FACULTY OF ENGINEERING
AND ARCHITECTURE

Knowledge Graph Creation and Embedding to Realize Explainable Hybrid AI Applications

Bram Steenwinckel

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Filip De Turck, PhD - Prof. Femke Ongenae, PhD
Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

January 2023

GHENT
UNIVERSITY

ISBN 978-94-6355-670-5
NUR 983, 984
Wettelijk depot: D/2023/10.500/2

Members of the Examination Board

Chair

Prof. Em. Luc Taerwe, PhD, Ghent University

Other members entitled to vote

Prof. Mehwish Alam, PhD, Télécom Paris, France
Ernesto Jiménez-Ruiz, PhD, City, University of London, United Kingdom
Prof. Jefrey Lijffijt, PhD, Ghent University
Prof. Erik Mannens, PhD, Ghent University
Prof. Sofie Van Hoecke, PhD, Ghent University

Supervisors

Prof. Filip De Turck, PhD, Ghent University
Prof. Femke Ongenae, PhD, Ghent University

Dankwoord

Wie ik in 2011 zou verteld hebben dat ik 11 jaar later een doctoraat in de computerwetenschappen zou behalen, zou eens goed gelachen hebben. Het was dan ook nooit het plan. Ik worstelde mezelf door mijn laatste jaren humaniora en was al blij dat ik met de hakken over de sloot kon beginnen aan een bachelor opleiding aan de Universiteit Gent. Wie ik na dat eerste jaar aan de universiteit zou verteld hebben dat ik 10 jaar later een doctoraat zou behalen, zou waarschijnlijk zelfs nog iets harder gelachen hebben. Mijn doel was dan ook veel concreter, ik wilde iedereen laten zien dat je met hard werken en de juiste mentaliteit wel degelijk eender welke universitaire richting aankan en dat je daarmee kan bereiken wat je maar wilt bereiken.

Nooit zou ik durven dromen dat ik ooit de kans zou krijgen om te doctoreren. Als ik terugdenk aan dat moment en het moment nu, dan ben ik fier op alles wat ik heb kunnen en mogen doen. Ik heb ontzettend veel geleerd en heel wat ervaringen mogen opdoen. Ik kan daarbij beamen dat niets vanzelf gaat. Maar ook hier waren hard werken en een juiste mentaliteit van cruciaal belang. Ik zou het zo opnieuw doen.

De dankbaarheid is dan ook groot voor heel wat mensen die me gesteund hebben en rechtstreeks of onrechtstreeks aan dit doctoraat hebben meegewerkt. Graag wil ik mijn promotor, Filip De Turck, bedanken voor de kans die ik gekregen heb. Oneindig veel dank ook voor mijn promotor Femke Ongenae. Femke, het was een hele eer om jouw PhD student te mogen zijn. Bedankt voor alle hulp 24 op 24, 7 dagen op 7. Bedankt voor al je adviezen, je visies op het leven en het (proberen) waken over mijn work-life balance. Ik was geen typische PhD student. Maar voor jou maakte dat dan ook helemaal niks uit. Through chaos as it swirls.

Ook zou ik graag al mijn juryleden en voorzitter willen bedanken. Bedankt Luc Taerwe, Sofie Van Hoecke, Jefrey Lijffijt, Erik Mannens, Mehwish Alam en Ernesto Jiménez-Ruiz om dit doctoraat te beoordelen en evalueren. Speciale dank ook aan Sofie, en dan vooral aan haar team van ongelooflijk competente mensen die ze heeft weten te verzamelen gedurende de periode waarin ik mocht doctoreren. Natuurlijk was dit hele verhaal niet mogelijk zonder de onderzoeksgroep waarin het zich afspeelde, IDLab, daarvoor wil ik Piet Demeester bedanken. Ook wil ik graag alle medewerkers binnen IDLab bedanken voor de administratieve steun die vaak nodig was de afgelopen 5 jaar.

Ik heb het genoegen gehad om voor vele projecten met industriepartners te mogen

werken. Ik wil daarom graag Mohamed Bentefrit, Bruno Van Den Bossche en Steven Lauwereins bedanken van Televic Rail, Pieter Crombez en Piet Verheye van Televic Healthcare, Steven Vandekerckhove van Renson, Haroen Vermylen van cumul.io, Jan Van Ooteghem en Koen Casier van Amaron, Joeri Ruyssinck en Joachim van der Herten van ML2Grow en Julie Wyffels van Z-Plus bedanken. Ook bedankt aan het Fonds Wetenschappelijk Onderzoek (FWO) voor hun steun in mijn strategisch basisonderzoek.

Ik heb tijdens mijn doctoraat mogen samenwerken met heel veel toponderzoekers en onderzoeksmedewerkers. Bedankt daarom aan Sander Vanden Hautte, Pieter Moens, Dieter De Paepe, Olivier Janssens, Pieter Heyvaert, Anastasia Dimou, Jasper Vaneesen, Nicolas Vandenbussche, Marija Stojchevska, Jonas Van Der Donckt, Jeroen Van Der Donckt, Vic Degraeve, Bruno Volckaert, Koen Paemeleire, Annelies Goris, Jelle Nelis, Ilja Rausch, Ruben Taelman, Pieter Colpaert, Pieter Simoens, Ben Demeester, Jarne Verhaeghe, Jerico Moeyersons, Tom Windels, Thomas De Corte, Stef Pletinck, Colin Soete, Sai Roberts, David Vander Mijnsbrugge, Nathan Vandemoortele, Diego Nieves Avendano, Stephanie Chen, Maria Torres Vega, Jeroen van der Hooft, Matthias Strobbe, Anna Lin, Christof Mahieu, Gertjan De Mulder en waarschijnlijk vele anderen die ik hier nu vergeet op te noemen. Ook wil ik nadrukkelijk Eric Laermans bedanken om me gedurende die 5 jaar toe te laten het vak Redeneren, Abstraheren en Formuleren mee te begeleiden.

Verder wil ik ook mijn collega's bedanken, waarmee ik samen op mijn bureau in iGent heb vertoeft. Bedankt Stijn, Philip, Michael, Kyana, Stéphanie, Ziye en mijn peter Pieter. Speciale dank ook voor 4 collega's. Femke De Backere, bedankt voor je jarenlange inzet gedurende de opleiding om uit iedere student het beste te halen. Het is onder andere door jouw begeleiding dat ik voor mijn masterthesis bij jou gekozen heb. Dat ik daarna een email stuurde naar de verkeerde Femke, is een fout die meerdere UGenters al hebben gemaakt. Toch wil ik je hier nog eens uitdrukkelijk bedanken voor alle begeleiding die je me ooit hebt gegeven. Mathias De Brouwer, wat hebben we 2 jaar ongelooflijk veel gefikst en gedebugd voor het Protego project. Oneindig veel dank voor alles wat je toen gedaan hebt. Gilles Vandewiele, bedankt om als senior PhD me alles te leren. Bedankt ook voor alle leuke momenten en voor side tracks te maken in mijn PhD. We waren het niet altijd op alle vlakken eens, maar dat maakte het net zo interessant om samen te werken. Joris Heyse, bedankt om dit avontuur samen met mij te starten en me te tonen hoe je een PhD afrond. Bedankt ook om samen de allereerste keer op conferentie te gaan en om te luisteren naar de vele verhalen, anekdotes en rare ideeën die ik gedurende die 5 jaar had. We hebben samen veel meegemaakt in die periode. Toch kan ik zeggen, met de woorden van wijlen jouw Oostendse held uit de film die waarschijnlijk onze jeugd heeft gekleurd: "Jij bent een vriend van mij".

Ook familie en vrienden zou ik nadrukkelijk willen bedanken voor de steun tijdens dit doctoraat. Bedankt aan alle volleybalploegen waar ik heb gespeeld en de fijne mensen die in zo'n verenigingen voor een goede werking zorgen. Bedankt ook aan Marie, Laurens, Joris en Stéphanie voor de spelletjesavonden. Bedankt ook aan Chris, Jonny,

Guy en Will voor de vele muzikale tijden samen en voor de titels van mijn papers. Bedankt ook aan mijn schoonouders, Jaap en Stephanie, voor al hun hulp tijdens deze 5 jaar en het zorgen voor een thuis. Ik heb me vaak afgevraagd wat ik jullie, mensen die alles al hebben, zou kunnen geven als bedanking hiervoor. Misschien dat een welgemeende en oprechte "dankjewel" wel de meeste waarde heeft. Bedankt ook om jullie dochter die waarden en normen mee te geven die belangrijk zijn om vandaag de dag kritisch naar de maatschappij te kijken. Ook wil ik mijn schoonbroers en schoonzus bedanken.

Lien, grote zus, bedankt om te tonen dat je met volharding en hard werken effectief overal kan geraken. Bedankt voor alle steun die je me hebt gegeven, en om een fiere grote zus te zijn bij alles wat ik deed. Gedurende de jaren zaten we misschien niet altijd meer zo kort en dicht bij elkaar als vroeger. Toch wil ik jou ook nog zeggen dat je kleine broer nog steeds fier is op alles wat je doet en dat er geen betere grote zus dan jij bestaat. Mama en papa, het moet wat zijn als ouder om zo twee kinderen te hebben. Die, ongeacht wat tegenslagen, bereikt hebben wat ze willen bereiken. Ik wil jullie bedanken voor alle steun en mij zo goed mogelijk af te zetten met de juiste bagage en kennis om dit te kunnen verwezenlijken. Bedankt ook voor de hulp bij weer een of ander project, bij het maken van een watersensor bijvoorbeeld of om me te komen halen als ik ergens vastzat in een station met geen trein meer naar huis. Ik hoop later te kunnen zeggen dat ik een even goede ouder ben voor Sepp en Isa. Een betere is gewoon onmogelijk.

Laura, wat hebben wij de afgelopen 5 jaar veel meegemaakt. We kregen twee kinderen. Je werd tweemaal ten huwelijk gevraagd en we trouwden half. We verloren beide geliefden maar hadden altijd elkaar voor steun en liefde. Jij zag en moest alle problemen ondergaan die dit doctoraat met zich meebracht. Jij weet dat, wat voor de buitenwereld vaak leek vanzelf te gaan, niet altijd vanzelfsprekend was. Jij zag de ontgoocheling, de irritaties maar ook de successen en het plezier dat ik had in mijn job. Ik wil je bedanken om, ondanks het vaak in je eigen nadeel was, mij te blijven steunen, mij graag te zien om wie ik ben en het beste in mij naar boven te halen. Om ervoor te zorgen dat ik alles heb kunnen doen in dit doctoraat wat ik maar wou doen. Ik heb je vele uren wakker gehouden door 's nachts te werken, was vaak heel laat thuis en plande vergaderingen in op de meest onmogelijke momenten. Zonder jou was dit doctoraat niet mogelijk en ik kan je daar niet dankbaar genoeg voor zijn. Ik hoop dat je beseft dat er veel van jou in dit boek zit. Onbewust is dit dus een werk van jou en mij. Je vroeg me gedurende die 5 jaar vaak wanneer ik klaar zou zijn met werken. Uiteraard doelend op het moment van de dag wanneer we zouden kunnen eten of samen konden zijn. Ik antwoordde toen vaak met: "nog 1 jaar, 5 maanden en een aantal dagen, want dan loopt mijn doctoraat af". Vandaag kan ik je zeggen: "Nu". Ik zie je graag en voor altijd en eeuwig is een hele lange tijd, maar voor altijd is met jou erbij nooit lang genoeg.

Als laatste wil ik Sepp en Isa bedanken. Veel mensen hadden een oordeel over het feit of het wel slim was om een doctoraat en een kind (laat staan twee) te combineren.

Zij weten dan ook niet hoe fantastisch jullie met twee zijn. Gewoon het feit dat jullie er zijn, gaf me vaak rust. Niets kan het gevoel van vader zijn overtreffen, dus ook dit doctoraat niet. Bedankt om met twee een lach op mijn gezicht te toveren wanneer het wat moeilijker ging. Om jullie mama te doen fronsen met ons gek gedoe. Bedankt om mij te laten inzien dat veel dingen maar bijzaak zijn en dat het dus oké is om zaken te laten liggen. Er zal nooit iets belangrijker zijn dan jullie twee. Ik hou van jullie.

Gent, december 2022
Bram Steenwinckel

Table of Contents

List of Figures

List of Tables

Samenvatting

Artificiële intelligentie (AI-toepassingen) maakt al enige tijd deel uit van ons leven, gaande van slimme thermostaten tot je smartphonescherm kunnen ontgrendelen op basis van gezichtsherkenning. Het is dan ook voornamelijk in deze meer persoonlijke en huishoudelijke sfeer dat AI vandaag de dag het bekendst is. Hier is AI vaak het tastbaarst en heeft het een impact op de dagelijkse routines. Maar naast deze meer consumentgerichte toepassingen, wordt AI meer en meer ingezet in domeinen ver buiten deze huiselijke sfeer. Er wordt terecht gesproken over een vierde industriële revolutie, met als hoofddoel: intelligente automatisering. AI is maar slechts één van de puzzelstukken in dit verhaal. Een andere puzzelstuk is het feit dat meer en maar objecten (of things) kunnen verbonden worden met het internet (Internet of Things of IoT). Deze apparaten sturen allemaal data door via het internet. Die data kan geanalyseerd worden of wordt zelfs gebruikt door AI-toepassingen om bepaalde beslissingen of voorspellingen te maken. In een industriële setting zijn dergelijke IoT toestellen niet meer weg te denken. Zo worden sensoren overal gebruikt in het productieproces en kan een AI applicatie die sensordata gebruiken om analyses te maken van het operationele systeem. Meer nog, AI toepassingen kunnen voorspellen wanneer er iets mis zal gaan met een bepaald onderdeel of wanneer onderhoud van een machine aangewezen is. Naast nieuwe toepassingen in de industrie, is er ook een groot applicatiedomein voor AI in de gezondheidszorg. Ook hier wordt er meer en meer gebruik gemaakt van sensoren en data om patiënten op te volgen en helpen AI toepassingen artsen om beslissingen te maken. De slimme toepassingen die nu al beschikbaar zijn voor de gezondheidszorg vormen een belangrijk middel om mensen langer thuis te laten wonen, met alle comfort en met minder zorgen voor zij die om hen geven.

Hoewel het bovenstaande klinkt alsof we vandaag alles in handen hebben om dergelijke applicaties te maken, is dit verre van waar. Hoewel AI vandaag de dag ons met verstomming slaat, leeft de perceptie dat iets wat beslist wordt door een AI-toepassing vaak onverklaarbaar is. Het lijkt wat op een doos waar bijvoorbeeld de data van patiënten of een productiesysteem wordt ingestoken en waar na hard schudden een bepaalde beslissing of voorspelling over die patiënt of dat productiesysteem uitkomt, zonder enige vorm van verklaring op wat net die beslissing of voorspelling gemaakt is. De consequenties van dit probleem zijn zichtbaar in zowel de gezondheidszorg als het bedrijfsleven. Artsen staan sceptisch op beslissingen die AI kan nemen zonder hen er een verklaring voor te geven. Een AI model dat een bloedvergiftiging in 70% van de gevallen kan diagnosticeren maar daarbij ook aangeeft op welke parameters die beslissing genomen is, zal verkozen worden boven een AI model dat dit met een 90% accuraatheid kan maar zonder enige vorm van uitleg. In de industrie kunnen de-

zelfde AI modellen voorspellen dat er iets mis zal gaan met het productieproces, maar kunnen aangeven waar of wat precies zal misgaan zal de stress en overlast van het onderhoudspersoneel gevoelig kunnen verlagen en kan zelfs zorgen dat het onderhoud van bepaalde systemen kan gebeuren net voor er iets zal misgaan met het product of het toestel dat gemaakt wordt.

De nood om verklaringen voor AI voorspellingen te voorzien werd zodanig groot, dat het AI-domein opgesplitst werd in een kennisgedreven en een datagedreven tak. Datagedreven AI leert hoe beslissingen of voorspellingen kunnen gemaakt worden door enkel gebruik te maken van de beschikbare data van bijvoorbeeld sensoren. Dergelijke AI applicaties hebben vaak een beperkte vorm van hoe de beslissing tot stand kwam. Bij kennisgedreven AI zijn het de experten in het vakgebied die regels en analyses maken op basis van hun kennis en de data. Dergelijke regels worden dan in een AI model gestoken en geven een resultaat terug wanneer ze geactiveerd worden. Doordat de expert weet welke regel net geactiveerd werd in een bepaalde situatie, weet men ook waarom een bepaald resultaat gegenereerd werd. De komst van meer en meer sensoren, en ook meer en meer situaties waarbij expertregels nodig zijn, maakte deze kennisgedreven modellen echter minder en minder populair. Aangezien de datagedreven varianten overweg konden met gigantische hoeveelheden data, werd deze vorm dominant in de laatste 5 jaar, ook omdat het onderhoud en toepassen van de regels in een kennisgedreven AI-model steeds langer en langer begon te duren. Voor de gezondheidszorg en veel industriële toepassingen leidde dit tot een minder groot geloof in de voordelen van AI.

De hoeveelheid data dat door deze slimme apparaten wordt geproduceerd, blijft echter groeien en meer experten moeten naast hun dagdagelijkse job, nu ook mee data analyseren om verworven inzichten te verifiëren of competitief te blijven ten opzichte van de concurrentie. De roep naar AI-modellen die zowel de kennis van deze experten kunnen incorporeren maar ook overweg kunnen met een grote hoeveelheid aan data werd dan ook groot. Oplossingen werden voorgesteld als een combinatie of aaneenschakeling van data- en kennisgebaseerde systemen. Ze worden om die reden ook hybride AI oplossingen genoemd. Zo'n aaneenschakeling van modellen is echter maar een tijdelijke oplossing. Niet alleen moeten er twee systemen afgesteld worden op elkaar, de mogelijkheid om dergelijk systeem aan te passen naar een nieuwe situatie is heel gelimiteerd net door die nauwe aaneenschakeling.

De belangrijkste onderzoeksbijdrage van dit proefschrift is dan ook om verder te gaan kijken dan deze aaneengeschakelde hybride AI-toepassingen. In dit proefschrift onderzoeken we namelijk hoe de output en input van data- en kennisgedreven AI modellen gecombineerd kan worden in een structuur die zowel op een efficiënte manier beslissingen kan maken als een uitleg kan geven waarom zo'n beslissing gemaakt werd, voor dynamisch veranderende domeinen te vinden in de industrie of gezondheidssector. Meer concreet vertaalt zich dit in de volgende vijf bijdragen.

De eerste uitdaging om een adaptief hybride AI systeem op te stellen is dat ook de kennis van de experten die nodig is om verklaringen te geven aan de beslissingen of voorspellingen op adaptieve wijze kan geïncorporeerd worden in een dergelijke systeem. De eerste bijdrage is om niet die kennis zelf te laten aanleveren door de experten via manuele input maar door gebruik te maken van documenten en reeds

bestaande kennismodellen om deze informatie rechtstreeks aan te bieden. Doordat de documenten kunnen worden aangepast, bestaat er nu de mogelijkheid om ook deze aanpassingen automatisch mee te nemen. Al deze kennis bundelen we nu in een zogenaamde kennisgraaf. Kennisgrafen zijn net gemaakt om informatie en connecties tussen deze informatie op te slaan.

De tweede uitdaging onderzoekt dan ook of zo'n kennisgraaf niet gebruikt kan worden om een centrale rol te spelen in hybride AI applicaties. Voor deze centrale rol moet niet alleen de expertkennis, maar ook de beslissingen van de modellen mee in de kennisgraaf beschreven worden. Op deze manier kan een kennisgedreven model gemakkelijk op basis van alle informatie aanwezig in de kennisgraaf, nieuwe inzichten gaan afleiden en zelfs voor de uitkomst van de datagedreven componenten verklaringen geven zonder dat de werking van deze datagedreven modellen hierdoor geaffecteerd worden. De kennisgedreven componenten krijgen hierdoor nog een extra adaptief karakter, omdat deze nu ook nieuwe en ongeziene inputs proberen te verklaren.

Om een hybride AI-applicatie nog verder adaptief te maken, onderzoekt een derde uitdaging of de regels die gebruikt worden voor een kennisgedreven component niet rechtstreeks afgeleid kunnen worden op basis van alle informatie in de kennisgraaf. Het leren van semantische regels is zo'n bestaande techniek. Echter zijn de huidig bestaande technieken niet echt gemaakt in het opzicht van een hybride AI-toepassing. Dit proefschrift beschrijft en evalueert een nieuwe manier om semantische regels te leren op basis van een interpreteerbare kennisgraafrepresentatie. Deze nieuwe techniek kan zowel meer relevante regels als meer accurate regels genereren dan de reeds bestaande technieken in een veel snellere tijd.

Omdat alle informatie nu centraal beschikbaar wordt gemaakt in een kennisgraaf, kijkt een vierde uitdaging naar hoe deze kennisgraaf kan gebruikt worden in de datagedreven componenten van een hybride AI-applicatie. Technieken bestaan om zo'n kennisgrafen om te vormen naar een representatie die ook gebruikt kan worden als input voor de datagedreven modellen. Echter hebben sommige van deze technieken moeilijkheden met hele grote kennisgrafen of met kennisgrafen die veranderen doorheen de tijd. Ook verliezen we bij deze omvormingen alle oorspronkelijke interpreteerbaarheid van de kennisgraaf, wat dergelijke technieken dus niet echt interessant maakt in een hybride AI-toepassing waar een verklaring voor een gemaakte voorspelling nodig is. In dit proefschrift gebruiken we dezelfde interpreteerbare kennisgraafrepresentatie om semantische regels te leren als input voor de datagedreven modellen. Niet enkel behielden we zo het inherent interpreteerbaar karakter van de kennisgraaf, deze techniek was ook accurater tijdens het maken van voorspellingen voor de datagedreven modellen ten opzichte van de bestaande technieken.

Zoals eerder vermeld, zijn de data in een hybride AI-applicatie binnen het domein van de industrie of de gezondheidszorg een bepaalde (sensor-) waarde gegenereerd op een bepaald tijdstip. Dergelijk type data wordt daarom ook vaak als tijdreeks bestempeld. Tijdreeksen mee incorporeren in een kennisgraaf leverde een vijfde en laatste uitdaging op. In dit proefschrift maakten we een kennisgraaf op basis van events. Bij elke event connecteerden we dan de beschikbare kennis voorhanden tijdens dat event, alsook de sensorwaarden die zich hadden voorgedaan op het tijdstip van dat

event. Events werden aan elkaar geschakeld in de kennisgraaf, zodanig dat er een connectie was tussen het huidige en de voorgaande events. Dezelfde kennisgraaf representatie, zoals beschreven in de vorige twee paragrafen, kon dan gebruikt worden om zowel regels als datagedreven modellen te leren, gebruikmakend van op tijd gebaseerde events. Voor een use case in de gezondheidszorg behaalden we met deze hybride oplossing een accuraatheid die meer dan 10% hoger ligt dan de traditionele datagedreven varianten.

De centrale rol van de kennisgraaf, en elke contributie die zowel de kennisgraaf verder adaptief maakte of de kennisgraaf mee hielp incorporeren in de data- en kennisgedreven componenten, geeft als resultaat dat een hybride AI-applicatie kan mee leren en evolueren doorheen de tijd, en dat een dergelijke applicatie nu kan ingezet worden in andere configuraties ver buiten de huidig beschikbare aaneenschakelingen van kennis- en datagedreven oplossingen.

Summary

Artificial intelligence (AI) applications have been a part of our lives for some time, ranging from smart thermostats to being able to unlock your smartphone screen based on facial recognition. It is therefore mainly in this more personal and domestic sphere that AI is best known today. It is here that AI is often the most tangible and has an impact on our daily routines. But besides these more consumer-oriented applications, AI is increasingly being used in domains far beyond this domestic sphere. We are encountering a whole fourth industrial revolution, with the main goal of intelligent automation. AI is only one of the pieces of the puzzle in this revolution. Another piece of the puzzle is the fact that more and more objects (or things) can be connected to the Internet (Internet of Things or IoT). These devices all send their data through the internet, which can be analysed or are even used by AI applications to make certain decisions or predictions. In an industrial setting, such IoT devices are indispensable. Sensors are used everywhere in the production process and an AI application can use that sensor data to make analyses of the operational system. Moreover, AI applications can predict when something will go wrong with a certain component or when maintenance of a machine is required. Besides new applications in industry, there is also a large application domain for AI in healthcare. Here, sensors and data are increasingly being used to monitor patients, and AI applications help doctors make decisions. The smart applications that are already available for healthcare are important to allow people to live longer at home, in comfort and with less worry for those who care about them.

While the above sounds like we have everything in place today to make such applications, this is far from correct. Although AI is ubiquitous, the perception is that anything decided by an AI application is often inexplicable. It is a bit like a box where, for example, the data from patients or a production line is put in and, after shaking hard, a certain decision or prediction about that patient or production comes out, without any kind of an explanation of how the decision or prediction was made. The consequences of this problem are visible in both healthcare and industry. Doctors are sceptical about decisions AI can make without giving them an explanation. An AI model that can diagnose sepsis correctly in 70% of the cases but also indicate on what parameters that decision was made will be preferred over an AI model that can do this with 90% accuracy but without any explanation, as doctors would never be able to assess whether an outcome belongs to the 90% correct ones or is a misjudgement by the model. In industry, the same AI models can predict that something will go wrong with the production process, but being able to indicate where or what will go wrong can significantly reduce the stress and inconvenience for maintenance staff,

and even allow maintenance of certain systems to be done just before something goes wrong with the production or the device being made.

The need to provide explanations for AI outcomes became so important that the AI domain was split into a knowledge-driven and a data-driven branch. Data-driven AI learns how to make decisions or predictions using only the data available from, for example, sensors. This group of AI applications often offers more limited insight into how the decision was made. In knowledge-driven AI, it is the experts in the field who make rules and analyses based on their knowledge and the data. Such rules are then plugged into an AI model and return a result when activated. By knowing which rule is activated in a given situation, the expert also knows why a certain result was generated. However, the usage of more and more sensors, and also more and more situations requiring expert rules, made these knowledge-driven models less and less popular as they are time-consuming and tedious to construct and maintain. As the data-driven variants could handle huge amounts of data easily, this form became dominant in the last 5 years. But the lack of explanations offered by these algorithms led to a lesser belief and trust in the benefits of AI within fields such as healthcare and many industrial applications.

However, the amount of data produced by these smart devices continues to grow and more experts need to analyse data in addition to their day-to-day job in order to verify new insights or to remain competitive with the competitors. The call for AI models that can both incorporate the knowledge of these experts and can handle a large amount of data became urgent. Solutions were proposed as a combination or chain of data- and knowledge-based systems. They are called hybrid AI solutions. However, such a combination of models is only a temporary fix. Not only do two systems have to be maintained, but the possibility of adapting such a system to a new situation is also very limited because of this tight coupling.

The main research contribution of this dissertation is therefore to look beyond these concatenated hybrid AI solutions. We investigated how the output and input of data- and knowledge-driven AI models can be combined in a methodology and architecture that can both efficiently make decisions and explain why these decisions were made for dynamically changing domains, such as predictive maintenance or healthcare. More concretely, this translates into the following five contributions.

The first challenge in building an adaptive hybrid AI system is that also the expert knowledge needed to explain the decisions or predictions can be easily and continuously incorporated into such a system, without needing the constant support of a knowledge engineer who is not proficient in the domain. The first contribution is not to let the experts perform the tedious task of building knowledge models themselves, but to use documents and already existing knowledge models to provide this information directly. Because the documents can be adapted, it is now possible to include these adaptations automatically. Moreover, it does not require domain experts to become knowledge engineers. The resulting knowledge is bundled in a so-called knowledge graph. Knowledge graphs represent the expert knowledge in a graphical format as nodes with a directed labelled edge between them, describing their relationship.

The second challenge is therefore to investigate whether such a knowledge graph

can be used to play a central role in hybrid AI applications. For this central role, not only the expert knowledge but also the decisions of all the AI models must be described in the knowledge graph. This way, a knowledge-driven model can easily derive new insights based on all the information present in the knowledge graph and even provide explanations for the outcome of the data-driven components without affecting the functioning of these data-driven models. The knowledge-driven components become adaptive because they now also try to explain new and unseen inputs. As such, this dissertation resulted in an architecture that outlines how knowledge- and data-driven models can ideally be intertwined into a hybrid AI application by using the knowledge graph.

To make a hybrid AI application even more adaptive, a third challenge investigates whether the rules used for a knowledge-driven component cannot be derived directly from all the information in the knowledge graph. Semantic rule mining is such an existing technique. However, the currently existing approaches are not optimized to operate in a hybrid AI application. This dissertation describes and evaluates a new way of learning semantic rules based on an explainable knowledge graph representation, called INK. INK keeps the explainable characteristics of the knowledge graphs by converting them to a two-dimensional binary representation. This new technique can generate both more relevant rules and more accurate rules than existing techniques in a much faster time.

Since all information is now made available centrally in a knowledge graph, a fourth challenge looks at how this knowledge graph can be used in the data-driven components of a hybrid AI application. Techniques, called knowledge graph embeddings, exist to transform such a knowledge graph into a representation that can also be used as input for data-driven models. However, some of these embedding techniques have difficulties with very large knowledge graphs or with knowledge graphs that change over time. Also, with these transformations, we lose all original explainability of the knowledge graph, which makes such techniques not interesting in a hybrid AI application where an explanation for a prediction is needed. In this dissertation, we used the INK representation to learn semantic rules as input for the data-driven models. Not only did we retain the inherently explainable aspects of the knowledge graph, but this technique was also more accurate in making predictions by the data-driven models compared to the existing approaches.

As mentioned earlier, the data within the domain of industry or healthcare can be a certain (sensor) value generated at a certain time. This type of data is therefore often referred to as time series. Incorporating time series into a knowledge graph provided a fifth and final challenge. In this dissertation, we created a knowledge graph based on events. For each event, we then connected the available knowledge during that event, as well as the (sensor) values that had occurred at the time of that event. Events were linked together in the knowledge graph, such that there was a connection between the current and previous events. By optimising the INK representation to work with this time-dependent knowledge graph, both rules and data-driven models using time-based events in a hybrid AI fashion could be designed. For a healthcare use case, we achieved an accuracy rate with this hybrid solution that is more than 10 % higher than the traditional simple data-driven variants.

The central role of the knowledge graph and every contribution that either made the knowledge graph further adaptive or helped to incorporate the knowledge graph into the data- and knowledge-driven components results in a hybrid AI solution than can explain the made decisions and that can learn and evolve and can now be deployed in other configurations far beyond the currently available concatenations of knowledge- and data-driven solutions.

1
Introduction

"Dad, I didn't listen to what you were saying, but can you just help me to balance my bike? I will ride to that stork over there myself."

–Sepp Steenwinckel (2022)

A couple of months ago, I started to teach my toddler how to ride a bike. Me, his dad, explained to him all the things he had to do or keep in mind in order to achieve the goal of staying upright and in control. I explained him the rules for what to do with his upper body to maintain balance, how, when, and at what speed he needed to pedal, that he had to observe his surroundings, the type of road he was on, whether the road was wet or sandy, whether there are obstacles in his way, what the likelihood was of a car coming around the next bend in the road, and whether it was light or dark out, cloudy or sunny, calm or windy, etc. All of this observation of context and knowledge on how to interpret it went into the act of staying upright long enough to keep him from falling.

The way I started to explain all the things I know about riding a bike is something typically human. Our whole learning mechanism is trained to capture the available knowledge efficiently and when needed, we try to reproduce this knowledge in the form of explainable instructions or rules. We, as humans, have the ability to analyse our actions or analyse our surroundings and we use these analyses to update our knowledge [1].

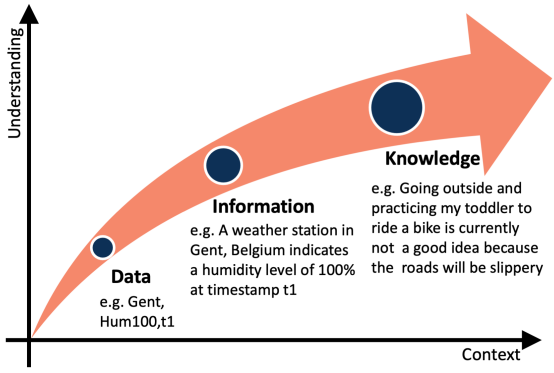


Figure 1.1: Going from data to knowledge. A simple example shows how data values from a weather station can be transformed into information that can be used to solve the current problem of whether or not to go outside.

1.1 Knowledge-based systems, an AI sub-field

Knowledge-based systems are a subfield within Artificial Intelligence (AI) that try to mimic what I described above. Their goal is to define an autonomous system which may be useful for assisting with human learning and making decisions [2]. To understand what this technique is capable of, we have to define what knowledge exactly is.

In general, AI starts from the principles of data as shown schematically in Figure 1.1. Data is typically a jumble of raw samples, and human effort is needed to properly interpret and organise the data. Data also comes in multiple formats. For example, images and videos can hold a lot of data that requires some sort of interpretation to extract information from them. In the example of Figure 1.1, the data represents a certain sensor output. Information, in contrast, is a collection of consistently organised and structured facts. Less human effort is needed to find relevant facts. A category of relevance or interest can be easily found within the information. This makes information more valuable than raw data. In our example, the original sensor data is translated to a high outside humidity level, originating from a weather station nearby. Knowledge, in the end, is the application of information to answer a question or solve a problem. In other words, information with context or meaning is knowledge. Again, in the example of Figure 1.1, more context is added to clarify why it is not a good idea to ride a bike, given the observed weather status.

The built-in problem-solving capabilities of knowledge-based systems allow them to understand the data input they process and let them make informed decisions based on the knowledge they store [3]. All knowledge-based systems provide specific explanations for the solutions of the problems they are trying to solve, which is their main

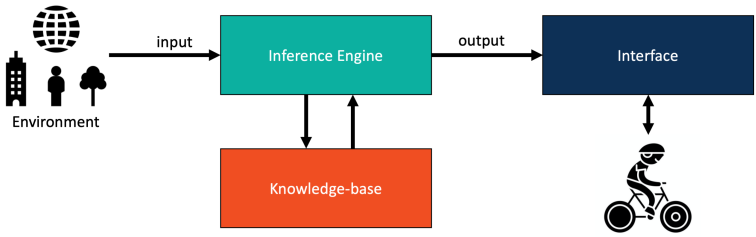


Figure 1.2: Architecture of a knowledge-based system. Three main components are linked to each other to decide and provide a solution to a given problem.

advantage. Knowledge-based systems fall in the range of symbolic AI [4]. They try to learn to understand the world by forming internal symbolic representations of its "world". Humans use symbols all the time to define things (bike, father, tree, etc.), but also abstract concepts (bank transaction) or things that don't physically exist (web page, blog post, etc.). They can also describe actions (biking) or states (inactive). Being able to communicate in symbols is one of the main things that make humans intelligent.

The typical architecture of a knowledge-based system, shown in Figure 1.2, consists of three main components: a knowledge base, an inference engine and an interface. [5]. The knowledge base is an established collection of information and resources, i.e. the known facts and rules, created by a so-called knowledge acquisition system. In our bike riding example, the knowledge base would hold all of our general knowledge about how to ride a bike. The system defines this knowledge base as its repository for the knowledge it uses to make decisions. The inference engine deduces insights from the knowledge within the knowledge base by using particular algorithms. The RETE algorithm [6] is, for example, such an algorithm to derive insights when the knowledge base consists of rules (so-called rule-based systems, which is a more specific knowledge-based system type). The inference engine can update the knowledge base with these insights, such as new rules. The user interface allows users to interact with the system and submit requests, also called queries.

1.1.1 From Knowledge Base, till Knowledge Graph

When people think about information or, more in general, data, they usually represent this data on a computer using a tabular format like in Excel or a, e.g., MySQL database. This representation is usually referred to as flat or two-dimensional as only rows and columns are provided to combine the information. An example of such flat data is weather station sensor values, such as the ones on the left in Figure 1.3, that decide whether or not it is a good idea to go outside. Those values can be stored in a simple Excel table. More complex structures are needed to combine the information from these flat representations together. Knowledge itself, however, is represented differ-

ently. It can still be structured, but instead of a tabular format, it uses objects with pointers to other objects, which in turn can have additional pointers. Instead of being flat, knowledge can also have a multi-relational aspect. The streets and neighbourhood where my toddler learns to ride a bike, visualised in the right part of Figure 1.3, is such a more complex network with crossings and streets linked to those crossings. For each street, we can define the type of street or the availability of a sidewalk to provide even more information.

Date	Time	Sky	Temp	Wind	Weather	Remarks
15/04/19	23:45	clear	7°c	30km/h	Good	Full moon
04/02/22	11:30	Cloudy	7°c	28km/h	Rain	New moon

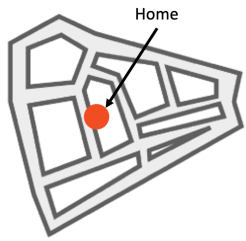


Figure 1.3: Flat weather data stored as a table (left) vs Complex neighbourhood street network (right)

When we want to represent this knowledge inside a knowledge base, the available knowledge should be defined in a structured format. When this structured knowledge is available as files, our system can interpret this knowledge in a similar way a computer can deal with a database. In the last decade, the Resource Description Framework (RDF) standard is frequently used to define knowledge in such a structured way [7]. RDF can be used to describe knowledge in a structured form of a collection of (subject, predicate, object) triples [8]. All these triples together form a knowledge base.

The multi-relation aspect within this knowledge base is defined by, e.g. two triples with a similar subject but different predicate and object values. To be able to visualise and represent this multi-relation aspect even better, this knowledge base can be represented in the form of a multi-relational directed graph, which is called a Knowledge Graph (KG) [9]. Each triple corresponds to 2 nodes (subject and object) in this graph with a labelled edge (the predicate) between them. KGs are used extensively in anything from search engines [10] and chatbots [11] to product recommenders [12] and autonomous systems today [13]. While other knowledge base structures exist, KGs are seen as the main data structure to encode domain, task and application knowledge [14].

In a database, an underlying data model typically provides a clear definition of every value and its semantic meaning in the database tables. KGs encode the meaning of the data for programmatic use in an ontology, which describes the different types of entities in the graph and their characteristics [14]. The ontology can be represented as a schema sub-graph. In contrast to a database, this means that the graph is both a place to organise and store the data, which is an advantage when reasoning what the data is about and deriving new knowledge. Ontologies are usually written in Web

Ontology Language (OWL) [15]. An example of how an ontology can be used in our bike example is shown in Figure 1.4. The ontology can be used to further specify the different types of nodes in our neighbourhood topology and it also defines the heterogeneous predicates that can be used to differentiate the relationships between those nodes. Now, we know that the orange node is a house, but this house is also near a park.

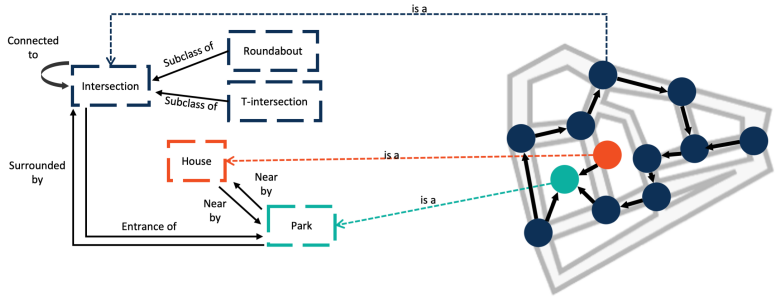


Figure 1.4: Example of a KG, with an ontology (left) and data linked to this ontology (right)

1.1.2 Inference engine

When all facts and knowledge about our domain are available in our knowledge base or KG, we still need a "brain" that tries to derive answers or actions from it. An inference engine can be such a "brain" that tries to derive answers from a knowledge base. It can reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions.

