encode much better semantic properties than structural properties. A deeper analysis of the results for each property is provided hereafter.

In-degree Table 7.2 shows that no GE model is able to capture the information about the number of incoming links for each node. Indeed, all the tested algorithms present comparable but very poor F1-scores. This unexpected result may be justified by the highly skewed distribution of this property for the nodes in the graph. Indeed, the evaluated embeddings have been computed from the subgraph described in Section 7.4.1 that contains a very large number of nodes with an in-degree of zero (i.e. without incoming links), even though the classifier has been trained with fairly balanced classes by undersampling the nodes with the most frequent degrees. By taking a closer look at the confusion matrices obtained for each algorithm, we notice that all the GE models expose a common behavior: **they are very good in identifying nodes with in-degree 0**, as it can be observed for

[3] http://snap.stanford.edu/index.html
[4] https://aws-dglke.readthedocs.io/en/latest/index.html



(a) Predicting the in-degree of a node.

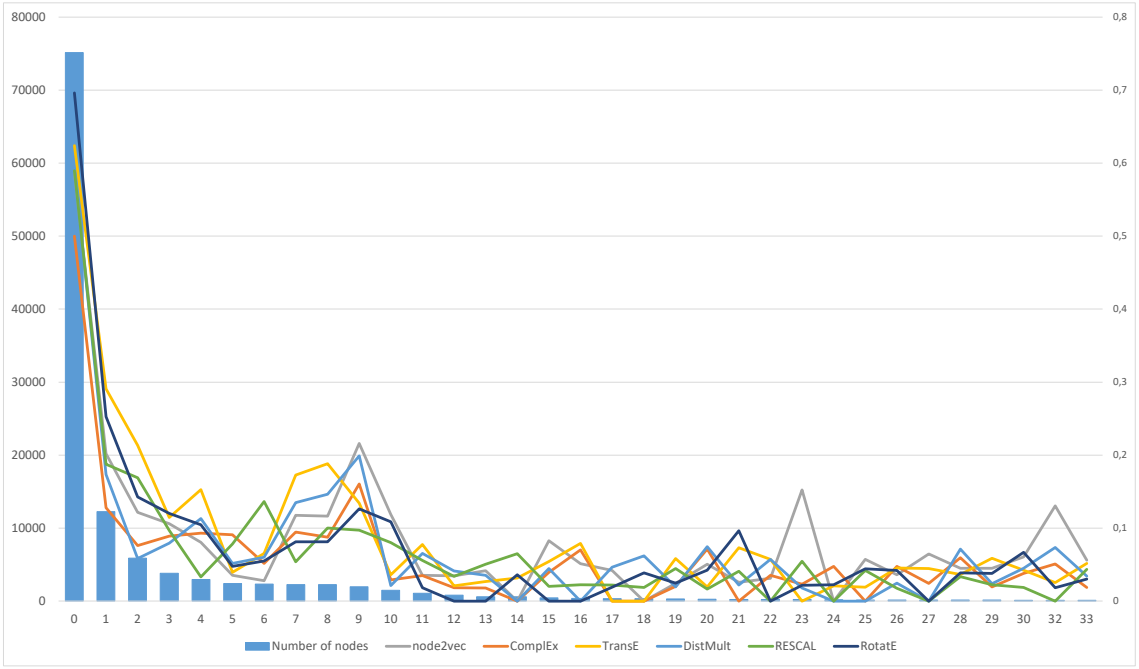
(b) Predicting the out-degree of a node.

Figure 7.2: Confusion matrices for the YAGO3 in-degree and out-degree classifiers with TransE GEs.

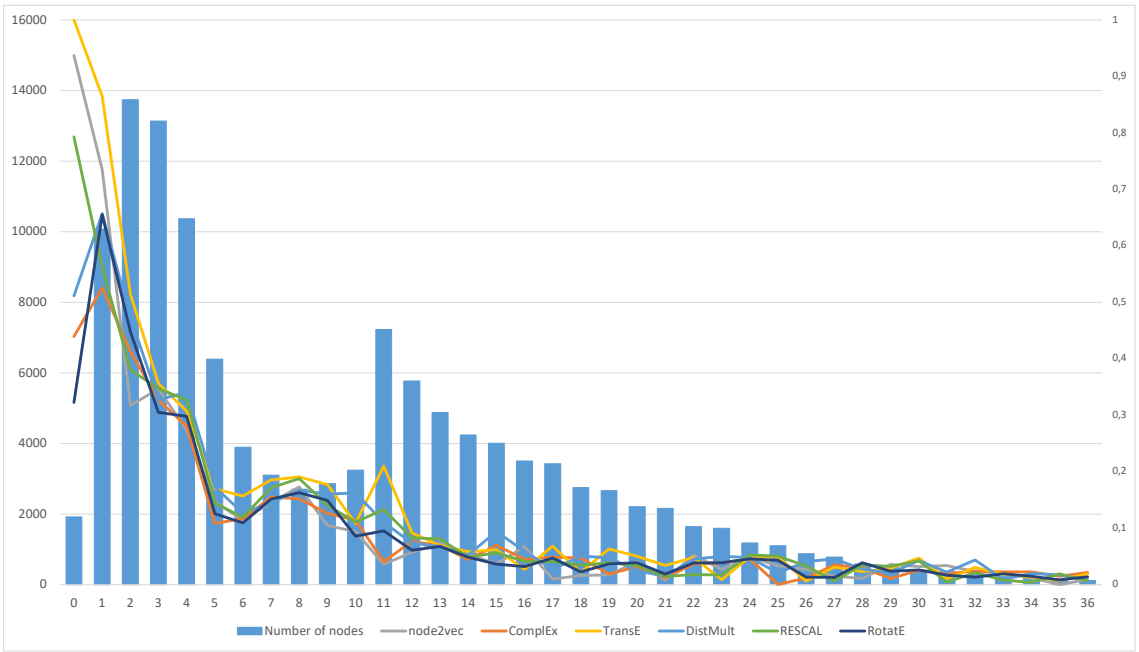
TransE in Fig. 7.2a. This last result could confirm that the information about the number of incoming links can be encoded in the vector representation for the most common in-degrees.

Out-degree As for the previous property, the figures in Table 7.2 highlight that the number of outgoing links is not encoded in the vector representation of the node by any of the GE models. Also in this case, **the different algorithms seem to behave similarly by being able to identify the nodes with low out-degrees, especially for values `0` and `1`** (see Fig. 7.2b). Nevertheless, this time, such behavior cannot be explained by the distribution of the property over the nodes in the graph. Indeed, by analyzing the structure of the graph we notice that the majority of the nodes are of out-degree 2, while the number of nodes having no outgoing links is rather small. To verify whether the distribution of the values of a property in the graph affects the capacity of a GE model to encode this property, we compare the F1-scores obtained for each class by every GE model with the distribution of the in-degree and out-degree classes. Fig. 7.3 shows that there is no clear correlation between the number of nodes in each class and the model's ability to encode the corresponding value, as all the models can identify nodes with no incoming or outgoing links. Nevertheless, we can conclude that, even though GEs do not encode the number of incoming/outgoing links per node, **they can still recognize leaves and roots nodes.** Moreover, we establish TransE to be the best model for such a task with F1-scores of 1 for out-degree and 0.63 for in-degree on class `0`, as it can be determined by looking at F1-score trends in Fig. 7.3.

Direct link Table 7.2 shows that **all the tested models are fairly good in encoding the information about the presence of a direct link between two nodes**, with F1-scores above 0.75 for every one of them. Being able to train a classifier for such property con-



(a) In-degree distribution.



(b) Out-degree distribution.

Figure 7.3: Distribution of classes and F1-scores per class for all GE models on YAGO3.

firms that embeddings of linked nodes present some common patterns that enable to determine the existence of a connection between the two nodes.

Entity type The F1-scores reported in Table 7.2 reveal the ability of GEs to encode semantic information. These results show that **all the models specifically designed for KGs are capable of identifying the correct type for the majority of nodes in the graph**, with the worst models (RESCAL and ComplEx) achieving a F1-score of 0.94. Although the performances of all these models are comparable, DistMult appeared to be the best in this task with almost perfect identification of the type for the tested entities. Surprisingly enough, also node2vec obtained fairly good results (F1 $= 0.79$) in this task, although it does not rely on any semantic information, such as the type of relation. A possible explanation for this behavior could lie in the fact that instances of the same class possibly present the same connectivity patterns (similar links, common nodes, etc.) that can be captured by the node2vec algorithm and transposed into the vector representation of such nodes. For example, several nodes representing people will always be connected to the class *Person* and very likely to a node representing their birthplace that will be linked to the class *Place*. Therefore, an instance of a person can be identified by the direct link to *Person* and the 2-hops path to *Place*.