Inference engines try to figure out new common facts and their relationships to other objects when applying these new rules. The inference engine can either perform descriptive or prescriptive reasoning [16]. Descriptive or task-agnostic reasoning uses general information in the KG or some general facts, which hold beyond this provided KG and in a rather general context applicable. The logical rules that are used in the inference engine can be e.g. *If X is Located in Y and Y has Weather Z, Then X perceives Z*, which is a generalisation of the fact that you perceive the weather where you are located. Such descriptive inference operations can be used for KG completion tasks [17, 18].

On the other hand, prescriptive reasoning is more task-specific and performs inferences on the current data, to make predictions in the future [19]. The engine uses logical rules which were derived from a positive set of nodes and tested against some (generated) negative set of counter-examples to confirm the applicability. An example prescriptive reasoning task could be to find one general rule to discriminate the intersection nodes from the other nodes in the example of Figure 1.4. One possible rule

could state that the *connected to ?x* relationship should hold for all intersection nodes.

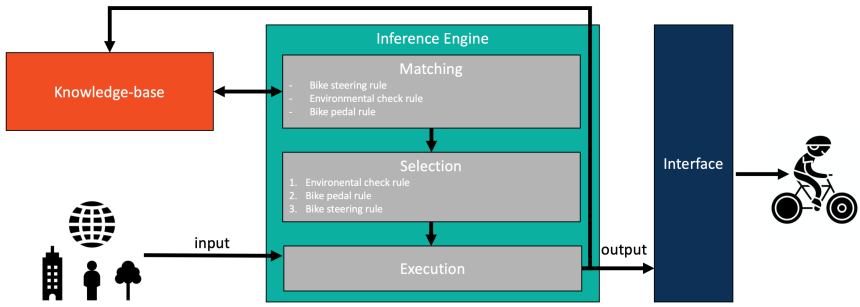


Figure 1.5: Overview of the Matching, Selection and Execution states within an inference engine. Within the knowledge base, the first three rules were matched with the current task of riding a bike. Second, the priority of these rules is defined. At last, the rules are executed one by one.

To perform such descriptive or prescriptive reasoning, the inference engine cycles between three action states as shown in Figure 1.5. Matching rules or facts, selecting rules or facts, and executing rules or facts [20]:

- **Matching:** In the first state, the inference engine finds all of the rules and facts that are satisfied by the current contents of the provided knowledge base. The matches that are found are all candidates for execution. The same rule or fact may appear several times in the selection set if it matches different subsets of inputted data items. In the example of Figure 1.5, the rules according to the task of riding a bike were matched given the provided knowledge base.

- **Selection:** The inference engine then passes along the matches to the second state. In this state, the inference engine applies some selection strategy to determine which rules will be executed and in what order. The selection strategy can be hard-coded into the engine or may be specified as part of the model. In the larger context of AI, these selection strategies are often referred to as heuristics. These heuristics can take into account the complexity of the rules and facts or favour rules with a higher priority. In our example, the environmental check rule is provided with a higher priority and should be executed first.

- **Execution:** Finally, the inference engine executes or fires the selected rules or facts with the provided input as parameters. The knowledge base is usually updated by firing rules and adding conclusions as facts. A different set of rules may match during the next cycle after these actions are performed. The inference engine stops either on a given number of cycles or when no rules match the input. The actions and rule outcomes may also trigger further processing outside of the inference engine (such as informing users through a graphical interface or calling remote functions to mitigate a certain triggered action).

1.1.3 Knowledge-driven problems

While I was explaining all the rules to my toddler to ride a bike, it was quite clear that I was unable to capture all the available knowledge in this domain and transfer this information to him. I'm sure he only listened to even a fraction of the rules that went into achieving his goal: riding his new bike. If his brain was represented by a knowledge-based system, it would also take some time for his inference engine to process all these available rules making it for such broad environments even impractical to infer what to do.

In many applications, where large amounts of inputs and situations are concerned or when performance time considerations are critical, the generation of rules or facts to apply is a non-trivial problem. The number of rules applied to determine whether or not an action should be taken has a negative impact on the time needed to come to such a decision [21].

Capturing the knowledge within such complex domains is also a challenging task. It is not always easy to transfer knowledge residing in the human brain towards such knowledge-based systems [21, 22].

1.2 Machine Learning, another AI subfield

Even before I could start to teach my toddler to ride a bike, we needed... a bike. So, I went over different advertisements for small bikes, both new and used ones. I started to see a pattern that bike prices depend on their age and drop on average 50 euros every year, but won't get lower than 25 euros.

People perform this decision process all the time, e.g. when trying to estimate a reasonable cost for a used iPhone or figure out how many cakes to buy for a birthday party. 2 pieces per person? 3? But in some cases, revealing such patterns can be hard. For the toddler's bike, not only the manufacturing date influenced the price, but also the technical condition, the number of options (brakes, bell, lights), the colour, and even more hidden factors I wasn't aware of.

For huge and complex tasks, humans, and definitely computer scientists, are lazy and want to let computers do the maths for them. So we provide the computer machine with all available bike data and ask it to find all hidden patterns related to the bike price. This form of AI, where the goal is to predict results based on incoming data, is called Machine Learning (ML) [23]. ML can be seen as a data-driven or a subsymbolic AI technique, as it performs calculations according to some principles that have been demonstrated to be able to solve problems [4]. ML consists of three main components: data, features and models.

1.2.1 Data to power the machine

Data can be represented in many formats. Images, texts, tabular or even KGs are all considered as data from an ML perspective. The rise of sensors being placed in homes, industrial plants, or even in so-called smart cities, led to the rise of data with a temporal effect, defined as time series data.

Based on this available data, Three main types of ML learning categories can be defined [24]. First, there is supervised learning, in which a label of interest is provided for each available data record of interest. The ML model learns from this historic labelled data and based on these learnings it tries to predict the label for new unknown records. The previous bike purchase example is such a supervised learning problem. The price of each bike can be seen as a label. Other examples of such applications are predicting the stress level of humans throughout the day using a smart wearable and environmental context [25] or the degradation of RNA [26]. Within supervised learning, a distinction is made based on the type of the label. If the label is discrete, the supervised ML approach links to a classification task. When the label is continuous, e.g. the purchase price of a bike, a regression problem is solved.

A second category is that of unsupervised learning, in which no label information is provided. The goal here is to form groups of common records and either try to describe these groups or represent the records which do not belong to these groups. Examples of such applications include clustering together similar papers about covid-19 using learned representations [27] or finding on which days a household was ventilated abnormally [28].

The third, and last category is reinforcement learning. Here, a learning agent interacts directly with an environment or a system. At each time step, the agent receives information in the form of data on the state of the system and the agent chooses an action to perform. Once in a while, the agent receives a reward or reinforcement signal, indicating whether or not the chosen actions were beneficial to obtain the predefined goal. No other evaluative feedback from the system other than the reward or failure signal (when a negative reward is obtained) is available. The ultimate goal of the agent is to learn a mapping from the states to the actions that maximise the discounted reward over time [29]. In our bike riding example, a reinforcement learning approach could perform actions like steering or pedalling and a negative reward could be returned when the agent falls. The number of states, obtained from the environment, is however quite large in this context. Reinforcement learning is applied in a wide variety of domains. It is also applied in the healthcare domain to determine which action a social robot should perform based on the emotional rewards of elderly with dementia [30].

1.2.2 The realm of feature engineering

Data can be seen as the fuel to run the ML model. But in every engine, the data must be transformed into something useful in order to get the right outcome. This transformation process can be a human engineering step starting from the data towards something more interesting and valuable for an ML model. As described previously, it was not the bikes themselves that were used in the ML model to reveal the price pattern. We used characteristics of the bike, such as the colour and the manufacturing date, to predict the price. These characteristics are called features and deducing features from the data is called feature engineering [31].

Engineering features from data might seem trivial, but in many cases, it opens a whole new area of research. ML likes vectors of various data types as input. Complex data structures such as KGs or time series require feature engineering to transform the original structure into such a vector representation. Simply converting the complex structure sample by sample or component by component to a flat, tabular representation would suffer from the curse of dimensionality which limits ML models to include any feature one can generate [32].

To solve the curse of dimensionality here, one can use embeddings. An embedding is a low-dimensional representation of high-dimensional data. Typically, an embedding won't capture all information contained in the original data. A good embedding, however, will capture enough to solve the problem or task at hand [33]. A well-known embedding technique word2vec transforms human-readable text into a feature vector, which can be later on used in an ML task [34]. Even for KGs, the nodes within such a graph can be embedded in a vector such that downstream ML models can use those vector embeddings to make predictions. Figure 1.6 shows the embedding for three nodes in KG, which can be used as features in an ML task. The big counterpart of embedding approaches in general is losing the originally explainable characteristics of complex data structures. As shown in Figure 1.6, the vector embedding itself does not keep the same relational expressiveness of each node within KG.

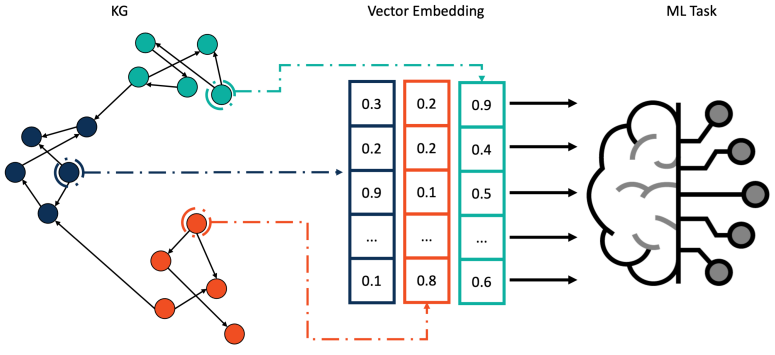


Figure 1.6: Overview of how nodes in KG can be represented as embedded vectors to be used in an ML task.

In the last decade, more automated methods to capture features inside the ML model were designed and researched. This led to a subarea of ML called Deep Learning (DL). DL directly takes the raw input data to make predictions. The features are learned together with the task [35]. Recurrent Neural Networks (RNN) is such a DL technique that can take into account the temporal effect in time series data [36]. To take graphs directly as input, Graph Convolutional Networks (GCN) were designed and applied in various applications [37]. DL techniques, by design, do not explain the obtained predictions.

1.2.3 Black and White-box

ML models where the raw data samples are directly used as input to the model, such as the DL models described in the previous section, or where the features are combined in a highly complex manner, are considered to be black-boxes [38]. Black-box ML models are unable to explain how a certain prediction was made. While in many cases black-box ML algorithms are highly valuable, they are not desirable in many applications, such as healthcare or predictive maintenance domain. Doctors are sceptical towards AI when the decision of the AI system can not be explained [39]. Similarly, a machine operator likes to know what is going wrong in order to rapidly fix the problem instead of just knowing that something is going wrong.

On the other side, white-box ML models are inherently explainable. The most well-known white-box ML algorithms are decision trees, where simple binary checks are performed on the provided features to come to a decision [40]. Figure 1.7 shows the example of which bike to buy using such a simple decision tree. The used approaches do require features to be engineered and their predictive performance is in certain cases lower than when black-box models are used.

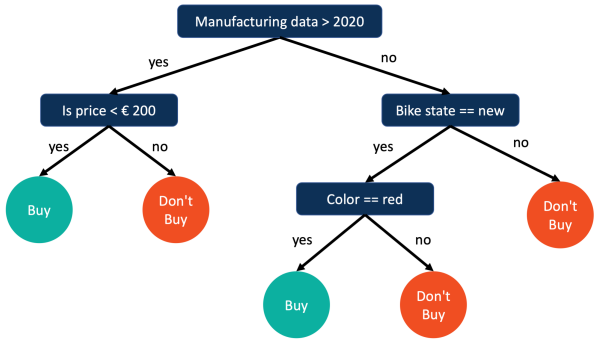


Figure 1.7: Decision tree example showing the binary checks that can be used to decide which bike to buy.

Some of these problems can be resolved by grey-box techniques or by providing local explanations of black-box techniques during a post-processing step. Grey-box

techniques can provide an explanation that is human-understandable but often still too complex to be interpreted efficiently [41]. Random forests [42] is such a technique, which consists of a large number of decision trees. Each individual decision tree is explainable, but the amount of trees is often so large that it becomes infeasible to look at each tree's prediction. Post processing techniques like Genetic Extraction of a Single Interpretable Model (GENESIM) and Born-Again Tree ensembles can resolve this issue by constructing a single decision tree of minimum size that reproduces the exact same behaviour as a given tree ensemble, like the Random forest approach, in its entire feature space [43, 44].

Post-hoc local explanations opened up a whole new emerging research area of eXplainable AI (xAI) [45]. Local Interpretable Model-agnostic Explanations (LIME) is such an xAI technique and works based on the assumption that every complex model is linear on a local scale [46]. LIME tries to fit a simple white-box model around a single observation that will mimic how the global model behaves at that specific locality. The simple model can then be used to explain the predictions of the more complex model locally. Another such post-hoc technique is shapley additive explanations (SHAP) [47]. SHAP quantifies the contribution that each feature brings to the prediction made by any model. All these techniques do, however, not provide a global explanation and do not expose all the internals of the complex model.

1.2.4 Data-driven problems

ML is a wonderful tool to know which bike to buy. It can try to learn patterns and deliver insights for a large number of available data records. But to teach a toddler how to ride a bike, this approach is somewhat limited. It is not that someone provides a bike to their child and lets it experiment with it for a large number of simulations, and label the outcome of these simulations to rather positive or negative experiences.

Providing labels for a certain dataset is, in a general context, not always possible. When using time series data, it is not possible to provide a label for each sample within the dataset. For complex tasks, it will even be necessary to have a team of experts to label the data due to the complex functioning of the system one is trying to examine. All these issues can make data labelling costly and slow and leads in many cases to labelling bias introduced in the data-driven pipeline [48]. In the end, models trained on such biased labels do not generalize over new, unseen samples which makes the trained ML models impractical for real-life cases.

Besides labelling bias being introduced in a data-driven context, bias can also be introduced during the creation of the dataset [49]. Again for complex tasks, it is not always easy to gather the correct data samples to learn an ML model that generalises over new, unseen data. For example, in the healthcare domain, data samples can be scarce due to a limited patient population. In a predictive maintenance domain, building a dataset with only correct and healthy system samples requires domain expertise.

In some fields or domains, it is also not possible due to the limited amount of data to achieve satisfying results with ML.

Data-driven approaches are still less explainable than the knowledge-driven variants. Correctly constructed datasets with, when possible, high-quality labels are needed to deliver this explainability. Once these aspects can be guaranteed, data-driven approaches can become an even more valuable decision-making tool for e.g. the eHealth and predictive maintenance tasks.

1.3 Hybrid AI

While it is not possible to transfer all of my bike riding knowledge to my toddler and it might cause him a lot of harm to just let him figure it out on his own, it seems that a combination of both some instructions that he could process in combination with his own observational trial and error approach leads to the eventual goal: riding a bike.

This is exactly what Hybrid AI (HAI) defines: it combines non-symbolic AI, such as ML and DL systems, with symbolic AI, such as inferences over KGs [50]. HAI integrates the fundamental cognitive abilities of intelligent agents, i.e. being able to learn from their environment, reason about what has been learned and share the acquired knowledge. In such integrated systems, ML approaches provide robust learning, whereas semantic models and knowledge representation allow for reasoning capabilities [51, 52].

An HAI system should ideally be adaptive and flexible to learn new concepts, interpret and relate concepts in the context of common and acquired knowledge, reason about hypotheses and consequences, take actions and communicate with other agents to collaboratively solve complex problems [53].

1.3.1 HAI Problems

The HAI solutions of today do incorporate expert knowledge within ML models [54]. These HAI applications fall in the realm of knowledge-infused learning [55] which helps ML and DL models in fields where a lot of expertise is required to solve a certain task (e.g. critical domain within healthcare). Knowledge is here only used to reduce the need for large qualitative datasets. **The available knowledge is not used to explain the decisions made.**

The expert knowledge used within these HAI applications is mainly static. This means that both the knowledge itself but also the knowledge-driven components are not updated when new insights or information is provided by the experts. Semantic rule mining is a technique which can derive those new insights or rules automatically. It is based on the principles of first-order inductive learning (FOIL) to obtain rules from a KG with accompanying ontology [56]. Such rule mining techniques are

however never performed on the combination of both knowledge- and data-driven insights in an HAI context.

Many of the applied HAI applications remain a loose combination of data- and knowledge-driven models that run in parallel. Many different hybrid learning patterns do exist to cover more complex cases [57]. But in practice, **no clear architecture implementation exists that complements knowledge-driven with data-driven techniques and vice versa such that both are strengthened.** Such an architecture is of high importance within domains where an HAI model has to adapt to new cases or in situations where the environment changes over time.

In many HAI applications, varying data such as time series are still being provided directly to the data-driven ML or DL components because they can be designed especially to make decisions based on these data types [58, 59]. **Only providing a specific type of data to a sub-component within an HAI application limits its applicability.** However, knowledge-based systems and especially the KGs were originally developed to be static. Incorporating time-dependent data within such a KG can be challenging [60].

1.3.2 HAI applications

The application field of HAI is very broad [53]. This dissertation covers two main of these HAI application domains: eHealth and predictive maintenance. Both fields have two things in common: First, the objective information inside these two fields comes from so-called sensors, which define the data as time series. In such a time series, every data sample is accompanied by timestamps, indicating when the obtained data value was measured by the accompanying sensor. Second, for both domains, a lot of domain knowledge is available, either as expertise built up by the predictive maintenance personnel or as evidence-based built-up medical knowledge by healthcare providers. Moreover, standardised ontologies exist for both domains

eHealth

Medical knowledge is widely available, enabling a huge impact on this research when it is automatically incorporated into AI tools. As more and more healthcare is shifted to homecare, systems are needed that can take into account the medical background of a patient in a monitoring system. When a certain deviation in the current behaviour of the patient is found by such a system, a nurse or doctor can be alerted to check and verify what is going wrong. Providing insights on which elements the HAI tools base their decisions on, will result in more trustworthiness by the staff. This has not only a societal impact but also an economic one, as it can lead to fewer unnecessary treatments and the use of fewer healthcare resources in hospitals.

Predictive maintenance

HAI applications in the predictive maintenance field encompass accurate Anomaly Detection (AD) and Root Cause Analysis (RCA), which are crucial to quickly detect malfunctioning components and plan appropriate interventions to ensure the continued correct functioning of an overall industry system. More insight into these anomalies and their causes will ensure that fewer resources are spent on identifying false alarms and that true alarms can be handled in a more controlled manner and with more support, communication and insights towards the end-users. The envisioned research outcomes will ensure that fewer unexpected situations occur, fewer unnecessary interventions are needed and less downtime happens, which has a huge economic impact. This also results in a societal impact as it leads to less travel and truck rollouts by service personnel and lower stress levels for the maintenance and operations personnel.

1.4 Research Challenges, Questions & Hypotheses

Based on the shortcomings of the data-driven, knowledge-driven and the identified gaps of HAI solutions discussed in the previous sections, we present five challenges, with according research question and hypotheses, which we will tackle in this dissertation and that are pertinent to solve and evaluate in order to realise truly value-adding HAI for healthcare & predictive maintenance applications.

C1: Reducing efforts to transform knowledge into machine-readable format.

To build accurate HAI applications, expert knowledge captured in a machine-readable format is of course needed. Capturing expert or domain knowledge can be a tedious task. Most of the time, an expert within a certain domain does not have the capabilities to construct a KG or an ontology. On the other hand, an ontology expert is not familiar with the expertise of the domain. A challenge of this dissertation is to capture and provide this machine-readable expert or domain knowledge in a more efficient manner costing less time and effort of the experts. The research question related to this challenge is defined as follows:

RQ1: Can existing, structured documents reduce the need for manually creating knowledge in HAI?

In many domains, existing ontologies cover parts of the operational field and describe concepts and relationships based on standardizations and general application fields, e.g. the SOSA/SSN ontology for Internet of Things/sensor applications [61]. These existing ontologies can be combined and used together to build up

an application-specific ontology. Most of the time, the expert's information is not only available in one's head, but additional structured documents are available describing the expertise and background knowledge on a particular system, its workflow and possible erroneous situations, e.g. anomalies, that can occur. We, therefore, investigated how these structured documents can be combined with existing ontologies to cover an application-specific domain of expertise. As those documents can be changed over time, a mechanism is needed to transform these updates and changes within the structured document into the ontologies or domain-specific KG. We hypothesise that this incorporation of knowledge results in a more complete and adaptable knowledge representation that can be constructed with less human effort.

RH1: The design of a methodology that can automatically extract knowledge from structured documents and link this knowledge to existing ontologies will lead to an adaptive and continuously evolving KG, without the need for an ontology expert to make these adaptations.

C2: Combine knowledge, data and predictions in one methodology.

To empower HAI to its fullness, guidelines on an overall methodology and architecture are required on how the data-driven and knowledge-driven techniques can be combined in such a fashion that they can iteratively exploit all the available knowledge, data and insights created by the different respective models in a performant and efficient manner. The research question related to this challenge is defined as follows:

RQ2: Can an architecture be devised that uses a KG to empower an entire HAI application and increase the explainability of its outcomes?

A combination of all knowledge and data is needed to exploit an HAI application. This means that not only the knowledge should be represented in a KG, but ideally, even the outcome of the ML models should be directly linked to the available knowledge and inputted data. It is also essential to exploit the available knowledge about the environment that the HAI model is operating in as it can lead to fewer false positives and limit the required computation resources, e.g. only the data linked to KG within a certain time interval should be analysed. The coupling of such contextual world knowledge on the one hand, and sensory interpretation or model outputs, on the other hand, is the real power of an HAI system. We hypothesise that when both the outputs from knowledge- and data-driven applications are combined, together with the available context data into one KG, HAI applications can provide the mechanisms to outperform knowledge- and data-driven techniques and are able to continuously improve their own system components.

RH2: A KG-central approach for both the data and knowledge-driven components

makes HAI outcomes explainable compared to its data-driven variants, and the HAI approach adaptable compared to its knowledge-driven variants.

C3: Performant and automated derivation of new knowledge for HAI.

As mentioned, manually generating facts or rules is a complex and time-consuming endeavour, which is a big drawback for expert systems. Moreover, it makes the knowledge-driven systems rather static as they cannot readily adapt to new situations or contexts. This requires manual interventions to add or update rules. In HAI applications, those new facts or rules would ideally be generated automatically based on the gained insights from the combination of data and knowledge. As such, a challenge of this dissertation is to investigate ways to incrementally expand the knowledge base used within HAI in an automated and performant manner. The research question related to this challenge is defined as follows:

RQ3: Can we optimise rule mining techniques for HAI applications by representing the KG in one data structure such that new insights and rules can be derived more accurately and efficiently?

As defined above in Section 1.1.2, reasoning mechanisms are either descriptive or prescriptive and the current rule mining techniques are so designed that the underlying representation of the KG has to be of a specific format to mine rules for one of these two reasoning methods. We, therefore, investigated how a new KG representation can be used to mine rules for both these two reasoning cases. We hypothesise that the creation of such a KG representation can be beneficial for both descriptive and prescriptive rules.

RH3: A KG rule mining technique which can mine rules for both descriptive and prescriptive reasoning cases, is in general 10% more accurate and requires less than 50% of the time to generate prescriptive rules while being able to generate 5% more rules in a descriptive setting on well-stated benchmark datasets and compared to the current state-of-the-art techniques.

C4: Explainable HAI

As discussed, a lot of performant ML models are black-box or grey-box, leading to non-explainable model outcomes. Moreover, to enable HAI complex data structures, such as KGs, modelling the knowledge needs to be incorporated into the ML pipeline. This requires feature engineering that is often performed by using embeddings that transform the complex data structure into a vector. However, this transformation leads to the features being no longer explainable, causing the overall HAI to be unexplainable. Therefore, a challenge is to research explainable KG feature engineering (or

embedding) techniques to enable explainable incorporation of knowledge into HAI. The research question related to this challenge is defined as follows:

RQ4: Can knowledge be incorporated in an ML model without losing the inherently explainable characteristics of the provided knowledge?

We investigated how KG could be represented such that they 1) keep their explainable aspects and can still operate in a knowledge-based system, and 2) the same structure can also be used in ML tasks. We hypothesise that the transformation of a KG into an ML readable format can both have a predictive advantage while retaining its explainable aspects.

RH4: A KG embedding technique keeping the inner explainable characteristics of the KG outperforms other KG embedding techniques with more than 2% in predictive performance on well-stated KG classification benchmark datasets.

C5: Incorporation of time-dependent data in HAI

Most applications in the eHealth and predictive maintenance domain provide volatile data in the form of sensor observations or more generally time series. While ML models have already proven to adequately incorporate these data formats, and methods exist to incorporate time-dependent information in KGs and accompanying inference engines, a challenge remains on how this data should be ideally modelled in a KG such that it simultaneously benefits the data-driven methods and knowledge-driven techniques within HAI applications. Moreover, a challenge is to perform this in such a manner that the explainability of the time-based information is not lost. The research question related to this challenge is defined as follows:

RQ5: Can time series and metadata efficiently be combined in a KG for HAI applications without losing explainability?

The incorporation of volatile sources into a KG is an interesting research domain. Especially in the case of many HAI applications, these sensor values and time series data are the mean input to be analysed. We hypothesise that when time series data can be integrated into a KG, HAI applications are able to outperform data-driven techniques while remaining explainable.

RH5: An HAI model relying on a KG linking time series data with domain knowledge is at least 10% more precise than single-model ML-based alternatives while maintaining explainability.

1.5 Outline

This doctoral dissertation consists of eight chapters, including this introduction chapter and a conclusion. Table 1.1 summarises how the different challenges from Section 1.3 are tackled by each research question and chapter.

Table 1.1: An overview of the challenges tackled by each research question (RQ) and the different chapters.

	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5
	RQ1	RQ2	RQ3	RQ4	RQ5
Ch 2	•				
Ch 3		•			
Ch 4			•		
Ch 5				•	
Ch 6					•
Ch 7					•
App A	•				
App B	•				

In what follows, we briefly introduce the contents of each chapter. Figure 1.8 positions the different contributions that are presented in each chapter and shows visually how the different chapters are linked. The different chapters of this book can be divided into three main parts: a knowledge creation, a knowledge incorporation and a knowledge exploitation part. The knowledge creation chapters and appendices will describe how a KG can be constructed given structured knowledge. The knowledge incorporation part covers the chapters that are responsible for transforming the available knowledge such that it can be used in the data- and knowledge-driven parts. The chapters inside the knowledge exploitation part describe how both the KG and the derived insights from the data and knowledge-driven models can be used in HAI applications and how the outcome of this overall application can be transferred back to the KG.

Each chapter is composed of one or several publications that were realised within the scope of this PhD research. The selected publications provide an integral and consistent overview of the work performed. The complete list of peer-reviewed publications that resulted from this work is presented in Section 1.6 below.

Chapter 2 describes a methodology to automatically transform structured documents to a KG for the predictive maintenance domain, requiring no expertise in knowledge modelling from the domain experts. The difference between this new paradigm and the expert or ontology central methodologies is discussed using a predictive maintenance use case within the railway domain. A second use case, describing how this methodology can be applied in a ventilation domain is provided in **Appendix A**. An explainable KG embedding technique can also help to provide additional knowledge by transforming structured expert documents in the form of tables

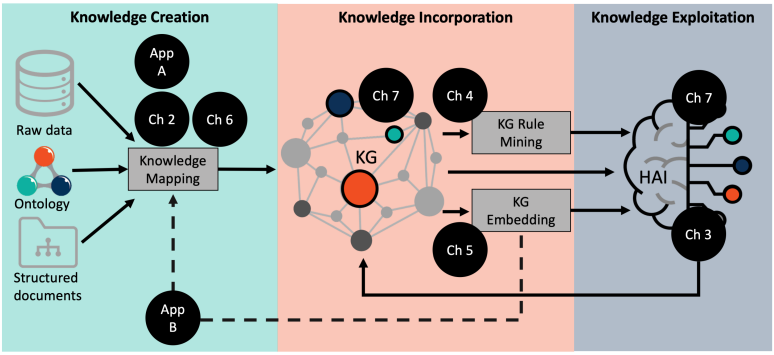


Figure 1.8: Schematic positioning of the different chapters in this dissertation

to existing linked data resources. **Appendix B** shows such an approach and even provides a technique to augment the given expert documents with new insights. This chapter and the appendices combined address challenge C1 and hypothesis RH1, and provide an answer to research question RQ1.

Chapter 3 proposes a new HAI architecture with a generated KG playing a central role. Both the data-driven and knowledge-driven techniques feed new knowledge into this KG. The newly imputed knowledge is later on used to generate new knowledge in the form of rules for the knowledge-driven models, while explanations are provided for the data-driven outcomes. The whole knowledge creation paradigm of Chapter 2 is also being used to define the initial knowledge in this HAI application. An evaluation of this new architecture was made using an anomaly and root cause analysis predictive maintenance use case. This chapter addresses challenge C2 and hypothesis RH2 and provides an answer to research question RQ2.

Chapter 4 goes beyond the rule-based limitations defined in this introduction and in Chapter 3. Here, a new rule mining technique, called INK, is described which can efficiently in terms of time and with good predictive performance generate rules for both descriptive and prescriptive cases. INK can be used to perform more accurate and efficient rule mining. An evaluation of INK for both these two descriptive and prescriptive cases is made on benchmark datasets. This chapter addresses challenge C3 and hypothesis RH3, and provides an answer to research question RQ3.

Chapter 5 uses the same underlying representation as in Chapter 4, namely INK, to describe and analyse an explainable KG embedding technique. This embedding is compared to other state-of-the-art embedding techniques, on a wide variety of benchmark datasets. It also shows how this embedding can be used in a predictive maintenance use case. This chapter addresses challenge C4 and hypothesis RH4, and provides an answer to research question RQ4.

Chapter 6 describes the data collection effort that was performed as part of this

dissertation to create a sensor dataset, with accompanying metadata, ontology and KG in a homecare setting. The data is built upon the eHealth use case, where elderly can be assisted by so-called smart homes to live longer at home and provide care in their homes. Preliminary technological prototype applications are suggested in this chapter that can benefit from the created connected eHealth KG, within such a use case. However, this dataset mainly filled the gap to work towards providing a dataset on which challenge C5 could be addressed, i.e. a data set with accompanying metadata and domain expertise for a real-life use case that contains time series data.

Chapter 7 uses the created resources of Chapter 6 as input for an eHealth HAI human activity recognition application and evaluation. Here, the sensor time series data is incorporated directly into the KG such that an HAI application can track indoor human activities. Knowledge-driven, time-dependent events are constructed such that the explainable embedding techniques of Chapters 4 & 5 can derive explainable features or rules to detect these activities within our HAI application while taking the time series information into account in an explainable fashion. This chapter addresses challenge C5 and hypothesis RH5, and provides an answer to research question RQ5.

1.6 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications obtained during my PhD research.

1.6.1 Publications in International Journals (listed in the Science Citation Index[1])

1. Sander Vanden Hautte, Pieter Moens, Joachim Van Herwegen, Dieter De Paepe, **Bram Steenwinckel**, Stijn Verstichel, Femke Ongenae, Sofie Van Hoecke, *A Dynamic Dashboarding Application for Fleet Monitoring Using Semantic Web of Things Technologies*, Published in Sensors, Volume 20, Issue 4, February 2020.

2. Dieter De Paepe, Sander Vanden Hautte, **Bram Steenwinckel**, Filip De Turck, Femke Ongenae, Olivier Janssens, Sofie Van Hoecke*A generalized matrix profile framework with support for contextual series analysis*, Published in Engineering Applications Of Artificial Intelligence, Volume 90, April 2020.

[1]The publications listed are recognised as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

3. Gilles Vandewiele, **Bram Steenwinckel**, Filip De Turck, Femke Ongenae*MINDWALC: mining interpretable, discriminative walks for classification of nodes in a knowledge graph*, Published in BMC Medical Informatics and Decision Making, Volume 20, Supplement 4, December 2020.

4. **Bram Steenwinckel**, Dieter De Paepe, Sander Vanden Hautte, Pieter Heyvaert, Mohamed Betefrit, Pieter Moens, Anastasia Dimou, Bruno Van Den Bossche, Filip De Turck, Sofie Van Hoecke, Femke Ongenae*FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning*, Published in Future Generation Computer Systems, Volume 116, March 2021.

5. Pieter Moens, Sander Vanden Hautte, Dieter De Paepe, **Bram Steenwinckel**, Stijn Verstichel, Steven Vandekerckhove, Femke Ongenae, Sofie Vanhoecke, *Event-Driven Dashboarding and Feedback for Improved Event Detection in Predictive Maintenance Applications*, Published in Applied Sciences, Volume 11, Issue 21, November 2021.

6. Dieter De Paepe, Sander Vanden Hautte, **Bram Steenwinckel**, Pieter Moens, Jasper Vaneesen, Steven Vanderkerckhove, Bruno Volckaert, Femke Ongenae, Sofie Van Hoecke*A Complete Software Stack for IoT Time-Series Analysis that Combines Semantics and Machine Learning-Lessons Learned from the Dyversify Project*, Published in Applied Sciences, Volume 11, Issue 24, December 2021.

7. **Bram Steenwinckel**, Gilles Vandewiele, Michael Weyns, Terencio Aggozino, Filip De Turck, Femke Ongenae,*INK: knowledge graph embeddings for node classification*, Published in Data Mining and Knowledge Discovery, Volume 36, Issue 2, January 2022.

8. Mathias De Brouwer,[2] Nicolas Vandenbussche[2], **Bram Steenwinckel**, Marija Stojchevska, Jonas Van Der Donckt, Vic Degraeve, Jasper Vaneesen, Filip De Turck, Bruno Volckaert, Paul Boon, Koen Paemeleire, Sofie Van Hoecke, Femke Ongenae,*mBrain: towards the continuous follow-up and headache classification of primary headache disorder patients*, Published in BMC Medical Informatics and Decision Making Volume 36, Issue 2, January 2022.

9. **Bram Steenwinckel**, Filip De Turck, and Femke Ongenae, *INK: Knowledge graph representation for efficient and performant rule mining*, Under review in Semantic Web Journal, May 2022.

10. Hannah K. Wayment-Steele, Wipapat Kladwang, Andrew M. Watkins, Do Soon Kim, Bojan Tunguz, Walter Reade, Maggie Demkin, Jonathan Romano, Roger Wellington-Oguri, John J. Nicol, Jiayang Gao, Kazuki Onodera,

[2]The first two authors share co-first authorship of this work.

Kazuki Fujikawa, Hanfei Mao, Gilles Vandewiele, Michele Tinti, **Bram Steenwinckel**, Takuya Ito, Taiga Noumi, Shujun He, Keiichiro Ishi, Youhan Lee, Fatih Öztürk, Anthony Chiu, Emin Öztürk, Karim Amer, Mohamed Fares, Eterna Participants, and Rhiju Das, *Predictive models of RNA degradation through dual crowdsourcing*, Published in Nature Machine Intelligence, September 2022.

11. Mathias De Brouwer, **Bram Steenwinckel**, Ziye Fang, Marija Stojchevska, Pieter Bonte, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae *Context-aware and privacy-preserving homecare monitoring through adaptive query derivation for IoT data streams with DIVIDE*, Revision submitted to Semantic Web Journal, August 2022.

12. Marija Stojchevska[2], **Bram Steenwinckel**[2], Jonas Van Der Donckt, Mathias De Brouwer, Annelies Goris, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae, *Assessing the added value of context during stress detection from wearable data*, Accepted to BMC Medical Informatics and Decision Making, September 2022.

13. **Bram Steenwinckel**, Mathias De Brouwer, Marija Stojchevska, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae, *TALK: Tracking Activities by Linking Knowledge*, Submitted to Engineering applications of artificial intelligence, October 2022.

1.6.2 Publications in Conference Proceedings

1. **Bram Steenwinckel**, Femke De Backere, Jelle Nelis, Femke Ongenae, and Filip De Turck, *Self-learning algorithms for the personalised interaction with people with dementia*, Published in the proceedings of the Artificial Intelligence Applied to Assistive Technologies and Smart Environments Workshop at the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI, New Orleans, USA, pages 153–158, June 2018.

2. **Bram Steenwinckel**, *[DC] Adaptive Anomaly Detection and Root Cause Analysis by Fusing Semantics and Machine Learning*, Published in the proceedings of the European Semantic Web Conference (ESWC), Heraklion, Greece, pages 272–282, June 2018.

3. **Bram Steenwinckel**, Pieter Heyvaert, Dieter De Paepe, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, Femke Ongenae, *Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses*, Published in the proceedings of the 9th international semantic sensor networks workshop, at the seventeenth International Semantic Web Conference (ISWC), Monterey, USA, pages 17–31, October 2018.

4. **Bram Steenwinckel**, Pieter Heyvaert, Dieter De Paepe, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, Femke Ongenae, *Automated extraction of rules and knowledge from risk analyses: a ventilation unit demo*, Published in the proceedings of the International Semantic Web Conference (ISWC) posters & demonstrations, Monterey, USA, pages 17–31, October 2018.

5. Gilles Vandewiele, **Bram Steenwinckel**, Femke Ongenae, Filip De Turck, *Inducing a decision tree with discriminative paths to classify entities in a knowledge graph*, Published in the proceedings of the 4th International workshop on semantics-powered data mining and analytics, at the eighteenth International Semantic Web Conference (ISWC) conference, Auckland, New Zealand, pages 1–6, October 2019.

6. **Bram Steenwinckel**[2], Gilles Vandewiele,[2], Filip De Turck, Femke Ongenae, *Csv2kg: Transforming tabular data into semantic knowledge*, Published in the Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, at the eighteenth International Semantic Web Conference (ISWC) conference, Auckland, New Zealand, pages 33–40, October 2019.

7. Michael Weyns, Pieter Bonte, **Bram Steenwinckel**, Filip De Turck, Femke Ongenae, *Conditional constraints for knowledge graph embeddings*, Published in the proceedings of the workshop on Deep Learning for Knowledge Graphs (DL4KG2020), co-located with the seventeenth Extended Semantic Web Conference (ESWC), online, June 2020.

8. **Bram Steenwinckel**, Gilles Vandewiele, Ilja Rausch, Pieter Heyvaert, Ruben Taelman, Pieter Colpaert, Pieter Simoens, Anastasia Dimou, Filip De Turck, Femke Ongenae, *Facilitating the Analysis of COVID-19 Literature Through a Knowledge Graph*, Published in the proceedings of the nineteenth International Semantic Web Conference (ISWC) conference, online, November 2020.

9. **Bram Steenwinckel**, Gilles Vandewiele, Pieter Bonte, Michael Weyns, Heiko Paulheim, Petar Ristoski, Filip De Turck, Femke Ongenae, *Walk extraction strategies for node embeddings with rdf2vec in knowledge graphs*, Published in the proceedings of the Database and Expert Systems Applications workshop, co-located with the International Conference on Database and Expert Systems Applications, online, pages 70–80, September 2021.

10. **Bram Steenwinckel**, Filip De Turck, Femke Ongenae, *MAGIC: Mining an Augmented Graph using INK, starting from a CSV*, Published in the proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the twentieth International Semantic Web Conference (ISWC) conference, online, November 2021.

11. Mathias De Brouwer, Nicolas Vandenbussche, **Bram Steenwinckel**, Marija Stojchevska, Jonas Van Der Donckt, Vic Degraeve, Filip De Turck, Koen Paemeleire, Sofie Van Hoecke, Femke Ongenae, *Towards Knowledge-Driven Symptom Monitoring and Trigger Detection of Primary Headache Disorders*, Published in the proceedings of the Web Conference (WWW), online, April 2022.

12. **Bram Steenwinckel**, Mathias De Brouwer, Marija Stojchevska, Jeroen Van Der Donckt, Jelle Nelis, Joeri Ruyssinck, Joachim van der Herten, Koen Casier, Jan Van Ooteghem, Pieter Crombez, Filip De Turck, Sofie Van Hoecke, Femke Ongenae, *Data Analytics For Health and Connected Care: Ontology, Knowledge Graph and Applications*, Published in the proceedings of the sixteenth EAI Pervasive Healthcare conference, Thessaloniki, Greece, December 2022.

13. Kyana Bosschaerts, Jeroen Stragier, **Bram Steenwinckel**, Lieven De Marez, Sofie Van Hoecke, Femke Ongenae, *Design of a social chatbot with gamification for user profiling and smoking trigger detection*, Published in the proceedings of the sixteenth EAI Pervasive Healthcare conference, Thessaloniki, Greece, December 2022.

References

[1] J. Joiner, M. Piva, C. Turrin, and S. W. Chang. *Social learning through prediction error in the brain*. NPJ science of learning, 2(1):1–9, 2017.

[2] J. McCarthy. *What is artificial intelligence*. URL: http://www-formal. stanford. edu/jmc/whatisai. html, 2004.

[3] F. Hayes-Roth. *The knowledge-based expert system: A tutorial*. Computer, 17(09):11–28, 1984.

[4] B. Campbell. *Towards Ontology-Guided Learning for Shepherding*. Shepherding UxVs for Human-Swarm Teaming, pages 115–130, 2021.

[5] K. Tripathi. *A review on knowledge-based expert system: concept and architecture*. IJCA Special Issue on Artificial Intelligence Techniques-Novel Approaches & Practical Applications, 4:19–23, 2011.

[6] C. L. Forgy. *Rete: A fast algorithm for the many pattern/many object pattern match problem*. In Readings in Artificial Intelligence and Databases, pages 547–559. Elsevier, 1989.

[7] A. Hogan. *Resource description framework*. In The Web of Data, pages 59–109. Springer, 2020.

[8] K. S. Candan, H. Liu, and R. Suvarna. *Resource description framework: metadata and its applications*. Acm Sigkdd Explorations Newsletter, 3(1):6–19, 2001.

[9] D. Fensel, U. Simsek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler. *Knowledge graphs*. Springer, 2020.

[10] X. Zhao, H. Chen, Z. Xing, and C. Miao. *Brain-inspired search engine assistant based on knowledge graph*. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[11] A. Ait-Mlouk and L. Jiang. *KBot: a Knowledge graph based chatBot for natural language understanding over linked data*. IEEE Access, 8:149220–149230, 2020.

[12] A. Breitfuss, K. Errou, A. Kurteva, and A. Fensel. *Representing emotions with knowledge graphs for movie recommendations*. Future Generation Computer Systems, 125:715–725, 2021.

[13] M. Yahya, J. G. Breslin, and M. I. Ali. *Semantic web and knowledge graphs for industry 4.0*. Applied Sciences, 11(11):5110, 2021.

[14] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al. *Knowledge graphs*. ACM Computing Surveys (CSUR), 54(4):1–37, 2021.

[15] G. Antoniou and F. v. Harmelen. *Web ontology language: Owl.* In Handbook on ontologies, pages 67–92. Springer, 2004.

[16] A. Fernandes. *Combining inductive and deductive inference in knowledge management tasks*. In Proceedings 11th International Workshop on Database and Expert Systems Applications, pages 1109–1114, 2000. doi:10.1109/DEXA.2000.875165.

[17] S. Ortona, V. V. Meduri, and P. Papotti. *Robust discovery of positive and negative rules in knowledge bases*. In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pages 1168–1179. IEEE, 2018.

[18] T. Ebisu and R. Ichise. *Graph pattern entity ranking model for knowledge graph completion*. arXiv preprint arXiv:1904.02856, 2019.

[19] N. Jain and V. Srivastava. *Data mining techniques: a survey paper*. IJRET: International Journal of Research in Engineering and Technology, 2(11):2319–1163, 2013.

[20] D. Liu, T. Gu, and J.-P. Xue. *Rule engine based on improvement rete algorithm*. In The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding, pages 346–349. IEEE, 2010.

[21] R. Akerkar and P. Sajja. *Knowledge-based systems*. Jones & Bartlett Publishers, 2009.

[22] R. Akerkar. *Introduction to artificial intelligence*. PHI Learning Pvt. Ltd., 2014.

[23] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[24] M. W. Berry, A. Mohamed, and B. W. Yap. *Supervised and unsupervised learning for data science*. Springer, 2019.

[25] M. Stojchevska, B. Steenwinckel, J. Van Der Donckt, M. De Brouwer, A. Goris, F. De Turck, S. Van Hoecke, and F. Ongenae. *Assessing the added value of context during stress detection from wearable data*. BMC Medical Informatics and Decision Making,, 1st revision.

[26] H. K. Wayment-Steele, W. Kladwang, A. M. Watkins, D. S. Kim, B. Tunguz, W. Reade, M. Demkin, J. Romano, R. Wellington-Oguri, J. J. Nicol, et al. *Deep learning models for predicting RNA degradation via dual crowdsourcing*. Nature Machine Intelligence,, 2nd revision.

[27] B. Steenwinckel, G. Vandewiele, I. Rausch, P. Heyvaert, R. Taelman, P. Colpaert, P. Simoens, A. Dimou, F. De Turck, and F. Ongenae. *Facilitating the analysis of COVID-19 literature through a knowledge graph*. In International Semantic Web Conference, pages 344–357. Springer, 2020.

[28] B. Steenwinckel, P. Heyvaert, D. De Paepe, O. Janssens, S. Vanden Hautte, A. Dimou, F. De Turck, S. Van Hoecke, and F. Ongenae. *Automated extraction of rules and knowledge from risk analyses: a ventilation unit demo*. In 17th International Semantic Web conference (ISWC 2018), volume 2180, 2018.

[29] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[30] B. Steenwinckel, F. De Backere, J. Nelis, F. Ongenae, and F. De Turck. *Self-learning algorithms for the personalised interaction with people with dementia*. In Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[31] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[32] M. Verleysen and D. François. *The curse of dimensionality in data mining and time series prediction*. In International work-conference on artificial neural networks, pages 758–770. Springer, 2005.

[33] Y. Kopotilov. *Using embeddings to make complex data simple*, Jun 2020. Available from: https://www.toptal.com/machine-learning/embeddings-in-machine-learning.

[34] K. W. Church. *Word2Vec*. Natural Language Engineering, 23(1):155–162, 2017.

[35] Y. LeCun, Y. Bengio, and G. Hinton. *Deep learning*. nature, 521(7553):436–444, 2015.

[36] H. Hewamalage, C. Bergmeir, and K. Bandara. *Recurrent neural networks for time series forecasting: Current status and future directions*. International Journal of Forecasting, 37(1):388–427, 2021.

[37] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. *Graph convolutional networks: a comprehensive review*. Computational Social Networks, 6(1):1–23, 2019.

[38] O. Loyola-Gonzalez. *Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view*. IEEE Access, 7:154096–154113, 2019.

[39] C. Farr. *Silicon Valley is trumpeting A.I. as the cure for the medical industry, but doctors are skeptical*, May 2017. Available from: https://www.cnbc.com/2017/05/27/ai-medicine-doctors-skeptical.html.

[40] S. B. Kotsiantis. *Decision trees: a recent overview*. Artificial Intelligence Review, 39(4):261–283, 2013.

[41] J. Wanner, L.-V. Herm, K. Heinrich, C. Janiesch, and P. Zschech. *White, Grey, Black: Effects of XAI Augmentation on the Confidence in AI-based Decision Support Systems.* In ICIS, 2020.

[42] G. Biau and E. Scornet. *A random forest guided tour.* Test, 25(2):197–227, 2016.

[43] G. Vandewiele, K. Lannoye, O. Janssens, F. Ongenae, F. D. Turck, and S. V. Hoecke. *A genetic algorithm for interpretable model extraction from decision tree ensembles.* In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 104–115. Springer, 2017.

[44] T. Vidal and M. Schiffer. *Born-again tree ensembles.* In International conference on machine learning, pages 9743–9753. PMLR, 2020.

[45] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek. *Explainable AI methods-a brief overview.* In International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers, pages 13–38. Springer, 2022.

[46] M. T. Ribeiro, S. Singh, and C. Guestrin. *" Why should i trust you?" Explaining the predictions of any classifier.* In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

[47] S. M. Lundberg and S.-I. Lee. *A unified approach to interpreting model predictions.* Advances in neural information processing systems, 30, 2017.

[48] R. S. Geiger, D. Cope, J. Ip, M. Lotosh, A. Shah, J. Weng, and R. Tang. *"Garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data?* Quantitative Science Studies, 2(3):795–827, 2021.

[49] R. Srinivasan and A. Chander. *Biases in AI systems.* Communications of the ACM, 64(8):44–49, 2021.

[50] M. Palmirani. *Hybrid AI to Support the Implementation of the European Directive.* In International Conference on Electronic Government and the Information Systems Perspective, pages 110–122. Springer, 2022.

[51] A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. *Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning.* arXiv preprint arXiv:1905.06088, 2019.

[52] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. *Building machines that learn and think like people.* Behavioral and brain sciences, 40, 2017.

[53] A. Meyer-Vitali, R. Bakker, M. van Bekkum, M. de Boer, G. Burghouts, J. van Diggelen, J. Dijk, C. Grappiolo, J. de Greeff, A. Huizing, et al. *Hybrid ai: white paper.* TNO Reports, 2019.

[54] H. Khayyam, A. Jamali, A. Bab-Hadiashar, T. Esch, S. Ramakrishna, M. Jalili, and M. Naebe. *A novel hybrid machine learning algorithm for limited and big data modeling with application in industry 4.0.* IEEE access, 8:111381–111393, 2020.

[55] U. Kursuncu, M. Gaur, and A. Sheth. *Knowledge infused learning (k-il): Towards deep incorporation of knowledge in deep learning.* arXiv preprint arXiv:1912.00512, 2019.

[56] B. Liu, X. Zhu, J. Wu, and L. Yao. *Rule reduction after knowledge graph mining for cyber situational awareness analysis.* Procedia Computer Science, 176:22–30, 2020.

[57] M. van Bekkum, M. de Boer, F. van Harmelen, A. Meyer-Vitali, and A. t. Teije. *Modular design patterns for hybrid learning and reasoning systems.* Applied Intelligence, 51(9):6528–6546, 2021.

[58] V. Nourani, Ö. Kisi, and M. Komasi. *Two hybrid artificial intelligence approaches for modeling rainfall–runoff process.* Journal of Hydrology, 402(1-2):41–59, 2011.

[59] F. Conte, F. D'Antoni, G. Natrella, and M. Merone. *A new hybrid AI optimal management method for renewable energy communities.* Energy and AI, 10:100197, 2022.

[60] T. Tietz, M. Alam, H. Sack, and M. van Erp. *Challenges of Knowledge Graph Evolution from an NLP Perspective.* In WHiSe@ ESWC, 2020.

[61] K. Taylor, A. Haller, M. Lefrançois, S. J. Cox, K. Janowicz, R. Garcia-Castro, D. Le Phuoc, J. Lieberman, R. Atkinson, and C. Stadler. *The semantic sensor network ontology, revamped.* In JT@ ISWC, 2019.

2

Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses

The construction of domain knowledge was previously limited to either two paradigms. In one paradigm, the ontology expert was in contact with a large number of people within the domain of interest and constructed the ontology and KG based on these findings. In another paradigm, tools and valuable training were provided to the experts within the domain to start and construct the ontology and accompanying KG by themselves. The result was always the same: neither solutions were adaptable to new scenarios as the ontology expert wasn't available after a designed use case or new experts weren't able to upgrade the used tools or follow new training sessions. In this chapter, we propose a new paradigm to resolve these adaptability issues by constructing the necessary domain knowledge from structured documents. In this way, experts are able to update the domain ontology and accompany KG by providing new information in the structured documents. The whole paradigm is tested and evaluated on a predictive maintenance use case in the railway domain. This paradigm and defined use case are also further used in Chapter 3 when a full KG-centred approach is being applied in an HAI setting. An additional evaluation of this methodology on a ventilation use case is made available in Appendix A. This chapter investigates research question 1: "Can existing, structured documents reduce the need for manually creating knowledge in HAI?" and validates hypothesis 1: "The design of a methodology that can automatically extract knowledge from structured documents and link this knowledge to existing ontologies will lead to an adaptive and continuously evolving KG, without the need for an ontology expert to make these adaptations.".

⋆ ⋆ ⋆

B. Steenwinckel, P. Heyvaert, D. De Paepe, O. Janssens, S. Vanden Hautte, A. Dimou, F. De Turck, S. Van Hoecke and F. Ongenae

Published in the Proceedings of the 9th International Semantic Sensor Networks Workshop, at the 17th ISWC Conference, October 2018.

Abstract
Connected sensors inside the device can analyse the environment and report possible unwanted behaviour. Current risk analysis tools, such as Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA), provide prior information on these malfunctions. A lot of people are involved during this process, resulting in disambiguations and incompleteness. Ontologies could resolve this issue by providing a uniform structure for the failures and their causes. However, domain experts are not always ontology experts, resulting in a lot of human effort to keep the ontologies up to date. In this chapter, a tool is developed to automate the mapping of the data from the FMEA to a domain-specific ontology and generate rules from a constructed FTA. The approach is demonstrated with a use case to investigate the possible failures and causes of reduced passenger comfort levels inside a train.

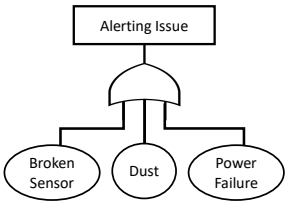
2.1 Introduction

Sensor monitoring systems are transforming industry, with game-changing applications in, e.g., transportation [1] and healthcare [2]. These systems can yield valuable insights into a company's physical assets and the interaction of these assets with their environment. However, sensors have limited added value without data analysis [3]. More and more, new methodologies are defined to specify the correct functioning of the system based on these sensor observations. Common methodologies for observing unwanted system behavior with this data are Anomaly Detection (AD) and Root Cause Analysis (RCA). AD is the identification process of events or observations, which do not adhere to the expected pattern or other items inside a dataset [2]. RCA guides the problem solver to deduce and understand the real causes of the anomalies [4]. Interest in AD & RCA will continue to grow as more relevant data is generated and tools become widely accessible that can handle data from diverse operating environments.

However, domain-specific knowledge needs to be leveraged to define the unwanted behavior and its causes inside these tools as sensor, or system behavior in general, varies wildly between application domains. This knowledge is often provided by domain experts by using formal documents, which define all the possible

Component	Function	Failure Mode	Failure Effect	Failure Cause	Control Method
Fire detector	Detects the fire	Unwanted evacuation	Falsely generated alarm	Broken sensor	None
Smoke sensor	Sense the smoke level in the environment	Values too high	Broken sensor	Dust accumulation	Clean system

(a)



(b)

Figure 2.1: Example of FMEA (a) and FTA (b)

failures and their (observable) effects on the system. Failure Mode and Effects Analysis (FMEA) [5] and Fault Tree Analysis (FTA) [6] provide templates to easily provide such so-called risk analyses. As shown in Figure 2.1 (a), FMEA captures, on multiple levels of the system, the potential failures that can occur to the components and their underlying causes and effects. FTA analyses the undesired states of a system using Boolean logic. This combination of low-level events, leading to system failures, can be visualised using a tree, as exemplified in Figure 2.1 (b).

Constructing these FMEA and FTA documents is a time-consuming process when applied thoroughly. A large number of experts are involved, who each have expertise on other parts of the system and interpret different parts of the risk analysis differently. Ambiguities, inconsistencies and duplicates are, therefore, quite common. This reduces the advantages of these risk analysis and makes it difficult for non-experts to interpret these document. Sharing, however, a common understanding about the structure of the system and contextual knowledge amongst the experts could help in separating the domain knowledge from the operational knowledge about the (mal)functioning of the system. Ontologies and accompanying inference rules have proven their worth in providing a common knowledge representation about a domain [7]. Consequently, ontology-based approaches have been proposed [8, 9] to structure the risk analyses, impose a common vocabulary and semantically link the different components, faults and causes. The information derived using these rules and ontologies is uniform. This reduces the previously mentioned ambiguities and inconsistencies of FMEA and FTA. However, most system experts are not familiar with ontology design, which makes these approaches difficult to implement. Semantic Web experts are required to constantly maintain and update these ontologies and rules with the domain-specific knowledge. Enabling domain experts to automatically generate ontologies and rules based on the domain knowledge captured in the FMEA and FTA documents lowers the barrier to use existing data analysis methodologies.

In this chapter, we propose an approach to automatically generate the required ontologies and inference rules from the aforementioned risk analysis outcomes. This removes the need for the involvement of ontology and rule experts in the risk analysis process. A first part of the approach uses declarative mapping rules to map FMEA

documents on domain-specific ontologies describing the components and their associated anomalies, causes and system effects. Second, predefined translation scripts are used to extract the inferences rules from the FTA trees. Both the mapping rules and scripts are generic and can be re-used for every new FMEA document and FTA tree. Only when the structure of the documents change, additional mappings or changes to the scripts will have to be provided. Our approach also provides methodologies to easily provide these changes with a minimum amount of human effort or knowledge about ontologies and inference rules. As such, the domain experts can focus on their primary task, i.e., applying their domain knowledge to accurately capture the unwanted behaviour of a system and its causes.

The remainder of the chapter is structured as follows. Section 2.2 situates our approach with respect to the related work. The designed approach is discussed in detail in Section 2.3, while Section 2.4 details the application of the approach on a real-life use case, i.e., investigating the possible failures and causes of reduced passenger comfort levels inside a train. Section 2.5 highlights the most important accomplishments and discusses the directions for future work.

2.2 Related work

As previously mentioned, ontology-based risk analysis methods have been proposed. Dittmann et al. [10] describe a process to capture the results of a FMEA in an ontology, instead of in a document, and highlighted the (dis)advantages. Rehman et al. [8] and Zhou et al. [11] designed high-level ontologies to model the main concepts of a FMEA and their relationships. The first applied it to model the results of a FMEA in the automotive domain. The second used it to capture the FMEA of wind turbines and developed a reasoning framework to perform intelligent fault diagnosis using the designed ontology capturing the domain-specific concepts. Both papers showed how an ontology can be used to easily trace the relationships between failures and their corresponding causes, making it easier to interpret the risk analysis. Ontologies to automatically link the observations made within a particular system to anomalies or faults that can occur, have also been proposed [12]. Although high-level concepts have been defined to model irregularities and link them to system components and effects, an ontology expert is required to model all the domain-specific anomalies that can occur and how they link to the sensor observations. None of the proposed ontologies are publicly available, hindering re-use. Moreover, all the approaches propose to replace the existing methodologies with a process in which the results of a FMEA are directly captured in an ontology. This requires extensive knowledge about ontology design from the system experts.

FTA has the advantage to be a more rigorous approach due to the step-by-step reasoning. Contrary to FMEA, FTA is a graphical method and already identifies the interrelations between concepts. As a result, FTA is more interpretable than FMEA as

the latter forces the analyst to decompose the system [13]. In an effort to automate the construction of the FTA trees, Venceslau et al. [9] defined an ontology to model the system components and failures and constructed a technique to automatically generate the FTA tree from the constructed ontology. The use of the ontology solves the issue of inconsistencies and ambiguities between FTA trees due to the lack of a common knowledge representation and the automatic generation of the tree ensures human understanding of the result. However, it again requires ontology design knowledge from the system experts.

While an ontology can capture the various concepts occurring within a domain and their intricate relationships, additional expressivity is required to derive that a fault has occurred out of the combination of various system observations. Rule languages, such as RuleML [14] and SWRL [15] can define inference rules, which are used inside a semantic reasoner to derive logical consequences. Recently, techniques have been designed to extract SWRL rules from text using NLP [16] or mine Semantic Web Association Rules from RDF data (SWARM) [17]. However, there are, to our knowledge, currently no techniques which allow the automated extraction of rules from risk analyses.

It can be concluded that currently no approaches exist that allow system experts to use their traditional risk analysis methodologies, i.e. FMEA tables and FTA trees, while still providing methods to automatically extract unambiguous and consistent ontologies and rules from them in a user-friendly manner.

2.3 User-friendly approach to extract knowledge from risk analyses

Defining rules which detect the failures based on the incoming sensor observations, in combination with a domain-specific ontology, enables the detection of irregularities and the derivation of their cause. To realize these rules and ontologies in a user-friendly manner, we propose an approach to automatically extract them from FMEA and FTA documents and trees, as visualized in Figure 2.2.

To ease the explanation of the different steps, a running example based on a smart fire detector will be used in this section. A part of the FMEA is visualized in Figure 2.1 (a) and it describes the possible failures of the available smoke sensor. A false alarm (failure effect) could be generated when dust accumulates in the device (failure cause), as it hinders the sensor from observing the environment correctly.

2.3.1 Folio ontology

Before we can define a methodology to extract knowledge from FMEA & FTA documents, a definition of the common concepts within the system risk analysis do-

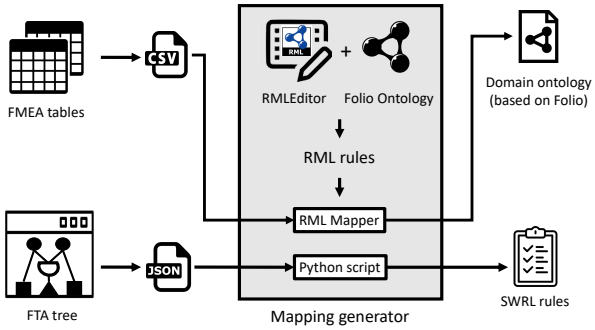


Figure 2.2: Overview of the approach to extract knowledge from risk analyses

main should be given. Therefore, an ontology was constructed, called Folio[1], which captures all application-independent concepts that occur within FMEA, FTA and anomaly detection methods. It is based on the aforementioned ontologies constructed by Zhou, et al. [11] and Pardo, et al. [12]. There are several concepts inside the FMEA template similar in the anomaly domain. The effects and causes of an anomaly can be related to the failure causes and effects, while both have detection methods and a degree of severity. Combining the concepts of both of them enables the derivation of the possible anomaly causes with the available knowledge inside the FMEA worksheets.

The `AnomalyKnowledge` concept defined inside Figure 2.3 and the related subclasses include all the possible anomaly information. These concepts were adapted to ensure applicability in a context of detecting anomalies for internet-connected devices and can determine the irregularities in streaming data. The Semantic Sensor Network (SSN) ontology[2] describes sensors and their observations for a diverse range of devices and is included in this upper ontology. The SSN architecture includes a lightweight, but self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties. With their different scope and different degrees of axiomatization, SSN and SOSA can support a wide range of applications and use cases. By using SSN & SOSA, the Folio ontology can describe the sensor's observations that are the basis for analyzing the system behavior. Relationships were defined in Folio to correlate the SSN concepts with possible failures and effects.

The FMEA concepts from Zhou, et al. were extended and related to the anomaly knowledge inside the Folio ontology. The `FailureEffect` and `Failure Cause` concepts are subclasses of the anomaly `Effect` and `Cause` concepts.

Relations between causes and effects are needed to describe the corresponding connections between multiple components. Figure 2.4 gives a detailed overview of

[1]Folio ontology: https://github.com/IBCNServices/Folio-Ontology/blob/master/Folio.owl
[2]SSN ontology: https://www.w3.org/TR/vocab-ssn/

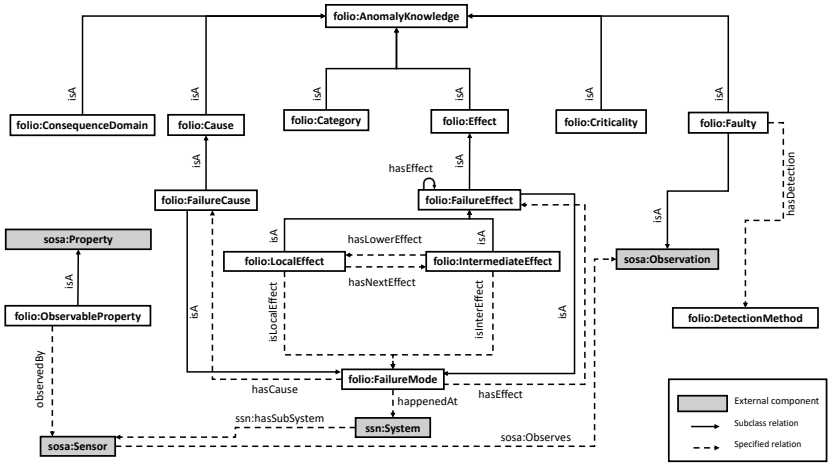


Figure 2.3: The Folio ontology.

these interrelationships. A `Cause` concept defines a concept with no further `hasNextEffect` relations. An `IntermediateEffect` concept will describe the influence of an intermediate component that is affected by, but not causing, the detected problem. The whole detection flow can have multiple `IntermediateEffect`s. The `LocalEffect` refers to the first detected effect onto the system. A `LocalEffect` will mostly be related to a faulty sensor observation itself, describing the current state of the device or system component. For the fire detector example, the accumulation of the dust will be defined as a `Cause`. The malfunctioning of the sensors are `IntermediateEffect`s and they could even have multiple causes. A `LocalEffect` could be a value too high notification, indicating something is wrong with the system.

2.3.2 Domain knowledge transformation

As shown at the top of Figure 2.2, the mapping of the entries of the FMEA tables on the defined Folio ontology consists of different steps, resulting in a domain-specific ontology. As such, anomaly knowledge can be extracted from the FMEA, and the causes of these anomalies can be derived by following the semantic links.

An FMEA is usually constructed using a spreadsheet program, resulting in a CSV document used for further analysis. The different possible elements of each record in the FMEA are fixed and defined by the column headers of the provided FMEA templates. Consequently, to enable the mapping of the FMEA on the Folio ontology, these column headers should be mapped on ontological concepts. To realize this, a mapping language was used, which enables the declarative definition of how to generate RDF from existing data sources through a set of rules. This approach is here preferred because mapping languages provide a reusable solution, while custom software

and mapping scripts are limited to a specific use case or implementation [18]. Another advantage is the adaptive character of the mapping rules: when making changes in the representation of the data (for instance, the risk analysts switch to a more advanced Failure Mode, Effect and Criticality Analysis method), updating the mapping rules will suffice to incorporate this extra information in the domain-ontology. Our approach uses the RML mapping language [18].

We defined the RML rules following the guidelines of the Folio ontology via the RMLEditor [19], which offers a graphical user interface to aid users in defining rules. The high levels steps we followed are as follows: (i) a sample of an FMEA table was loaded in the RMLEditor, (ii) the rules were created by an ontology expert, (iii) the corresponding RDF triples were generated, (iv) if the triples are not as expected the rules are updated, and (v) the rules are exported[3]. Afterwards, the RMLMapper[4], a tool to execute RML rules, is used to generate the ontologies for all FMEA tables. The mappings ensure that for each cell in the FMEA table, a new concept is created in the ontology, which is a subconcept of the concept on which the column is mapped according to the rules. For example, if we consider the 5th cell on the second row of Figure 2.1 (a), the RMLMapper will create a new concept `DustAccumulation` in the ontology, which has as superclass the `FailureCause` class.

As such, these mappings can be re-used to translate any FMEA table that is created according to the standard FMEA structure. Changes and additions to the FMEA documents do not affect the generation of the domain ontologies at all. If a new column is added, a new rule can easily be created to map this column to the Folio ontology by using the RMLEditor. Due to the frequently used FMEA templates, this will not happen often. In our fire detector example this means that updating the FMEA documents, by adding additional causes and failures, does not affect the generation process. The outcome of our mapping approach is a fire detector ontology in OWL, relating the failures of the temperature and smoke sensors to the general system effects, using the relationships of Folio.

2.3.3 Rule generation

Rules are helpful in determining the irregularities in the data through defining patterns. For example, a smoke detector can be defined as faulty, when it measures impossibly high values due to dust accumulation. This requires experts to adequately define the normal value ranges for these sensors. The Folio ontology and the previously explained FMEA mapping approach already allow to define the possible observations made by sensors. This section describes how rules can be extracted from the FTA trees to link these observations to possible faults that occur, by using the process visualized at the bottom of Figure 2.2.

[3] RML rules: https://github.com/IBCNServices/Folio-Ontology/blob/master/mapping.rml.ttl
[4] RMLMapper: https://github.com/RMLio/RML-Mapper

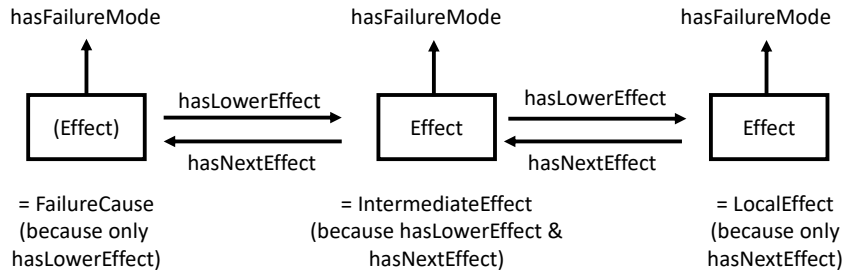


Figure 2.4: Overview of the relationships between the different effects.

While original FTA trees describe the relationship between the components of the system, they usually do not allow to differentiate the observations from their possible failures. In the case of the fire detector, the link between the sensor observations and all the possible failures shows the interaction of the different system components, but does not capture the difference between the accumulation of dust or, for example, a broken sensor. A FTA tree restricts the analysis to the relations between the components inside the system solely. Therefore, a combination of a decision tree, which is capable of modelling the decision from observations to failure with the possible consequences, together with the general FTA tree, is used here. This so-called decision fault tree (DFT) provides tests on the intermediate edges of the tree, visualising the basic rules for further analysis.

A user interface was designed to built such DFTs, as shown in Figure 2.7. In this editor, descriptions of the observation and failure nodes can be given. These different node concepts should align with the concepts defined in the FMEA. Tests describing the relations between these observations and failures can be added or adapted. Several representations are possible for such DFTs. The user interface outputs JSON file to describe the nodes and the rule-specific edges.

To translate the rules inside the tree to SWRL Rules, a rule generator script was designed in Python[5]. In a first step, the rules and nodes are gathered from the DFT inside JSON format. Second, RDF syntax rule definitions are used as mock-ups for the SWRL rules. These definitions specify all the basic boolean operations, as well the logical operators ($<, \leq, ==, \geq, >$). The JSON DFTs are then provided as input to these definitions, resulting finally in specific SWRL rules. These SWRL rules can be attached to the FMEA RDF document or can be saved separately. Again, the python script is defined once and is able to operate on all generated DFTs. When new fault tree knowledge becomes available, i.e. new types of operations, the script can be enhanced for further use. Changes and additions inside the DFT do not affect the rule generation engine at all. For example, the generated SWRL rule specifying a

[5]Script: https://github.com/IBCNServices/Folio-Ontology/blob/master/swrl_builder.py

`ValuesTooHigh` failure in the fire detector example looks as follows:

```
SmokeObservation(?o) ^  hasResult(?o, ?result) ^
swrlb:greaterThan(?Value, 50)  -> ValuesTooHigh(?o)
```

This rule describes the inference of a `ValuesTooHigh` failure when an observation is a `SmokeObservation` and the result of this observation is greater than 50.

2.3.4 Enabling adaptive AD and RCA

The ontologies and rules generated by using the full translation approach are the building blocks to determine unwanted behavior. They can be incorporated in a knowledge-based monitoring system to continuously identify anomalies and their causes. For example, the generated ontology and rules for the fire detection example were integrated in MASSIF, a data-driven platform for the semantic annotation of and reasoning on internet-connected data, allowing complex decision-making processing [20]. When new (sensor) observations are generated by the system, MASSIF semantically annotates them using the domain-specific ontologies, i.e. the fire detector ontology, generated by mapping the FMEA tables. MASSIF then uses a semantic reasoner to process the generated SWRL rules and links defined in the ontologies to determine whether failures are occurring and what their possible causes are. As such, the sensed data can be combined on the fly with background knowledge, resulting in enhanced and adaptive context-aware AD and RCA applications.

2.4 Use case: Measuring Train Passenger Comfort

The growing requirements for quality of service put new challenges on the operation and development of trains and railway tracks. Therefore, research on the passenger comfort levels has reached high interest in the last decade [21]. As shown in Figure 2.5, train bogies are now equipped with accelerometers and gyroscope sensors, able to detect the shocks and damping effect of the train on the tracks. Multiple sensor observations of different train cars can be combined on a server to indicate the passenger comfort inside the train. Maintenance alerts are given to both the train or track staff to resolve the issues.

The company installing these train monitoring units, i.e. Televic Rail, performed risk analyses. The resulting FMEA table, visualised in Figure 2.6, shows the possible failures of a disallowed comfort level that result in the effect of multiple falsely generated warnings for the train driver. Two possibilities are a broken or malfunctioning sensor. The FMEA table shows that the cause of the latter is varying outdoor temperatures while degradation causes the broken sensor. Replacing or recalibrating it could solve these issues.

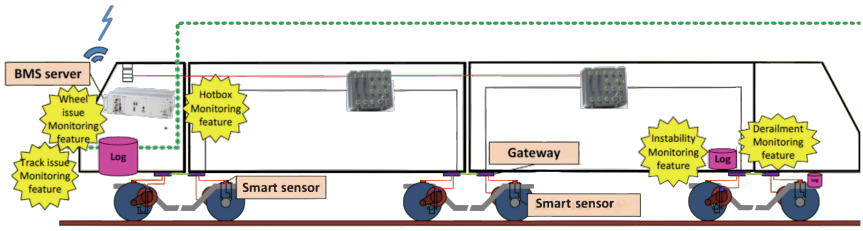


Figure 2.5: Schematic overview of a train.

Component	Function	Failure Mode	Failure Effect	I	Failure Cause	O	Control Method	D	RPN	Containment Action
Passenger Comfort Unit	Detects the level of Comfort	False warning	Indicating impossible comfort level	2	Broken sensor	6	None	2	24	None
Accelerometer Sensor	Measures changes in gravitational acceleration	Values too high	Broken sensor	6	Degradation of the sensor	6	None	4	144	Replace Sensor
			Malfunctioning sensor	8	Rapid temperature changes	7	None	8	448	Calibrate sensor
Gyroscope Sensor	Measure the smoke level	Values too high	Broken sensor	6	Degradation of the sensor	6	None	4	144	Replace Sensor
			Malfunctioning sensor	8	Rapid temperature changes	7	None	8	448	Calibrate sensor

Figure 2.6: Train passenger comfort FMEA example

A DFT was also modeled by Televic in the designed web interface, as shown in Figure 2.7. This tree describes the relationship between the temperature observations of the accelerometer unit and the humidity observations of the gyroscope sensor unit with their possible failure modes. A `ValuesTooHigh` failure can occur when the temperature of the accelerometer has either a value higher than 125 degrees Celsius or lower or equal than minus 40 degrees Celsius, or the humidity of the gyroscope has a value higher than 85%. All other observations are classified as normal in this simple use case.

The corresponding JSON file of the DFT and the CSV file of the table can be given as input to the mapping engine. The RML rules are here already predefined (same rules as defined in the fire detector example) and map the specific input fields to an RDF train-specific ontology. A schematic overview of the generated ontology is given in Figure 2.8 and visualises the major concepts of Figure 2.6. The inferred rules of the DFT, given in Figure 2.7, are visualised in Listing 2.1. This listing describes three SWRL rules corresponding with the paths from the sensor observations to the single failure mode. When an accelerometer temperature observation reaches the reasoning engine, and its value is greater than to 125 degrees Celsius, the observation will be classified as a `ValuesTooHigh` failure, and further actions can be taken.

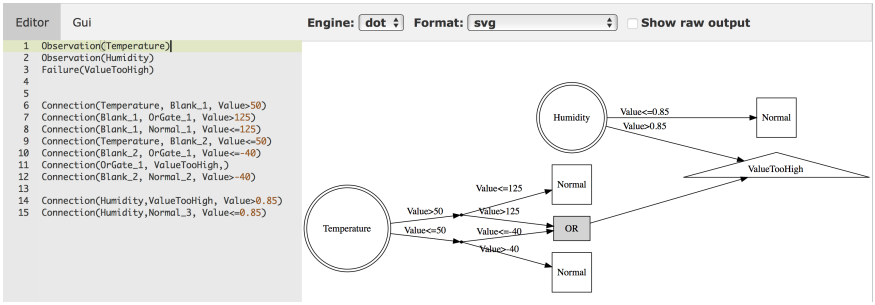


Figure 2.7: Train sensors DFT example

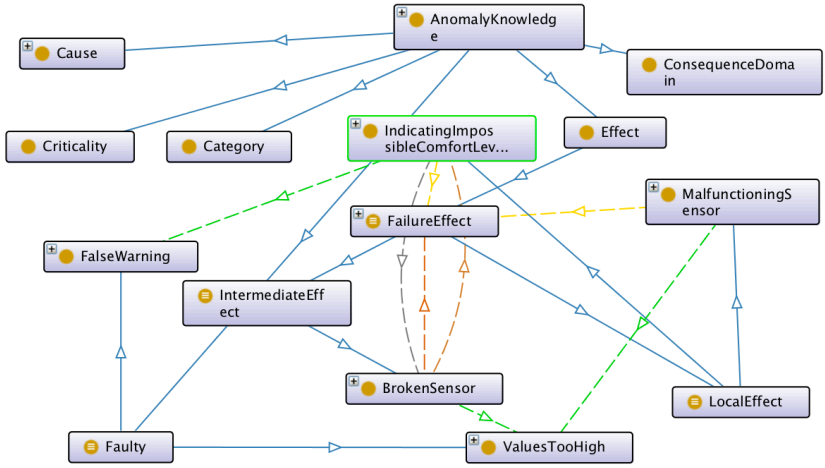


Figure 2.8: Ontograf visualisation of the passenger comfort FMEA ontology

```
HumidityObservation(?o) ^
hasResult(?o, ?result) ^
swrlb:greaterThan(?Value, 0.85) ^
hasValue(?result, ?Value)
-> ValuesTooHigh(?o)

hasResult(?o, ?result) ^
swrlb:greaterThan(?Value, 50) ^
hasValue(?result, ?Value) ^
TemperatureObservation(?o) ^
swrlb:greaterThan(?Value, 125)
-> ValuesTooHigh(?o)

swrlb:lessThanOrEqual(?Value, -40) ^
hasResult(?o, ?result) ^
hasValue(?result, ?Value) ^
TemperatureObservation(?o) ^
swrlb:lessThanOrEqual(?Value, 50)
-> ValuesTooHigh(?o)
```

Listing 2.1: SWRL rules derived from the DFT in Figure 2.7

2.5 Conclusion and Future work

In this chapter, a tool is proposed to enable the automatic knowledge extraction out of risk analyses into domain-specific ontologies and accompanying inference rules. This allows system experts to use the risk analysis methodologies and tools they are used to. The mapping of the resulting documents to ontologies and accompanying rules ensures that a common vocabulary and consistency check is maintained. Moreover, they can be used to enable on the fly detection of anomalies and their causes through semantic reasoning. It enables the system experts to focus on the risk analysis task, instead of on a knowledge modelling task for which they do not have the adequate ontology design expertise. Future research can now use the designed ontologies, together with accompanying rules to derive or reason on the possible causes.