Relation The results for this property are the most heterogeneous among the different models. **All the tested KGE algorithms, despite some differences in the F1-score, seem able to encode the relation between two entities in the nodes' embeddings**. This conclusion does not come unexpectedly. In fact, KGE models take into account the presence of different types of relations and compute a vector representation for each of them. Therefore, it is reasonable to assume that part of this information is reported in the connected nodes' embeddings as well, and, consequently, the type of relationship can be recovered by jointly studying the embeddings of two nodes. As for node2vec, similar considerations as the ones made for the entity type hold in this case. If by relying on structural information we are able to identify the type of two nodes, we will, at the same time, limit the set of possible relations existing between these nodes. For example, a *Person* and a *Place* might be linked with a limited set of predicates like *diedIn*, *livesIn* and *wasBornIn*, but not through *hasWonPrize*.

7.4.4.2 Evaluation on OntoSIDES

Structural and semantic properties As shown by Table 7.3, the results for structural and semantic properties obtained on OntoSIDES are compatible and mainly aligned with the ones achieved for YAGO3. The experiments on this KG underline the difficulty of capturing the number of incoming and outgoing links for each node. At the same time, the analysis of the confusion matrices confirms the ability of all the GE models to distinguish nodes without incoming/outgoing connections. The better results in identifying the out-degree could be due to the higher homogeneity and smaller number of classes: for YAGO3 the maximum number of outgoing links was 36, while for SIDES it is 20. Similarly to

Table 7.3: Weighted F1-scores on the different probing tasks testing structural and semantic properties for KGE computed on OntoSIDES.

GE algorithm	Structural properties			Semantic properties	
	in-degree	**out-degree**	**direct link**	**entity type**	**relation**
node2vec	**0.22**	**0.51**	**0.87**	**0.98**	**1.00**
ComplEx	0.11	0.37	0.74	0.90	0.99
TransE	0.13	0.41	0.80	0.96	**1.00**
DistMult	0.12	0.41	0.83	0.94	**1.00**
RESCAL	0.13	0.37	0.83	0.87	0.99
RotatE	0.11	0.37	0.67	0.95	**1.00**

Table 7.4: Weighted F1-scores on the different probing tasks testing context-specific properties for KGE computed on OntoSIDES.

GE algorithm	Context-specific properties	
	student's university	**question's topic**
node2vec	**1.00**	**0.69**
ComplEx	0.51	0.18
TransE	0.05	0.03
DistMult	0.34	0.10
RESCAL	0.08	0.06
RotatE	0.57	0.37

the experiments on YAGO3, all the models showed good F1-scores when encoding the presence of a direct link between two nodes.

Table 7.3 shows that the patterns discovered in YAGO3 for semantic properties also hold for OntoSIDES. All the GE models perform very well when encoding the information about entity type and relation. The global improvement in the values of F1-score is very likely due to the lower heterogeneity of the knowledge represented in this graph. In such a specific context, i.e. higher education for medical studies in France, the information to be described is sensibly less diverse (e.g. smaller number of classes and properties to be encoded) and more clearly separated (e.g. a student and a question have less in common than a singer and an actor). It is worth pointing out that node2vec seems to encode much more meaningful information than in the previous case. This might be explained by the higher importance of the graph structure in such a simple context. For example, answers can be identified only relying on their structural role as nodes: an answer is always a node linking a single student to a single question.

Context-specific properties On the one hand, Table 7.4 shows that context-specific properties are on average hardly captured by GE methods specifically designed for KGs. While for a human being it is straightforward to understand that the attended university

is a relevant concept when summarising information about students, it is much more difficult for the GEs to grasp such a correlation uniquely from the graph. If no additional information (e.g. edge weights) is provided, it is not possible to recognize that, for a student node, the link towards the university is more meaningful than the connection with a given answer. The poor results in terms of F1-score shown by most of the models in Table 7.4 confirm that **this kind of knowledge presents a much higher level of abstraction (complexity) than mere semantic and, therefore, cannot be easily interpreted by such techniques**. Possibly considering longer embeddings could facilitate the incorporation of this kind of knowledge.

On the other hand, we can observe that node2vec achieves good performance also in this task. This could be motivated by the fact that this algorithm strongly relies on the structure of the graph by creating fixed-length paths starting in each node, hence it can recognize the more important role of universities' nodes under a structural point of view. Universities nodes behave as hubs for students, i.e. several students are directly linked to a single university, therefore all the embeddings of students attending the same university will share similar patterns as they are connected in the graph through the university node. The same considerations hold for questions and their corresponding topics, with the difference that encoding topics information is a more challenging task as questions can be linked to several topics and the number of different topics is much larger than the number of universities (more than 500 topics vs. 33 universities). This result highlights the crucial role of the graph structure for the meaningfulness of the final GEs and suggests that careful ontology modeling choices are of the uttermost importance. It is interesting to point out that RotatE and ComplEx perform much better on this task when compared to the other KGE methods. This could be related to their ability to represent entities through Complex numbers, but further experiments would be needed to confirm this intuition.

7.5 Decoding Graph Embeddings for Students' Outcomes Prediction

So far, we have introduced our approach for decoding the knowledge captured by Graph Embeddings relying on probing tasks in Section 7.3 and showed how such an approach can be applied to benchmark and compare the information borne by different Knowledge Graph Embeddings in Section 7.4.1.

The analysis of the behavior of our DeepFM model with different Graph Embeddings, carried out in Chapter 5, led us to the conclusion that the quality of the students' outcome prediction was improved when using answers' Graph Embeddings computed through node2vec and TransE because they encode the information about the number of selected proposals of answers. In the following, we illustrate the use of probing tasks for verifying this hypothesis. We define, therefore, a new classification task whose aim is to predict the number of selected proposals for each answer uniquely relying on the Graph Embeddings of such answers' nodes. Table 7.5 shows the results, in terms of F1-score, obtained on this probing task by all the Graph Embeddings computed and employed for

GE model	node2vec	RDF2vec	RESCAL	DistMult	ComplEx	TransE	TransR
F1-score	**0.94**	0.71	0.33	0.39	0.33	**0.92**	0.31

Table 7.5: F1-score obtained by all the tested Graph Embedding model on probing task for the encoding of the number of selected proposals of answers.

the prediction of students' outcomes in Section 5.3. The results confirm that **the number of selected proposals of answer is captured and encoded in the answers' embeddings computed through node2vec (using the directed graph) and TransE**, as these models achieve very high F1-scores (0.94 and 0.92 respectively). As expected, RESCAL, DistMult, ComplEx and TransR exhibit very low values of F1-score on this task (around 0.30), validating our claim according to which they do not encode such a property. Surprisingly enough, the F1-score achieved by RDF2vec is significantly higher that the other models that bring no contribution to the DeepFM prediction. This could mean that this property is *partially encoded* by RDF2vec but it cannot be exploited by DeepFM. In general, we can affirm that our hypothesis is validated and the DeepFM model exploits the information about the number of selected proposals, inferred through the Graph Embeddings of answers' nodes, to predict the correctness of a given answer.

7.6 Conclusions

In this chapter, we presented a methodology based on probing tasks to verify whether GEs are able to encode certain properties of the graph elements and we introduced a categorization of such properties. To test our approach we proposed a first set of properties that could be encoded by GEs and we relied on them to analyze the knowledge captured by GEs on two different graphs: YAGO3 and OntoSIDES. We found out that, in our use cases, GEs can encode information about characteristics of graph elements at any level of abstraction: structural, semantic, and context-specific. We have shown that, even with some differences, all the tested GE models are able to capture the type of a given node, the presence of a direct link between two nodes, and the type of relationship between two nodes. Moreover, we discovered that some models, e.g. TransE, are very good at identifying roots and leaves nodes, although no model can encode the in and out-degree. Additionally, we observed that context-specific properties are better captured by node2vec embeddings. We hypothesized that this is because, in many cases, properties that are considered relevant from a human perspective are represented in the graph by nodes with a common structural pattern. This conclusion highlights the importance of the graph structure and ontology design. Finally, we used the presented methodology based on probing tasks to probe the hypothesis explaining the behavior of Graph Embeddings formulated in Chapter 5. More precisely, we could confirm that, among all the Graph Embeddings employed for the students' outcomes prediction in Chapter 5, the answers' nodes embeddings computed through node2vec and TransE encoded the number of selected proposals for each answer.

CHAPTER 8

Prediction of Learners' Performance based on Link Prediction in a Knowledge Graph

In this chapter, we present a new model for Knowledge Tracing based on Knowledge Graphs, which tries to overcome the issues discovered during the analysis of our first KG-based Knowledge Tracing model. To do so, we reformulate the Knowledge Tracing task as a Link Prediction problem on a Knowledge Graph and we predict students' outcome to questions by determining the most probable link between each answer and its correct or wrong realizations. Our first experiments on three real-world datasets show that the proposed approach yields promising results comparable with state-of-the-art models. This chapter is based on the work published at the International Conference on Artificial Intelligence in Education (AIED2022) (Ettorre et al., 2022c).

8.1 Introduction

In the previous chapters of this thesis, we proposed a method to exploit Knowledge Graphs through their embeddings to predict students' outcomes to questions and we focused, afterwards, on analyzing whether, when and why Knowledge Graph Embeddings could actually be useful to improve such prediction and provide meaningful explanations for it.

This analysis helped us highlight several shortcomings in the proposed method. First and foremost, we discovered that adding Graph Embeddings as input features of our external deep predictive model, DeepFM, was beneficial for the prediction only because of data leakage. Indeed, Graph Embeddings injected into the model information about the number of selected proposals of answers, which should not be available at prediction time. Secondly, throughout the proposed method, we are not able to exploit all the information available in the Knowledge Graph, as only embeddings of few types of nodes, i.e. students, questions and answers, are employed as input features. Instead, other elements in the Knowledge Graph can bear meaningful knowledge which is not embedded in the vector representation of the selected nodes, e.g. topics involved in the questions or organization in tests. Additionally, since the input of the DeepFM model is a combination of basic and extracted features, our approach still requires an important work of features engineering, which must be carried out based on the application context and on the available information. Finally, relying on an external deep model for the prediction increases the overall complexity of the approach, as it introduces an additional element which needs to be properly tuned and trained. At the same time, it makes the explanation of the produced results difficult, since it requires not only to decode the information captured by the Graph Embeddings used as input, but also to verify whether and how such information is exploited by the predictive model.

In this chapter, we try to overcome these issues by proposing a new approach for Knowledge Tracing entirely and uniquely based on the exploitation of Knowledge Graphs through their embeddings. As a first step, we exploit Semantic Web technologies by modeling the complete learning ecosystem as an RDF Knowledge Graph. Thanks to the expressiveness of this representation, we can define and describe concepts such as students, questions' types, institutions, etc. Secondly, we translate the Knowledge Tracing task into a Link Prediction (LP) problem on the newly generated KG. Finally, by solving this problem, we identify the most probable links between the answers whose outcomes need to be predicted and their correct or wrong results, which corresponds to the solution of the binary classification task for answers' correctness. We think that reformulating the problem of predicting students' outcomes to questions in such a way allows us to take advantage of a much wider amount and variety of information about the learning environment while reusing widely-known and well-established Deep Learning methods for KGs and avoiding features engineering. Indeed, thanks to the use of such techniques, we do not need to explore the data to find possible relevant features to use as input for the model, because the factors affecting students' outcomes will be inherently identified during the embedding process. Furthermore, this new formulation for the Knowledge Tracing task does not require us to train and tune an additional external model. This will further simplify

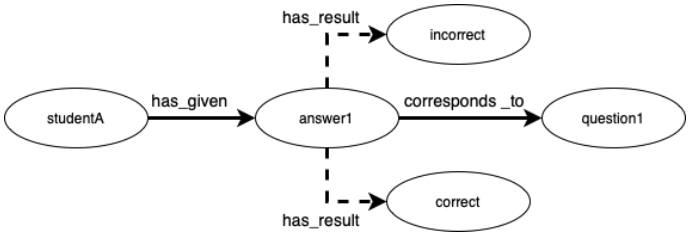


Figure 8.1: Example of a KG representing the answer of a student to a question. The dashed lines represent the links we aim to predict.

the predictions' explanation process as we can possibly rely on well-known explanation methods for Link Prediction. To the best of our knowledge, the approach we present hereafter is the first attempt to make predictions of students' outcomes uniquely based on a Knowledge Graph representing and exploiting a 360-degrees description of the learning ecosystem. To empirically confirm the validity of the proposed approach, we apply it on three real-world datasets and compare its performance with state-of-the-art Knowledge Tracing models.

The remainder of this chapter is organized as follows. In Section 8.2 we detail our approach, while in Section 8.3 we evaluate it by presenting and discussing the results of our experiments. Finally, conclusions are discussed in Section 8.4.

8.2 Link Prediction for Students' Outcomes

The approach presented in this chapter is mainly based on the hypothesis that it is possible to turn the Knowledge Tracing task into a Link Prediction problem, after modeling the learning environment as a KG. In other words, instead of predicting the probability that a given student correctly answers a specific question, we evaluate the plausibility that an implicit link (i.e. a triple) exists (in the KG) between the expected student's answer and its correct or incorrect result. Finally, an answer is labeled as "1" (correct) if the link towards the correct result has a higher score than the link towards the incorrect one, while it is labeled as "0" (incorrect) in the opposite case. For example, to predict the positive or negative outcome of the answer given by student A to question 1 (Figure 8.1), we compute the score of the two triples $\langle answer1, has\_result, correct\rangle$ and $\langle answer1, has\_result, incorrect\rangle$, and we predict that A's answer will be correct if the first triple has a higher score than the second one. To empirically validate our hypothesis, we designed and developed an end-to-end pipeline depicted in Figure 8.2, which takes as input the traces of the students' learning history possibly enriched with contextual knowledge and the list of the students-questions interactions whose outcomes must be predicted. The framework implements four steps further described below.

1. Graph Building. The first step of our approach is the creation of a KG describing the students' activity, enriched with all possible relevant side information. This step is

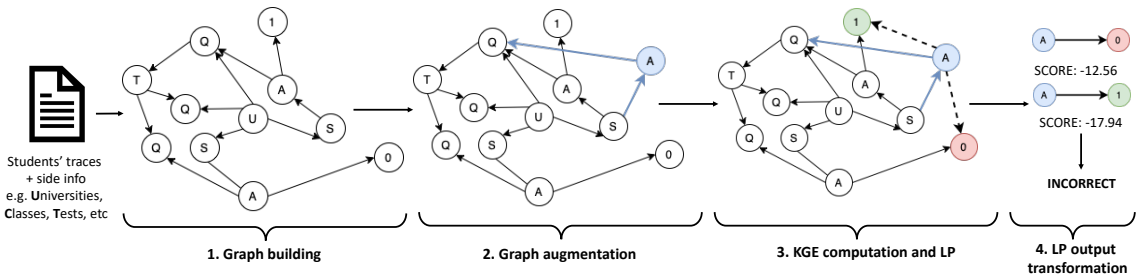


Figure 8.2: Depiction of the different steps for the proposed Link Prediction-based approach.

crucial for achieving high quality of the final prediction as the graph content and structure chosen for the representation of the learning environment will strongly influence the elements' interactions that KGE models are able to capture, and therefore the quality of the link prediction. The modeling choices are strongly dependent on the context, system, and data to be represented, i.e. a higher education learning system involves concepts and actors which are not present in a platform for vocational training. Nevertheless, a general model for the definition of the most common learning elements, such as students, questions, and answers can be provided. For example, although the datasets used for the experiments presented in this chapter describe two very different educational contexts (see Section 8.3.1), our approach achieved similar performance, when using the same KG structure for student-question interactions in the two cases.

2. Graph Augmentation for Prediction. The computation of KGEs, necessary for carrying out the Link Prediction task, is typically a non-inductive operation, meaning that, to be able to evaluate the presence of a link between two nodes, the corresponding elements must be originally present in the KG. This implies that, in order to predict the outcome of any future interaction between a student and a question, a fictitious node representing such interaction must be added to the Knowledge Graph (i.e. blue node *A* and links in Figure 8.2). This step takes as input the list of answers to be predicted and the original KG representing the previous students activities, and outputs the final graph to be used for the KGEs computation in the next step.