Acknowledgment: This research is part of the imec ICON project Dyversify, co-funded by imec, VLAIO, Renson Ventilation NV, Televic Rail & Cumul.io.

References

[1] E. Camossi and et al. *Semantic-based Anomalous Pattern Discovery in Moving Object Trajectories*. CoRR, abs/1305.1, 2013.

[2] I. Souiden and et al. *A survey on outlier detection in the context of stream mining*. In Advances in Intelligent Systems and Computing. 2017.

[3] YE. *Big data: Changing the way businesses compete and operate*, 2014.

[4] M. Solé and et al. *Survey on Models and Techniques for Root-Cause Analysis*. Clinical Orthopaedics and Related Research (CoRR), 2017.

[5] M. Ben-Daya. *Failure mode and effect analysis*. In Handbook of maintenance management and engineering, pages 75–90. Springer, 2009.

[6] C. A. Ericson. *Fault tree analysis*. Hazard analysis techniques for system safety, pages 183–221, 2005.

[7] J. Ye and et al. *Semantic web technologies in pervasive computing*. Pervasive and Mobile Computing, pages 1–25, 2015.

[8] Z. Rehman and C. V. Kifor. *An Ontology to Support Semantic Management of FMEA Knowledge*. International Journal of Computers, Communications & Control, 2016.

[9] A. Venceslau and et al. *Ontology for computer-aided fault tree synthesis*. In Emerging Technology and Factory Automation (ETFA), 2014 IEEE, pages 1–4. IEEE, 2014.

[10] L. Dittmann and et al. *Performing FMEA using ontologies*. In 18th International Workshop on Qualitative Reasoning. Evanston USA, pages 209–216, 2004.

[11] A. Zhou and et al. *A research on intelligent fault diagnosis of wind turbines based on ontology and FMECA*. Advanced Engineering Informatics, 29(1):115–125, 2015.

[12] E. Pardo and et al. *A Framework for Anomaly Diagnosis in Smart Homes Based on Ontology*. Procedia Computer Science, 83, 2016.

[13] J. Peeters and et al. *Improving failure analysis efficiency by combining FTA and FMEA in a recursive manner*. Reliability engineering & system safety, 172:36–44, 2018.

[14] H. Boley and et al. *Design rationale of RuleML: A markup language for semantic web rules*. In Proceedings on Semantic Web Working, pages 381–401. CEUR-WS, 2001.

[15] I. Horrocks and et al. *SWRL: A semantic web rule language combining OWL and RuleML*. W3C Member submission, 21:79, 2004.

[16] S. Hassanpour and et al. *Framework for the automatic extraction of rules from online text*. In Workshop on Rules and Rule Markup Languages. Springer, 2011.

[17] M. Barati and et al. *Swarm: approach for mining association rules from semantic web data*. In Conference on Artificial Intelligence, pages 30–43. Springer, 2016.

[18] A. Dimou and et al. *RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data.* In LDOW, 2014.

[19] P. Heyvaert and et al. *RMLEditor: a graph-based mapping editor for linked data mappings*. In International Semantic Web Conference, pages 709–723. Springer, 2016.

[20] P. Bonte and et al. *The MASSIF platform: a modular and semantic platform for the development of flexible IoT services.* Knowledge and Information Systems, 2017.

[21] T. G. Karimpanal, H. M. Gadhia, R. Sukumar, and J. Cabibihan. *Sensing discomfort of standing passengers in public rail transportation systems using a smart phone.* CoRR, 2017.

3

FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning

Based on the provided knowledge gathered using structured documents in the previous chapter, an HAI application can be constructed by fusing knowledge-driven and data-driven AI components. This chapter explains the methodology to create such HAI applications and is evaluated using a predictive maintenance use case. The methodology shows the central role of a KG to store expert information. The KG also stores the outputs of the data-driven components, which can further on being used as inputs in the knowledge-driven parts to provide useful insights. Knowledge can be further extended using the adaptive paradigm provided in Chapter 2, but new knowledge in the form of rules is also being derived automatically, based on this KG using a semantic rule mining technique. Based on the findings in this HAI application, a new semantic rule mining method was developed that can better handle the underlying representation of knowledge within such HAI applications. This new semantic rule mining technique is further discussed in Chapter 4. This chapter investigates research question 2: "Can an architecture be devised that uses a KG to empower an entire HAI application and increase the explainability of its outcomes?" and validates hypothesis 2: "A KG-central approach for both the data and knowledge-driven components makes HAI outcomes explainable compared to its data-driven variants, and the HAI approach adaptable compared to its knowledge-driven variants.".

⋆ ⋆ ⋆

B. Steenwinckel, D. De Paepe, S. Vanden Hautte, P. Heyvaert, M. Betefrit, P. Moens, A. Dimou, B. Van Den Bossche, F. De Turck, S. Van Hoecke, F. Ongenae

Published in Elsevier Journal of Future Generation Computer Systems, Volume 116, March 2021.

Abstract
Anomalies and faults can be detected, and their causes verified, using both data-driven and knowledge-driven techniques. Data-driven techniques can adapt their internal functioning based on the raw input data but fail to explain the manifestation of any detection. Knowledge-driven techniques inherently deliver the cause of the faults that were detected but require too much human effort to set up. In this chapter, we introduce FLAGS, the Fused-AI interpretabLe Anomaly Generation System, and combine both techniques in one methodology to overcome their limitations and optimize them based on limited user feedback. Semantic knowledge is incorporated in a machine learning technique to enhance expressivity. At the same time, feedback about the faults and anomalies that occurred is provided as input to increase adaptiveness using semantic rule mining methods. This new methodology is evaluated on a predictive maintenance case for trains. We show that our method reduces their downtime and provides more insight into frequently occurring problems.

3.1 Introduction

Sensor monitoring systems are transforming the industry, steered by the so-called Internet of Things (IoT) and Artificial Intelligence (AI) fields, through game-changing applications in, e.g., transportation [1], security [2], ventilation [3] and healthcare [4]. For example, in the railway domain the number of sensors deployed on a single train bogie ranges from 10 to 50. A wide variety of sensors are used, such as accelerometers to monitor vibrations, ultrasonic, inductive and draw-wire range sensors, shock pulse sensors or gyroscopes for rotational speed. Monitoring these sensors can deliver valuable insights into the physical assets, the performance, and the interaction with the environment. For example, a bogie monitoring system can be used to assess the state of the wheel bearings, fatigue in the bogie, driving comfort for the passengers and the train body tilting.

Sensor monitoring systems analyze so-called sensor networks and can be used to detect faulty or deviating system behavior using methodologies such as Anomaly Detection (AD), Fault Recognition (FR) and Root Cause Analysis (RCA).

In the area of sensor networks and streaming data, AD consists of finding unknown patterns or outliers in unlabeled data when something unusual occurs or when the conditions deviate from the normal behavior [5]. FR captures the stronger pat-

terns as the condition develops, and the system's operation deteriorates towards failure. Once a pattern for a specific fault has been identified, it can be referenced in the future when the pattern emerges again. This approach can be used to explain what is currently going wrong [6].

RCA is the process of deducing and understanding the underlying cause of the occurring anomalies or faults [7].

Combined, AD, FR and RCA allow end-users to accurately pinpoint problems, mediate them and prevent further escalations. For example, proprietary analysis tools are used by train maintenance personnel to analyze the accelerometer data from bogie-mounted sensors and to identify possible wheel issues.

Two main techniques exist to perform AD, FR and RCA in sensor networks: those that are data-driven and those that are knowledge-driven. The first technique derives anomalies or recognizes faults directly from the streaming data by identifying unusual patterns using machine learning (ML). The second technique encodes expert knowledge about the systems, e.g., expected sensor ranges during normal behavior, to detect known or unknown behavior in the data streams.

As discussed in detail in Section 3.3, both have limitations. The knowledge-driven techniques are highly interpretable, context-aware and have a low false-positive rate. However, they cannot update new fault- or anomaly-related knowledge on the fly. They are also unable to learn new anomalies automatically and require much human effort to construct and maintain. When used in sensor monitoring systems, data-driven techniques are adaptive and require minimal human effort to start analyzing a data stream. Nevertheless, the data-driven techniques lead to many false positives as they are not context-aware and are often uninterpretable.

The drawbacks of both techniques cancel each other out. The fuse of both allows extracting interpretable alerts in highly dynamic environments with a reduction in human involvement. However, developing an efficient methodology that combines both data- and knowledge-driven techniques remains a considerable challenge.

Therefore, the objective of this chapter is the introduction of a methodology to efficiently combine data- and knowledge-driven techniques towards optimizing AD, FR and RCA for sensor monitoring systems. This fused methodology tackles the above mentioned drawbacks of current AD, FR and RCA systems. A prototype implementation of this methodology, called FLAGS (Fused-AI interpretabLe Anomaly Generation System), is also presented. FLAGS is evaluated on a predictive maintenance use case to illustrate its benefits, i.e., decreased number of falsely generated alerts, almost no human involvement needed to adapt towards new environments and providing interpretable causes for the occurred anomalies.

Our approach makes the following contributions:

- The output of both the data- and knowledge-driven techniques are combined to (i) pinpoint the unwanted behavior and (ii) reduce the number of falsely generated and missed alerts when compared to the data-driven techniques, while

(iii) requiring less human involvement than the knowledge-driven techniques.

- Expert knowledge is fused into data-driven techniques, resulting in RCA algorithms that deliver the most likely causes of the anomalies or derive interpretations to start data-driven FR.

- AD, FR & RCA are combined and take the available context into account to readily adapt to new contexts of deployment and configurations, while reducing the amount of required data about this new context.

- Dashboard applications can gather non-intrusive user feedback on the detected faults, outliers and their causes. Here, user feedback is crucial for dealing with the continually changing or unknown deployment environments and enables the optimization of AD, FR, and RCA algorithms. This feedback is used by many components to improve the whole monitoring system automatically.

The remainder of the chapter is structured as follows. Section 3.2 gives an overview of the requirements needed to fulfill a fused AD, FR and RCA methodology. Section 3.3 gives an overview of the two most common techniques to perform AD, FR and RCA and describes why not all of the requirements mentioned in the previous section can be met. The proposed end-to-end methodology for the derivation of highly accurate anomalies and their interpretable causes from sensor monitoring streams through adaptive and context-aware AD and RCA is discussed in Section 3.4. The methodology itself is evaluated using a train bogie monitoring use case in Section 3.5. Section 3.6 shows the results in terms of efficiency and performance of the FLAGS methodology based on the train monitoring use case. Section 3.7 discusses the methodology with respect to the requirements while Section 3.8 summarizes the approach and lists future work.

3.2 Requirements

Several requirements should be met to analyze anomalies, faults and determine their causes in a sensor network.

Sensor monitoring systems are deployed on a plethora of devices, with various configurations and within varying contexts. This makes AD, FR & RCA a challenging task. No upfront knowledge is available about the actual configuration or the complex environments or domain they are being deployed. However, knowledge about this deployment environment and used configuration can have a severe impact on whether an anomaly or fault has been occurred. For example, bogie monitoring systems are deployed on various types of trains, driving on a wide variety of tracks all over the world. The threshold indicating whether a temperature observation is anomalous varies widely between these locations.

Moreover, knowledge also influences the assessment of these anomalies and provides more information about a possible cause. For example, when multiple trains report vibration anomalies at the same location, the track is probably faulty and not the trains.

The knowledge about the anomaly or fault, together with the raw sensor data, gives greater interpretability to those who follow up on the problem posed. For example, maintenance can be requested faster when a temperature sensor of the wheel axle reports high values given the context that the train is riding through Siberia during the winter months.

As more information is provided to the end-users or operators, their actions on the detected anomalies and faults provide useful information for future investigations. As sensor environments can change rapidly, the sensor monitoring systems must be updated regularly with this new information to guarantee the correct functioning of the monitoring units. In our example, whether or not the operator requests maintenance for a particular detected anomaly or fault can indicate the urgency of a detected anomaly. However, a lot of human involvement is required when the operator has to label all the data points corresponding to the occurring event. These tasks are even seen as useless to the operator when no direct improvement is notified while relabeling.

Fusing knowledge into data-driven techniques can improve the detection rate of interesting anomalies. In the previous example, incorporating knowledge about the external temperature would improve the detection rate of wheel axle faults or anomalies. Analysis of a plethora of industry defined use cases for AD, such as the one outlined above, have led to the following requirements:

- **Precise:** The detection of anomalies should be accurate to reduce the number of false positives, which usually overwhelms the operator with annoying alerts. These alerts now lead to less timely interventions, increased stress and less detailed investigations of real problems. On the other hand, accurately detecting anomalies should also reduce the number of false negatives, as this lead to undetected problems that can escalate. In critical domains, the false positive rate should be lower than 70% to reduce the current burden and stress of the operators trying to resolve them [8].

- **Require minimal human involvement:** Human involvement should be minimized to deal with the lack of continuous access to the deployment environment. Industry partners state that the operators should be able to steer the functioning of the monitoring unit by providing simple feedback on the system's outcomes for more than 50% of the time.

- **Context-aware:** Providing context to data-driven detections results in the reduction of false positives as more information is available to discriminate wrong

from correct behavior. Industry partners estimated that context incorporation is only beneficial when the reduction in false positives can be noticed.

- **Adaptive:** An effective AD, FR & RCA sensor monitoring system should be capable of adapting the detection behavior to changing conditions in the deployment environment or system configuration while still recognizing anomalous or faulty activities. If not, the system is either not operable in a streaming environment or large numbers of fault positives will be generated due to the lack of adaptability [9]. The system should be able to adapt its detection behavior in less than 10 seconds after the operator or end-user has indicated changes in behavior. This upper limit is required to avoid losing the users' attention completely [10].

- **Interpretable:** The detection model is highly interpretable if end-users, i.e., operators, are able to quickly plan the appropriate mediation actions for the detected anomalies. Interpretable results should increase the efficiency of an operator's intervention effectiveness with more than 15% [11].

To adhere to all requirements, on the one hand, we need to deliver cost-effective and value-adding AD, FR & RCA techniques. On the other hand, these techniques must be combined and made to reinforce each other. By doing this, the human involvement can be reduced which is required to tune them for long-term tasks.

3.3 Related Work

This section gives an overview of the two current techniques, i.e., data- & knowledge-driven, to perform AD, FR & RCA in the context of sensor networks. We compare both approaches with respect to the requirements of Section 3.2.

3.3.1 Knowledge-driven FR & RCA

When knowledge is provided by field experts to describe a particular system problem, such a problem is referred to as known faults. Knowledge-driven FR & RCA methodologies consist of two steps, as shown in Figure 3.1: knowledge acquisition and knowledge transformation. The first aims to capture the existing guidelines and knowledge of domain experts on the expected normal behavior of a certain system or the faults that can occur with their possible underlying causes. During the second step, this information is transformed into software, e.g., rules, that can extract insights from the incoming data. These insights are the description of the detected faults and their causes.

Knowledge acquisition can be performed through risk analysis. Two types of risk analysis are prevalent, i.e. Failure Mode and Effects Analysis (FMEA) [12] and Fault

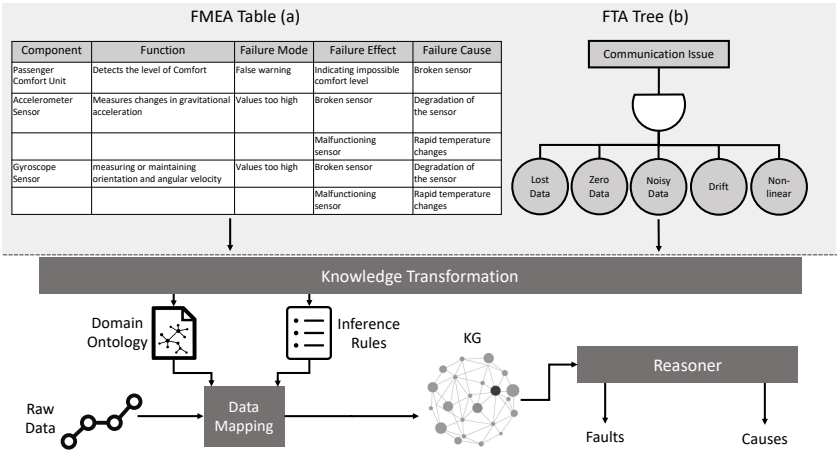


Figure 3.1: The knowledge-driven AD & RCA methodology. The top part illustrates an FMEA table (a) and FTA tree (b). The lower part shows how this knowledge from both risk documents and the raw sensor data is incorporated through a mapping script into a KG. At last, a rule-based reasoner can be performed to detect faulty behavior with the accompanying cause.

Tree Analysis (FTA) [13]. Both are visualized in Figure 3.1(a) and (b), respectively. FMEA is an inductive technique[1] that captures the potential failures in the system components, together with their underlying causes and effects. By contrast, FTA is a deductive technique that uses Boolean logic to analyze the undesired states of a system to see which lower-level event caused it.

Constructing these FMEA and FTA documents is a time-consuming process as many experts are involved. As each of them has expertise in other parts of the system, they interpret the risk analysis differently. All these different interpretations result in ambiguities, inconsistencies and duplicates. Moreover, no links between the specified anomalies, causes, the expected system behavior and the contextual deployment are defined within these documents. Without such links, it is rather hard to get a clear overview of the system's functioning.

Ontology-based risk analysis methods have been designed to overcome these problems [14]. They provide high-level ontologies, e.g., the FOLIO ontologies[2], and rules that allow to semantically model the expert knowledge from FMEA & FTA analyses. By employing the Linked Data approach, this expert knowledge can then easily be related to knowledge on the incoming data by enriching these streams through domain & system ontologies, e.g. the Semantic Sensor Network (SSN) ontology [15].

[1] An inductive method starts with many observations, to find a few, powerful statements. It is the opposite of a deductive technique, which starts with a few true statements (axioms) to prove many true statements (theorems) that logically follow from them.

[2] https://github.com/IBCNServices/Folio-Ontology

Combining SSN and FOLIO enables the consolidation of the data streams and makes the device properties and the gathered context explicit. However, ontology engineers are required to improve and update these models with new domain knowledge constantly. As system experts are not familiar with ontology design, methods have been investigated to transform the existing FMEA tables, FTA trees or even free text automatically to ontological models and inference rules [16].

When the constructed ontologies and rules have been generated, they can be used to annotate incoming raw data. The appropriate metadata is used to link them to the available background knowledge, as shown in the bottom part of Figure 3.1. This results in a so-called Knowledge Graph (KG), where the data is linked with the domain metadata. Commonly available semantic reasoners, e.g., Hermit [17] and Pellet [18], can be used to interpret this semantic data to detect possible faults and infer causes using the ontology and rules.

In sensor networks, reasoning on a stream of semantic data is more commonly known as Semantic Complex Event Processing (SCEP) [19]. The available rules can describe the semantic complex event, which in our case signifies a fault. A full comparison of all available SCEP systems is out of scope for this chapter. There are already exist systems which can process complex events over high-velocity streams consisting of up to hundreds of events per second [20].

As these models are interpretable, they can explain how they derive conclusions or offer more insight for the end-user. Moreover, the in-depth knowledge of the domain experts, their evidence and the carefully curated and maintained guidelines lead to a low number of false positives. No data on normal system behavior is required to build the FR and RCA system. By explicitly linking the defined faults and causes to the system properties and the context in which they are valid, the operating systems can readily adapt to other known contexts and deployments.

This methodology also has its limitations. The logic models can range from relatively simple rules, e.g., setting thresholds on measured parameters and statistically derived features, to highly complex, e.g., intricate ontologies with associated rule sets. A profound understanding of the domain and a large amount of human effort are required to construct and maintain these ontologies and rules. Moreover, these systems cannot learn new anomalous behavior without providing new knowledge manually. They are unable to adapt automatically to dynamic environments or previously unknown contexts. Not being able to adapt the detection behavior leads to undetected anomalies and the constant need for human involvement in changing the logical model based on new expert insights, their know-how and their efforts.

3.3.2 Data-driven AD & RCA

Data-driven AD and RCA employ ML to directly learn a model from the data collected by the sensor monitoring systems [21]. A preprocessing step can be required to

transform the data into a set or vector of useful features. The overview in Figure 3.2 shows how the raw data must be first preprocessed before both the AD and RCA modules can use it.

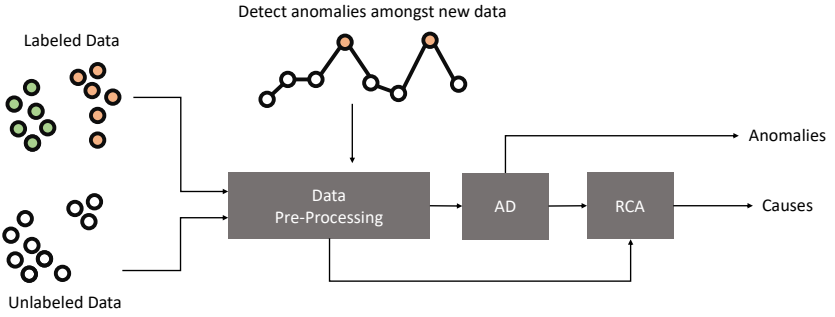


Figure 3.2: The data-driven AD & RCA methodology. After prepossessing either labeled or unlabeled data, the corresponding features can be given to the AD module which detects anomalies. The RCA module takes both the features and anomalies to generate causes.

Specialized ML techniques are required to perform AD in the context of sensor networks. As most IoT systems provide streaming sensor data, methods are required to cope with time series data either directly or by preprocessing the time series to discrete events [22]. Anomalies occur infrequently and can be either uni- or multivariate. Techniques are needed that can cope with such inherent imbalance. For multivariate anomalies, multiple sensor streams need to be investigated simultaneously to make a correct decision. Based on these properties, three general categories of ML-based AD can be discerned, i.e. supervised, unsupervised and semi-supervised [23].

Supervised AD requires data where the instances have been labeled as normal or anomalous. Note that more than one normal or anomalous class can exist. Traditional classification techniques can be used by leveraging methods to deal with the inherent imbalance of the data set. Common methods are undersampling of the normal data, oversampling of the anomalous data or cost-sensitive learning punishing the misclassification of anomalies harder than that of normal samples. Supervised AD also assumes that new anomalies will be similar to past ones. Popular supervised AD methods that can deal with time-series data are Recurrent Neural Networks (RNN, e.g., LSTM) and Support Vector Machines (SVM) [24].

In many sensor streaming situations, class labels are not available due to the relatively large number of samples needed to train the model. Unsupervised AD techniques try to detect outliers by assuming that most of the data is normal behavior. Here, the objective is to assign a score (or a label) to each instance that reflects its degree of normalcy. For unsupervised AD to be successful, anomalies must be distinct from one another, as well as from the normal behavior. A prevalent methodology for

performing unsupervised AD on time series is discovering motifs and discords by, e.g., Matrix Profiling [25] or HOT SAX [26]. Motifs are the best matching subsequences in time series, while discords are subsequences that maximally differ from the other ones, i.e., the anomalies. Autoregressive models (e.g., ARMA, ARIMA and VAR), isolation forests and clustering-based techniques (e.g., DBSCAN) are also popular [27].

Finally, datasets will sometimes only contain data that is labeled as normal. Semi-supervised AD methods construct a model representing the normal behavior from a given training data set with only positive labels and test the likelihood of a new sample to be generated by the learned model. The presence of many related outliers in the set of objects to be scored does not impact the model's evaluation. Popular methods for semi-supervised AD which can deal with time series are clustering algorithms [28].

The application of RCA for sensor systems and the IoT is rather under-explored in contrast to AD [29]. Different ML techniques for RCA are discussed in the literature [30–32]. Particularly association rule mining (ARM) and Bayesian networks are well-recognized data mining techniques for knowledge discovery and exploring relations between failure events embedded in large datasets. Techniques from the field of statistical process control, such as contribution plots based on principal component analysis (PCA) or partial least squares, can also be used for RCA.

Several advantages of these data-driven AD and RCA techniques can be noted. As they learn directly from the data, little human intervention is required. Moreover, they are able to discover new anomalies in the provided data that are not known to the domain experts yet.

Despite these advantages, huge amounts of data are required to learn the normal behavior accurately and distinguish it from the anomalies. Providing labeled data will have the advantage that several models can be trained or developed in parallel. When not available, more human interventions will be required to check whether or not the anomalies are of interest. Both approaches entail a huge effort to collect this data and correctly label it or provide useful feedback afterward. These data sets are hard to come by. ML-based AD in sensor streams is often trained only once on a limited, labeled dataset collected from the environment and deployed in various contexts and configurations. A limited data set can result in many false positives, mainly due to patterns that were not detected in the past. The problem of dealing with such missing patterns can be reduced by grouping the patterns together based on similar feature values [33]. However, this technique only reduces the false positives for those patterns that are available in the provided dataset and not for the new undetected ones.

Data-driven AD and RCA can take into account the environment they are operating in and know how to optimize their performance to relate the system components to each other [34, 35]. However, this kind of context is limited to some additional features and makes it difficult to adapt them to new types of sources and environments. Many false positives are prone to overwhelm the user with redundant notifications, and anomalies that go undetected (false negatives) occur.

Table 3.1: Overview of the data- and knowledge-driven techniques in comparison with the requirements of Section 3.2

	Data-driven	Knowledge-driven
Precise		X
Minimal human involvement	X	
Context-aware	(X)	X
Adaptive	X	
Interpretable		X

The lack of domain and background knowledge also makes it difficult to accurately pinpoint the real causes of anomalies. The best performing and most expressive ML-based AD methods for analyzing sensor streams are black-box and, therefore, not interpretable. This leaves the operators guessing why an alert was raised as no interpretation can be given by the model. Recent advances in eXplainable AI (XAI) have made it possible to give some interpretations of the trained models' outcome. Techniques such as Shapley values show which features are important for our ML model or on which parts of the input a decision is based [36]. However, such feature importance methods give only a small amount of interpretability and correlate features based on the given data. Moreover, the applicability of XAI is rather limited in streaming environments [37].

3.3.3 Comparison regarding the requirements

Both the data-driven and knowledge-driven AD, FR & RCA techniques can be summarized regarding the requirements listed in Section 3.2. As can be seen in Table 3.1, none of these two techniques met all stated requirements. As discussed above, the data-driven techniques adapt to new, unseen anomalies and applying such a model in the context of streaming data requires minimal effort. The incorporation of additional features makes these techniques contextual, but they cannot guarantee to fully incorporate the system or operating environment dynamics based on the data solely. By contrast, knowledge-driven techniques do include the inherent dynamics and provide an interpretable and precise FR mechanism. As a counterpart, new expert knowledge must be incorporated manually, which requires human expertise. Such human involvements make these techniques less attractive to operate in rapidly changing environments such as sensor networks.

From Table 3.1, it can be seen that combining both approaches could resolve the stated drawbacks. An efficient way to fuse both the data- and knowledge-driven AD, FR & RCA techniques for sensor streaming data in one methodology is proposed in Section 3.4.

3.3.4 AD, FD & RCA within predictive maintenance

While the requirements are defined in the general context of AD, FD & RCA, they also impose several challenges within the predictive maintenance and IoT domain [38, 39].

Adequate information about possible failures is challenging to acquire, as most system failures are rare or vary a lot for different types of systems and equipment [40]. For the data-driven paradigm, it is more common for models to be trained on the sensor data of one single machine instead of a more global approach as this is less effective. Predictive maintenance is therefore limited to health monitoring tasks, where they identify machine failures based on the deviations from the healthy machine or system data [41]. These health monitoring tasks are almost always unsupervised due to the unavailability of labeled data. Despite their effectiveness in both accurately finding anomalies and efficient usage of the available computational resources, still a lot of falsely introduced alerts or missed interventions occur. This almost always results in unwanted maintenance costs. The operators are unable to reduce these costs because their insights into the problems are limited to comparing the deviated signals with the healthy system behavior [42].

In contrast, expert systems have been the most used fault diagnostic technology during the previous decade as faults must be detected precisely and timely in critical systems [43]. However, experts have a rather global view on the system and its functioning. As more and more sensors are attached to subparts of the system, it becomes difficult for experts to give concrete explanations to the deviations in sensor values and how they affect all the other system components. Knowledge in the form of facts and rules is available from mainly two types of sources: diagnostic-based knowledge and documented information. Both introduce difficulties in assimilation and requires a lot of work to become computerized [44]. Tools exist to limit these drawbacks, but human efforts are still required to maintain the acquired knowledge base [16]. Knowledge-driven models are in this perspective complex, resulting in longer processing times, which makes them impractical for real-time or nearly real-time purposes.

To overcome the problem of knowledge acquisition, a procedure has recently been proposed to select the input variables for soft sensors based on both data-driven and knowledge-driven input selection methods [45]. Soft sensors are generally built through data-driven approaches that exploit industry historical databases [46]. This new procedure allows designing soft sensors with good prediction accuracy and a low number of inputs, which reduces the complexity of the model and increases its maintainability. Currently, this is the only technique that is able to combine expert information with raw data. However, by limiting the expertise to the selection of input variables, these soft sensors deliver only a limited amount of interpretability. Another drawback is a possible reduction in adaptiveness when different systems require different input variables.

3.4 Proposed methodology

To meet all the requirements stated in Section 3.2, we propose FLAGS, the Fused-AI interpretabLe Anomaly Generation System. This system operates in 3 phases:

- In the first phase, both data- and knowledge-driven techniques are used in parallel. They take as input the data streams provided by one or more sensors, together with case-specific context data. Faults (knowledge-driven) or outliers (data-driven) are outputted. If possible, an interpretation of the detected anomalies is provided. The output of both techniques is stored inside a KG.

- During the second phase, the detected anomalies are shown in a comprehensive dashboard. Both the associated raw data and an interpretation, if available, are shown as well. The user can then provide feedback, e.g., confirm the anomalies and faults, merge them, or edit them. The feedback is also stored inside the KG.

- In a third phase, the information in the KG, i.e., the detected anomalies, the feedback provided by the user through the dashboard and all contextual meta-information, is used to improve the data- and knowledge-driven AD, FR & RCA techniques. User feedback is used by the data-driven methods to derive new interpretations for the outliers, while the contextual information is used to adapt the detection algorithms itself. Semantic rule mining is employed by the knowledge-driven methods to generate new knowledge given the updated KG. The newly derived knowledge is of the form of rules that indicate when particular faults occur, based on the historical information stored in the KG. This new knowledge is used to update the knowledge-driven detection tools automatically.

To let the different phases interact with each other, the FLAGS framework is built around the consumer-producers' principles. Each component can input data of interest and make their results available for other components that benefit from this produced output. Details on how such a consumer-producer approach is implemented, are given in Section 3.5. The following subsections explain in more detail the various modules that make up these 3 phases and how they interact.

3.4.1 Phase 1: Data- and knowledge-driven AD, FR & RCA resulting in semantic & interpretable anomalies and faults

In Figure 3.3, Knowledge (light grey boxes) and data-driven (black boxes) AD, FR and RCA are applied separately, as shown in the top and bottom parts.

As detailed in Section 3.3, the knowledge-driven FR & RCA techniques rely on expert knowledge captured in FMEA & FTA documents. By employing ontology-based risk analysis methods, this know-how is captured in semantic models and linked

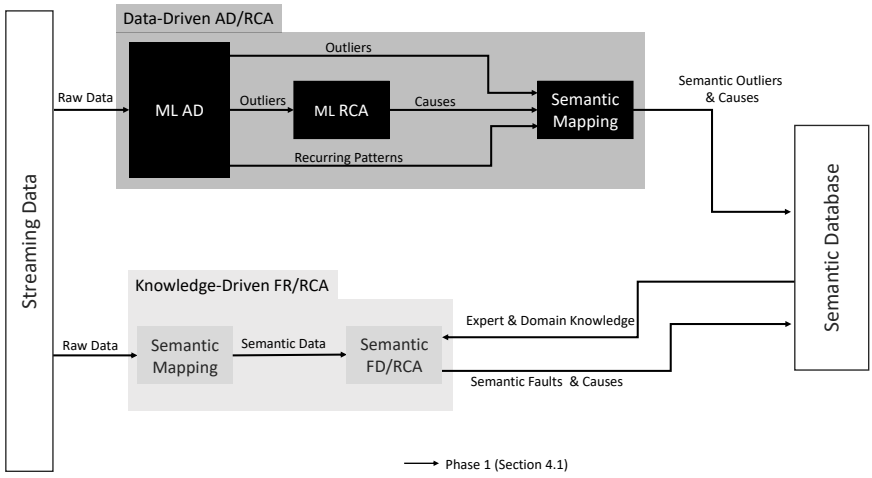


Figure 3.3: Phase 1 of the FLAGS end-to-end methodology. Both the output of the data- and knowledge-driven techniques are combined in a semantic database.

to the available background information about the system and context in which it operates. This information, i.e., the KG, is stored in a semantic database, e.g., a triple store.

The Semantic Mapping module is responsible for enriching the incoming data streams through these domain models. Mapping raw data to semantic observations provides an additional interpretable layer, and is an effective way to give a richer semantic meaning to the sensor data. Semantic mapping realizes the sharing, reuse and fusion of that sensor data [47]. Different mapping languages and paradigms can be used in the FLAGS methodology. An example of such a mapping module is discussed in Section 3.5.2.

This enriched stream is fed to the Semantic FD/RCA module. This module keeps track of a window of semantic observations and uses a semantic reasoner on these windows to infer semantic rules provided by the experts. When such a rule is triggered, a known fault is detected. The semantic reasoner combines the semantic observations with the profound domain knowledge in this perspective. This profound knowledge is available in the semantic database as an ontology with accompanying rules. As these faults originate from given semantic rules, the detected faults can be explained by providing the rule that was fired. This description resembles to what is more commonly known as Semantic Stream Reasoning (SSR) or SCEP. Section 3.5.3.2 gives more information on how such an SSR module can be implemented and interacts in the FLAGS methodology. An evaluation of the most common SSR systems is left out of scope.

The semantic FD/RCA module outputs faults and accompanying causes, which

are also added to the KG stored in the semantic database, such that they can be used in future reasoning tasks. This whole knowledge-driven approach is shown schematically in the lower part of Figure 3.3.

The ML-based AD modules take the raw data as input and use a ML technique to derive unique patterns in the streaming data, assuming these are outliers. From this perspective, any ML module which follows the guidelines of being operational in streaming environments and output outliers or uncommon patterns can be used in the FLAGS methodology. Details about such a learning module are provided in Section 3.5.3.1. Note that the different modules can output different types of anomalies.

The output of all these ML AD techniques is fed to the ML RCA module, which tries to link the anomalies and give some basic data-driven explanations. The ML RCA module is triggered once enough anomalies, all sharing the same confirmation label, have been recorded. An anomaly is from this perspective an event with a timestamp. The ML RCA module searches which combination of events can predict the occurrence of this anomalous event. Pattern matching and association rule mining techniques are used to find the relations between these events and the specific triggering anomaly. Once such a rule or pattern has been found, an explanation can be given to the anomalies triggering this event. These alerts continue to pop up until a better rule has been found using new incoming events, or the confidence of the rule drops as new anomalies have been registered. The implementation of such a RCA module is explained in Section 3.5.3.1.

A Semantic Mapping module is used again to semantically enrich the outputs of the ML AD & RCA modules. This module links the causes to the associated anomalies and let them relate to the underlying correlated data, system and context in which they occurred. These enriched anomalies are also stored in the semantic database. The data-driven AD/RCA module is shown in the top part of Figure 3.3.

At the end of phase 1, the anomalies and faults are mapped in semantic format and stored in a KG. Storing them inside the KG has the advantage that background knowledge can be linked to the detected anomalies, giving more information about how and why they happened. Also, each AD component can semantically query and filter anomalies, either ML or rule-based. This eases the uniform visualization of all these detected anomalies, which is important for phase 2. The different techniques might pick up similar or correlated anomalies, which can be detected by exploiting the created links in the KG. Overall, the KG adds a layer of interpretability to the detected anomalies and their causes.

3.4.2 Phase 2: Comprehensive dashboard to easily capture valuable user feedback

In the first phase, both the data- and knowledge-driven techniques operate separately. A dashboard can give an overview of all detected anomalies, faults and causes, as

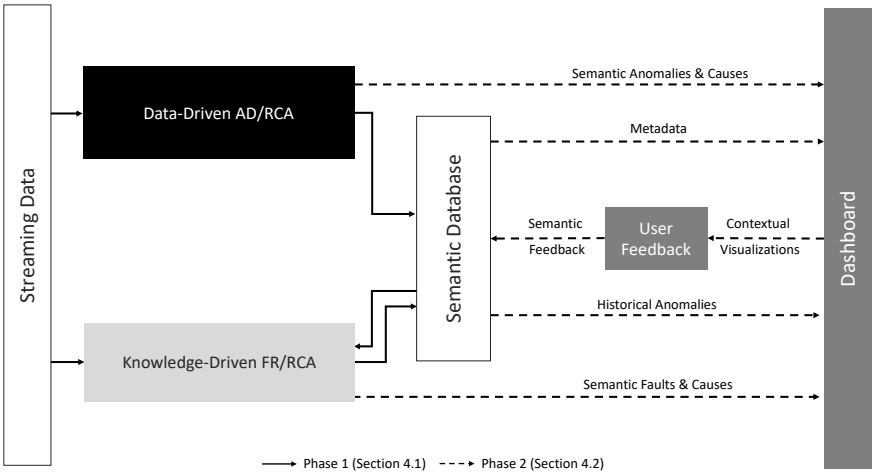


Figure 3.4: Phase 2 of the FLAGS end-to-end methodology. The semantic faults, anomalies and causes are represented in a dashboard application. The end-users can provide feedback on the represented views.

shown by the striped arrows and dark grey boxes in Figure 3.4. When new anomalies or faults are inserted in the KG, a dashboard visualization can be made to investigate the problem.

By exploiting all the semantic background information captured in the KG, comprehensive visualizations can easily be created. For example, anomalies can easily be grouped according to the context, system, configuration or the time frame they occur in. Such visualizations allow the end-users to quickly get a view on the system behavior without being overloaded with information.