3. Knowledge Graph Embeddings Computation and Link Prediction. In this phase, KGEs are computed for each entity and relation in the KG. Then, for each answer to be predicted, the scores of the two triples representing a correct and a wrong realization are evaluated based on the computed embeddings. The output of this step is the list of the $2n$ possible triples, where n is the number of answers to be predicted, ranked according to their score. This phase is the most time-consuming in the whole pipeline because of the KGEs computation. Also, in the current implementation the embeddings cannot be simply updated when the KG is modified, but they need to be recomputed from scratch. Therefore, this step needs to be repeated, together with the previous and the next one, for

Table 8.1: Statistics of the datasets.

Dataset	# records	# students	# questions	# skills
SIDES DS1	354.109	1.099	2.381	414
SIDES DS2	295.529	1.098	2.493	332
ASSIST09	283.105	4.163	17.751	123

every batch of answers to predict. However, the computational overhead introduced by this operation can be reduced by predicting large batches of answers at once. Moreover, having to periodically recompute the KGEs allows us to update the KG with the most recent students' traces to keep the prediction as reliable as possible.

4. Link Prediction Output Transformation. Once the Link Prediction task is solved, we need to turn its results into the corresponding solution for the binary classification of students' answers (correct or incorrect). This process is simply carried out by identifying, for each answer, the triple with the highest score computed in the previous step, i.e. link towards correct or wrong option, and labeling the answer accordingly. This produces the final prediction of students' outcomes according to our model.

8.3 Evaluation

8.3.1 Datasets

To empirically validate our approach, we evaluated it on two different datasets extracted from the OntoSIDES KG and on an additional dataset obtained from the ASSISTments data described in Chapter 3.

For the SIDES data, we chose two subsets that contain the learning traces of the medical students in the last year of studies at the University of Lyon in a.y. 2018/2019 and 2019/2020. We limited our experiments to their activities on questions related to the medical specialties of pediatrics for the first dataset, and infectious diseases for the second one, respectively named **SIDES DS1** and **SIDES DS2**. These two specialties have been chosen because they are the ones with the highest number of answers for the selected subset of students.

To assess the generalization capabilities of the proposed approach and to be able to compare it with state-of-the-art Knowledge Tracing models, we decided to additionally test our method on the ASSISTments data, previously described in Section 3.2. More precisely, we used the ASSITments 2009-2010 skill builder dataset (Feng et al., 2009), which is one of the best known and most widely used benchmarking datasets.

The main statistics for the above-described datasets are shown in Table 8.1.

8.3.2 Knowledge Graphs

The first step to apply our approach is to model all the available information related to the learning ecosystem in the form of a KG. Hereafter, we detail how the KGs representing the data in SIDES and ASSISTment have been designed.

SIDES. As described in Section 3.1, the SIDES dataset includes information about the attempts of students to questions, but also rich additional details further describing such questions and students. Nevertheless, part of the information stored in OntoSIDES is very specific (e.g. links to external sources defining medical concepts) and the modeling choices are sometimes unsuitable for our final goal. For example, OntoSIDES includes nodes representing selected options for multiple-choice questions, that are not necessary in our use case, and it uses Literals to indicate the level of correctness of each answer, which can not be directly exploited by the employed embedding models. Therefore, we slightly modified the structure and content of the KG describing the extracted SIDES datasets to obtain a representation better suited for our approach. The KG representing our SIDES subsets is illustrated in Fig. 8.3. Concerning the questions, this new KG differs from the original one for the absence of the nodes representing the proposals of answer, while it still links questions to their medical specialties, learning objectives and sub-objectives, universities, and possible multimedia resources. Another difference is in the modeling of the students who are now linked directly to the university they attend through the property `registered_in`. The most interesting and impactful part of the modeling concerns the interactions between students and questions and their outcomes. As in the original graph, each attempt of a student to a question is represented by a node of type `answer` in the KG, which is connected to the corresponding student and question, through properties `done_by` and `correspond_to`. Additionally, we connect every answer to its outcome, i.e. correct or incorrect, which is represented by the two nodes for the correct and the incorrect options for each question. Therefore, the goal of our Link Prediction task is to find the most probable object of the triple $\langle A1, has\_selected, ?\rangle$. Let us highlight that we are considering answers as either correct or incorrect. Therefore, partially correct answers will be treated as incorrect. The described KG designing choices have been identified, through a first experimentation, as the most effective ones to optimize the quality of the students' outcomes prediction.

ASSISTment 2009-2010. We designed the KG describing the ASSISTment data similarly to the SIDES one. In this case, the side information is represented by different types of problems, i.e. either main or scaffolding problems on one side, and algebra, single choice, multiple choice or fill-in problems on the other side; organization of such problems in assistments and assignments; division of students in classes; and teachers scheduling. We defined one class for describing each one of the following concepts: answers, students, skills, teachers, assignments, assistments, and students classes; and 7 additional classes to represent the different types of problems: `MainProblem`, `ScaffoldingProblem`, `AlgebraProblem`,

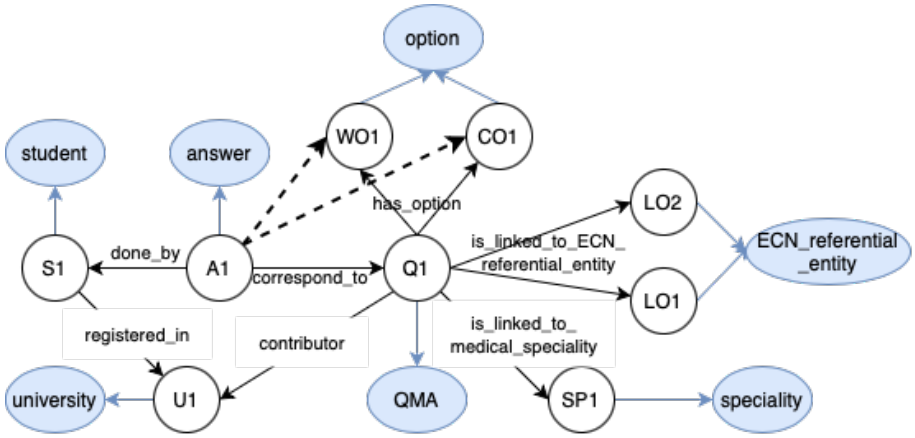


Figure 8.3: KG representing the SIDES learning environment. Blue bubbles are OWL classes, while white bubbles are instances of such classes. Blue links are property `rdf:type`. Dashed edges are property `has_selected` whose object we aim to predict.

`SingleChoiceProblem`, `MultiChoiceProblem`, `FillInProblem`, and the parent class `Problem`. We reused some of the properties previously defined for SIDES to describe relations between students, answers, problems and skills. Additionally, we linked each problem to the related assistments and those to the corresponding assignments. Each assignment is then connected both to the teacher who gave it, through property `assigned_by`, and to the students' class solving the assignment through `assigned_to`. Finally, a major difference with the KG built for SIDES lies in the modeling of the outcomes of students' attempts. Indeed, we noticed that, when linking correct and wrong outcomes to both the question and the answer by using separate properties, respectively `has_correct_option`, `has_incorrect_option`, `has_correctly_selected` and `has_incorrectly_selected`, the prediction performance was significantly better than when using only two properties that do not distinguish between correct and incorrect outcomes, i.e. `has_option` and `has_selected`. Therefore, the task we solve through Link Prediction is, in this case, the identification of the most probable triple between $\langle A1, has\_correctly\_selected, CO1\rangle$ and $\langle A1, has\_incorrectly\_selected, WO1\rangle$.

8.3.3 Knowledge Graph Embedding Models

For the resolution of the Link Prediction task for the prediction of students' outcomes, we tested several KGE models, implemented by the AWS DGL-KGE library (Zheng et al., 2020): DistMult (Yang et al., 2015), TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2017) and RotatE (Sun et al., 2019). We empirically identified TransE as the best one in recognizing the real links in all the KGs. Therefore, we chose to rely on this model for the following experiments. All the KGEs employed for the experiments presented in this chapter have been computed using hidden dimension 100 and 100.000 max iterations.

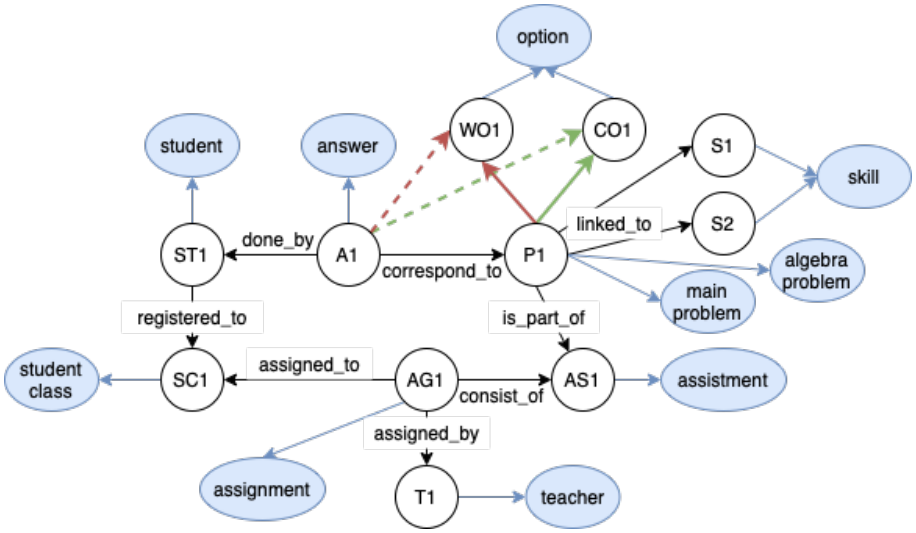


Figure 8.4: KG representing the ASSISTment learning environment. Blue bubbles are OWL classes, while white bubbles are instances of such classes. Blue links are property `rdf:type`. The solid green and red edges represent respectively properties `has_correct_option` and `has_incorrect_option`, while the dashed green and red ones indicate the two properties `has_correctly_selected` and `has_incorrectly_selected` to be predicted.

8.3.4 Evaluation Setup

To validate our approach, we decided to compare the performance of our model with the original DKT model. This choice is motivated by the fact that DKT has been widely used as a baseline by recent research works in the area and that several implementations are available online. For the experiments presented in Section 8.3.5, we relied on the DKT TensorFlow implementation[1] provided by (Xiong et al., 2016). For the evaluation, we split the three datasets into a train set, including the learning traces of 70% of the students, and a test set, containing the remaining 30%. We used exactly the same splits for training and testing the two approaches. For the experiments with DKT, we had to perform an additional preprocessing step. Since DKT makes predictions relying on the input couples $(student, skill)$, when a student answers a question tagged with multiple skills, the corresponding record would be duplicated as many times as the number of involved skills. To avoid this issue, we created fictitious joint skills for the questions associated with multiple skills as in the solution proposed by (Xiong et al., 2016).

Finally, to evaluate the two approaches we used several metrics commonly employed in classification problems: AUC, F1 scores, and balanced Accuracy (Brodersen et al., 2010) which is invariant w.r.t. the distribution of the target values in the dataset. In order to properly compute the AUC for the Link Prediction task, we had to convert the scores of each link into a probability estimator for the correctness of the corresponding answer,

[1] https://github.com/siyuanzhao/2016-EDM

Table 8.2: Results of Link Prediction and DKT approach.

Dataset	Link Prediction				DKT			
	AUC	F1 (0)	F1 (1)	bACC	AUC	F1 (0)	F1 (1)	bACC
SIDES DS1	**0.759**	0.724	0.656	**0.692**	0.706	0.690	0.609	0.650
SIDES DS2	**0.759**	0.691	0.683	**0.687**	0.680	0.646	0.607	0.627
ASSIST09	**0.717**	0.544	0.734	**0.657**	0.691	0.512	0.755	0.640

valued between 0 and 1. To do so, we normalized the scores of each complementary couple of triples (indicating the correct and incorrect outcome of the same student's attempt) by dividing the score of the link towards the correct outcome by the sum of the scores of the two possible links (correct and incorrect outcomes). Equation (8.1) shows the formula used for the probability computation, where p_c indicates the probability of correctness of each answer, S_I is the score of the link towards the incorrect outcome, while S_C is the score of the link towards the correct outcome. Let us highlight that we consider, as estimator for the probability, the complement of the normalized score for the correct answer because, being the scores negative, the highest score, and, thus, more probable link is the one with the lowest normalized score.

$$p_c = 1 - \frac{S_C}{S_I + S_C} \tag{8.1}$$

Moreover, to be able to fairly compare the results obtained by Link Prediction and DKT over the other selected metrics, i.e. F1-scores and bACC, we chose, for the DKT method, the threshold that maximizes the balanced accuracy.

8.3.5 Results and Discussion

Table 8.2 shows the results in terms of AUC, F1-scores and balanced Accuracy of both Link Prediction and DKT on the three datasets described in Section 8.3.1[2].

As it can be seen from Table 8.2, the newly proposed Link Prediction approach consistently outperforms DKT on almost all the metrics for the three datasets. In particular, the Link Prediction approach achieves an improvement of 4-6% in terms of bACC on the two SIDES datasets, while it does almost 2% better on the ASSISTment dataset. Concerning AUC, the improvement is slightly more important as Link Prediction outperforms DKT by 6-8% on the SIDES datasets and by 2.5% on ASSISTment. We believe that the main reason for this improvement is the ability of our method to take into account a greater variety of knowledge about the learning ecosystem. Indeed, we are able to exploit, for the students' outcome prediction, relations between skills, students, tests, questions, and institutions.

It is important to also note that, while the results of the Link Prediction approach are very close for the two SIDES datasets ($bACC = 0.692$ and 0.687) they are importantly

[2]The code and the data for reproducing the following experiments can be found at https://github.com/antoniaettorre/kt_link_prediction.

lower for the ASSISTment09 dataset, $bACC = 0.657$. This behavior can be explained by the different characteristics of the datasets. Taking a closer look at the statistics in Section 8.3.1, we can notice that the number of students and questions included in the SIDES datasets are sensibly lower than the ones in ASSISTment data. This is a direct consequence of the context described by these data. The SIDES datasets include traces of two small groups of students over 6 years of highly specialized training, therefore all the students must face a very specific, although relatively small, set of questions. While the ASSISTment dataset reports the training of many students on a wider and more general set of questions. This difference results in a higher density of answers per question in the SIDES datasets (148 answers/question and 118 answer/question respectively for DS1 and DS2), when compared to ASSISTment (16 answers/question), which directly corresponds to a larger number of links towards the positive and negative outcomes of questions in the KG. Therefore, it is reasonable to think that the Link Prediction algorithm performs better on the SIDES datasets because the corresponding KGs include more examples of links similar to the ones we aim to predict. Another important difference between the SIDES and the ASSISTment data lies in the balance of the dataset with respect to the distribution of the target values. Indeed, while the SIDES datasets resulted to be almost balanced in the number of correct and incorrect answers (respectively around 47% and 49% of correct answers for DS1 and DS2), the ASSISTment dataset contains almost two times more correct answers than incorrect ones (around 65% of correct answers). This explains the strong difference between the negative and positive F1 scores achieved by both DKT and our model based on Link Prediction (see Table 8.2).

A last interesting remark regards the execution time since, as explained in Section 8.2, our method is based on 4 steps and could present a bottleneck in the embedding computation phase. Nevertheless, we observed that, with the same evaluation setup and similar hardware settings, both DKT and Link Prediction methods exhibit comparable total execution time (around 30 minutes on a machine equipped with 4 Nvidia Quadro RTX 8000 GPU cards).

8.4 Conclusions

In this chapter, we presented a novel approach to predict students' outcomes to questions uniquely and entirely relying on the use of Knowledge Graphs and Knowledge Graph Embeddings. More specifically, we reformulated the Knowledge Tracing task as a Link Prediction problem on a KG describing such system. We developed an end-to-end framework implementing all the steps to execute our approach: from KG building to the final prediction. We tested our approach on two very different use cases and compared its performance with the state-of-the-art DKT Knowledge Tracing model. We showed that Link Prediction achieves competitive results in terms of prediction accuracy, but also that its performance is partially dependent on the characteristics of the dataset: the more examples of answers per question are present in the data, the better the quality of the prediction will be. In conclusion, we proposed a method to predict students' outcomes that allows

us to account for all the possible knowledge about a learning environment, even if highly complex, while avoiding features engineering. Moreover, thanks to the KG representation, the Link Prediction based approach offers a homogeneous interface independent from a specific educational system.