Aided by the visualizations, the user can then quickly provide feedback on these anomalies, faults and causes by indicating their correctness or relabeling them based on the user's expert knowledge. By exploring the semantic links between the detected faults and outliers, e.g., whether they originate from the same sensor data, the user can also merge anomalies together. Merged anomalies are of huge interest for the ML RCA module as it provides similar events from which patterns can be found. The merge actions cause the outliers to be relabeled as known faults, which again is additional information to reduce possible false alerts.

All this feedback is also linked to the anomalies in the KG and stored in the semantic database. A fully functional dashboard that can provide feedback based on the visualization is discussed in depth in Section 3.5.4.

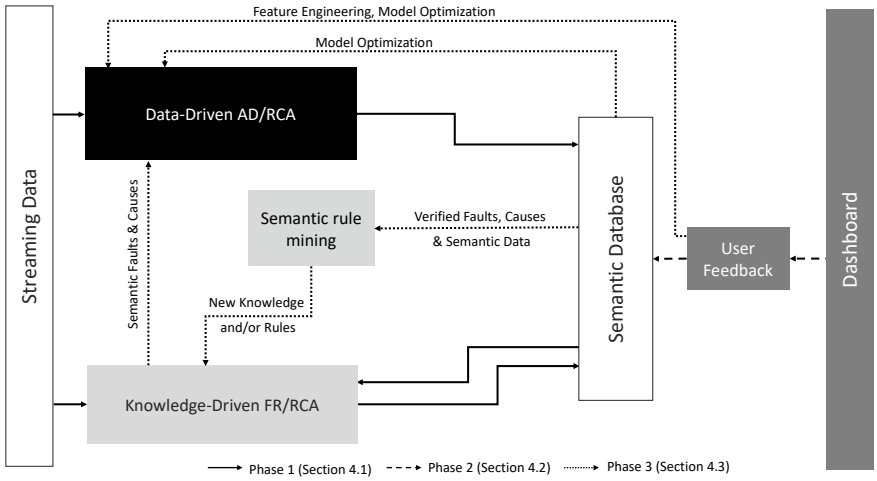


Figure 3.5: Phase 3 of the FLAGS end-to-end methodology. The user feedback is used to update the data-driven detection behavior and to provide new expertise to the knowledge-driven FR using a semantic rule mining approach.

3.4.3 Phase 3: Optimizing the data- and knowledge-driven AD & RCA through fused AI and user feedback

All the background knowledge, the context information, the anomalies, the causes and the feedback stored in the KG are used to optimize both the ML AD and Semantic FD/RCA module. These steps are indicated by the dotted arrows in Figure 3.5.

For the data-driven techniques, contextual metadata is used first to enrich the preprocessing and feature extraction steps, by extracting additional features or by combining features into more informative ones. Background knowledge can also be used to choose the ML model type or tune this model to operate in varying contexts, i.e., use a separate model per context or configuration. In addition, the feedback generated by the user through the dashboard can also be taken into account. The outliers are (semi) automatically relabeled by merging them with detected faults for the same data points using the knowledge-driven techniques. This information can be used by the ML RCA to give a more detailed cause description of future detected anomalies of the same type. Another option is to use this information to build pattern detection tools to provide ML-based FR. Moreover, user feedback about the correctness of the anomalies is used by the ML AD module to further optimize its detection rate. The false positives and false negatives are now quickly addressed and stored alongside the input data.

Combining the outliers and faults in one KG also allows pinpointing faults that were not discovered by the semantic FD. These combinations of ML and Semantic

FD output are ideal cases for finding new rules, i.e., new explanations for such new anomalies and making them detectable and explainable in the future. Semantic rule mining is employed on the KG containing all the semantic observations, their links to (merged) known (faults) and unknown (outliers) anomalies, and the links to the context and background knowledge. As indicated by the Semantic rule mining box in Figure 3.5, this method derives inference rules describing the situations in which the outliers occur. These rules can then be added to the semantic FD and applied right after they are made available.

Semantic rule mining is done by applying sequential pattern mining techniques on the semantic data. Predefined support and confidence parameters are set as thresholds to filter those rules of interest. Different semantic rule mining approaches exist and can operate in the FLAGS framework. An implementation of such a semantic learner is discussed in detail in Section 3.5.5. A more in-depth analysis of all possible semantic mining frameworks is out of scope for this chapter.

Again, these newly derived explained anomalies or faults, together with their accompanying context and inference rules when available, can be visualized to the end-user in an interpretable fashion. These visualizations enable a constant feedback loop between the detected events and the (learned) inference rules. The feedback loop allows the KG to evolve gradually with minimal human intervention.

3.5 Use case: Train bogie monitoring

The proposed methodology of Section 3.4 can be used in a wide range of applications. To showcase its potential, the methodology was applied to the railway domain, in close collaboration with the experts of Televic Rail[3]. They provided realistic datasets captured by their train monitoring systems, risk analysis information captured in FMEA & FTA documents and ML algorithms they already employ to perform AD. The following subsections first outline the use case, followed by a detailed explanation of how the FLAGS methodology was employed and the various modules that were implemented to realize the optimization of AD, FR & RCA by fusing knowledge-driven algorithms, data-driven techniques with minimal user feedback. The application of the methodology to this specific use case is visualized in Figure 3.6 and shows the different components of each phase in one diagram. As seen at the top right, the Matrix profile technique is used as an ML AD module to detect outliers in the raw data. At the bottom, the raw data is being semantified using RML and a Stardog Rule reasoner was built to detect known faults. All outliers and semantic metadata were stored inside a Stardog database, as shown on the right of Figure 3.6. This Stardog database also communicates with a so-called Dynamic Dashboard [48]. Here, the database provides semantic information and stores the user feedback. At last, as rule mining techniques,

[3]https://www.televic-rail.com/en

we used some standard episode mining techniques, as well as DL-Learner, to discover new information. The rule mining interactions are shown in the center of Figure 3.6. In the following sections, we will give more details about each of the used techniques.

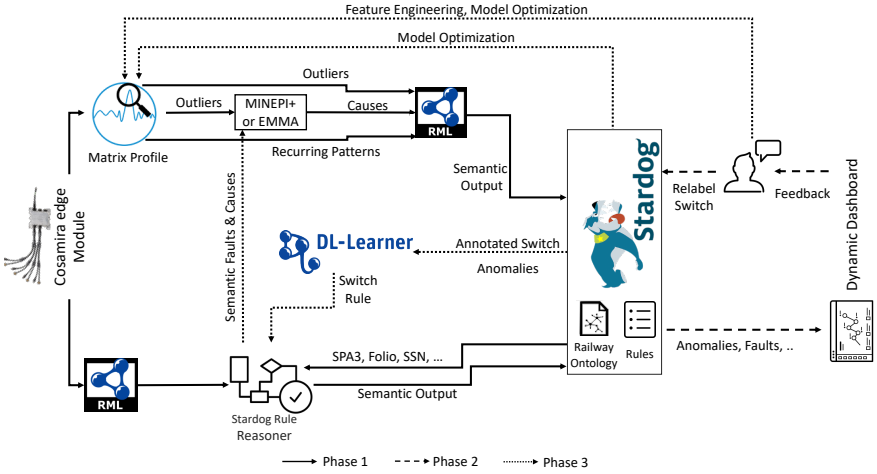


Figure 3.6: Train monitoring use case overview, each component of the methodology in Section 3.4 is now specified by functional modules.

As discussed in Section 3.4, this whole approach is designed around the principles of consumers and producers. In this use case, we used the Apache Kafka 2.4.1 platform[4] to let the data flow from one component to another. The components defined in this use case are either consumers, producers, or both as visualized in Figure 3.7. The arrows show how these components interact with the available Kafka topics. All Kafka topics were divided into two main groups, one for the raw data and one for the semantic data. The raw.events topic is the one that is being used to let the company push its data. Both the data-driven ML modules and semantic mapping component consume the data from this raw.events topic. The mapping component transforms the raw events into semantic events and pushes these events to the semantic.events topic. Later on, the semantic FD module will consume the data from this semantic.events topic and produces events to the anomaly topic. The anomalies in this topic are also transformed semantically and added to the semantic.anomalies topic. This last topic is consumed by the semantic database, which stores all the semantic anomalies into the database. Similarly, the feedback and semantic.feedback topics are created to define the difference between raw and semantic feedback.

[4]https://kafka.apache.org

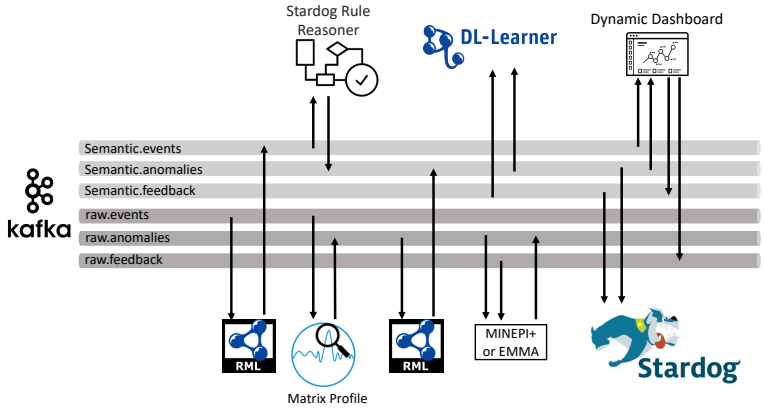


Figure 3.7: Overview of each producer/consumer component and their interaction with the different Kafka topics.

3.5.1 Use Case Description

Trains operate under various conditions. To ensure that both passenger comfort and equipment quality are adequate, more and more monitoring devices are being used to verify whether pre-defined standards are met [49]. One such sensor monitoring device is the Cosamira edge[5] designed by Televic Rail. This sensor monitoring unit is placed on the train bogie to capture the train's current location, the wheel axle temperature and the primary[6] & secondary suspension[7] of a train.

The data produced by this module can be analyzed to find irregularities in the train's behavior or the surrounding environment, e.g., to find anomalies. The main idea underlying this use case is that anomalies detected at the same location by different trains are most likely due to problems in the environment, e.g., faulty tracks or obstacles. The end-to-end methodology will make it possible to automatically discern anomalies due to degrading train equipment from the ones caused by irregularities in the environment and accurately pinpoint their cause, e.g., faulty wheel, bridge present, etc.

[5]https://www.televic-rail.com/en/cosamira-rail-bogie-monitoring-data-analysis
[6]The primary suspension stabilizes the running behavior of the bogie to ensure low track forces, low wear and proper behavior in curves.
[7]The secondary suspension measures the connection between the car body and bogie to isolate excitations transmitted from track irregularities via the wheelsets and bogie frames.

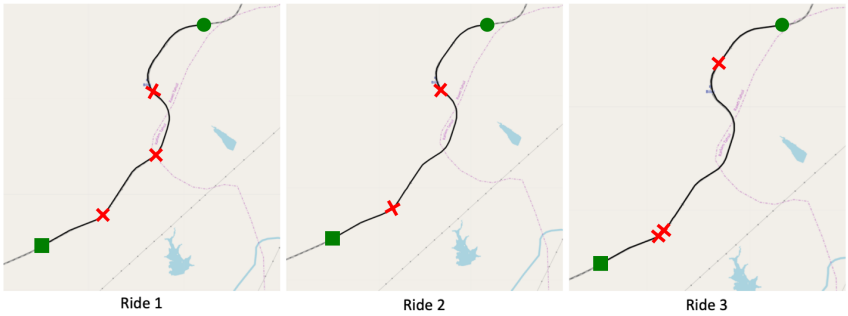


Figure 3.8: Use case data for three train rides at the same location, at different moments in time performed by different trains. The train rides from the top of the figure (starting at the green circle) to the bottom (green square). The red crosses indicate the location of found anomalies by the ML AD technique specified in Section 3.5.3.

3.5.2 Knowledge Graph and mappings

Televic prepared three datasets that each contain one hour of data. The datasets have been recorded at the same location, but for three different trains and moments in time. Every dataset contains:

- Latitude & longitude data sampled at 1Hz (Global Navigation Satellite System or Gnss).
- Wheel axle temperature data sampled at 0.1Hz.
- 3-axis accelerometers, data sampled at 1KHz (Inertial Measurement Unit IMU).
- 3-axis gyroscopes, data sampled at 1KHz (Inertial Measurement Unit or IMU).
- 1-axis shock pulse accelerometers, data sampled at 1KHz (Shock Pulse Methods or SPM).

The datasets are known to contain track anomalies and to be different enough from one another in terms of accelerations. Three segments of five-minute data from these three different train rides are used here to show the benefits of our used approach. These segments are shown in Figure 3.8. To provide a real-time detection mechanism, the sensor observations are streamed.

A railway ontology was made, in collaboration with the domain experts at Televic Rail who manufactured the Cosamira edge. This product-specific ontology extends the existing SPA3 and InteGRail ontologies [50]. In this ontology, the sensors, their interactions with the car body, wheel axle or track were defined.

Metadata of the trains and associated Cosamira modules, for which the data was collected, was also provided by Televic Rail. Track and train route meta information

```
PREFIX folio: <http://IBCNServices.github.io/Folio-Ontology/Folio.owl>
PREFIX cosamira: <http://scope.icon.dyversify/televic/Cosamira#>
IF {
 ?a1 cosamira:fromTrain ?c1 .
 ?a2 cosamira:fromTrain ?c2 .
 Filter(?c1 != ?c2)

} THEN {
 ?a1 folio:cause 'Track problem' .
}
```

Listing 3.1: Track issue rule: comparing two anomalies or faults if they are from a different train. Whether these two anomalies occur near each other is defined by the GeoSPARQL functionality inside the semantic database.

was captured using OpenStreetMap[8]. We created RML mapping scripts [51] to map all of this metadata about the tracks, trains and Cosamira module to the designed domain ontology in an automated fashion.

The OpenStreetMap railway-related nodes were transformed into semantic notations using the GEO-spatial functionalities to define their exact position and gather additional information such as crossings or stations provided by the SPA3 and INTEGRail ontologies. We store the resulting KG in a semantic database. For this use case, Stardog[9] was used. The GeoSPARQL functionality was enabled in this semantic database to query location-based semantic data using the latitude and longitude samples.

Knowledge about possible anomalies, their causes and their effects on the train were provided by the Cosamira manufacturer using the discussed FMEA documents and FTA trees of Section 3.3. In this FMEA document, two main categories of anomalies were defined: track and train issues. A track issue is characterized by a certain irregularity on a specified location affecting more than one train. A train issue was specified by similar problems occurring multiple times, originating from the same Cosamira edge module (and so from the same train). By using the FOLIO ontology (see Section 3.3), these documents were translated into a semantic model linked to the railway domain ontology with accompanying rules.

A rule linking the location data from the GNSS sensor with the track anomalies and a rule specifying anomalies occurring at a single train were generated. Both of these rules are visualized in Listing 3.1 & 3.2. The anomaly information and the generated rules are stored in the same Stardog database as discussed above.

During the previous steps, schema information and expert knowledge were incorporated as an ontology with accompanying rules and populated with concepts borrowed from the railway and anomaly domain. RML mapping scripts were also created to map the incoming stream of raw observations from the Cosamira sensors

[8]https://www.openstreetmap.org/
[9]https://www.stardog.com

```
PREFIX folio: <http://IBCNServices.github.io/Folio-Ontology/Folio.owl>
PREFIX cosamira: <http://scope.icon.dyversify/televic/Cosamira#>
IF {
 ?a1 cosamira:fromTrain ?c1 .
 ?a2 cosamira:fromTrain ?c2 .
 ?a1 folio:description ?d1 .
 ?a1 folio:description ?d2 .
 Filter(?c1 == ?c2 & ?d1 == ?d2)
} THEN {
 ?a1 folio:cause 'Train problem' .
}
```

Listing 3.2: Train issue rule: comparing two anomalies or faults if they are from the same train.

```
{
 "metricId": "sensor.axle.temperature::number",
 "timestamp": 1537060680000,
 "timeUnit": "MILLISECONDS",
 "sourceId": "COSAMIRA.BOX.0",
 "value": 23.81766845703125,
}
```

Listing 3.3: Example of a JSON value produced by the Cosamira Edge

to the SSN and the railway domain ontology. Concretely, all the sensor values listed above arrived at our platform in a JSON representation. Each JSON string included the actual data as a floating-point, the originating sensor, the corresponding metric and a timestamp when the value was generated. An example of such a JSON string is shown in Listing 3.3. A corresponding JSON-LD representation was made using the mapping file in Listing 3.4 to format these raw data samples. As a single number just specified the sensor ID, the corresponding JSON-LD makes the identifier link to additional sensor information and makes it possible for a semantic reasoner to infer knowledge by following this link. The other values, such as the timestamp, were matched in a similar fashion. As these transformations must occur in real-time, a streaming variant of the RML mapper was used [52]. The mapping script in Listing 3.4 was made manually using YARRRML[10], which is a human-readable text-based representation for declarative Linked Data generation rules. It can be used to represent R2RML and RML rules in a human-friendly manner.

A snippet of the resulting overall ontology is visualized in Figure 3.9. In this snippet, the case-specific components, such as the Cosamira Shock sensor, are indicated as black boxes. Reusing concepts of the FOLIO ontology, the railway ontology and SSN show how the case-specific instances interact with each other and how they relate to the provided faults (a Train issue in this example). A simple rule is also visualized, showing the link between the observations and the AnomalyKnowledge concept.

[10]https://rml.io/yarrrml/

```
prefixes:
 sosa: http://www.w3.org/ns/sosa/

mappings:
 observation:
 sources:
 - [input.json~jsonpath, $]
 s: http://.../observations/$(timestamp)
 po:
 - [a, sosa:Observation]
 - [sosa:observedProperty, http://.../$(metricId)~iri]
 - [sosa:resultTime, $(timestamp), xsd:dateTime]
 - [sosa:hasSimpleResult, $(value), xsd:float]
```

Listing 3.4: Example of a mapping file to transform the JSON value of Listing 3.3 to a semantic format.

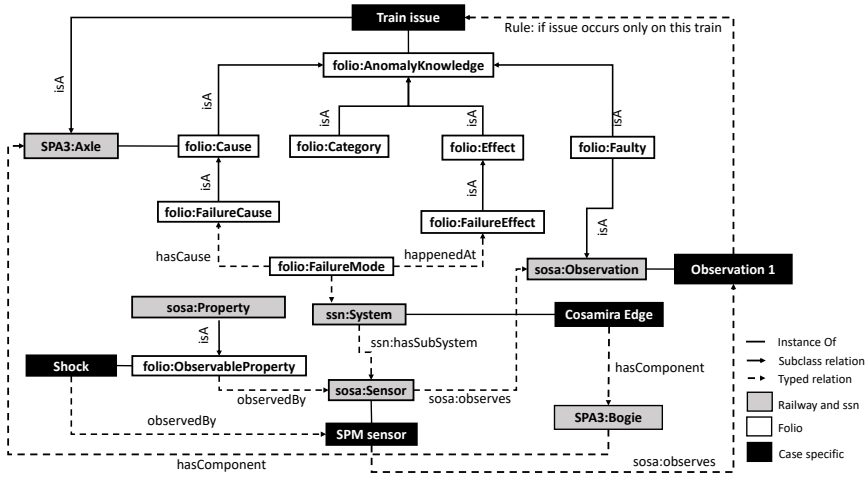


Figure 3.9: A snippet of the overall ontology. This snippet shows how the SPM sensor observes Shock Observations and how it is connected to the Cosamira Edge system through the bogie and wheel axle. A rule which relates observations to train issues is also given.

3.5.3 Data- and knowledge-driven AD & RCA through semantic reasoning and Matrix Profiling

When all metadata is available in our system and sensor data streams in, both the data- and knowledge-driven modules can start detecting anomalies.

3.5.3.1 ML AD & RCA:

Our main focus in this use case is to find either track or train issues. Therefore, the SPM signal is the most informative for this case as the producing sensors are closest to the track. The ML AD module aims to find abnormal patterns, i.e., discords,

in the observed time series made by the sensors, and to match incoming patterns against previous confirmed to be anomalous by the dashboard. Matrix Profiling was selected because it is unsupervised, well suited to streaming data, has support for multidimensional data and can work in real-time. As visualized in Figure 3.10, it works by moving a sliding window (W1) over the time series and tracking the best match to any other previous window (W2). The z-normalized euclidean distance is used as a distance measure to determine possible matches [26]. The minimum values for each of these matching fragments, in this case columns (F), are used for the resulting Matrix Profile vector.

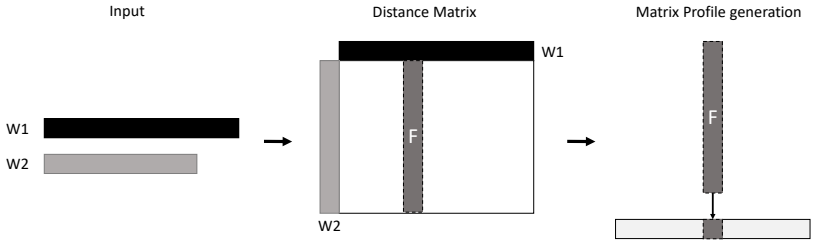


Figure 3.10: Overview of the ML AD Matrix Profile approach. Starting from two input windows (W1, W2), the z-normalized Euclidean distance generator iteratively creates fragments (columns F) from the distance matrix of all subsequences. Each of these fragments is processed by the Matrix Profile consumer, storing the minimum value for each column in the resulting Matrix Profile vector. This Figure has been adapted from [25].

After preprocessing the SPM signal, by subtracting the sensor offset and then squaring and smoothing it, the values are passed into a streaming-enabled Matrix Profiler that tracks the last 30 seconds of data (30k sensor values).

We implemented an additional constraint in the Matrix Profile so that only windows with a similar standard deviation could be compared. This ensures that the technique could distinguish between visually different patterns. As the input data is being processed, the resulting stream of distances is checked against a threshold, where high values indicates anomalies. Afterwards, we merge nearby high values into a single, longer anomaly of up to 10 seconds. This technique was implemented as an extension of the Series Distance Matrix framework [25].

All train rides followed the same trajectory. Therefore, the main difference between these three rides was the train speed and corresponding vibrations along the way. The red crosses in Figure 3.8 indicate anomalies detected for the 5 minute use case by the Matrix Profiling.

Matrix Profiling was used to detect new outliers in the data, while the ML RCA module wants to provide an interpretation for the outliers of interest on the streaming data. To determine which outlier has to be mapped, rule mining techniques can

be used to provide possible explanations, transforming them into more faults after anomalies have been detected.

Because anomalies in the SPM data consisted entirely of outliers rather than faults, the rule mining technique lacked proper input and did not deliver interesting results. This is a good example of where data-driven methodologies are limited in the amount of interpretation they can give or the knowledge they can derive.

Both the anomalies and causes derived by the ML AD and ML RCA module are semantified using RML mapping scripts and stored in the Stardog database. This semantification step is similar to the one described to semantify the raw data. The output of both the ML AD and ML RCA module is in a JSON format, and an RML mapping script transforms these JSON strings into a JSON-LD representation, based on the FOLIO ontology.

3.5.3.2 Semantic FD/RCA:

As mentioned in Section 3.5.2, RML mapping scripts were created to semantically enrich the raw observations coming from the Cosamira sensors. To continuously map the incoming streaming data, the RMLStreamer [52] was used. All the raw events are stored in a data warehouse and it is not required to store all the semantic observations inside the semantic database. A replay of the data is possible on request. Therefore, the semantic database can be kept small and does not require to be reindexed when new data samples are coming in.

Data-driven AD is good at processing sensor data streams fast. Moreover, the domain experts had limited expert knowledge about which values or trends in the raw data streams indicated interesting events or anomalies. However, expert knowledge was available about how the detected anomalies could be related to the background knowledge, the context (e.g., the environment of the rails) and rolling stock (e.g., the dynamics of the train) to classify them as particular faults. Therefore, the semantic FD includes rules to reason directly on the anomalies generated from the ML AD module and classify them as faults. The code to perform such Semantic FD has been made open-source[11]. Once processed, the information of multiple outliers detected by the ML AD and train & track metadata becomes available inside the Stardog database. This semantic data can be used inside the semantic FD/RCA to filter generated anomalies or provide them with additional data such as more knowledge-based descriptions and causes. This filtering approach is visualised in Figure 3.11.

More concretely, this filtering approach consumes the semantically transformed output of the ML-based Matrix Profiler. When the Matrix Profile module detected an anomaly, the coordinates of this anomalous event are used in a SPARQL query. The query which has been executed, is shown in Listing 3.5. This query finds all the information of interest in a predefined range around this anomaly. Among the

[11] Semantic Fault detector: https://github.com/IBCNServices/StardogStreamReasoning

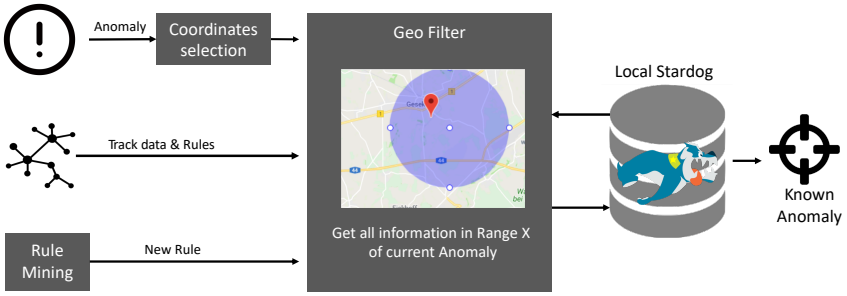


Figure 3.11: Overview of the semantic AD/RCA approach. The coordinates of the detected anomaly, the track data, and accompanying rules are used inside the GEO Filter to determine whether or not an explanation can be given to the detected anomaly. A local Stardog database is used to filter those events. Additionally, a component is available to update newly mined rules when they become available.

possibilities are included:

- Previously occurred anomalies, which are confirmed or known faults.
- Semantically enhanced OpenStreetMap nodes which deliver railway information.
- Rules triggered by anomalies, which were composed based on Televic's expertise.

When any such additional information is available and gets triggered by the executed query, the inputted anomaly is enhanced with this information and stored semantically in the Stardog database.

In the perspective of the given use case, during the first train ride only additional information provided by the experts and extracted from the OpenStreetMap data can be delivered. The coordinates from the ML detected anomalies during the second train ride are given to the GEO Filter. This filter executes the query in Listing 3.5 on the semantic database. The rules specified in Listing 3.1 and 3.2, describing either a track or train issue, will be triggered to return the corresponding cause description. When no additional information can be provided, the anomaly is kept in its original state and reported, as is, in the dashboard.

The nodes and rules specified by the Geo Filter are all loaded into a separate local Stardog database to perform rule-based reasoning on a particular subset of the data. This Stardog database is a completely separate instance, isolated from the general semantic database which stores the semantic anomalies, faults, causes and user feedback. Loading a set of observations and the corresponding metadata in separate semantic databases has two main advantages. First, multiple local databases can be set

```
prefix geo: <http://www.opengis.net/ont/geosparql#>
prefix geof: <http://www.opengis.net/def/function/geosparql/>
prefix unit: <http://qudt.org/vocab/unit#>

SELECT distinct ?feature {
 {
 ?geom geof:nearby ( $lat $long $range unit:Meter) .
 ?geom <http://IBCNServices.github.io/dyversify/Televic#type> ?feature.
 } UNION {
 ?an <http://IBCNServices.github.io/dyversify/Televic#track> ?feature.
 }
}
```

Listing 3.5: Geo Filter query: This query filters on nearby nodes given a latitude, longitude and range. The lat and long parameters are extracted from the current occurring anomaly, the range parameter is set upfront to 100 meters.

up when multiple new anomalies arrive at the same time. This enabled us to perform reasoning in parallel to speed up the whole procedure. Multiple GEO filters can be made operational to query the additional data for multiple occurring anomalies as the Matrix Profiler can produce them faster than a single GEO filter can reason. Second, reasoning operations did not affect the general Stardog database, which is preferable when more than one application depends on it.

During a second train ride, some anomalies are close to the location of the anomalies detected in the first train ride. In the meantime, semantic background knowledge became available and is provided in the semantic database. The semantic FD will use the information about the anomalies detected during the first train ride, together with meta-information given by the railway experts to identify these new anomalies as track issues according to the rule shown in Listing 3.1. An example of such a triggered rule is shown in Figure 3.12 (1 or red cross). The anomaly that occurred during the second train ride is close to the one that occurred during the first ride (2 or green cross). Since both anomalies originated from different trains, both were reclassified as track issues by our semantic FD module.

3.5.4 Dynamic semantic dashboard enabling user feedback

For visualization purposes, the so-called Dynamic Dashboard [48] is being used to show those visualizations that correspond with sensors selected by a user (user-driven visualization) and those visualizations that correspond with occurring anomalies or faults (anomaly-driven visualization).

In user-driven visualization, the Dynamic Dashboard helps the user select visualizations that are interesting to present, e.g., a user may want to visualize the route of a train on a map by selecting a widget for the sensor which publishes locations. The user is presented with the visualization options that the software deduced. Because all sensors and their corresponding metrics have a semantic description, each widget can

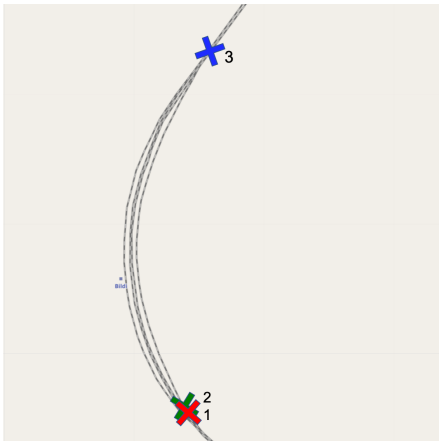


Figure 3.12: Detailed overview of the three top anomalies from train ride 1 (red), 2 (green) and 3 (blue) in Figure 3.8. They all occur near a switch.

reason whether or not it is useful to display the observations generated by the sensors. In this example, a gauge widget can be suggested to the operator to visualize the wheel axle's temperature sensor. More information about the workings of this dashboard can be found in Vanden Hautte, et al. [48].

In anomaly-driven visualization, the Dynamic Dashboard automatically creates a dashboard tab to investigate a selected anomaly. The dashboard tab is built with widgets that are found to be appropriate given the type of the selected anomaly and according to the sensor properties that are linked to the anomaly. The dashboard tab is built with appropriate widgets given the type of the selected anomaly and according to the linked sensor properties. Like the user-driven widgets, the anomalies are represented in a semantic format and can, therefore, interact with these widgets. In this example, the anomalies are all associated with a location and can thus be visualized on a map as shown in Figure 3.12.

Creating a dashboard with multiple widgets of interest requires less time with this Dynamic Dashboard, since several suggestions are made using an underlying semantic reasoner. A possible dashboard for our use case can be seen in Figure 3.13. Two different panes are visible: one left width five different widgets and one showing the listing on the right. Each of the five widgets represents a different sensor from our monitoring unit. From left to right and from top to bottom, the dashboard shows:

- The Y-axis acceleration of the IMU sensor as a time series.
- The Z-axis acceleration of the IMU sensor as a time series.
- The Yaw values of the IMU sensor as a bar plot.
- The temperature sensor as a raw value with a color-schemed background.

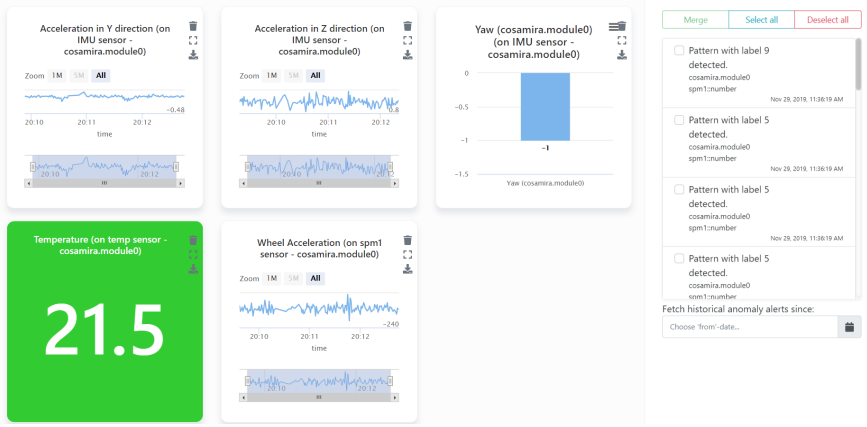


Figure 3.13: Dynamic dashboard example for a train bogie monitoring case. The left pane shows 5 different widgets, displaying the sensor values. The right pane is populated with the detected anomalies.

- The wheel acceleration of the SPM sensor as a time series.

. The listing on the right side of this dashboard shows the occurring anomalies. An anomaly-driven dashboard tab will be opened once we click on one of the anomalies.

Creating widgets, interacting with these widgets and setting parameters all indicate possible interesting parts of the data an end-user deals with. These interactions can be useful feedback for our detection modules. Two types of feedback can be captured in this dashboard and are of interest during this evaluation. Confirming or rejecting anomalies will adapt the operating AD modules in our methodology. Merging anomalies will cluster similar anomalies together, which is useful for the rule mining modules.

During the first train ride, only the ML AD can detect outliers, which are visualized to the end-users using anomaly-driven visualizations. This means that, based on the semantic annotations of the anomaly (in this case, unknown anomalies generated by the Matrix Profiler) and its origin (the SPM sensor with 1-axis accelerometer metric), a widget pops up on the dashboard making all the information visible to the end-user. The operator can already correct or label these outliers.

After the second train ride is finished, the semantic FD/RCA already outputs some faults, which differentiate between train and track faults. As the anomaly-driven widget visualizes the locations of the occurring anomalies as shown in Figure 3.12, an operator was able to determine that some of the inferred track issues could be explained with the occurrence of a switch. The relabeled anomalies can then be used as input for the semantic rule mining procedure of phase 3.

At last, similar to the raw data, the user feedback captured in the dashboard is

semantified using RML and stored in the semantic database. All feedback is provided in a JSON format and semantified to JSON-LD, a procedure similar to semantifying the anomalies or raw observations.

A newly designed feedback ontology was used to generate this semantic feedback and differentiates between intrusive and non-intrusive feedback[12]. Intrusive feedback comprises manual operations performed by the operator or dashboard user, such as relabelling anomalies. Examples of non-intrusive feedback are closing and opening dashboard widgets. All the semantic feedback is stored in the Stardog database.

3.5.5 Optimizing data- and knowledge-driven techniques through user feedback and semantic rule mining.

Despite several occurrences of (geographically) nearby abnormal patterns detected by the Matrix Profiler over the three train rides, as visualized in Figure 3.14, there was almost no similarity between their shapes. This made it challenging to build a pattern matching or fault detection tool purely based on the ML AD. The differences in these patterns are most likely due to the spiky nature of the signal and slight variations in train speed, causing high distances when comparing subsequent shapes [53]. Therefore, the feedback indicating switches occurred at the specified locations did not affect the ML AD module. This again shows the importance of combining the data-driven techniques with expert knowledge and user feedback to optimize AD, FR & RCA.

When enough anomalies have been labeled in the Dynamic Dashboard and become available in the Stardog database, the semantic rule mining component will become active to mine rules for those cases of interest. In this train use case, two anomalies from two different train rides occur close to each other and are labeled as switches through the Dynamic Dashboard, as shown in Figure 3.12.

The rule miner starts to search for a possible explanation and an accompanying rule for these similar anomalies. The supervised framework DL-learner [54] was used to search for rules to describe the detected anomalies. DL-learner takes the labeled switch events, some nearby nodes linked to the anomalies and some non-related observations as input. It then searches a rule based on all this information, together with all available metadata of the track and train. The miner's input thus contains both positive samples (switch events), labeled as anomalies, and negative samples (other, normal events). The goal is to find the OWL class expression R such that all or many positive samples are instances of of a class C and as few as possible negative examples. As explained, R should be learned such that it generalizes to unseen individuals and is readable. R is seen as a rule in our system and is applied to find the positive events. The process described above and used by DL-Learner is called the Class Learning Problem (CLP).

[12]also available at https://github.com/IBCNServices/Folio-Ontology

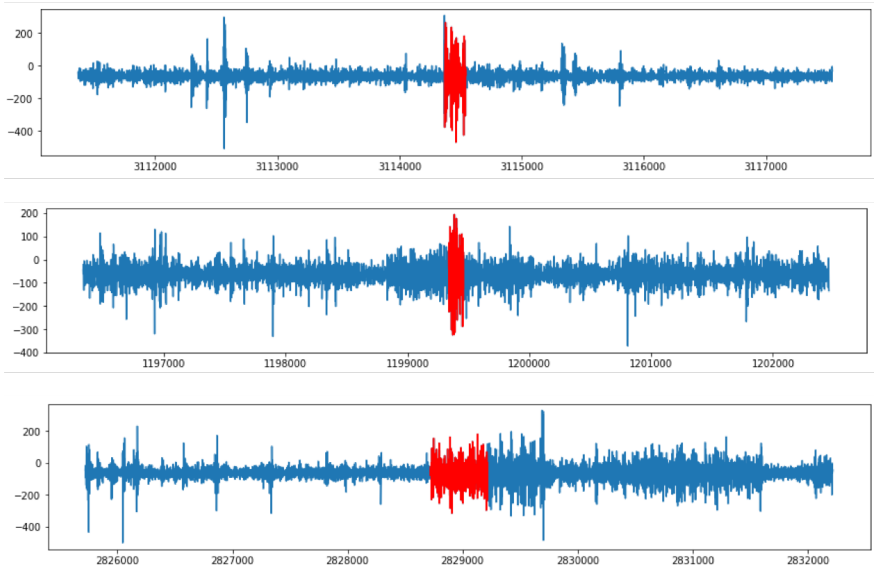


Figure 3.14: Examples of detected anomalies for the SPM signal.

```
IF {
 ?w1 a <http://www.integrail.info/ont/SP3A.owl#Way>.
 ?w2 a <http://www.integrail.info/ont/SP3A.owl#Way>.
 ?w1 <http://IBCNServices.github.io/dyversify/Televic#Contains> ?node1 .
 ?w2 <http://IBCNServices.github.io/dyversify/Televic#Contains> ?node1 .
 FILTER(?w1 != ?w2)
} THEN {
 ?node1 <http://IBCNServices.github.io/dyversify/Televic#type> 'Switch' .
}
```

Listing 3.6: Newly generated rule applied after the second train ride. This rule was mined using the anomalies found in the first two train rides, as shown in Figure 3.12 and combines track segment information available in the metadata.

The Geo filter set-up defined in Figure 3.11 can consume new rules that were altered and applies them directly after the semantic rule miner has generated them.

The generalized rule, mined for these two switch events, is given in Listing 3.6 and defines a switch as a 'transition' in track segments using the OpenStreetMap's metadata. An additional component applying newly learned rules derived by the semantic rule miner was added, as shown in Figure 3.11. Before the Geo filter is executed, this component is used to insert newly generated rules into the local Stardog databases. This update module ensures that newly mined information can automatically influence the derivation of anomalies in the future.

During a third train ride, a newly generated anomaly is classified as a switch event, as shown in Figure 3.12. This is due to the newly applied rule (blue cross, or 3).

With this rule, the semantic FR/RCA was able to deliver new insights provided by the metadata. A new type of fault can be filtered now, without needing additional human involvement.