CHAPTER 9

Conclusions and Perspectives

9.1 Summary of the contributions

The work done during this thesis is a first effort at investigating whether and how Symbolic AI, in the form of Knowledge Representation and Reasoning, and, more precisely, Knowledge Graphs can help optimize the resolution of the Knowledge Tracing task, overcoming the limitations of state-of-the-art Deep Learning approaches. These limitations, already illustrated in Chapter 1, include, as two major shortcomings, the impossibility of modeling and exploiting complete information about learning environments, which can be highly complex and very diverse depending on the described educational context, and the difficulty of providing meaningful explanations for the generated Knowledge Tracing predictions.

In our fist contribution, we tried to overcome these issues by proposing a new model for tracing students' learning based on the well-known DeepFM architecture, extended to include, as input features, Graph Embeddings computed from a Knowledge Graph describing the learning environment. In this contribution, we proved that Knowledge Graph Embeddings can capture meaningful information from the Knowledge Graph representing the learning environment which can subsequently be exploited by DeepFM to provide better predictions for students' outcomes to questions. Through further extensive experimentation, we realized that our approach led to the problem of data leakage. Nevertheless, the identification and resolution of such issue brought us to the realization of two important contributions useful for the general interpretation and understanding of Knowledge Graph Embeddings.

Firstly, we developed *Stunning Doodle*, a tool for the visual analysis of Knowledge Graph Embeddings based on the joint visualization of the Graph Embeddings and the Knowledge Graph from which they are generated. The idea is that by visualizing, at the same time, the structure and the semantic of the Knowledge Graph, and the distances between the graph elements in the embedding space, we can help the user get insights into the information encoded by the embeddings. Secondly, inspired by the work in the NLP community, we proposed an approach to systematically analyze Knowledge Graph Embeddings and decode the information they capture from the Knowledge Graph. This approach consists of testing hypothesis on the encoding of specific features by the latent

representation of each graph element, through the use of classification tasks, known as probing tasks. Both, the developed visualization tool and the probing task methodology, helped us analyze different Graph Embeddings and identify the data leakage issue in our Knowledge Tracing approach, allowing us to confirm the hypothesis on the information encoded by the embeddings, and, thus, better understand their behavior. Furthermore, these two contributions have a more general aim, as they can be exploited by the global Semantic Web community to gain a better understanding of the information captured by Graph Embeddings in any application, which can lead to a more aware use of such embedding models.

Finally, we presented a Knowledge Tracing method which relies entirely and uniquely on the exploitation of Knowledge Graphs and their embeddings. This method consists of treating the Knowledge Tracing task as a Link Prediction problem on a Knowledge Graph representing the whole learning environment. Such formulation allows us to represent and exploit all the available information about any learning environment while keeping a uniform interface for the students' outcome prediction. Indeed, no feature engineering or any adaptation of the Link Prediction based method is necessary to apply it to a new Knowledge Graph in a different educational context.

This last contribution represents a good first lead for the exploitation of Knowledge Graphs for the Knowledge Tracing task resolution, as, even in its basic settings, it shows good prediction performance, while accounting for the complexity and variety of any learning environment and application context. For example, concepts such as soft and hard skills can be defined in a Knowledge Graph, hierarchical organization of questions, topics and skills can also be described, as well as students personal traits and information about their learning history and habits. And such complex knowledge can be further exploited through the Knowledge Graph Embeddings. Additionally, it is possible to easily obtain a first explanation of the generated predictions by reusing explanatory methods for Link Prediction.

In conclusion, with respect to the research questions formulated at the beginning of this thesis (Section 1.2), we provided the following answers:

- **RQ1: How can we exploit Knowledge Representation for Knowledge Tracing purposes?** We explored, analyzed and tested the use of Graph Embedding techniques to take advantage of the knowledge represented in a Knowledge Graph. We proposed two approaches relying on the use of such embedding models to predict students' outcome to questions in different educational contexts. Additionally, we showed how such Knowledge Graph based approaches allow us to represent and exploit all the possible information about any learning system providing a simple and homogeneous interface.

- **RQ2: Under what conditions and in which context does our Knowledge Graph based approach to Knowledge Tracing work? Can it be generalized across different learning environments and to other Knowledge Graphs?** Through extensive experimentation, we discovered which Knowledge Graph Embedding settings and Knowledge Graph content and structure were optimal for the resolution of the

Knowledge Tracing task in the OntoSIDES use case and we formulated hypothesis on the reasons behind their behavior. At the same time, we found out that several factors can heavily affect the quality of Knowledge Graph Embeddings and, therefore, impact the performance on the resolution of the final task. Moreover, to optimize the quality of the students' outcome prediction, KG-based approaches require extreme care in the design of Knowledge Graphs describing diverse learning environments. Indeed, although general guidelines can be provided for the modeling of such Knowledge Graphs, design choices need to be fine-tuned based on the represented educational context.

- **RQ3: How to interpret and explain the prediction produced by our Knowledge Graph based model?** A major part of the work carried out for this thesis was dedicated to the analysis and interpretation of the information encoded by Graph Embeddings. To this end, we first developed *Stunning Doodle*, a tool for the visual analysis of Knowledge Graph Embeddings, and subsequently we proposed an approach to verify hypothesis on the the properties encoded by Graph Embeddings.

9.2 Future works and Perspectives

In this thesis, we took a significant first step towards the exploitation of Knowledge Graph Embeddings for supporting e-education, and pushed further the interpretation of the information encoded by such embeddings. Nevertheless, all the proposed solutions present several limitations and much work is still to be done for their improvement and extension to be able to fully exploit them.

Concerning the visual analysis of Knowledge Graph Embeddings, *Stunning Doodle* could be enhanced and extended to include better and more comprehensive visualization options and advanced functionalities for the analysis of Graph Embeddings. For example, we could implement the possibility of visualizing a Knowledge Graph by querying a SPARQL endpoint, and we could provide high-quality statistics on the neighboring nodes in the embedding space, e.g. identifying commonalities among close nodes. Moreover, one major weakness of *Stunning Doodle* is its performance on large Knowledge Graphs, thus the optimization of the back-end pipeline is necessary to more easily and seamlessly use this tool in real-world scenarios.

Also our approach for the prediction of students' outcomes to questions based on Link prediction needs to be further analyzed, improved and extended. Firstly, our experiments on two different use cases highlighted how much the Knowledge Graph designing choices impact the quality of the link prediction. Indeed, we saw that best Knowledge Graph structure found for the SIDES scenario was not optimal for the students' performance prediction on the ASSISTments data. Therefore, for a more aware utilization of our Link Prediction based approach, it is necessary to further analyze these differences, and try to provide guidelines for the representation of learning environments with diverse characteristics and in different contexts. Moreover, we pointed out the possibility of applying explanatory methods for Link Prediction to interpret the solutions provided by our

model, but we did not test such methods and analyze whether the generated explanations would actually be meaningful. This would surely be an essential study to be carried out in the close future. Furthermore, one of the most important limitations of the proposed approach is the impossibility of taking into consideration the time dimension and, therefore, account for the temporal evolution of students' knowledge. Indeed, even though the outcomes prediction is done based on the previous activities of the students, such activities are not temporally ordered. Thus, an important research direction for the future is the introduction in the Knowledge Graph and exploitation, through the embeddings, of temporal information.

Finally, to confirm the results presented in this thesis and provide a deeper analysis and further meaningful insights into the representation of diverse learning environments, we plan to extend our experimentation to a different educational domain. In the context of an ongoing collaboration with the TeachOnMars [1] company, we are currently working on the implementation and test of the proposed models on data obtained from professional training of employees in various activity sectors.

In conclusion, we showed how Knowledge Graphs and Knowledge Graph Embeddings can be used and useful in the field of e-education for the resolution of the Knowledge Tracing task. We can move forward in this direction trying to exploit these technologies for other tasks to support global education, both online and offline. For example, we can think about employing Knowledge Graphs for extending the scope of the learning environment description, representing additional information such as social interactions among students and cognitive or emotional characteristics of learners. We can also use Knowledge Graphs to relate different educational resources among several learning systems. Lastly, we can exploit all this knowledge for tasks such as recommendation of educational resources, learning partners and careers but also to enable smarter automatic correction systems and provide more fine-grained statistics and advanced learning analytics to teachers.

[1] https://www.teachonmars.com/fr/

Bibliography

Ghodai Abdelrahman and Qing Wang. Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 175–184, 2019.

Ghodai Abdelrahman, Qing Wang, and Bernardo Pereira Nunes. Knowledge Tracing: A Survey. *arXiv preprint arXiv:2201.06953*, 2022.

Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. 2016.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 1638–1649, 2018.

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010, 2020.

Prithviraj Ammanabrolu and Mark O Riedl. Transfer in deep reinforcement learning using knowledge graphs. *arXiv preprint arXiv:1908.06556*, 2019.

Francesco Antoniazzi and Fabio Viola. RDF Graph Visualization Tools: a Survey. 11 2018. doi: 10.23919/FRUCT.2018.8588069.

Luigi Asprino, Christian Colonna, Misael Mongiovì, Margherita Porena, and Valentina Presutti. Pattern-based Visualization of Knowledge Graphs. *arXiv preprint arXiv:2106.12857*, 2021.

Mark A Barton and Frederic M Lord. An upper asymptote for the three-parameter logistic item-response model. *ETS Research Report Series*, 1981(1):i–8, 1981.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

Rajarshi Bhowmik and Gerard de Melo. Explainable link prediction for emerging entities in knowledge graphs. In *International Semantic Web Conference*, pages 39–55. Springer, 2020.

Nikos Bikakis, John Liagouris, Maria Kromida, George Papastefanatos, and Timos Sellis. Towards scalable visual exploration of very large RDF graphs. In *European Semantic Web Conference*, pages 9–13. Springer, 2015.

Nikos Bikakis, John Liagouris, Maria Krommyda, George Papastefanatos, and Timos Sellis. GraphVizdb: A scalable platform for interactive large graph visualization. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1342–1345. IEEE, 2016.

Allan Birnbaum. Statistical theory for logistic mental test models with a prior distribution of ability. *Journal of Mathematical Psychology*, 6(2):258–276, 1969.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Journal of web semantics*, 7(3):154–165, 2009.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global, 2011.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, December 2017. ISSN 2307-387X.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26:2787–2795, 2013.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The Balanced Accuracy and Its Posterior Distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010.

Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. LodLive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 197–200, 2012.

Giovanna Castellano, Vincenzo Digeno, Giovanni Sansaro, and Gennaro Vessio. Leveraging Knowledge Graphs and Deep Learning for automatic art analysis. *Knowledge-Based Systems*, 248:108859, 2022.

Hao Cen, Kenneth Koedinger, and Brian Junker. Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement. In *Intelligent Tutoring Systems*, pages 164–175, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-35160-3.

Hao Cen, Kenneth R Koedinger, and Brian Junker. Is Over Practice Necessary?-Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. *Frontiers in artificial intelligence and applications*, 158:511, 2007.

Lijia Chen, Pingping Chen, and Zhijian Lin. Artificial Intelligence in Education: A review. *IEEE Access*, 8:75264–75278, 2020.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, Dec 1994. ISSN 1573-1391. doi: 10.1007/BF01099821.

Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. Location Embeddings for Next Trip Recommendation. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 896–903, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366755.

Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050750.

Gerben KD de Vries. A fast approximation of the Weisfeiler-Lehman graph kernel for RDF data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 606–621. Springer, 2013.

Federico Desimoni and Laura Po. Empirical evaluation of Linked Data visualization tools. *Future Generation Computer Systems*, 112:258–282, 2020.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

Natalia Díaz-Rodríguez, Alberto Lamas, Jules Sanchez, Gianni Franchi, Ivan Donadello, Siham Tabik, David Filliat, Policarpo Cruz, Rosana Montes, and Francisco Herrera. EXplainable Neural-Symbolic Learning (X-NeSyL) methodology to fuse deep learning representations with expert knowledge graphs: The MonuMAI cultural heritage use case. *Information Fusion*, 79:58–83, 2022.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2, 2016.

Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs. International Joint Conferences on Artificial Intelligence, 2017.

Mingyu Feng, Neil Heffernan, and Kenneth Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009.

Giuseppe Futia and Antonio Vetrò. On the integration of knowledge graphs into deep learning models for a more comprehensible AI—Three challenges for future research. *Information*, 11(2):122, 2020.

Manas Gaur, Keyur Faldu, and Amit Sheth. Semantics of the black-box: Can knowledge graphs help make deep learning systems more interpretable and explainable? *IEEE Internet Computing*, 25(1):51–59, 2021.

Theophile Gervet, Ken Koedinger, Jeff Schneider, Tom Mitchell, et al. When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining*, 12(3):31–54, 2020.

Aritra Ghosh, Neil Heffernan, and Andrew S Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2330–2339, 2020.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

Harold Gulliksen. Theory of mental tests. 1950.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1725–1731, Melbourne, Australia, August 2017. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/239.

R.K. Hambleton, H. Swaminathan, and H.J. Rogers. *Fundamentals of Item Response Theory*. Measurement Methods for the Social Science. SAGE Publications, 1991. ISBN 9780803936478.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Fabian Hoppe, Danilo Dessì, and Harald Sack. Deep learning meets knowledge graphs for scholarly data classification. In *Companion proceedings of the web conference 2021*, pages 417–421, 2021.

Lan Huang, Congcong Yu, Yang Chi, Xiaohui Qi, and Hao Xu. Towards smart healthcare management based on knowledge graph technology. In *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, pages 330–337, 2019.

Nitisha Jain, Jan-Christoph Kalo, Wolf-Tilo Balke, and Ralf Krestel. Do Embeddings Actually Capture Knowledge Graph Semantics? In *European Semantic Web Conference*, pages 143–159. Springer, 2021.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1715–1724, 2016a.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Encoding temporal information for time-aware link prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2354, 2016b.

Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Explaine: An approach for explaining network embedding-based link predictions. 2019.

Tanja Käser, Severin Klingler, Alexander G Schwing, and Markus Gross. Dynamic Bayesian networks for student modeling. *IEEE Transactions on Learning Technologies*, 10(4):450–462, 2017.

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.

Kevin H. Wilson and Yan Karklin and Bojian Han and Chaitanya Ekanadham. Back to the Basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, Raleigh, NC, USA, 2016. Association for Computational Linguistics.

Mohammad Khajah, Rowan Wing, Robert Lindsey, and Michael Mozer. Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. In *Educational Data Mining 2014*, 2014.

Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

Kenneth R Koedinger, Julie L Booth, and David Klahr. Instructional complexity and the science to constrain it. *Science*, 342(6161):935–937, 2013.

Freddy Lecue. On the role of knowledge graphs in explainable AI. *Semantic Web*, 11 (1):41–51, 2020.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015. doi: 10.1038/nature14539.

Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1576–1585, 2018.

Jung In Lee and Emma Brunskill. The Impact on Individualizing Student Models on Necessary Practice Opportunities. *International Educational Data Mining Society*, 2012.

Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1): 1–20, 2016.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

Qi Liu, Shuanghong Shen, Zhenya Huang, Enhong Chen, and Yonghe Zheng. A survey of knowledge tracing. *arXiv preprint arXiv:2105.15106*, 2021.

Yang Liu, Qingguo Zeng, Joaquín Ordieres Meré, and Huanrui Yang. Anticipating stock market of the renowned companies: a knowledge graph approach. *Complexity*, 2019, 2019.

FM Lord, MR Novick, and Allan Birnbaum. Statistical theories of mental test scores. 1968.

Rose Luckin, Wayne Holmes, Mark Griffiths, and Laurie B Forcier. Intelligence unleashed: An argument for AI in education. 2016.

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.

Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

András Micsik, Sándor Turbucz, and Attila Györök. Lodmilla: a linked data browser for all. 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Koki Nagatani, Qian Zhang, Masahiro Sato, Yan-Ying Chen, Francine Chen, and Tomoko Ohkuma. Augmenting knowledge tracing by considering forgetting behavior. In *The world wide web conference*, pages 3101–3107, 2019.

Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 156–163. IEEE, 2019.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. pages 809–816, 01 2011.