3.5.6 Evaluation setup

To evaluate this use case, a Kubernetes[13] cloud setup was used to integrate each component mentioned in Figure 3.6 and to let them communicate with each other. In total, two separate clusters were used:

- One cluster with 40 cores and with 40G RAM was used for all the ML and semantic AD, FR and RCA modules, the rule mining components, the Stardog database and dashboard. Resources were shared according to the needs of the modules.
- A second cluster with two cores and 24G RAM was used for the Kafka delegation.

This separation was needed to ensure the throughput of the streaming data.

The five-minute raw sensor data for all three train rides took 855 MB (285 MB each). This dataset contained 1,200,000 SPM sensor values, which all have to be transformed into a semantic notation. For each such sensor value, five triples were generated by the RMLStreamer and pushed to the corresponding semantic Kafka topic. This RMLStremaer module, based on Apache Flink, was initialized with four task slots and one job manager with a heap size of 2048 MB to ensure the throughput. The manually constructed semantic metadata was stored inside the Stardog database and consisted of 326 586 triples. This Stardog database module was set with a heap size of 15G to ensure all queries could be executed. The Dynamic Dashboard module was initialized using the default configurations as described in [48].

Working within a streaming environment required some of the modules to be initialized upfront. The Matrix Profile module compared windows with a length of 100ms and was initialized on the first 100 seconds of the original one-hour dataset. This initialization step is required to determine the sensor offset and noise parameters. Since the presence of noise can have a significant effect on the distance measure used by the Matrix Profiler, the noise elimination technique described in De Paepe, et al. [53] was used. The Matrix Profile itself is calculated at 25 ms intervals, to ensure sufficient throughput. This approach can process up to 6000 events per second, enough to match 4000 incoming train events every second (calculating the matrix profile at 25ms interval, at a rate of 1 kHz).

The semantic FD module was initialized with the metadata and rules described in Section 3.5.3.2. Ten local Stardog modules were used within this module to ensure

[13] https://kubernetes.io

```
// knowledge source definition
ks.type = "OWL File"
ks.fileName = "data.nt"

// reasoner
reasoner.type = "closed world reasoner"
reasoner.reasonerComponent = embeddedReasoner

embeddedReasoner.type = "OWL API Reasoner"
embeddedReasoner.reasonerImplementation = "Hermit"
embeddedReasoner.sources = { ks }

// CELOE
alg.type = "celoe"
alg.maxNrOfResults = 300
alg.maxExecutionTimeInSeconds = 500
alg.expandAccuracy100Nodes = true
alg.maxDepth = 45

// learning problem
lp.type = "clp"
lp.classToDescribe = folio:AnomalyKnowledge (iri)
accuracyMethod.type = "fmeasure"
```

Listing 3.7: DL-learner configuration file to relate the positive instances in the AnomalyKnowledge class together.

multiple anomalies could be filtered at the same time. The SL reasoning level and GeoSPARQL functionalities were enabled within these local Stardog databases. The default parameters were used for all other settings.

A configuration file was required to initialize the semantic rule mining by DL-Learner. Listing 3.7 shows this configuration. The default Class Expression Learner or Ontology Engineering (CELOE) algorithm was initialized with the Hermit reasoner and limited to return 300 results. The maximum execution time of this algorithm was set to 500 seconds. The rules were evaluated using the F1-measure.

All defined parameters for these data- and knowledge-driven techniques were defined by using expert knowledge. Each module runs in a different Docker container[14], and does not affect other ones' execution when enough resources are available. To measure the duration of the code within each module, the standard libraries within Python and JavaScript for time measurements were used. Grafana [15] was used to investigate both the performance in terms of used resources for each module, as well the lag and interactions between the multiple Kafka topics.

3.6 Results

In total eight anomalies were found in the three segments, visualized in Figure 3.8. An operator of the industry partner inspected all of them. Three anomalies were found

[14]https://www.docker.com
[15]https://www.grafana.com

near bridges and were indicated as low priority track issues. Two anomalies were relabeled as switches as described above, which led the third one to be automatically labeled as a switch event. The other two anomalies could not be clarified, and as they did not reoccur during the other train rides, the operator classified them as low priority train issues for which further investigations are needed.

The Matrix Profiler found all these anomalies. The semantic FD classified 75% of the found anomalies as track issues as they occurred near each other. For the bridges and switches, the operator used the available context information of OpenStreetMap to verify whether the occurring anomalies could be explained. After the new switch rule has been applied, one anomaly was even explained automatically. In total, 75% of the anomalies received an explanation and the operator reduced their priority based on these explanations. 25% of the anomalies are now classified as false positives as the operators could not provide clear explanations for them.

To show the FLAGS system's adaptation rate, we investigated how long it would take to enable a new rule based on the feedback provided by the operator inside the Dynamic Dashboard. Time measures were taken to get an indication of the time it might take until the moment when the additional dashboard information becomes available in the semantic database. As discussed above, both the semantic rule miner and ML algorithms benefit from this feedback. The time between the moment the merge or relabel button was pressed until the moment the feedback was made available in the Stardog database was measured. On average, this action took 1694 ms (measured over 5 different runs std: 239.68 ms).

When enough semantic feedback is available, the semantic rule miner generates new knowledge to be incorporated in the semantic FD. This mining operation was limited in time, as indicated in Listing 3.7. Automatically applying a rule found by the rule mining module and making it active inside the semantic FD requires 31ms on average (std: 2ms over 5 rules). The semantic FD can filter the found anomalies based on the provided rules and metadata in 730ms on average (std: 500ms over 5 runs, using a cluster node with 8G ram and 10 CPUs).

3.7 Discussion

As stated in the requirements of Section 3.2, to deliver cost-effective and value-adding AD, FR & RCA, the proposed methodology must be adaptive, context-aware, precise, interpretable and should necessitate as little interventions needed as possible. The methodology that fulfills these requirements is FLAGS, which fuses the two known data- and knowledge-driven techniques together and incorporates valuable user feedback to adapt the models' internal functioning to new contexts or environments.

As explained in Section 3.3, relying on either one of the data- or knowledge-driven methodologies by itself would result in a less functional system. The knowledge-driven setup would not be able to detect faults because it cannot deal with the flood

of sensor data and the experts do not have enough knowledge to verify manually which patterns in the raw data might lead to possible faults. The data-driven technique would be able to detect outliers, but almost all detected outliers are different from each other. Pattern matching tools, delivering some additional information by grouping the anomalies, gave no impressive results and obeyed the ML-based RCA to find useful causes. In the end, it comes down to a choice between almost no detected faults at all or giving anomalies to an operator without any explanation.

In this perspective, FLAGS fuses both approaches taking only their advantages. The evaluation of this system shows that most of the set requirements are met, as summarized in Table 3.2:

- **Precise:** The combination of knowledge-driven expertise with data-driven detections improved the detection rate.

  The Matrix Profiler operational in our evaluation use case will produce anomalies as outliers without updating its behavior after the initial 100 seconds. The semantic FD decides by using a GEO filter whether or not the detected outlier is of interest for an operator. FLAGS, therefore, filters a lot of false positives compared to the data-driven technique.

  The combination of both methods resulted in 75% correct filtered anomalies and only 25% of the anomalies were classified as misleading or false alerts. FLAGS again benefits from the combination of both the data- and knowledge-driven techniques. 100% of the found anomalies would have been false alerts using only the data-driven technique. The currently operating knowledge-driven approach would not even be able to make a single detection.

- **Require minimal human involvement:** FLAGS does not require to implement all the expert knowledge or provide fully trained ML models to generate useful insights. The evaluation section, for example, uses the Matrix Profiler instantiated on the first 100 seconds of the dataset. Almost all of the knowledge about possible faults was provided using FMEA tables and automatically mapped as described in Section 3.3. Fusing both a data- and knowledge-driven approach leads to better results than the approaches either would have exhibited on their own. Anomalies are filtered, reducing the need to improve the ML models themselves. New anomalies are merged and/or relabeled, resulting in automated adaptations in failure recognition, and thus enriching the knowledge-driven components.

  The evaluation shows the benefits of this fusion by demonstrating that no human effort is needed to retrain or adapt the knowledge to start detecting switch events after the first two train rides are finished. To thoroughly verify this requirement, deploying FLAGS within the operational environment is needed to perform realistic tests by the operators.

- **Context-aware:** The overall FLAGS methodology incorporates context information from the central KG in both the data- and knowledge-driven modules. From this perspective, the ML modules can benefit from the detected faults provided by the knowledge-driven parts and vice versa.

  In our evaluation, this contextual enhancement reduces the manual inspections needed for those anomalies related to the track issues. As Televic is interested in scheduling predictive maintenance for trains, other companies have to handle the anomalies related to track issues. However, detecting track issues reduces the number of falsely requested maintenance actions or, more in general, the number of false positives. FLAGS was able to explain four anomalies (two related to the bridge and two near the switch) based on the context of the ones that were found earlier.

  Without the use of context, all these anomalies would have been classified according to their signal shape as shown in Figure 3.14. The Matrix Profile only considers the raw signals, and as already discussed, limited information could be extracted from these patterns due to the different behavior of the trains in this track segment.

- **Adaptive:** In general, the FLAGS methodology copes directly with the user-provided feedback and adapts the models by merging, deleting or relabeling the detected faults or anomalies.

  During the evaluation, relabeling two track anomalies as switches enabled the system to detect switch events for the third train ride. On average, 8.36 minutes are required to apply a newly learned rule in the semantic FD (1694ms to store the user feedback in the database, 500s to perform the rule mining and 31ms to install this found rule). These are all non-blocking operations, so the operator was not aware of these delays at all.

  Without the FLAGS methodology, changing the behavior of the semantic FD would require more time to adapt. The operator would have to translate his findings into a user-defined rule, which must then be applied in the semantic FD rule base. In most cases, this would even require stopping the semantic FD for a while to make sure the update was sufficient. To adapt the behavior of the data-driven components without FLAGS requires retraining or even a reinitialization of the ML algorithms.

- **Interpretable:** The FLAGS approach combines all the output of the generated modules in a semantic format, which provides the end-user or operator with a lot more interpretability than the fault or anomaly could give by themselves.

  During the evaluation, all results were shown in the Dynamic Dashboard. As this dashboard benefits from the semantically generated output, an operator only has to perform several clicks to investigate the anomaly's occurrence. The

preferred visualization was decided by reasoning about the occurrence of these anomalies. Out of the eight anomalies, the operator could easily identify which of the track issues were related to a switch event, as visualized in Figure 3.8. Additionally, the semantic FD within FLAGS could cope with the third track issue and automatically relabel it as a switch event. This use case already showed the efficiency of investigating anomalous occurrences using FLAGS.

For six of the eight anomalies, a useful interpretation was given by FLAGS. The data-driven AD module was unable to explain these cases, and they only became useful for the operator when the semantic FD filtered them.

However, the FLAGS system also has some limitations. As the adaptation of the whole approach is orchestrated by providing feedback, falsely pinpointing parts as normal behavior or wrongly classifying events as anomalies, would require multiple components to reset. This is mainly due to the fact that these components depend on the provided feedback and adapt their behavior based on them. Another drawback is the automatic application of new rules in the semantic FD. While this makes the knowledge-driven part adaptable, it reduces the experts' control of the system's internal functioning. At last, the amount of data floating through FLAGS is doubled as the data-driven approaches prefer raw instead of semantified data. Such huge amounts of data and the need to transform them into a semantic representation require high-end cloud set-ups.

Table 3.2: Overview of the proposed FLAGS methodology regarding the requirements made in Section 3.2.

	Data-driven	Knowledge-driven	FLAGS
Precise		X	**X**
Minimal human involvement	X		**(X)**
Context-aware	(X)	X	**X**
Adaptive	X		**X**
Interpretable		X	**X**

3.8 Conclusion

To deliver cost-effective and value-adding AD, FR & RCA within streaming sensor environments, the used algorithms must be adaptive, context-aware, precise, interpretable and should require minimal human intervention. None of the currently available methodologies meet all these mentioned requirements. Knowledge-driven techniques are not adaptive and too time-consuming to construct & maintain. Data-driven methods are not context-aware and are often not interpretable. In this work, we have proposed FLAGS, a methodology that covers all those requirements.

First, we have combined the results of both data- and knowledge-driven techniques, and by using semantic filters enriched by metadata, a lot of the detected anomalies can be classified as known behavior. The opportunity to interpret occurring, data-driven anomalies by using semantic data, reduces the human involvement needed for the operators to find the correct alerts. Second, explaining the output of a data-driven model based on expert information is way more intuitive than deriving them through an experimental approach. Data-driven techniques are more expressive and can give valuable model insights when taking the metadata into account. Inspecting the data by visualizing the data with user- and anomaly-driven widgets gives the operators a necessary tool to investigate unknown system behavior.

Concretely, the whole end-to-end methodology with ML AD, ML RCA, Semantic FD/RCA, Semantic rule mining and a Dynamic dashboard which provides user feedback, is being tested using a predictive maintenance case in the railway domain. Differentiating between train or track problems is quite common in these domains. By applying a smart IoT device on multiple train rides, a single operator can already receive valuable insights. The proposed methodology gives the operator a new tool to investigate possible errors in the system. Their knowledge about the railway domain is used as the primary input, together with the feedback and information on possible anomalies and possible causes. Further evaluation of multiple cases is needed, as well as an in-depth evaluation to determine the reduction in human involvement. Additional future work will investigate how the rule mining component can be made more dynamic and can become visually available for the operator such that they can verify the newly generated knowledge and accompanied rules.

Code availability:

The different components of the proposed FLAGS methodology are made available online in different repositories. This section gives an overview of each component/-tool and the link where additional information can be found:

- Mappings tool (YARRML, RML Mapper, RMLStreamer):
  https://github.com/RMLio

- Matrix Profile code:
  https://github.com/IDLabResearch/seriesdistancematrix

- Semantic FD code:
  https://github.com/IBCNServices/StardogStreamReasoning

- FOLIO ontology + transformation scripts FMEA and FTA:
  https://github.com/IBCNServices/Folio-Ontology

Acknowledgment

This research is part of the imec ICON project Dyversify, co-funded by imec, VLAIO, Renson Ventilation NV, Televic Rail & Cumul.io. Televic Rail, the manufacturer of the Cosamira Edge module, delivered the use case data, expert knowledge and additional metadata to evaluate the proposed methodology.

References

[1] S. Latif, H. Afzaal, and N. A. Zafar. *Intelligent traffic monitoring and guidance system for smart city.* In 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pages 1–6, 2018.

[2] R. Saia. *Internet of Entities (IoE): a Blockchain-based Distributed Paradigm to Security*, 2018. arXiv:1808.08809.

[3] J. Ren and S.-J. Cao. *Incorporating online monitoring data into fast prediction models towards the development of artificial intelligent ventilation systems.* Sustainable Cities and Society, 47:101498, 2019.

[4] M. T. Mardini, Y. Iraqi, and N. Agoulmine. *A Survey of Healthcare Monitoring Systems for Chronically Ill Patients and Elderly.* Journal of medical systems, 43(3):50, 2019.

[5] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. *Unsupervised real-time anomaly detection for streaming data.* Neurocomputing, 262:134–147, 2017.

[6] *Using predictive analytics in anomaly detection and fault recognition*, Apr 2019. https://www.turbomachinerymag.com/using-predictive-analytics-in-anomaly-detection-and-fault-recognition/.

[7] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada. *Survey on Models and Techniques for Root-Cause Analysis*, 2017. arXiv:1701.08546.

[8] S. Sendelbach and M. Funk. *Alarm fatigue: a patient safety concern.* AACN advanced critical care, 24(4):378–386, 2013.

[9] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. *A comparative study of anomaly detection schemes in network intrusion detection.* In Proceedings of the 2003 SIAM International Conference on Data Mining, pages 25–36. SIAM, 2003.

[10] R. B. Miller. *Response time in man-computer conversational transactions.* In Proceedings of the December 9-11, 1968, fall joint computer conference, part I, pages 267–277, 1968.

[11] E. Tjoa and C. Guan. *A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI*, 2019. arXiv:1907.07374.

[12] D. H. Stamatis. *Failure mode and effect analysis: FMEA from theory to execution.* ASQ Quality press, 2003.

[13] W.-S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. *Fault Tree Analysis, Methods, and Applications A Review.* IEEE transactions on reliability, 34(3):194–203, 1985.

[14] I. Lykourentzou, K. Papadaki, A. Kalliakmanis, Y. Djaghloul, T. Latour, I. Charalabis, and E. Kapetanios. *Ontology-based operational risk management*. In 2011 IEEE 13th Conference on Commerce and Enterprise Computing, pages 153–160. IEEE, 2011.

[15] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. *The SSN ontology of the W3C semantic sensor network incubator group*. Web semantics: science, services and agents on the World Wide Web, 17:25–32, 2012.

[16] B. Steenwinckel, P. Heyvaert, D. De Paepe, O. Janssens, S. Vanden Hautte, A. Dimou, F. De Turck, S. Van Hoecke, and F. Ongenae. *Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses*. In 9th International Semantic Sensor Networks workshop, co-located with 17th International Semantic Web conference (ISWC 2018), volume 2213, pages 17–31, 2018.

[17] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. *HermiT: an OWL 2 reasoner*. Journal of Automated Reasoning, 53(3):245–269, 2014.

[18] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. *Pellet: A practical owl-dl reasoner*. Web Semantics: science, services and agents on the World Wide Web, 5(2):51–53, 2007.

[19] M. Schaaf, S. G. Grivas, D. Ackermann, A. Diekmann, A. Koschel, and I. Astrova. *Semantic complex event processing*. Recent Researches in Applied Information Science, pages 38–43, 2012.

[20] P. Bonte, R. Tommasini, E. Della Valle, F. De Turck, and F. Ongenae. *Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams*. Sensors, 18(11):3832, 2018.

[21] K. A. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque. *Internet of Things: A survey on machine learning-based intrusion detection approaches*. Computer Networks, 151:147 – 157, 2019.

[22] I. Cramer, P. Govindarajan, M. Martin, A. Savinov, A. Shekhawat, A. Staerk, and A. Thirugnana. *Detecting Anomalies in Device Event Data in the IoT*. In IoTBDS, pages 52–62, 2018.

[23] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. *Outlier detection for temporal data: A survey*. IEEE Transactions on Knowledge and Data Engineering, 26(9):2250–2267, 2013.

[24] D. J. Hill and B. S. Minsker. *Anomaly detection in streaming environmental sensor data: A data-driven modeling approach*. Environmental Modelling & Software, 25(9):1014–1022, 2010.

[25] D. D. Paepe, S. V. Hautte, B. Steenwinckel, F. D. Turck, F. Ongenae, O. Janssens, and S. V. Hoecke. *A generalized matrix profile framework with support for contextual series analysis*. Engineering Applications of Artificial Intelligence, 90:103487, 2020.

[26] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. *Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets*. In 2016 IEEE 16th international conference on data mining (ICDM), pages 1317–1322. IEEE, 2016.

[27] M. Hushchyn, A. Sapronov, and A. Ustyuzhanin. *Machine Learning Algorithms for Automatic Anomalies Detection in Data Storage Systems Operation*. Advances in Systems Science and Applications, 19(2):23–32, 2019.

[28] N. Kant and M. Mahajan. *Time-Series Outlier Detection Using Enhanced K-Means in Combination with PSO Algorithm*. In Engineering Vibration, Communication and Information Processing, pages 363–373. Springer, 2019.

[29] P. Chemweno, L. Pintelon, P. Muchiri, et al. *i-RCAM: Intelligent expert system for root cause analysis in maintenance decision making*. In 2016 IEEE International Conference on Prognostics and Health Management (ICPHM), pages 1–7. IEEE, 2016.

[30] M. Shokoohi-Yekta, Y. Chen, B. Campana, B. Hu, J. Zakaria, and E. Keogh. *Discovery of meaningful rules in time series*. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 1085–1094. ACM, 2015.

[31] A. Alaeddini and I. Dogan. *Using Bayesian networks for root cause analysis in statistical process control*. Expert Systems with Applications, 38(9):11230–11243, 2011.

[32] P. Van den Kerkhof, J. Vanlaer, G. Gins, and J. F. Van Impe. *Contribution plots for statistical process control: Analysis of the smearing-out effect*. In 2013 European Control Conference (ECC), pages 428–433. IEEE, 2013.

[33] R. Saia, S. Carta, D. R. Recupero, G. Fenu, and M. M. Stanciu. *A Discretized Extended Feature Space (DEFS) Model to Improve the Anomaly Detection Performance in Network Intrusion Detection Systems*. In Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Vienna, Austria, pages 17–19, 2019.

[34] H. H. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta. *Ensembles of incremental learners to detect anomalies in ad hoc sensor networks.* ad hoc networks, 35:14–36, 2015.

[35] H. H. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta. *Spatial anomaly detection in sensor networks using neighborhood information.* Information Fusion, 33:41–56, 2017.

[36] S. M. Lundberg and S.-I. Lee. *A unified approach to interpreting model predictions.* In Advances in neural information processing systems, pages 4765–4774, 2017.

[37] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, and P. Eckersley. *Explainable machine learning in deployment.* In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pages 648–657, 2020.

[38] P. Kamat and R. Sugandhi. *Anomaly Detection for Predictive Maintenance in Industry 4.0-A survey.* In E3S Web of Conferences, volume 170, page 02007. EDP Sciences, 2020.

[39] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque. *Artificial Intelligence-Driven Mechanism for Edge Computing-Based Industrial Applications.* IEEE Transactions on Industrial Informatics, 15:4235–4243, 2019.

[40] S. S. Khan and M. G. Madden. *One-class classification: taxonomy of study and review of techniques.* The Knowledge Engineering Review, 29(3):345–374, 2014.

[41] V. M. Janakiraman and D. Nielsen. *Anomaly detection in aviation data using extreme learning machines.* In 2016 International Joint Conference on Neural Networks (IJCNN), pages 1993–2000. IEEE, 2016.

[42] S. K. Bose, B. Kar, M. Roy, P. K. Gopalakrishnan, and A. Basu. *ADEPOS: anomaly detection based power saving for predictive maintenance using edge computing.* In Proceedings of the 24th Asia and South Pacific Design Automation Conference, pages 597–602, 2019.

[43] G. F. Luger. *Artificial intelligence: structures and strategies for complex problem solving.* Pearson education, 2005.

[44] H. Long. *Aircraft Oil System Fault Detection Expert System.* In 2015 International Conference on Electromechanical Control Technology and Transportation, pages 196–199. Atlantis Press, 2015. doi:https://doi.org/10.2991/icectt-15.2015.38.

[45] F. Curreri, S. Graziani, and M. G. Xibilia. *Input selection methods for data-driven Soft sensors design: Application to an industrial process.* Information Sciences, 537:1 – 17, 2020.

[46] F. A. Souza, R. Araújo, and J. Mendes. *Review of soft sensor methods for regression applications.* Chemometrics and Intelligent Laboratory Systems, 152:69–79, 2016.

[47] J. Liu, Y. Li, X. Tian, A. K. Sangaiah, and J. Wang. *Towards Semantic Sensor Data: An Ontology Approach.* Sensors, 19(5):1193, 2019.

[48] S. Vanden Hautte, P. Moens, J. Van Herwegen, D. De Paepe, B. Steenwinckel, S. Verstichel, F. Ongenae, and S. Van Hoecke. *A dynamic dashboarding application for fleet monitoring using semantic web of things technologies.* Sensors, 20(4):32, 2020.

[49] Y. Jiang, B. K. Chen, and C. Thompson. *A comparison study of ride comfort indices between Sperling's method and EN 12299.* International Journal of Rail Transportation, 7(4):279–296, 2019.

[50] P. Umiliacchi, R. Shingler, G. Langer, and U. Henning. *A new approach to optimisation through intelligent integration of railway systems: the InteGRail project.* In Proceedings of the 7th WCRR, World Conference on Railway Research, 2006.

[51] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. *RML: a generic language for integrated RDF mappings of heterogeneous data.* In 7th Workshop on Linked Data on the Web, Proceedings, page 5, 2014.

[52] G. Haesendonck, W. Maroy, P. Heyvaert, R. Verborgh, and A. Dimou. *Parallel RDF generation from heterogeneous big data.* In SBD2019, the International Workshop on Semantic Big Data, pages 1–6. ACM Press, 2019.

[53] D. De Paepe, O. Janssens, and S. Van Hoecke. *Eliminating noise in the matrix profile.* In ICPRAM2019, the 8th International Conference on Pattern Recognition Applications and Methods, pages 84–93, 2019.

[54] J. Lehmann. *DL-Learner: learning concepts in description logics.* Journal of Machine Learning Research, 10:2639–2642, 2009.

4

INK: Knowledge Graph Representation for Efficient and Performant Rule Mining

Transforming structured documents into a KG is only one aspect to reduce the efforts needed by an expert to create an adaptive HAI application. As shown in Chapter 3, semantic rule mining can be used to transform the information outputted by the data and knowledge-driven models into new rules or new insights which can be taken into account automatically in a next analysing iteration. Different rules exist based on the underlying reasoning mechanism, as stated in Section 1.1.2. Inductive reasoning uses generalised rules based on specific facts while deductive reasoning uses rules based on generalised facts to form a specific conclusion. The currently available semantic rule mining techniques to generate such rules for deductive and inductive reasoning are all designed for rather static graphs and the underlying knowledge representation is designed such that more than one technique is needed to generate both deductive and inductive rules from one KG. In this chapter, INK is proposed and creates a representation that can be used to mine both deductive and inductive rules. The whole approach is for both reasoning cases evaluated on several well-stated benchmark datasets. This chapter investigates research question 3: "Can we optimise rule mining techniques for HAI applications by representing the KG in one data structure such that new insights and rules can be derived more accurately and efficiently?" and validates hypothesis 3: "A KG rule mining technique which can mine rules for both descriptive and prescriptive reasoning cases, is in general more than 10% accurate and requires less than 50% of the time to generate prescriptive rules while being able to generate 5% more rules in a descriptive setting on well-stated benchmark datasets and compared to the current state-of-the-art techniques.".

⋆ ⋆ ⋆

B. Steenwinckel, F. De Turck, F. Ongenae

Under review in Semantic Web Journal, submitted in May 2022.

Abstract
Semantic rule mining can be used for both deriving task-agnostic or task-specific information within a Knowledge Graph (KG). Underlying logical inferences to summarise the KG or fully interpretable binary classifiers predicting future events are common results of such a rule mining process. The current methods to perform task-agnostic or task-specific semantic rule mining operate, however, a completely different KG representation, making them not suitable to perform both tasks or incorporate each other's optimizations. This also results in the need to master multiple techniques for both exploring and mining rules within KGs, as well losing time and resources when converting one KG format into another. In this chapter, we present INK, a KG representation based on neighbourhood nodes of interest for improved decision support. By selecting one or two sets of nodes of interest, the designed rule miner on top of this INK representation will either mine task-agnostic or task-specific rules. In both subfields, the INK miner outperforms the currently state-of-the-art semantic rule miners on 14 different benchmark datasets within multiple domains.

4.1 Introduction

Knowledge graphs (KGs) are increasingly used as data structures to combine domain expertise with raw data values [1]. In general, a KG is a multi-relational directed graph, $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ are the vertices or entities in our graph and $\mathbb{E}$ the edges or predicates. The example KG represented in Figure 4.1 shows eight interlinked nodes describing four members of the band Coldplay. Three of these members have a common subgraph as they all studied and were born in England. One member was born in Scotland which is, at time of writing, still a part of the United Kingdom (UK).

Numerous applications are built upon these KGs, covering various domains such as industry 4.0, pervasive health and smart cities [3–5]. These applications interact with the KGs directly or transform the graph into a vector representation to perform Machine Learning (ML) related tasks [6]. Rule mining is also such a KG application, where the goal is to find logical rules in a given KG. For example, a rule mining task for the given example KG in Figure 4.1 could find the logical rule: *If X has Alma mater Y and Y is Located In Z, Then X is born in Z*. Such logical rules will come with a certain confidence score, defining the general applicability of the rule. The more reliable rules can then be used to complete the KG, perform downstream applications such as fact prediction, fact checking or anomaly and error detection.

Rule mining is part of data mining in general, where two broad subfields exist. On the one hand, there is task-agnostic or descriptive mining, where one wants to mine

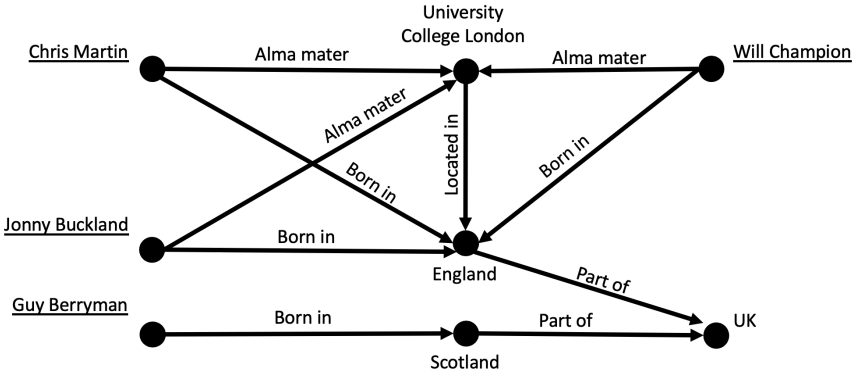


Figure 4.1: Simple example of a KG, extracted from DBpedia [2]. Eight nodes are defined, linked to each other by four unique labelled edges.

some general information about the KG or some general facts, which hold beyond this provided KG and are generally applicable.

This subfield was originally created to discover hidden knowledge from transactional data, such as relational databases. Association Rule Mining (ARM) is the best-known descriptive technique. A transaction in ARM is an observation of the co-occurrence of a set of items. ARM can be applied to semantic data or KGs by converting the internal representation to a set of transactions. ARM thus identifies the transactions that identify co-occurrences of items that appear frequently in the KG by calculating the associated metrics to quantify this, such as the confidence and the support of the transactions. The logical rule defined, *If X has Alma mater Y and Y is Located In Z, Then X is born in Z*, is such a possible hidden rule that could be mined with an ARM application. ARM for KGs is used in data integration and KG completion tasks [7, 8].

On the other hand, prescriptive mining is more task-specific and performs inferences on the current data, to make predictions in the future [9]. Inductive Logic Programming (ILP) is the best-known paradigm in this subfield. The ILP techniques deduce logical rules from a positive set of nodes and require some (generated) negative set of counter-examples. An example ILP task could be to find one general rule to describe all four members of the Coldplay band in Figure 4.1. The positive set of nodes selected for this mining task are underlined in Figure 4.1. One possible rule could state the *born In ?x and ?x Part of UK* relationships hold for all members.

Both subfields are complementary. ILP performs the task-specific cases, as specific facts are needed for those cases. Therefore, ILP directly captures the available related information in the KG to generate the rules. The ILP program will immediately use the available predicate-object information to discriminate between the provided

positive and negative set. ILP can, however, be relatively slow and can therefore not handle the huge amount of data that KGs provide today. ARM can handle large KGs and generate rules for fully task-agnostic problems. It is fast and scales to large graphs. This often results in the fact that ARM generates a lot of nonsense or too generally applicable rules as it considers all triples or facts. The generated ARM rule for our example KG is such a rule with limited effect, because people do not always study where they were born.

There doesn't exist a technique which can perform both prescriptive (task-specific) and descriptive (task-agnostic) rule mining for KGs. The main reason, to our knowledge, is that the current techniques available for both tasks require a different internal representation of the KG. These transformations are performed in relation to the subfield they are operating on. ARM mainly requires the KG to represent as transactions, which reduces the linked aspects of the existing KG. ILP directly works on the graph representation itself, leading to the earlier discussed performance issues.

In this work, we propose a technique to perform rule mining on KGs by Instance Neighbouring using Knowledge (INK). INK is a new way to represent a KG by analysing the neighbourhoods of selected nodes of interest. In the case of mining rules over the whole KG, the neighbourhood of all nodes in the KG are used inside the representation. When a more specific task is given, neighbourhoods of the nodes which have to be considered are only taken into account. In this perspective the INK representation combines the best of both worlds, as it can be used both to mine specific rules in a descriptive rule mining task and reduce the time needed to perform prescriptive mining tasks. In both cases, INK is able to capture the complexity of the KG in an efficient manner.

The remainder of this chapter is structured as follows. Section 4.2 provides an overview of the current available semantic rule mining techniques and discusses their internal representation of the KG. Section 4.3 details the INK KG representation. Section 4.4 shows how INK can be incorporated into a rule mining system. Both the implementation of INK and the accompanying rule miner are discussed in Section 4.5. In Section 4.6, we evaluate INK for both the task-agnostic and task-specific rule mining, and compare the results with the current state-of-the-art. Section 4.7 discusses the advantages and drawbacks of INK in the perspective of task-agnostic and task-specific mining. At last, the conclusion of this work is provided in Section 4.8.

4.2 Related work & background

Rule mining has a long history, but the existing techniques can be either based on ARM or ILP. ARM searches for implication (if ... then ...) rules, such as *"If a person X has an Alma mater Y, and Y is located in Z, then X is born in Z"*. ILP techniques deduce logical rules from ground facts and require negative statements as counter-examples. Both techniques were already applied in the context of KGs. This section discusses the two

state-of-the-art techniques for KGs, together with some recent advances, namely the ARM technique AMIE and the ILP technique DL-Learner.

4.2.1 AMIE

AMIE mines task-agnostic rules in the form of horn clauses [10]. Horn clauses are denoted as horn rules when they contain an implication. They usually consist of a head and a body, where the head is a single atom:

$$B_1 \wedge B_2 \wedge ... \wedge B_n \Rightarrow r(x, y)$$

with head $r(x, y)$ and body $B_1 \wedge B_2 \wedge ... \wedge B_n$. The latter is frequently represented as $\vec{B}$. The rules state that if all instantiated body atoms appear in the KG, the head atom can be derived. AMIE reduces the search space by searching for connected and closed rules that are not reflexive:

- A rule is connected if every atom is connected transitively to every other atom of the rule.
- A rule is closed if all its variables are closed. A variable is closed when it appears at least twice in the rule.
- A rule is reflexive if it contains atoms of the form r(x,x)

The example rule *"If a person X has an Alma mater Y, and Y is located in Z, then X is born in Z"* is an example of a connected and closed, not reflexive rule.

4.2.1.1 Pruning of AMIE candidate rules

Additionally, support and head coverage measures are defined to further prune candidate rules. The support of a rule quantifies the number of correct predictions in the existing KG. AMIE defines the support of a rule as the number of distinct pairs of subjects and objects in the head of all the instantiations that appear in the KG. More in general, the support of a rule R in a KG $\mathcal{G}$ is the number of true derivations r(x, y) (with r(x, y) a the head atom as explained above) that the rule makes in the KG:

$$\begin{aligned}\text{support}(R) = &|\{r(x, y) : (\mathcal{G} \wedge R \models r(x, y)) \\ &\wedge r(x, y) \in \mathcal{G}\}|\end{aligned}$$

By providing a threshold on this support value, rules and facts which are less common can be pruned.

Support is an absolute number, such that one has to know the absolute size of the KG to give a meaningful value for the threshold. To avoid this, AMIE proposed the head coverage measure to quantify the ratio of the known true facts that are implied

by the rule. It is the ratio of instantiations of the head atom that are derived by the rule:

$$hc(\vec{B} \Rightarrow r(x,y)) = \frac{\text{support}(\vec{B} \Rightarrow r(x,y))}{|\{(x',y') : r(x',y') \in \mathcal{G}\}|}$$

Based on the example KG of Figure 4.1, AMIE mined the following rule:

$$alma\_mater(?x,?y) \wedge located\_in(?y,?z) \\ \Longrightarrow born\_in(?x,?z)$$

This rule has a support of 3 because three band members fulfil this rule. The head coverage of this rule is 3/4 as one member has the *born_in* relation, but without a link to the *alma_mater* relation.

Both the support and head coverage only consider the correct predictions of the generated rule. Confidence is a measure that also takes false predictions into account. The standard confidence of a rule is the ratio of all its predictions that are in the KG. All facts that are not in the KG are seen as negative evidence.

$$conf(\vec{B} \Rightarrow r(x,y)) = \frac{\text{support}(\vec{B} \Rightarrow r(x,y))}{|(x,y) : \vec{B}|}$$

However, it must be treated with care within the context of KGs. ARM operates under the closed world assumption: a statement that is true is also known to be true and conversely, what is not currently known to be true, is false and introduces negative evidence for those cases which are not available in the dataset. KGs can, however, follow the open world principle: the truth value of a statement may be true irrespective of whether or not it is known to be true [11]. As ARM does not take the difference between "*unknown*" and "*false*" into account, this can lead to surprising results when mining rules inside a KG.

AMIE resolves this problem by generating negative examples for a rule by means of the Partial Completeness Assumption (PCA). Here, they assume that if one y for a given x and r in r(x,y) head is known, then all the y's are known for that x and r. Under the PCA, the denominator of the confidence formula is not the size of the entire set of conclusions derived from the body of the rule, but the number of facts that we know to be true together with the facts that we assume to be false.

$$conf_{pca}(\vec{B} \Rightarrow r(x,y)) = \frac{\text{support}(\vec{B} \Rightarrow r(x,y))}{|(x,y) : \vec{B} \wedge r(x,y')|}$$

This PCA has been applied in the Google Knowledge Vault and is more commonly known as the "local completeness assumption" [12].

In our example, the confidence of our rule is 1, as no negative facts can be found that contradict the rule. If we would add an additional relationship in our example KG, which states that Guy also studied at the University College London, one negative fact would be available and the confidence score would drop to 3/4.

4.2.1.2 Implementation AMIE Rule Mining

Mining rules with AMIE is an iterative process. The algorithm maintains a queue of possible rules, initially filled with all head atoms of length 1. It then iteratively dequeues a possible rule. If it meets certain criteria, i.e. closed and of high quality regarding the used pruning technique, it is pushed to the output. If the rule does not exceed the maximum number of atoms maxLen, it goes through a refinement process which expands the rule (the parent) to produce a set of new rules (the children). This refinement procedure is an efficient way to explore the search space as enumerating all possible combinations of conjunctions of atoms is infeasible for large KGs. Hence, they explore the search space by iteratively extending rules using one of the following refinement operators:

- Add Dangling Atom: Add an atom that uses a fresh variable for one of its two arguments. The other argument is a variable that is shared with the rule.
- Add Instantiated Atom: adds a new atom to a rule that uses an entity for one argument and shares the other argument (variable) with the rule.
- Add Closing Atom: adds a new atom to a rule so that both of its arguments are shared with the rule.

The rules who are not already used before or are not pruned to single atoms after this refinement process are pushed into the queue and can be extended in a next iteration.This process is repeated until the queue is empty.

4.2.2 Optimizations of AMIE approach

To efficiently calculate the pruning metrics, such as support and head coverage, AMIE first used multiple count projection queries inside the mining operators, which translate into very inefficient queries in both SPARQL and SQL. To resolve these inefficiencies, an in-memory database was explicitly designed geared towards this type of queries. This database stores for the S, P and O index of the triple (S,P,O), a nested hash table providing the information of the two other triple parts. More specifically, for each subject, two hash tables are created: (a) one with as keys the predicates and as values the objects, and (b) one with as keys the objects and as values the predicates. This process is repeated for all six combinations. Searching the relevant values in these hash maps can then be performed in constant time.

Several improvements were made to increase the efficiency of AMIE even further. A new version of AMIE, called AMIE+ [13], speeds up 2 different parts of the main rule mining algorithm: the refinement phase and the confidence evaluation. Recently, the AMIE framework was even further upgraded, becoming AMIE3 [14], and incorporates new pruning strategies, parallelization, and a lazy computation of confidence scores to allow the system to scale effortlessly to large KGs. When we refer to AMIE

in the next sections of this chapter, we refer to the most recent version AMIE3, unless indicated differently.

4.2.3 DL-learner

DL-Learner is defined in an open software framework, employing methods that are able to use the complex structure of available background knowledge when learning hypotheses [15]. The rule mining capabilities of DL-Learner can solve three types of learning problems:

- **Standard Supervised Learning:** Given a set of positive and negative labels, DL-Learner tries to find an OWL class expression such that all, or as many as possible, positive examples are instances of C and none, or as few as possible, negative examples are instances of C.
- **Positive Only Learning:** Similar as the above, but searching for a class expression that covers the positive examples without providing negative samples. The goal is to still generalise sufficiently.
- **Class Learning:** Given an already existing class A within the ontology, DL-Learner finds an expression to describe A, using the instances of the class as positive examples. Both the existing knowledge about A in the ontology and the positive examples are used to describe A.

The algorithms to solve these learning problems are derived from ILP and genetic programming, extended with theoretical and algorithmic foundations going from learning complex definitions in ontologies to generic schema enrichment and fuzzy description logics.

Almost all of these algorithms are also, similar to AMIE, based on the principle of refinement operators. Within DL-Learner, learning can be seen as the search for a correct concept definition in an ordered space. In such a setting, one can define suitable operators to traverse the search space. The goal is to use those operators that have many useful properties like finiteness, non-redundancy, properness and completeness, while still allowing to efficiently traverse through the search space in pursuit of good hypotheses.

4.2.3.1 Implementation DL-Learner

The OWL Class Expression Learner (OCEL) is such a common algorithm, which uses a proper and complete refinement operator to build a search tree, while using heuristics that control how the search tree is traversed [16]. In this tree, the nodes are annotated with a score and the number of times they have been expanded. Based on these two values, OCEL defines weak nodes, i.e. when the number of uncovered positive examples is above a given threshold. They are never visited, which allows the

algorithm to ignore those parts of the search space resulting in improved efficiency. All the other algorithms made available by DL-Learner follow the same procedure to traverse the search space, but are optimised for a certain problem or use Description Logics to infer more knowledge within the KG. The Class Expression Learner for Ontology Engineering (CELOE) is built on the OCEL algorithm, but uses a different heuristic and introduces some bias to shorter class expressions. All algorithms return the expressions with the highest scores and sort them based on the number of expansions until a certain time threshold parameter is exceeded [16].

If we for example want to learn the class expressions that describe the four band members in our KG (underlined nodes), we can instruct DL-Learner's CELOE algorithm to mine rules for only these specific nodes. Fives rules are mined and ordered based on their accuracy scores:

```
born_in some Thing 100.0%
(alma_mater some university_college_london) or (born_in some Scotland) 100.0%
(alma_mater some university_college_london) or (born_in some Thing) 100.0%
(alma_mater some Thing) or (born_in some Scotland) 100.0%
(alma_mater some Thing) or (born_in some Thing) 100.0%
```

4.2.3.2 Optimizations of DL-Learner approach

As the DL-Learner algorithms are developed to, preferably, only output the optimal solutions, some optimizations such as divide and conquer search techniques are provided to reduce the number of sub-optimal solutions within a certain time interval. In all cases, the goal is to restrict the set of nodes for expanding a fixed size of candidates within a set.

To operate efficiently, DL-Learner requires to store the inferred knowledge within memory as the reasoning procedures afterwards check the provided instances based on this inferred knowledge. This storage in memory is needed to speedup the whole process, but restricts DL-Learner from being run on large and complex KGs. Statistical sampling approaches were used to resolve these memory issues and to test hypotheses in the presence of a large number of nodes within the KG [17].

4.2.4 Other semantic Rule miners

The recent advances in the area of embeddings and KG vector representations resulted in some additional semantic rule mining methods. The main goal of such miners is to deal with the possible incompleteness or large scale of the KGs, which reduces the need for partial completeness calculations as discussed in Section 4.2.1. One such miner is RuLes [18]. It iteratively constructs rules over a KG and collects feedback for assessing the quality of (partially constructed) rule candidates through specific queries issued to a precomputed embedding model. Within the Rules framework, the confidence measures capture the rule quality better than other techniques because they now reflect the patterns in the missing facts. The improved confidence measures,

therefore, improve the ranking of rules. An embedded version of the KG is used here to define the quality of the rule and is not used to mine the task-agnostic rules themselves.

Another such technique is RLvLR (Rule Learning via Learning Representations) miner, an embedding-based approach to rule learning focusing on descriptive rule mining [19]. This miner specifies a target predicate in a KG to mine quality rules whose head has that predicate. The combination of the technique of embedding in representation learning together with a new sampling method results in more quality rules than major systems for rule learning in KGs such as AMIE+. The main focus of the RLvLR miner is defined in the scope of only mining specific rules for a given predicate. The RLvLR miner is, however, not made publicly available, except for an compiled executable to reproduce the fixed experimental setup.

4.3 INK representation

While AMIE and DL-learner use refinement operators to traverse the search space, INK builds its internal representation by transforming the neighbourhood of the nodes of interest into a binary matrix representation. With this binary matrix representation, column operations based and comparisons of columns for a large number of nodes of interest can be easily performed to reveal new patterns or rules. To explain how this binary representation is built, we use the example KG visualised in Figure 4.1 throughout this section.

4.3.1 Neighborhood dictionary

INK operates by selecting nodes of interest. This can be both all nodes within a graph (for task-agnostic mining), as well as some nodes specified upfront (task-specific mining). In our example KG, we select two nodes of interest: `Chris Martin` and `Guy Berryman`. INK will first query the neighbourhood of a given depth for all these nodes of interest. If we define the depth parameter K to be two, the neighbourhood for `Chris Martin` will exist of the Alma mater and Born in relations, together with the neighbourhood of the *University College London* node providing the Located in relation and the neighbourhood of the *England* node with the Part of relation. To store these neighbourhoods efficiently, a dictionary representation is used. For a given node of interest, this dictionary is built in an iterative fashion. The predicates in a neighbourhood of depth one are inserted first into our dictionary, together with their corresponding objects as values. These dictionary values are lists, as a single predicate can occur multiple times with different objects in the neighbourhood of a node. For our given example node `Chris Martin`, we represent the neighbourhood at depth

one by:

$$\{\text{Alma mater} \rightarrow [\textit{University College London}],$$
$$\text{Born in} \rightarrow [\textit{England}]\}$$

To add the neighbourhoods of depths > one, INK concatenates the predicates together. By concatenating these relations, INK provides a path from the node of interest to another node within our graph without providing detailed information about all intermediate nodes on that path. However, this information is still available in the (key, value) pairs added to our dictionary at the lower neighbourhood's depths. In our example node, the previous dictionary will be extended with the following (key, value) pairs at depth two:

$$\text{Alma mater.Located in} \rightarrow [\textit{England}]$$
$$\text{Born in.Part of} \rightarrow [\textit{UK}]$$

Here, we see a link from the node of interest to the *UK* node over the Born in relation. The Born in object value is not represented in this (key, value) pair, but was specified at the previous depth 1.

In several cases, it is also beneficial to indicate that the relationship itself within the neighbourhood of a node of interest is provided. The current dictionary structure does not indicate this presence explicitly in the dictionary values list. To ensure multiple nodes of interesting with similar relationship edges within their neighbourhoods can be compared against each other on a predicate level, the transformation step will also explicitly state particular relationships are available:

$$\text{Alma mater} \rightarrow [\textit{True}],$$
$$\text{Born in} \rightarrow [\textit{True}],$$
$$\text{Alma mater.Located in} \rightarrow [\textit{True}]$$
$$\text{Born in.Part of} \rightarrow [\textit{True}]$$

Again, lists were used as values for our dictionary as a single node can have the same predicate multiple times, but with different object values.

Completely similar, the dictionary representation of `Guy Berryman` until depth 2 is:

$$\{\text{Born in} \rightarrow [\textit{Scotland}],$$
$$\text{Born in.Part of} \rightarrow [\textit{UK}]$$
$$\text{Born in} \rightarrow [\textit{True}],$$
$$\text{Born in.Part of} \rightarrow [\textit{True}]\}$$

More in general, combined for all nodes of interest $\mathcal{N}$, the initial data structure of INK uses the following format:

$$list(tuple(n, neighbourhood(n, k))), \forall \text{ n in } \mathcal{N}$$

where the $neighbourhood(n, K)$ is the function which outputs the dictionary representation for our node n till a defined depth K.

4.3.2 Binary format

As the $list(tuple(n, neighbourhood(n, k)))$ representation is 3 dimensional (one axis for the nodes of interest, one for the dictionary relation keys and one for dictionary object list values), an additional transformation is required to provide a binary representation of this data. All of the object's values inside our neighbourhood dictionary are combined using a delimiter § to their corresponding key. In the strict sense, the binary format is created by unravelling the value lists within our dictionary by concating them with the corresponding dictionary key. For the example nodes `Chris Martin` and `Guy Berryman`, we transformed both depth one and depth two neighbourhoods into:

Alma mater§*University College London* (4.1)

Born in§*England* (4.2)

Born in§*Scotland* (4.3)

Alma mater.Located in§*England* (4.4)

Born in.Part of§*UK* (4.5)

Alma mater§*True* (4.6)

Born in§*True* (4.7)

Alma mater.Located in§*True* (4.8)

Born in.Part of§*True* (4.9)

These transformations can now be easily represented as a binary matrix. The rows are defined by the nodes of interest, such that each cell indicates whether or not the subject of interest contains the relation(s)§object value. The binary INK representation for the example above is visualised in Table 4.1. Here, (3) is a specific relation for `Guy Berryman` and could be of interest to differentiate `Guy Berryman` from the other team members.

4.3.3 Extension modules

While the binary representation of INK reflects the whole KG, it can derive additional information based, e.g. datatype properties or the amount of relationships that

Table 4.1: INK's binary representation of `Chris Martin` and `Guy Berryman` nodes in the example graph of Figure 4.1

	(4.1)	(4.2)	(4.3)	(4.4)	(4.5)	(4.6)	(4.7)	(4.8)	(4.9)
C. Martin	1	1	0	1	1	1	1	1	1
G. Berryman	0	0	1	0	1	0	1	0	1

are available. This subsection describes two optional extension modules which are available in INK.

4.3.3.1 Numerical inequality

To deal with numerical data, a preprocessing module will check if the values corresponding to a specific relation are all floats or integers for all the corresponding objects and nodes of interest. When such a relation is found, we build a set of all possible inequalities using all the found objects for that relation. In our example KG, we could add the birth year of all our Coldplay members, which would be an integer value. When this extension module is enabled, all these integer values will be stored inside a set. INK compares for each node of interest the value of the birth year relation with all possible values in our set and adds a new entry to our neighbourhood dictionary as follow:

$$\begin{aligned} &\text{birth year} < l \rightarrow [\textit{True or False}] \ \ \& \\ &l >= \text{birth year} \rightarrow [\textit{True or False}], \\ &\forall\, \text{l in inequality set} \end{aligned}$$

Concrete, eight new entries for will be added, describing if the birth year of `Band Member X` is smaller than the birth year of the `Band Member Y`, or if birth year of `Band Member X` is greater than or equal the birth year of the `Band Member Y`, with both X and Y $\in$ `Chis Martin, Will Champion, Guy Berryman, Jonny Buckland`.

4.3.3.2 Relation count

Another preprocessing module is available in INK to deal with relations having more than one object value. It can be beneficial to indicate how many similar relations are available for a given node of interest. Therefore, a counting module adds new entries to the neighbourhoods dictionary when two or more similar relations are available. More specifically, if in our example graph `Chis Martin` would have a second alma mater relation, this module would add the following entry:

$$\text{count.alma mater} \rightarrow [2]$$

This entry can be directly transformed to

count.alma mater§*2*

as described above. The previous inequality module can also use these counting values, as they are stored as integer values.

4.4 INK Rule Mining

The matrix representation discussed above can be used to perform both task-specific and task-agnostic rule mining. More in general, rules will be built based on the column description of our binary matrix, as shown in Table 4.1. The fourth column in this example, i.e. Alma mater.Located in§*England*, already introduced implicitly a variable to ignore the specific alma mater located in England. This fourth column states that there is a relation from our nodes of interest about a non-specified university, school, or college that one formerly attended which is located in England. This column can be interpreted more formally by:

$$\text{Alma Mater}(?i, ?x) \wedge \text{Located in}(?x, \textit{England})$$

with $?x$ and $?i$ a variable.

As both task-specific and task-agnostic techniques use this representation, the only difference between them is how they interact and extract the relevant information. The task-agnostic miner operates on the columns themselves, comparing the Boolean values to build so-called frequent itemsets and create the rules. The task-specific miner operates on the rows to differentiate between the nodes of interest. The task-specific miner uses the columns as features within its model to define a more specific rule given the task it wants to solve.

4.4.1 Task-agnostic mining

Similar to all ARM techniques, the INK task-agnostic mining component defines frequent itemsets. Here, the frequent itemsets are, however, based on the columns of INK's binary representation. The most important aspect to calculate these frequent itemsets is the support level. But in order to build these frequent itemsets, we first have to extract the neighbourhood for all subject nodes containing a fact in our KG. For our example graph in Figure 4.1, this means that not only the neighbourhoods for `Chris Martin` and `Guy Berryman` will be extracted, but also for all other nodes in our KG as they are also a subjects of facts.

While the relation§object columns were by default extracted by INK, the task-agnostic miner is more interested in the relation only columns. These columns already introduce a variable near the end, for example we can write alma mater(?x) to indicate

that those columns contain an, not specified, alma mater, indicated through the variable x. Based on these relation columns, the frequent itemsets are determined by all the possible combinations of a predefined length.

In traditional frequent itemset miners, the number of items inside the set must be defined upfront. For INK, we fix the number of items within an itemset to two, as the depth parameter of the neighbourhood already implicitly introduces additional preconfigured items in our itemsets with possible lengths greater than one. In our example graph, the relation Alma mater.Located in is already such a predefined item, combining the Alma mater and Located in relationship. As this combined relation can already be found in the neighbourhood of the nodes of interest, there is a high chance that they can occur frequently together. When we add to this combined relation, a second, single relation, we implicitly have a frequent itemset of length 3. For example, if we combine Alma mater.Located in, with Born in, we get an itemset stating (Alma mater(?x, ?y) $\wedge$ Located in(?y, ?z), Born in(?x, ?z)). However, purely algorithmic, the length of the itemset remains to two. We just provided additional variables within the items themselves to chain relationships, occurring frequently together, within the KG. INK is in this perspective also not limited to mine closed rules while the rule can still be connected. The head atom can also contain additional free variables due INK's item representation within the frequent itemsets. Both advantages eventually lead to more and more advanced rules.

Whether two items within an itemsets represent an interesting rule, depends on the calculated support value and corresponding threshold. The support for each itemset within INK is calculated using the following rule:

$$|\forall_{R_1} \text{ in } C_{ink}, \forall_{R_2} \text{ in } C_{ink} \sum_{\forall \text{ o in } O_{R_1 \cap R_2}}^{R_1 != R_2} R_1 \S o \ \& \ R_2 \S o|$$

Where C_{ink} are all the relation-only columns of our INK representation, $O_{R_1 \cap R_2}$ contains the intersection of all object values of the two relations R_1 and R_2 and & is the bitwise and operator. Note that both R_1 and R_2 can be a chain of relationships as discussed above. The individual support measures for each of these items within the itemsets is calculated by counting the number of true values for each relation. Based on these itemsets, we can select both an antecedent and consequent to get rules of interest. Measures such as confidence, lift and conviction are calculated from the support values and can be used to filter these rules.

4.4.2 Task-specific mining

The task-specific mining approach is based on the Bayesian rule set mining technique described by Wang et al. [20]. In this approach, the model consists of a set of rules and each rule is a conjunction of conditions. The model predicts that an observation is in a positive class when at least one of these rules is satisfied. In contrast, the

observation belongs to the negative class if none of the rules apply. The approach described by Wang et al. optimises the search for these rule sets by relying on Bayesian analysis. A globally optimised rule set is learned by considering both the accuracy and the interpretability of a model, while keeping computation simple. By controlling the parameters of the Bayesian prior, large models are penalised. For the Bayesian Rule Set model, this results in a a smaller number of rules. Since a small number of rules must cover the positive class, each rule in this model must cover as many observations as possible. To enforce this, a threshold on the number of examples satisfying the rule, more commonly known as the support of a rule, is introduced. It is due to this threshold that a significant reduction of the rule set's search space can be made.

The required input for this Bayesian rule set mining is a binary matrix, which fit with the proposed INK representation of Section 4.3. All the extension modules enrich this binary representation, such that all of them can be used as well. To train this model, INK will extract the neighbourhoods from two sets of nodes of interest, for a given depth parameter. One set contains all the positive nodes, the other set contains all negative ones. For task-specific cases, it is therefore required to specify these sets upfront. The labels for each node are stored in a different array. Optional parameters, such as the support, maximum length of the concatenation and the maximum number of rules in the rule set can be provided as input for the algorithm.

The output of the Bayesian rule set mining module contains both the rules learned on the given training dataset to discriminate both positive and negative nodes of interest, as well the mechanism to evaluate the rules on the new unseen nodes.

4.5 Implementation

The INK representation described in Section 4.3 and the INK rule mining module of Section 4.4 are both implemented in Python. To extract the neighbourhoods for a given set of nodes of interest, a component was implemented which can query these relations iteratively. Two options are currently available inside this component: either a KG or a SPARQL endpoint is given as input. When a KG file is given, RDFLib [21] will be used to load the graph in the internal memory of the operating system. However, some large KGs can be hard to fit within the internal memory. Therefore, INK can use RDFLib in combination with the Header, Dictionary Triples (HDT) file format [22]. HDT compresses big RDF datasets while maintaining basic search operations, such as providing the neighbourhood of a node of interest. Listing 4.1 shows the query used for both options to extract the neighbourhood nodes and relation. The variable subject <ind> starts with the nodes of interest, but differs in each iteration given the graph. The datatype of the object within this query is used to determine if queried objects can be used as subjects in the next iteration (when the neighbourhood depth is not reached yet). The predicates and objects in each iteration

are stored as described in Section 4.3. Python's internal multiprocessing library is used to speed up the extraction of the neighbourhoods, as this operation can be performed over multiple processors given the amount of nodes of interest.

To transform the initial representation into a binary matrix, we used the Scikit-learn DictVectorizer [23] with the sparse option set to true and specifying the data type to be Boolean. This is necessary when we want to deal with large KGs and a large number of nodes of interest.

If target values are specified together with the nodes of interest, the INK miner assumes a task-specific mining operation can be executed. Code from Wang et. al. [20] was adapted to operate on our representation.

When a target array is not provided, task-agnostic mining is executed on the neighbourhoods of all nodes of interest. The task-agnostic code uses the MLxtend library [24] to produce the rules based on the calculated frequent itemsets, based on the INK representation.

The whole INK package is made available on GitHub[1].

```
SELECT ?p ?o ?dt
WHERE {
 <ind> ?p ?o.
 BIND (datatype(?o) AS ?dt)
}
```

Listing 4.1: SPARQL query

4.6 Evaluation set-up & results

Both the task-specific and task-agnostic mining capabilities are evaluated on multiple benchmark datasets as specified below. To extract the neighbourhoods of interest, all benchmark datasets were transformed to an HDT format such that the SPARQL query of listing 5.1 can be executed performant. All evaluations were performed on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz processor with 32 cores and 128gb RAM.

4.6.1 INK vs AMIE

To compare the task independent rule mining capacities, we made a comparison between INK and AMIE on five benchmark datasets, which were already frequently used during various AMIE evaluations. All these datasets are available in a TSV format (preferred by AMIE) and are transformed into HDT (after being transformed into NTRIPLES) for INK. YAGO (2 and 2s) is a semantic knowledge base derived from Wikipedia, WordNet and GeoNames [25]. The latest version, YAGO2s, contains 120M facts describing properties of 10M different entities. The DBpedia datasets (2.0

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

Table 4.2: Comparison between INK and AMIE3 on 5 benchmark datasets. Both the confidences measures for the top 10, 25 and top min number of rules of either INK or AMIE3 is visualized

	Confidence Top 10		Confidence Top 25		Confidence Top N		# Rules	
	INK	AMIE3	INK	AMIE3	INK	AMIE3	INK	AMIE3
Yago2	**0.553 (0.09)**	0.507 (0.08)	**0.421 (0.13)**	0.353 (0.15)	**0.12 (0.15)**	0.086 (0.13)	**294**	166
Yago2s	**0.927 (0.05)**	0.898 (0.08)	**0.787 (0.14)**	0.707 (0.18)	**0.31 (0.24)**	0.221 (0.23)	**754**	405
DBpedia 2.0	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	**0.329 (0.27)**	0.238 (0.3)	**16957**	8963
DBpedia 3.8	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)	**0.162 (0.22)**	0.126 (0.21)	**16499**	9383
Wikidata	1.0 (0.0)	1.0 (0.0)	0.998 (0.0)	0.998 (0.0)	**0.287 (0.29)**	0.223 (0.3)	**3993**	2121

and 3.8) are a subset of the crowd-sourced community effort to extract structured information from Wikipedia [2]. DBpedia 2.0 and DBpedia 3.8 contain 6.7M and 11.02M facts respectively. The Wikidata dataset is a Wikidata dump from December 2014 and contains 8.4M facts. Wikidata is a free, community-based knowledge base maintained by the Wikimedia Foundation with the goal to provide the same information as Wikipedia but in a computer-readable format [26]. All 5 benchmark datasets are made available by the Max Planck Institute[2].

Both AMIE and INK prune rules based on both the default support level of 100 and a default max rule length of 3. To mine rules of length 3, the INK neighbourhood's depth parameter was set to 2. This could result in rules containing atoms for both the head and body of length 2. When the support level of those atoms is above the provided thresholds, they can both be combined into a rule which implicitly results in a rule with length of 4. To make a fair comparison towards the mined AMIE rules of length 3, we filtered all those length 4 rules.

For all datasets, the cofidence of the top 10, top 25 and top N, with N the smallest number of rules from either INK or AMIE are compared as shown in Table 4.2. The total number of filtered rules is also listed for both AMIE and INK.

For all benchmark datasets, INK mined a lot more rules compared to the AMIE results. The confidence values are also higher, indicating that the additional rules can be of high value.

4.6.2 INK vs DL-Learner

To compare the INK miner in the context of task-specific mining, we used the Structured Machine Learning benchmark framework (SML-Bench) [27]. This framework enables some specific tasks where structured hypotheses are learned from data with a rich internal structure or knowledge representation, usually in the form of one or more relations. The systems within this framework might differ in the knowledge representation languages they support and the programming languages they are written in. DL-Learner has already been used in many evaluations within the SML-Bench framework. By incorporating INK within this same framework, a fair comparison

[2]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie

between INK and DL-Learner could be made. The SML-Bench framework already evaluated seven structured machine learning models. The code to incorporate INK within the SML-Bench framework is also provided online[3] such that INK can also be used in future evaluations.

Table 4.3: Overview of the datasets that are part of SML-Bench with their number of axioms (#A), classes (#C), object properties (#O), datatype properties (#D) and their description

Dataset	#A	#C	#O	#D	N.o.i.	pos/neg	Prediction of
Carcinogenesis	74,566	142	4	15	298	1.19	Carcinogenic drugs
Hepatitis	73,114	14	5	12	500	0.70	Hepatitis patients
Lymphography	2,187	53	0	0	148	1.21	Lymphography patients
Mammographic	6,808	19	3	2	961	0.86	Breast cancer
Mutagenesis	62,066	86	5	6	42	0.44	Chemical Mutagenicity
NCTRER	92,861	37	9	50	224	1.41	Estrogen receptor
Prem. League	214,566	10	14	202	81	0.97	Goal keepers
Pyrimidine	2,006	1	0	27	40	1.0	Pyrimidines inhibition
Suramin	13,506	46	3	1	17	0.70	Suramin inhibits

In total, nine different datasets are available in the SML-Bench 3.0 version, all containing an OWL knowledge base and a single task based on two sets of files indicating the positive and negative nodes of interest.

The SML-Bench framework also contains other learning systems beyond DL-Learner and INK, but those systems do not take any semantic knowledge into account and were therefore neglected. The default SML-Bench configuration options were used within all our evaluations: 10-fold cross validation was used with a maximum execution time of 15 minutes for each fold. DL-learner version 1.5 was used with the by SML-Bench default parameters for each learning task: For all tests, the CELEO algorithm was used to traverse the search space guided by the Pellet reasoner. INK was initialised with a maximum neighbourhood depth of 3 such that the neighbourhoods of the neighbours from our start nodes were taken into account during the rule generation phase. The numerical levels and relation count extension modules described in Section 4.3.3 were also enabled. The OWL datasets were transformed into the HDT format, which was used as input to generate the INK representation. Four different metrics are reported in Table 4.4:

- **Accuracy score:** The number of correct predictions divided by all predictions. All learning tasks are binary classification problems, but can be unbalanced. We report the average accuracy score between 0 and 1 together with the standard deviation across the 10 folds.

- **F1 score:** $F1 = 2 * \frac{precision*recall}{precision+recall}$ or the harmonic mean of the precision and recall. Again, the average and standard deviation over 10 folds are reported.

[3]https://github.com/IBCNServices/INK

- **Matthews Correlation Coefficient (MCC):** This metric takes into account the true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes: $MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$. MCC returns a value between -1 and $+1$: A coefficient of $+1$ represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation. Again, averages and standard deviations over 10 folds are reported.

- **Duration:** The time needed to find the most descriptive rule, based on the nine out of 10-folds + the time to evaluate this rule on one holdout fold. Averages and standard deviations over 10 folds are reported in seconds.

The results for each learning task are provided in Table 4.4. The task-specific rule mining results showed that INK is highly competitive with DL-Learner.

Table 4.4: SML-Benchmark comparison between INK and DL-Learner for 4 metrics on 8 benchmark datasets. The results show both the average and standard deviation for a 10-fold cross validation.

	Accuracy (std)		F1 (std)		MCC (std)		Duration (std)	
	INK	DL-Learner	INK	DL-Learner	INK	DL-Learner	INK	DL-Learner
carcinogenesis	**0.56 (0.12)**	0.54 (0.02)	0.47 (0.13)	**0.70 (0.01)**	**0.18 (0.28)**	0.00 (0.09)	**191.4 (23.31)**	888.3 (0.46)
hepatitis	**0.78 (0.03)**	0.49 (0.06)	**0.73 (0.07)**	0.61 (0.03)	**0.56 (0.09)**	0.21 (0.07)	**55.4 (1.28)**	879.0 (0.0)
lymphography	0.80 (0.09)	**0.82 (0.1)**	0.82 (0.07)	**0.86 (0.07)**	0.64 (0.15)	**0.67 (0.18)**	**33.3 (1.27)**	873.7 (0.46)
mammographic	**0.83 (0.04)**	0.49 (0.02)	**0.80 (0.05)**	0.64 (0.01)	**0.66 (0.07)**	0.12 (0.1)	**85.8 (2.52)**	874.1 (0.3)
mutagenesis	**0.98 (0.06)**	0.94 (0.13)	**0.97 (0.1)**	0.93 (0.13)	**0.96 (0.12)**	0.9 (0.2)	**31.4 (0.8)**	883.0 (0.0)
nctrer	**0.99 (0.2)**	0.59 (0.04)	**0.99 (0.02)**	0.73 (0.02)	**0.98 (0.04)**	0.01 (0.12)	**201.9 (3.14)**	885.2 (0.87)
premierleague	**0.99 (0.04)**	DNF	**0.99 (0.04)**	DNF	**0.98 (0.07)**	DNF	**167.9 (2.7))**	DNF
pyrimidine	**0.95 (0.1)**	0.82 (0.16)	**0.93 (0.13)**	0.84 (0.14)	**0.92 (0.17)**	0.69 (0.3)	**28.0 (0.89)**	874.0 (0.0)
Suramin	0.65 (0.32)	**0.71 (0.25)**	0.33 (0.42)	**0.71 (0.33)**	0.2 (0.4)	**0.43 (0.49)**	**26.7 (0.9)**	875.0 (0.0)

- For the carcinogenesis dataset, DL-Learner founds a rule describing the node of interest has minimal five hasAtom relationships which are not linked to the `Iodine-95` object node (hasAtom min 5 (not (Iodine-95))). The INK rule triggers when at least three `Hydrogen-3` objects nodes are linked to the hasAtom relationship (count.hasAtom.type.Hydrogen-3 >= 3).

- The DL-Learner rule defined for the hepatitis dataset states the sample must be either Male or having a colisteraseActivityLevel <= 5. The mined INK rule is more precise: the sample must have either a colisteraseActivityLevel <=5 or less or equal to 19 zincSulfateTurbidityTestLevel.2 screenings, more than 8 and less or equal to 12 totalBilirubinLevel.2 screenings with an in total of less than 24 screenings.

- For the lymphography dataset, DL-Learner founds a rule describing the sample should be of the class CIN14_Lac_Margin or (CIN14_Lacunar and (not (SF16_Vesicles))) or (NON19_n0-9 and (not (LNE11_3))). INK only founds the rule stating the sample should be of the class CIN14_Lac_Margin.

- In the mammographic dataset, the DL-Learner mined rule states the sample must have an hasAge >= 23.5. The INK rule miner generates the following rule: hasBiRads >= 5.0 or hasShape of type irregular with hasAge >= 69.
- The rules for the mutagenesis dataset for the DL-Learner and INK technique are quite similar: act >= -0.22 for DL-Learner, act >= 0.3 for INK.
- For the nctrer dataset, the DL-Learner rule states the sample should have minimal two atoms with minimal three bindings. The INK rule looks at the coordinate positions of the first_bound_atom and second_bound_atom in combination with ActivityScore_NCTRER>=32.
- The INK rule in the premierleague dataset is defined as a sample having an action with a goalkeeper_distribution >= 5.
- For the pyrimidine dataset, DL-Learner found a rule where a sample must have a p1_size >= 0.34 or p3_size >= 0.233 and p2_size >= 0.18. The mined INK rule defines a similar rule where the sample must have a p1_polarizable>=0.367, p1_pi_donor<=0.5 and a p2_size>=0.26.
- DL-Learner finds the following rule for a sample within the Suramin dataset: hasAtom with minimal 4 (Hydrogen-8 or Nitrogen-32). The INK rule states the sample should have the hasBond relationship with an inBond charge <= -0.622.

4.7 Discussion

Based on the results provided in Section 4.6, the INK representation and defined INK miners show for both task-specific and task-agnostics rule mining interesting results. In this section, we provide an explanation on why INK is beneficial in the context of rule mining, but also discuss some of the obtained results to show its limitations.

4.7.1 INK within task-agnostic rule mining

Compared to AMIE3, INK mined in all datasets more rules and most of these rules have also a high confidence level. The larger number of rules are mainly due to the fact that INK is also capable of analysing head atoms with more than 2 variables. While in previous works the perception raised that those rules could be neglected as their confidence level should be extremely low, INK showed that some of these rules do occur quite frequently in large datasets. An example of such a rule in DBpedia 3.8: $?a\ isCitizenOf\ ?b \Rightarrow ?a\ wasBornIn\ ?x \wedge ?x\ isLocatedIn\ ?b$ (confidence: 0.27). INK also mines non-closed or open rules here as a variable inside the rule can only occur only once.

Beyond the scope of the current evaluation, INK does have some disadvantages compared to AMIE3. INK consumes a large amount of RAM in order to build the internal representation and to generate the frequent itemsets. AMIE3 is designed to mine rules iteratively and therefore uses less RAM to obtain rules. INK's configuration settings are currently also limited as AMIE can also take into account constants, PCA confidence, removals of perfect rules, etc. INK does however have the capability to mine long rules (rules with a large amount of atoms) without expanding the frequent itemsets. The internal INK representation concatenates relationships within the neighbourhood of the nodes together and represent them already as sub rule parts within the binary INK representation (crf the *Alma Mater.Located In* path or sub rule within our example graph of Figure 4.1). The support of those rule parts can be easily calculated by analysing how many nodes have this sub part in its neighbourhood. A frequent itemset can combine this rule part directly with another rule part to generate a confident rule. By following this approach, INK always has frequent itemsets with maximum 2 items within each set. This simplifies a large amount of calculations as only 2 rules (one for A$\Rightarrow$ B and B $\Rightarrow$ A) have to be generated and evaluated within each itemset.

4.7.2 INK within task-specific rule mining

The INK miner holds both a predictive and time advantage compared to DL-Learning in the context of task-specific rule mining, given a large enough positive and negative set of instances. DL-Learner typically traverses the search space until the predefined amount of time is reached. This is a first advantage of INK over DL-learner, as it does not need to tune this time parameter.

More in depth, within the Carcinogenesis dataset, the accuracy measures for both INK and DL-Learner are similar. DL-Learner, however, optimises its rules to benefit the instances of the majority class. These cases are reflected in a MCC score close to zero, which indicates that the used rules hold the same predictive performance as a random classifier. For both the rules of the Carcinogenesis and NCTRER datasets, DL-Learner obtained such a MCC score of zero. INK obtains a positive MCC score for these datasets.

In contrast, for the Lymphography and Suramin dataset, INK's MCC scores is lower than the MCC scores of DL-Learner. The explanation is two-fold. First, DL-Learner introduces negation within its rules. By explicitly stating within a rule, a concept must not be available, DL-Learner is able to obtain a predictive advantage. Within the INK transformation steps, no such negation operator is defined for now. By doing this, INK postpones the decision to close the world (crf. Closed-World Assumption) to the learning task and accompanying learning model. The INK representation is able to define Not a Number (NaN) for the unknown values inside the neighbourhoud of a node of interest. INK opens up the possibility to adjust future rule miners to deal

with these NaN values or, more generally, open-world cases based on its representation. Second, some of the benchmark datasets have a too small set of nodes of interest for INK to be operational. While DL-Learner is able to correctly define generic rules for the Suramin dataset, INK's strengths lie within larger datasets, with more nodes of interest to mine rules from.

For the Prem. League dataset, DL-Learner was unable to finish the training procedure within the time limit of 15 minutes. In contrast, the most interesting rules generated from the INK miner were available within less than 3 minutes. In general, DL-Learner always searches for better, more descriptive and generic rules when enough time is left. This behaviour is also stated in the obtained results of Table 4.4. Here, nevertheless the used dataset, the duration of the DL-Learner training and evaluation phase is almost always the same. The difference in time across multiple datasets is due to the loading phase of the dataset itself before the actual rule mining starts. The maximum execution time is a parameter within the DL-Learner configuration file. INK does not have such a timing constraint, but is constrained in rule mining's search space by limiting the neighbourhood's depth. As shown in the performed experiments, high quality rules can already be found when limiting the neighbourhood depth to three.

The INK rules for the Hepatitis, Mammographic, Mutagenesis and Pyrimidine datasets extend in some sort the obtained DL-Learner rules. In most of these cases INK finds additional information within the neighbourhood and adds one or two extra rule atoms or sub rules to achieve a better predictive performance. INK is also able to better define the numerical properties within a rule. DL-Learner tries to minimise the full integer or floating point range when mining such rules, while INK uses the available data within the neighbourhood to already limit the ranges upfront in the rule mining process.

4.8 Conclusion

In this work, we addressed the current problems of both task-specific and task-agnostic semantic rule mining and the need for one technique which can perform both. The main contribution to fulfil this need is the development of an internal representation benefiting both techniques. INK is such a representation, where the neighbourhood of nodes in a KG are represented as a binary matrix. Combining this INK representation with a Bayesian Rule miner resulted in outperforming the current state of the art methods to perform structured machine learning, both in prediction performance and in time. The same representation can be used to mine frequent itemsets of nodes of interest and build general rules filtered by confidence and a given support level. Compared with the filtered results of AMIE, more confident and new rules were mined by INK for several benchmark datasets.

The INK representation resembles a binary vector matrix, and can be used in

several other situations going beyond the general purpose of rule mining. Future work will adapt INK to mine rules with both constants or a wider range scalar data in combination with a temporal aspect. This would enable INK to mine temporal rules, originated from a sensor or more broader, Internet of Things (IoT) streaming data domain.

Acknowledgement:

Bram Steenwinckel (1SA0219N) is funded by a strategic base research Grant of the Fund for Scientific Research Flanders (FWO).

References

[1] L. Ehrlinger and W. Wöß. *Towards a Definition of Knowledge Graphs.* SEMANTiCS (Posters, Demos, SuCCESS), 48, 2016.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data.* In The semantic web, pages 722–735. Springer, 2007.

[3] B. Steenwinckel, D. De Paepe, S. Vanden Hautte, P. Heyvaert, M. Bentefrit, P. Moens, A. Dimou, B. Van Den Bossche, F. De Turck, S. Van Hoecke, and F. Ongenae. *FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning.* Future Generation Computer Systems, 116:30–48, 2021. Available from: https://www.sciencedirect.com/science/article/pii/S0167739X20329927, doi:https://doi.org/10.1016/j.future.2020.10.015.

[4] G. Vandewiele, F. De Backere, K. Lannoye, M. V. Berghe, O. Janssens, S. Van Hoecke, V. Keereman, K. Paemeleire, F. Ongenae, and F. De Turck. *A decision support system to follow up and diagnose primary headache patients using semantically enriched data.* BMC medical informatics and decision making, 18(1):1–15, 2018.

[5] P. Bonte, R. Tommasini, E. Della Valle, F. De Turck, and F. Ongenae. *Streaming MASSIF: cascading reasoning for efficient processing of IoT data streams.* Sensors, 18(11):3832, 2018.

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

[7] T. Ebisu and R. Ichise. *Graph pattern entity ranking model for knowledge graph completion.* arXiv preprint arXiv:1904.02856, 2019.

[8] S. Ortona, V. V. Meduri, and P. Papotti. *Robust discovery of positive and negative rules in knowledge bases.* In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pages 1168–1179. IEEE, 2018.

[9] N. Jain and V. Srivastava. *Data mining techniques: a survey paper.* IJRET: International Journal of Research in Engineering and Technology, 2(11):2319–1163, 2013.

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

[11] J. Minker. *On indefinite databases and the closed world assumption*. In International Conference on Automated Deduction, pages 292–308. Springer, 1982.

[12] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. *Knowledge vault: A web-scale approach to probabilistic knowledge fusion*. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 601–610, 2014.

[13] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. *Fast rule mining in ontological knowledge bases with AMIE+*. The VLDB Journal, 24(6):707–730, 2015.

[14] J. Lajus, L. Galárraga, and F. Suchanek. *Fast and Exact Rule Mining with AMIE 3*. In European Semantic Web Conference, pages 36–52. Springer, 2020.

[15] J. Lehmann. *DL-Learner: learning concepts in description logics*. Journal of Machine Learning Research, 10(Nov):2639–2642, 2009.

[16] J. Lehmann. *Learning OWL class expressions*, volume 22. IOS Press, 2010.

[17] L. Bühmann, J. Lehmann, P. Westphal, and S. Bin. *Dl-learner structured machine learning on semantic web data*. In Companion Proceedings of the The Web Conference 2018, pages 467–471, 2018.