Melvin R. Novick. The axioms and principal results of classical test theory. *Journal of Mathematical Psychology*, 3(1):1 – 18, 1966. ISSN 0022-2496. doi: 10.1016/0022-2496(66)90002-2.

Andrea Giovanni Nuzzolese, Valentina Presutti, Aldo Gangemi, Silvio Peroni, and Paolo Ciancarini. Aemoo: Linked data exploration based on knowledge patterns. *Semantic Web*, 8(1):87–112, 2017.

Matteo Palmonari and Pasquale Minervini. Knowledge graph embeddings and explainable AI. *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, 47:49, 2020.

Olivier Palombi, Fabrice Jouanot, Nafissetou Nziengam, Behrooz Omidvar-Tehrani, Marie-Christine Rousset, and Adam Sanchez. OntoSIDES: Ontology-based student progress monitoring on the national evaluation system of French Medical Schools. *Artificial Intelligence in Medicine*, 96:59–67, 2019.

Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. Knowledge Graph Embeddings with node2vec for Item Recommendation. In *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, pages 117–120, 2018. doi: 10.1007/978-3-319-98192-5_22. URL https://doi.org/10.1007/978-3-319-98192-5_22.

Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

Shalini Pandey and Jaideep Srivastava. RKT: relation-aware self-attention for knowledge tracing. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1205–1214, 2020.

Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

Philip I. Pavlik, Hao Cen, and Kenneth R. Koedinger. Performance Factors Analysis –A New Alternative to Knowledge Tracing. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*, pages 531–538, Amsterdam, The Netherlands, 2009. IOS Press. ISBN 978-1-60750-028-5.

Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. Investigating robustness and interpretability of link prediction via adversarial modifications. 2019.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep Knowledge Tracing. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 505–513. Curran Associates, Inc., 2015.

Laura Po and Davide Malvezzi. High-level Visualization Over Big Linked Data. In *International Semantic Web Conference*, 2018.

G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Studies in mathematical psychology. Danmarks Paedagogiske Institut, 1960. ISBN 9780598554512.

Mark D. Reckase. The Past and Future of Multidimensional Item Response Theory. *Applied Psychological Measurement*, 21(1):25–36, 1997. doi: 10.1177/0146621697211002.

Steffen Rendle. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4256-0. doi: 10.1109/ICDM.2010.127.

Petar Ristoski and Heiko Paulheim. Rdf2vec: RDF graph embeddings for data mining. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, pages 498–514, 2016. doi: 10.1007/978-3-319-46523-4_30. URL https://doi.org/10.1007/978-3-319-46523-4_30.

Ido Roll and Ruth Wylie. Evolution and revolution in artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 26(2):582–599, 2016.

Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Trans. Knowl. Discov. Data*, 15(2), January 2021. ISSN 1556-4681.

Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. Learning a health knowledge graph from electronic medical records. *Scientific reports*, 7(1):1–11, 2017.

Tong Ruan, Lijuan Xue, Haofen Wang, Fanghuai Hu, Liang Zhao, and Jun Ding. Building and exploring an enterprise knowledge graph for investment analysis. In *International semantic web conference*, pages 418–436. Springer, 2016.

Idafen Santana-Pérez. Graphless: Using Statistical Analysis and Heuristics for Visualizing Large Datasets. *VOILA@ ISWC*, 2187:1–12, 2018.

Guus Schreiber, Yves Raimond, Frank Manola, Eric Miller, and Brian McBride. RDF 1.1 Primer, W3C Working Group Note. *World Wide Web Consortium (W3C)*, 2014.

Aditya Sharma, Partha Talukdar, et al. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 122–131, 2018.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Shashank Sonkar, Andrew E Waters, Andrew S Lan, Phillip J Grimaldi, and Richard G Baraniuk. qdkt: Question-centric deep knowledge tracing. 2020.

Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris Ding, Si Wei, and Guoping Hu. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. 2019.

Ilaria Tiddi and Stefan Schlobach. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence*, 302:103627, 2022.

Shiwei Tong, Qi Liu, Wei Huang, Zhenya Hunag, Enhong Chen, Chuanren Liu, Haiping Ma, and Shijin Wang. Structure-based knowledge tracing: an influence propagation view. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 541–550. IEEE, 2020.

Bernie Trilling and Charles Fadel. *21st century skills: Learning for life in our times*. John Wiley & Sons, 2009.

Théo Trouillon, Christopher R. Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge Graph Completion via Complex Tensor Factorization. *The Journal of Machine Learning Research*, abs/1702.06879, 2017.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction, 2016.

Georgia Troullinou, Haridimos Kondylakis, Kostas Stefanidis, and Dimitris Plexousakis. RDFDigest+: A Summary-driven System for KBs Exploration. In *International Semantic Web Conference (P&D/Industry/BlueSky)*, 2018.

Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist*, 46(4):197–221, 2011.

Jill-Jênn Vie. Deep Factorization Machines for Knowledge Tracing. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, Louisiana (USA), 2018.

Jill-Jênn Vie and Hisashi Kashima. Knowledge Tracing Machines: Factorization Machines for Knowledge Tracing. In *Proceedings of the 33th AAAI Conference on Artificial Intelligence (AAAI-19)*, Honolulu, Hawai (USA), 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.

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, 2014a.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601, 2014b.

Zhiwei Wang, Xiaoqin Feng, Jiliang Tang, Gale Yan Huang, and Zitao Liu. Deep knowledge tracing with side information. In *International conference on artificial intelligence in education*, pages 303–308. Springer, 2019.

Kevin H Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. Back to the Basics: Bayesian Extensions of IRT Outperform Neural Networks for Proficiency Estimation. *International Educational Data Mining Society*, 2016.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Image-embodied Knowledge Representation Learning. In *IJCAI*, pages 3140–3146, 2017. URL https://doi.org/10.24963/ijcai.2017/438.

Xiaolu Xiong, Siyuan Zhao, Eric G Van Inwegen, and Joseph E Beck. Going deeper with deep knowledge tracing. *International Educational Data Mining Society*, 2016.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases, 2015.

Yang, Yang and Shen, Jian and Qu, Yanru and Liu, Yunfei and Wang, Kerong and Zhu, Yaoming and Zhang, Weinan and Yu, Yong. GIKT: a graph-based interaction model for knowledge tracing. *arXiv preprint arXiv:2009.05991*, 2020.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

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. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–748, 2020.

List of Figures

List of Tables

Towards an interpretable model of learners in a learning environment based on Knowledge Graphs

Antonia ETTORRE

Abstract

In recent years, society has demonstrated increasing need for more effective, comprehensive and easily accessible educational resources. A growing number of people around the world have gained access to online education and demand more efficient tools to enable learning anything, anywhere, at any moment. This requires the development of smarter educational systems, which should be able to improve users' learning curves and effectively assist them in their knowledge acquisition process, possibly relying on no, or very little, human support. A major step to concretize this vision lies in personalizing the learning process to be specifically adapted to every single user, taking into account their background, learning style, personal needs and objectives. To create such adaptive and customized environments, a major requirement is represented by the ability to trace user knowledge over time and assess whether they have the capacity to face a specific problem, exercise or question. This problem, known in the Education community as *Knowledge Tracing*, has been widely investigated in the last 50 years and several resolution approaches have been proposed. Though their performance improved sensibly over the last decade, such approaches present several shortcomings: from the excessive simplicity in the representation of the learning environment, which does not account for complex scenarios such as acquisition of soft skills or solution of group assignments; to the impossibility of interpreting the provided predictions, e.g. explaining why a student will fail while trying to answer a given question. In this thesis, we try to overcome these issues by exploring and proposing approaches based on the use of Symbolic AI approaches focusing on Graph based Knowledge Representation and Reasoning. Firstly, we propose a Knowledge Tracing approach that extends an existing model by introducing, as additional input features, *Knowledge Graph Embeddings*. Secondly, we investigate the explainability of the proposed approach by seeking the interpretation of the employed Graph Embeddings. This leads us to the implementation of a tool for the joint visual analysis of Knowledge Graphs and Graph Embeddings and to the development of an approach to verify the information encoded by such Graph Embeddings. Finally, we present a Knowledge Tracing model exclusively relying on the representation of the learning environment in the form of a Knowledge Graph, which does not require any additional external model for the prediction.

Keywords: Semantic Web, Knowledge Graphs, Education, Knowledge Tracing