[18] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, and G. Weikum. *Rule learning from knowledge graphs guided by embedding models*. In International Semantic Web Conference, pages 72–90. Springer, 2018.

[19] P. G. Omran, K. Wang, and Z. Wang. *An embedding-based approach to rule learning in knowledge graphs*. IEEE Transactions on Knowledge and Data Engineering, 33(4):1348–1359, 2019.

[20] T. Wang, C. Rudin, F. Velez-Doshi, Y. Liu, E. Klampfl, and P. MacNeille. *Bayesian rule sets for interpretable classification*. In 2016 IEEE 16th International Conference on Data Mining (ICDM), pages 1269–1274. IEEE, 2016.

[21] D. Krech. *RDFLib: A Python library for working with RDF*, 2006.

[22] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. *Binary RDF Representation for Publication and Exchange (HDT)*. Web Semantics: Science, Services and Agents on the World Wide Web, 19:22–41, 2013. Available from: http://www.websemanticsjournal.org/index.php/ps/article/view/328.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. *Scikit-learn: Machine learning in Python*. the Journal of machine Learning research, 12:2825–2830, 2011.

[24] S. Raschka. *MLxtend: providing machine learning and data science utilities and extensions to Python's scientific computing stack.* Journal of open source software, 3(24):638, 2018.

[25] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum. *YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames*. In International semantic web conference, pages 177–185. Springer, 2016.

[26] D. Vrandečić and M. Krötzsch. *Wikidata: a free collaborative knowledgebase.* Communications of the ACM, 57(10):78–85, 2014.

[27] P. Westphal, L. Bühmann, S. Bin, H. Jabeen, and J. Lehmann. *SML-Bench–A benchmarking framework for structured machine learning.* Semantic Web, 10(2):231–245, 2019.

5

INK: Knowledge Graph Embeddings for Node Classification

In chapter 4, a KG representation was created which could be used in deductive and inductive semantic rule mining. The interpretable KG representation could also be directly provided to an ML classifier. The representation can be seen as a feature set or an embedding for the ML model and predictions are made for these features. This chapter shows how such a KG embedding can be used for a classification task. A predictive maintenance use case is used to show the benefits of this new embedding technique. Combined with a white-box classifier such as e.g. a decision tree classifier, this new KG representation can provide interpretable results. This chapter investigates research question 4: "Can knowledge be incorporated in an ML model without losing the inherently explainable characteristics of the provided knowledge?" and validates hypothesis 4: "A KG embedding technique keeping the inner explainable characteristics of the KG outperforms other KG embedding techniques with more than 2% in predictive performance on well-stated KG classification benchmark datasets.". Additionally, this interpretable KG representation can also be used to annotate structured documents and transform and augment the information in these documents to knowledge. Appendix B provides more information regarding this embedding-based semantic annotator. The INK representation of Chapter 4 is described again in Section 5.3. Readers who are already familiar with this INK representation can skip this Section 5.3.

⋆ ⋆ ⋆

B Steenwinckel, G. Vandewiele, M. Weyns, T. Aggozino, F. De Turck, F. Ongenae

Published in Springer Journal of Data Mining and Knowledge Discovery, Volume 36, Issue 2, January 2022.

Abstract
Deep learning techniques are increasingly being applied to solve various machine learning tasks that use Knowledge Graphs as input data. However, these techniques typically learn a latent representation for the entities of interest internally, which is then used to make decisions. This latent representation is often not comprehensible to humans, which is why deep learning techniques are often considered to be black boxes. In this chapter, we present INK: Instance Neighbouring by using Knowledge, a novel technique to learn binary feature-based representations, which are comprehensible to humans, for nodes of interest in a knowledge graph. We demonstrate the predictive power of the node representations obtained through INK by feeding them to classical machine learning techniques and comparing their predictive performances for the node classification task to the current state of the art: Graph Convolutional Networks (R-GCN) and RDF2Vec. We perform this comparison both on benchmark datasets and using a real-world use case.

5.1 Introduction

Knowledge graphs (KGs) are becoming more and more recognised as valuable data structures as they are a structured representation of facts, consisting of entities, relationships, and semantic descriptions [1]. They can be represented in the Resource Description Framework (RDF), a standard for data interchange, developed and agreed upon by W3C [2]. RDF exists to provide a standardized way to interlink and access all that we know in a formal, machine-processable language. The way RDF connects data pieces together is via triples. As the name suggests, a triple is a set of three entities representing a statement about the graph in the form of subject-predicate-object expressions. Multiple of these statements or triples form a KG. The underlying semantics of a KG are usually provided in a so-called ontology. Ontologies can be seen as the data schema of the graph and define the shared understanding of the data and its meanings [3]. In most cases, the semantic information residing in the ontology is also expressed in RDF and incorporated into the KG. This process is more commonly known as the materialization of the ontology [4]. This chapter assumes both the materialized ontology and RDF data when the term KG is used.

KGs are now frequently being used in multiple Machine Learning (ML) tasks [5]. The main motivation why KGs are so popular today is because an increasing amount of data, containing semantic relationships, is available [6]. One of these tasks, called

node classification, classifies nodes into one out of a set of discrete classes using its neighbourhood information. Node classification belongs to the broader category of link prediction tasks. In link prediction, the goal is to predict the existence of a link between two entities in a graph. Node classification restricts this prediction to the occurrence of a predefined class type. An example of such a node classification task is shown in Fig. 5.1. Here, the goal is to determine the type of two masterpieces of Michelangelo based on their linked information. `The Creation of Adam` belongs to the class of Fresco's, while the statue of `David` belongs to the class of Sculptures. More in general, the goal in this example is to predict the link between the two nodes of interest (`The Creation of Adam`, and `David`) and two predefined classes (Fresco and Sculptures). Node classification became very popular since existing classification tasks can be augmented with the additional data residing in popular KGs such as DBpedia [7] or Wikidata [8].

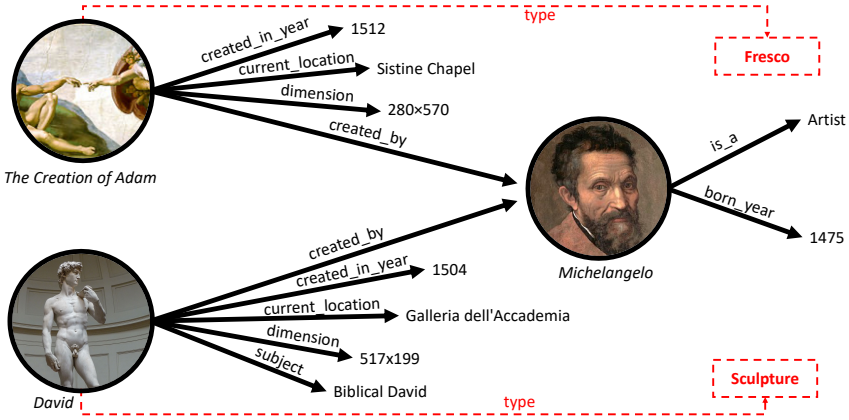


Figure 5.1: Node classification example. Two masterpieces of Michelangelo need to be classified in either Fresco or Sculpture based on their linked information. The "type" relations are, for obvious reasons, not taken into account during this evaluation.

The simplified RDF triple representation of the example in Figure 5.1 is defined as follows:

<The Creation of Adam>	<created_in_year>	”1512”^^xsd:int .
<The Creation of Adam>	<current_location>	<Sistine Chapel> .
<The Creation of Adam>	<dimension>	”280x570” .
<The Creation of Adam>	<created_by>	<Michelangelo> .
<David>	<subject>	<Biblical David> .
<David>	<dimension>	”517x199” .
<David>	<current_location>	<Galleria dell”Academia> .
<David>	<created_in_year>	”1504”^^xsd:int .
<David>	<created_by>	<Michelangelo> .
<Michelangelo>	<is_a>	<Artist> .
<Michelangelo>	<born_year>	”1475”^^xsd:int .

Various techniques exist to perform node classification on KGs, but none of them can use the inherent semantic aspects of the KG while maintaining the ability to actually augment an existing dataset, hold their predictive performance and keep the interpretable aspects. Therefore, in this chapter, we present a binary node embedding technique that can classify unseen entities or nodes from the KG, given some already labelled entities. It builds these embeddings by using INK, a technique used to transform the neighbourhood of several nodes within a KG to a binary matrix. By providing this binary matrix to standard ML classifiers, semantic information can be easily incorporated into a ML pipeline while still delivering competitive and interpretable results. This chapter evaluate INK based on three requirements: (predictive) performance, interpretability and applicability towards different tasks.

We have organised the remainder of this chapter in the following way: Section 5.2 describes the currently state-of-the-art techniques to perform a node classification task on KGs. This section also lists the current drawbacks of these techniques. The binary INK representation and how it can be constructed is fully explained in Section 5.3. In Section 5.4, we describe how the binary INK representation can be used to perform a node classification task. Section 5.5 delivers the implementation details of the node classification techniques and how they can be used. In Section 5.6, we explain our node classification benchmark setup and show the achieved predictive performances, as well the time and memory consumption for the best results on seven proposed benchmark tasks. As node classification tasks are relevant in an industry setting, Section 5.7 describes INK's performance for a supervised anomaly detection use case in a sensor environment. A discussion on both the acquired results, the interpretable aspects and the applicability of our approach to more general tasks are given in Section 5.8. Section 5.9 concludes this work.

5.2 Related Work

Three broad categories of node classification techniques on KGs exist. All are described in this section and further compared concerning their interpretable aspects.

5.2.1 Graph-based Deep Learning

A first approach adapts the Deep Learning (DL) architectures to work with graph-based data directly [9]. The current state-of-the-art technique is Relational Graph Convolution Networks (R-GCN) [10]. R-GCN were used to solve two fundamental tasks on KGs: Entity or node classification and link prediction. Both problems use a common Graph Convolutional Network (GCN), extended with multi-edge encoding to compute the entities' embedding, but with different downstream processing. For node classification, GCN layers are stacked in a ML model together with a softmax activation for each node on the last layer's output. The goal is to minimise the cross-entropy loss on all labelled nodes (while ignoring unlabeled nodes) [9]. The key difference between R-GCN and GCN is that in R-GCN, edges can represent different relations. In GCN, the weights are shared by all the edges in a layer. In contrast, in R-GCN, different edge types use different weights and only edges of the same relation type are associated with the same projection weights [11].

5.2.2 Representation Learning

The second approach is based on representation learning [12]. Here, the goal is to create a mapping from the KG to low-dimensional numerical vectors. This vector is then further on used by a traditional ML classifier. To generate these vectors, one can use tensor factorisation techniques or apply unsupervised deep-learning techniques on the walks extracted from the KG [13, 14]. The current state-of-the-art representation learning technique for node classifications with KGs is RDF2Vec [15]. RDF2Vec creates a numeric vector for each node in an RDF graph. These numeric vectors or embeddings are generated from a set of walks extracted from the nodes' neighbourhood. The embeddings, which are based on the principles of word2vec [16], represent the node's neighbourhood as latent feature vectors and therefore lose some information. Also, both the Continues Bag Of Words (CBOW) and Skip-gram architectures can be used. As the RDF2Vec features are generated per node, they can be easily given to any ML classifier to perform a node classification task. Besides node classification, these embeddings can also be used for other tasks as they do not directly involve the learning process. This idea comes from the more general definition of graph kernels, which promise a more flexible approach by providing a powerful framework for decoupling the representation from the learning task [17]. Relevant research already investigated the differences and usefulness of these RDF graph kernel approaches [18, 19]. However, RDF2Vec took into account these recent advantages and is quite often still preferred for various mining tasks due to the scalability advantages of the generated walks in large RDF graphs [20–22]. Recent research also investigated why some of the provided graph kernel methods provide some misleading results in the context of RDF graphs [23].

5.2.3 Direct encoding

Bilinear models like RESCAL [24], DistMult [25], ComplEx [26], and SimplE [27] make use of tensor factorisation to capture multiplicative interactions between pairs of entity vectors. They were proposed for the general field of link prediction, but can also be applied for generating propositional features from graphs. The most common techniques in this field are TransE and its successors [28–30]. TransE builds entity and relation embeddings by defining the relation as a translation from the subject to the object entity. This assumes that some relationships between entities could be computed by their vector difference in the embedding space. TransE, however, cannot deal with reflexive, one-to-many, many-to-one, and many-to-many relations. TransR first introduced the idea of relation-specific entity spaces next to a generic relation-agnostic entity space, to capture the different ways entities might behave according to their semantic roles [30]. TransR differs from TransH in that the latter was constructed to overcome TransE's inability to deal with relations more generic than 1-1 [29]. TransR, however, treats relationships as different embedding spaces. Related to TransR is TransG, which deals with the problem of hidden relation polysemy, whereby relationships display different latent semantic characteristics depending on the varying nature of the participants [31]. GTrans generalises from most of these previous attempts to model the complexity of varying semantic roles by acknowledging that any relationship is solely defined by what features its participants jointly have when they participate in the relationship [32]. Significantly, even though GTrans makes use of different embedding spaces it is able to construct a single representation for each entity and relation through a linear combination.

These methods only capture the linear relationships between entities. Recent research has raised interest in applying deep neural networks to triple-based prediction problems. ConvKB models the provided entries at the same dimension of the given triple using a Convolutional Neural Network (CNN), where presumably each dimension captures some relation-specific attribute of entities [33]. CapsE explores, in contrast, the advancements of CapsNet [34] on triple-based data to model the entries at the same dimension in the entity and relation embeddings [35]. CapsE forms unique k-dimensional embeddings of the triple, which is fed to the convolution layer where multiple filters of the same shape are repeatedly operated over every row of the embedding matrix to produce k-dimensional feature maps. Entries at the same dimension from all feature maps are then encapsulated into a so-called capsule. Each capsule encodes many aspects of the embedding triple.

5.2.4 Interpretability

While some of the above mentioned techniques effectively classify nodes in a KG, they reduce the initial interpretable aspects of the KG. R-GCN, and more in general all DL-based graph embedding techniques, are black-box techniques. The provided

insights from these models are, therefore, somewhat limited. The original method was also not designed to work with multi-relational and multi-modal graphs. Adaptations were made to assign the importance of the node's neighbours while doing aggregation, instead of giving fixed normalisation constant to every node's neighbor [36]. While adding such attention layers to the whole R-GCN setup can help to learn the influence of an edge during training [37], it still limits the possible interpretable aspects of the nodes themselves. Techniques, however, exist to provide some sort of explainability to GCN [38]. They are, to our knowledge, not yet adapted to work with KGs.

In contrast, the traditional, feature-based methods and encoding based techniques do deliver interpretable results. Their performance on simple node classification tasks is, however, still limited and do not outperform techniques such as R-GCN and RDF2Vec [15]. The use of content features makes the model interpretable, but projecting the nodes and relations in a new vector space of latent features makes the final result not directly interpretable [39]. The main disadvantage of creating low-dimensional feature vectors is that no one-to-one mapping between the used content features and the latent variables exists. Techniques exist to inject semantic features into the learning process to retain the original informativeness of the items available in the dataset [39]. As direct interpretable results can be important in certain critical domains, recent advances started to use class discriminative node substructures together with interpretable Decision Trees [40]. In this way, the extracted substructures can be seen as derived features which keep their interpretable aspects. Combined with an interpretable ML classifier, useful insights can still be guaranteed.

5.2.5 Contribution

The contribution of this chapter is to further extend the idea of using discriminative node substructures proposed by Vandewiele et al. [40]. Instead of generating substructures and depending on the interpretable aspects of a decision tree classifier, the proposed method INK will generate an interpretable representation of the neighbourhoods of those nodes which need to be classified. While direct encoding techniques can provide similar information regarding the task at hand, their predictive performance on entity or node classification tasks is not competitive to the current state-of-the-art techniques [15, 41]. Encoding information further away from the node of interest seems to be crucial in this perspective [42]. While R-GCN is currently the best technique in terms of predictive performance for node classification tasks on KGs, the RDF2Vec embeddings provide a general way to perform multiple tasks based on KGs. This means that the RDF2Vec embeddings are more generic and not tailored to the task at hand. Both approaches do not deliver interpretable or explainable results directly. The goal, here, is to build a node embedding technique that holds the similar benefits of RDF2Vec while being as accurate as R-GCN for the node classification task and keeping the interpretable aspects of the KG as delivered by direct encoding

methods.

5.3 Instance Neighbouring by using Knowledge (INK)

In this section, we present "Instance Neighboring using Knowledge" (INK), a novel technique to learn binary feature-based representations for nodes in a KG.

5.3.1 Neighborhood dictionary

In a node classification task, we will typically define those nodes which we want to classify. In our example KG, we choose two nodes of interest: `The Creation of Adam` and the `David` node. INK will first query the neighbourhood of a given depth for all these nodes of interest. If we define the depth parameter K to be two, the neighbourhood of `The Creation of Adam` will exist of the created_on, location, dimension and created_by relations, together with the neighbourhood of Michelangelo. To store these neighbourhoods efficiently, a dictionary representation is used. For a given node of interest, this dictionary is built in an iterative fashion. All the predicates in a neighbourhood of depth one are inserted first into our dictionary, together with their corresponding objects as values. These dictionary values are lists, as a single predicate can occur multiple times with different objects in the neighbourhood of a node. For our given example node `The Creation of Adam`, we represent the neighbourhood at depth one by:

$$\{\text{created\_in\_year} \rightarrow [1512],$$
$$\text{current\_location} \rightarrow [\text{Sistine Chapel}],$$
$$\text{dimension} \rightarrow [280\text{x}570],$$
$$\text{created\_by} \rightarrow [\text{Michelangelo}]\}$$

To add the neighbourhoods of depths > 1, INK concatenates the predicates together. By concatenating the relations, INK provides a path from the node of interest to another node within our graph without providing detailed information about all intermediate nodes on that path. However, this information is still available in the (key, value) pairs added to our dictionary at the lower neighbourhood's depths. In our example node, the previous dictionary will be extended with the following (key, value) pairs at depth 2:

$$\text{created\_by.is\_a} \rightarrow [\text{Artist}]$$
$$\text{created\_by.born\_year} \rightarrow [1475]$$

Here, we see a link from the node of interest to the Artist node over the created_by relation. The created_by object value is not represented in this (key, value) pair, but was specified at the previous depth 1.

To indicate a certain predicate is present for our node of interest, we can also add another transformation to explicitly state certain relationships are available:

created_in_year → [True],
current_location → [True],
dimension → [True],
created_by → [True]
created_by.is_a → [True]
created_by.born_year → [True]

Again, lists were used as values for our dictionary as a single node can have the same predicate multiple times but with different object values.

Completely similar, the dictionary representation of `David` until depth 2 is:

{created_in_year → [1504],
current_location → [Galleria dell'Accademia],
dimension → [517x199],
subject → [Biblical David],
created_by → [Michelangelo]
created_by.is_a → [Artist]
created_by.born_year → [1475]
created_in_year → [True],
current_location → [True],
dimension → [True],
subject → [True],
created_by → [True]
created_by.is_a → [True]
created_by.born_year → [True]}

More in general, combined for all nodes of interest $\mathcal{N}$, the initial data structure of INK uses the following format:

$$list(tuple(n, neighbourhood(n, k))), \forall \text{ n in } \mathcal{N}$$

where the $neighbourhood(n, k)$ is the function which outputs the dictionary representation for our node n till a defined depth k.

5.3.2 Binary format

As the $list(tuple(n, neighbourhood(n, k)))$ representation is 3 dimensional (one axis for the nodes of interest, one for the dictionary relation keys and one for dictio-

nary object list values), an additional transformation is required to provide a binary representation of this data. All of the objects inside our neighbourhood dictionary are combined using a delimiter § to their corresponding (key, value). For the example nodes `The Creation of Adam` and `David`, we transformed both depth one and depth two neighbourhoods into:

created_in_year§1512 (5.1)

created_in_year§1504 (5.2)

current_location§Sistine Chapel (5.3)

current_location§Galleria dell'Accademia (5.4)

dimension§280x570 (5.5)

dimension§517x199 (5.6)

subject§Biblical David (5.7)

created_by§Michelangelo (5.8)

created_by.is_a§Artist (5.9)

created_by.born_year§1475 (5.10)

created_in_year§True (5.11)

current_location§True (5.12)

dimension§True (5.13)

subject§True (5.14)

created_by§True (5.15)

created_by.is_a§True (5.16)

created_by.born_year§True (5.17)

These transformations can now be easily represented as a binary matrix. The rows are defined by the nodes of interest, such that each cell indicates whether or not the subject of interest contains the relation(s)§object value. The binary INK representation for the example above is visualised in Table 5.1. Here, (14) is a specific relation for `David` and could be of interest to classify sculptures.

Table 5.1: INK's binary representation of the example graph of Fig. 5.1.

	(5.1)	(5.2)	(5.3)	(5.4)	(5.5)	(5.6)	(5.7)	(5.8)	(5.9)	(5.10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
The Creation of Adam	1	0	1	0	1	0	0	1	1	1	1	1	1	0	1	1	1
David	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

5.3.3 Extension modules

While the binary representation of INK reflects the whole KG, it can also add advanced, calculated features based on, e.g. datatype properties. In this subsection, we

describe two optional extension modules which are available in INK. They provide only additional information about the KG and are not required to operate to create the binary representation discussed above.

5.3.3.1 Numerical inequality

To deal with numerical data, a preprocessing module will check if the values corresponding to a specific relation are all floats or integers for all the corresponding objects and nodes of interest. When such a relation is found, we build a set of all possible inequalities using all the found objects for that relation. In our example KG, we have the created_in_year relation that contains for all our nodes of interest, an integer value. When this extension module is enabled, all these integer values will be stored inside a set. INK compares for each node of interest the value of this created_in_year relation with all possible values in our set and adds a new entry to our neighbourhood dictionary as follow:

$$\begin{aligned} &\text{created\_in\_year} < l \rightarrow [\text{True or False}] \ \ \& \\ &l >= \text{created\_in\_year} \rightarrow [\text{True or False}], \forall \text{ l in inequality set} \end{aligned}$$

Concrete, 4 new entries for `The Creation of Adam` will be added, describing 1) if his created_in_year is smaller than the created_in_year of the `David` sculpture, 2) if his created_in_year year is greater than or equal the created_in_year of `The Creation of Adam` (itself), 3) if the given created_in_year is smaller than the one of the `David` sculpture and 4) if the year is greater than or equal the year of `The Creation of Adam` (itself).

5.3.3.2 Relation count

Another preprocessing module is available in INK to deal with relations having more than one object value. It can be beneficial to indicate how many similar relations are available for a given node of interest. Therefore, a counting module adds new entries to the neighbourhoods dictionary when two or more similar relations are available. More specifically, if in our example graph `The Creation of Adam` would have a second created_by relation, this module would add the following entry:

$$\text{count.created\_by} \rightarrow [2]$$

This entry can be directly transformed to count.created _by&2 as described above. The previous inequality module can also use these new entries as these counting values are stored as integer values.

Table 5.2: INK's binary representation of a numerical columns.

	created_in_year§1512	created_in_year§1504
The Creation of Adam	1	0
David	0	1

5.3.3.3 Non-binary format

Instead of transforming all columns into a binary format, keeping certain columns in a float or integer representation can also be beneficial. If we take for example the created_in_year relationship, both the values for `The Creation of Adam` and the `David` sculpture are represented as integers.

Therefore, the non-binary INK representation will obey the created_in_year§1512 and created_in_year§1504 columns as shown in Table 5.2 but will create the created_in_year_real_value column with values [1512, 1504]. Such a non-binary column is shown in Table 5.3

Table 5.3: INK's non-binary representation of a numerical columns.

	created_in_year_real_value
The Creation of Adam	1512
David	1504

This extension module will actively search for such non-binary columns, by evaluating if all the dictionary representation values can be represented as floats. When this is not the case for a certain column, the classical approach of adding multiple columns representing each value is being used. It depends on the used ML model, whether these integers or floats can be beneficial for the task at hand.

5.4 Node classification

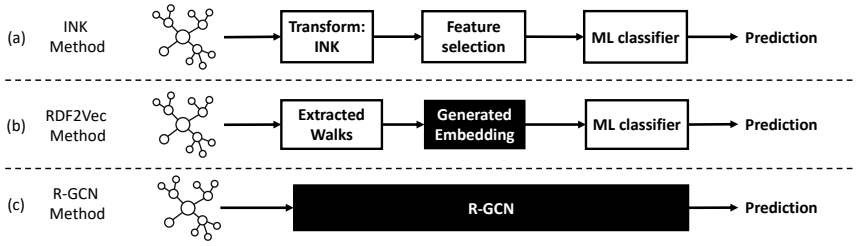


Figure 5.2: Three different techniques to perform KG node classification. (a) describes the INK pipeline discussed in this chapter. (b) The RDF2Vec generated embeddings and (c) the rather black-box graph convolutional neural networks.

When given a multi-relational directed KG, the goal of a node classification task is to construct a model or hypothesis based on a set of selected entities with corresponding, discrete labels. This model or hypothesis minimises a loss function such that it generalises well to unseen entities. Based on the INK representation in the previous section, a more traditional ML pipeline can be used to define the model or hypothesis, as visualised in Fig. 5.2(a). INK is here used as a preprocessing step to transform the given KG into binary feature vectors. As INK represents the KG in a binary format, some features can be less useful within a node classification task. Binary features which, e.g. hold a positive value for only one node of interest will not generalise well. Therefore, a feature selection technique can be added. This selection strategy should work unsupervised, which means that columns can be filtered independently of the task or labels within our binary matrix representation. Removing duplicate columns and a variance-threshold selection strategy are two such techniques that can be used. After this selection phase, the feature vectors are operated by a traditional ML classifier during the training phase to optimise the model and determine which of these binary features separate the classes the best. These features are also used during the testing phase to classify new unseen entities.

As described in the related work in Section 5.2, other techniques exist to determine a possible model to perform the node classification task. The RDF2Vec embedding and R-GCN Deep-Learning techniques are also visualised in Fig. 5.2(b) and (c) respectively.

5.5 Implementation

To make a fair comparison between all the previously mentioned approaches, we describe the used implementations for each of them in this section.

5.5.1 INK implementation

The INK representation described in the previous section is fully implemented in Python. To extract the neighbourhoods for a given set of nodes of interest, a component was implemented to query these relations iteratively. Two options are currently available inside this component: either a KG is given as input or a SPARQL endpoint is provided. SPARQL is the RDF query language and can retrieve and manipulate the data stored in a KG. A SPARQL endpoint can be seen as a remote database or triple store from which semantic data can be queried. When a KG file is given, RDFLib [43] will be used to load the graph in the operating system's internal memory.

```
SELECT ?p ?o ?dt WHERE
{
 <ind> ?p ?o.
 BIND (datatype(?o) AS ?dt)
}
```

Listing 5.1: SPARQL query

```
input: KG, NOIs (nodes of interest), depth k

# create list(tuple(n, neighbourhood(n,k)))
neighbourhoods = []
for n in NOI:
  neighbourhoods.append((n,extract_neighbours(n,k,KG))

# create binary representation
key_values = []
for tup in neighbourhoods:
  # create new dictionary:
  n_dct = {}
  # select the neighbourhood dictionary
  for key in tup[1]:
    for value in tup[1][key]:
      if len(tup[1][key]) > 0:
        # relation feature
        n_dct[key] = True
        # value feature
        n_dct[key + '§' + value] = True
  key_values.append(n_dct)

vec = DictVectorizer(sparse=True, dtype=bool)
binary_matrix = vec.fit_transform(key_values)

return binary_matrix, NOIs, vec.feature_names_
```

Listing 5.2: Implementation binary INK representation

For both options, Listing 5.1 shows the SPARQL query to extract the neighbourhood nodes and relations. The variable subject <ind> starts with the nodes of interest but differs in each iteration given the graph. The data type of the object within this query is used to determine if queried objects can be used as subjects in the next iteration (when the neighbourhood depth is not reached yet). The output of these queries are provided in the JSON format. The predicates and objects in each iteration are stored as described previously. Python's internal multiprocessing library can speed up the neighbourhoods' extraction, as this operation can be performed over multiple processors given the number of nodes of interest. The INK implementation also caches the SPARQL queries to reduce the overhead of dealing with multiple, similar requests. The actual transformation step to create the binary matrix representation is visualised in pseudo-code in Listing 5.2. In this code, the neighbourhood is first created until a particular depth using the SPARQL query above. Next, for the neighbourhood of each node of interest, we generate the relation(s)§object representation. To transform this representation into a binary matrix, we used the Scikit-learn DictVectorizer [44] with the sparse option set to True and specifying the data type to be Boolean. This is necessary when we want to deal with large KGs and many nodes of interest.

The numerical inequality and counts modules add additional (key, value) pairs to the original neighbourhood dictionary and are transformed as discussed above. Only two small modifications are required to receive the non-binary matrix, as described in

Section 5.3.3.3. The DictVectorizer option dtype is changed to float and for each key in our neighbourhood dictionary, we check whether all values can be transformed into floats. If this is the case, we create the a non-binary column by setting `n_dct[key+"_real_vale"] = value`.

5.5.2 pyRDF2Vec

To evaluate the RDF2Vec results, we used the pyRDF2Vec Python package (version 0.1.0) and also tweaked it to work with both a SPARQL endpoint, cache optimisation and the Python's multiprocessing library [45]. The implementation of the used RDF2Vec walker is provided in Listing 5.3. In this code, the `_proc` function is exactly the same as the original RandomWalker defined by Ristoiske et al. We used this walker code inside the `_extract` function of pyRDF2vec in a multiprocessing fashion in order to embed multiple instances or nodes of interest-based on the number of available cores (4 in this example).

```
class MultiProcessingRandomWalker(RandomWalker):
  def _proc(self, t):
    kg, instance = t
    walks = self.extract_random_walks(kg, instance)
    canonical_walks = set()
    for walk in walks:
      canonical_walk = []
      for i, hop in enumerate(walk):  # type: ignore
        if i == 0 or i % 2 == 1:
           canonical_walk.append(str(hop))
        else:
          digest = md5(str(hop).encode()).digest()[:8]
          canonical_walk.append(str(digest))
      canonical_walks.add(tuple(canonical_walk))
     return {instance:tuple(canonical_walks)}

  def _extract(self, kg, instances):
    canonical_walks = set()
    seq = [(kg, r) for _,r in enumerate(instances)]
    with Pool(4) as pool:
      res = list(pool.imap_unordered(self._proc, seq))
    res = {k:v for el in res for k,v in el.items()}
    for r in instances:
       canonical_walks.update(res[r])
    return canonical_walks
```

Listing 5.3: Implementation RDF2Vec walker

5.5.3 Pytorch R-GCN

The R-GCN implementation is directly taken from the Pytorch-based implementation of R-GCN for semi-supervised node classification on directed relational graphs repository[1]. This code is adapted from Kipf's Keras-based implementation[2], but made compatible for Python 3 and optimised towards speed and efficiency.

[1]https://github.com/berlincho/RGCN-pytorch
[2]https://github.com/tkipf/relational-gcn

5.6 Benchmark evaluation

To compare the predictive performance of INK with the RDF2Vec and R-GCN described in Section 5.2, we will analyse the accuracy, performance and memory consumption on several benchmark datasets. As both INK and RDF2Vec require an additional ML classifier to make predictions, a comparison between several well-known classification models is also made.

5.6.1 Datasets

Seven datasets, from varying domains, were already defined in a public repository set up by Ristoski et al. [46]:

- BGS: describes three geological measurements in Great Britain. The task of this dataset is to predict the lithogenesis property of named rock units. In total 146 samples are available in this binary classification problem (with 53 samples having the lithogenesis property and 93 not). This dataset has 100k triples with 105 relation types and around 280 different predicate-object pairs per node.

- AIFB: describes the AIFB research institute in terms of its staff, research groups, and publications. The task here is to predict for the people in this dataset to which affiliation they belong. The dataset contains 178 members of four research groups. However, the smallest group contains only four people and is removed from the dataset. This results in a 3-class classification problem (with 73 people in the first, 28 in the second and 70 in the third group). The AIFB dataset has around 29k triples, 47 different relation types and around 250 different predicate-object pairs per node.

- MUTAG: contains information about 340 potentially carcinogenic complex molecules, which is given by the `isMutagenic` property (129 samples which are mutagenic and 211 not). This task resembles a binary classification problem as the molecules have to be classified as ”mutagenic” or ”not mutagenic”. The MUTAG dataset consists of around 75k triples, with 24 unique relation types. Around 250 different predicate-objects pairs are available per node.

- AM: contains semantic information about artefacts in the Amsterdam Museum [47]. Each artefact in the dataset is linked to other artefacts and details about its production, material, and content. In this task, 1000 samples are available for an 11-class classification problem where the goal is to predict to which category an artefact belongs (347, 127, 116, 86, 81, 56, 55, 50, 42, 25, 15 samples for each class respectively). With more than 5,7 million triples and 100 different relations, each node contains around 300 predicate-object pairs.

- DBpedia Cities: The Cities dataset contains a list of 212 cities and their quality of living, as captured by Mercer [48]. Based on this quality of living value, three classes (low, medium, high) are provided (39 samples indicated as low, 106 as medium and 67 as high). Additional to these values, the DBpedia URI is provided for each city.
- DBpedia Albums: A dataset retrieved from the Metacritic website, which contains the average rating of all-time reviews for 1600 albums [49]. In this dataset, the albums are classified as either containing a positive or a negative rating. 800 albums have a positive label and 800 albums a negative one. For each album, the DBpedia URI is provided.
- DBpedia Movies: A dataset retrieved from the Metacritic website, which contains an average rating of all-time reviews for a list of movies [50]. 2000 movies are available in this dataset and are classified as either containing a positive rating or a negative rating. 1000 movies have a positive label and 1000 movies a negative one. Similarly, the DBpedia URI is provided.

The three DBpedia datasets will use the same DBpedia KG to query useful information. The 2015-10 DBpedia version is used in these evaluations. This KG contains more than 650 million triples and almost 8000 different relations. To limit the amount of extracted paths for both INK and RDF2Vec, we do not extract the neighbourhoods of the nodes containing the following incoming relationships:

- http://dbpedia.org/ontology/abstract
- http://dbpedia.org/ontology/wikiPageExternalLink
- http://www.w3.org/2002/07/owl#sameAs
- http://dbpedia.org/ontology/wikiPageWikiLink
- http://purl.org/dc/terms/subject
- http://www.w3.org/2000/01/rdf-schema#seeAlso
- http://dbpedia.org/property/wikiPageUsesTemplate
- http://www.w3.org/2000/01/rdf-schema#comment
- http://www.w3.org/2000/01/rdf-schema#label
- http://dbpedia.org/ontology/wikiPageWikiLinkText

We use the same train-test partitioning for each of the benchmark datasets as provided by the original repository. In general, 80% of the total dataset was trained. All the results below and the evaluations are based on the other 20%. These dataset splits were initially stratified such that the train and test set class distributions are similar.

5.6.2 Evaluation pipeline

As both INK and RDF2Vec generate an embedded version of the original graph, several parameters were set to make a comparison. For INK, we evaluate the standard representation without extension modules, a representation with the numerical inequality module, a representation with the counts module and the non-binary variant of the INK representation as discussed in Section 5.3. For RDF2Vec, we both generated 500-dimensional embeddings with the Skip-gram and CBOW architecture for 500 generated paths. Both the embeddings for INK and RDF2Vec were generated, taking into account multiple node neighbourhood depths. We generated embeddings for the depths from 1 until 4 during this evaluation.

When both the INK and RDF2Vec embeddings are available, seven different classifiers were used to make predictions according to the task at hand. Five of these classifiers, and the defined hyperparameters, were also used in the original RDF2Vec paper [15]. All these classifiers are implemented in Scikit-learn [44]:

- Naive Bayes: A Gaussian Naive Bayes classifier that is suitable for classification with discrete features.
- Nearest Neighbour: Classifier implementing the k-nearest neighbour's vote algorithm. The number of neighbours parameter was set fixed to 3.
- Decision Tree: Tree-based classifier defined by the CART (Classification and Regression Trees) algorithm.
- Support Vector Machine: Support Vector Classification using a regularization parameter C. This parameter was optimized in cross-validation using the following values: $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$.
- Logistic Regression: The logit or MaxEnt classifier. The regularization parameter is optimized in cross-validation using the following values: $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$.
- Extra-trees: Meta estimator that fits several randomised Decision Trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The default number of estimators was kept to 100.
- Random forest: Meta estimator that fits several Decision Tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The default number of estimators was kept to 100.

The R-GCN classifier parameters are based on those provided by the original research paper [9] but were slightly adapted to ensure the models could run on our

evaluation setup. The number of hidden nodes were set to 10 for all the datasets, while the number of bases was set to 40, 0, 30 and 40 for the BGS, AIFB, MUTAG, AM respectively. As R-GCN require to load the KG into memory, no analyses were performed for the DBpedia tasks. The l2 penalty is set to 0 for the AIFB dataset, while for all other datasets this parameter is set to $5*10^{-4}$. The number of layers changed in a range of 1 till 4 to incorporate more information in the nodes' neighbourhood. This is similar to the changing depth parameter for the RDF2Vec and INK embeddings.

All the KGs were preloaded in a Stardog triple store [51]. For the INK and RDF2Vec evaluations, the embeddings were generated by locally querying a SPARQL endpoint. R-GCN takes the RDF representation directly as input. Further evaluations will always neglect the times to either load the data into the Stardog triple store or the R-GCN module's loading time.

5.6.3 Comparison

Three different comparisons are made. First, a global comparison based on the predicted accuracy on the test set is made over all the dataset, different classifiers and different embedded characteristics. Second, we select the best results of INK, RDF2Vec and R-GCN for each dataset and we compare the time needed to create the internal representation (embedding for INK and RDF2Vec), the training and test time over multiple depths or layers. Finally, the best results are also used to compare the memory consumption of the internal representation over multiple depths or layers. All individual results, as well as additional results regarding weighted F1 scores, weighted precision and weighted recall scores are available online[3].

5.6.4 Results

The whole pipeline is tested using a 6 core Intel Core i5-9400 CPU @ 2.90GHz, with 32 GB of RAM. The SPARQL endpoint was removed for the R-GCN tests. The detailed results for the first evaluation for each dataset are visualized in the first addendum section provided at the end of this chapter. For each classifier, the best mean accuracy over 5 runs was analysed, together with the standard deviations In Table 5.4, the best results are summarised between each of the three techniques.

- The best R-GCN setup for the BGS dataset only required one layer, RDF2Vec results were obtained using CBOW at a depth of 4 with the SVM classifier. The overall best score of this BGS dataset was obtained by the INK non-binary variant, using a Decision Tree classifier at depth 3.
- The R-GCN classifier obtained the overall best results for the AIFB dataset at depth 2. The best RDF2Vec results were obtained with a logistic regression

[3]https://github.com/IBCNServices/INK/tree/master/ink_bennchmark

Table 5.4: Summary of the accuracy scores of R-GCN and RDF2Vec and our proposed approach INK on the seven benchmark datasets. The AM results for R-GCN reported by the original research paper were 89.3 but using a setup with more RAM to load the initial graph. Similarly, the best reported results for the RDF2Vec for this AM dataset were 88.3 (hence the * in this table) [9]. The / in this table indicate the missing results for R-GCN, as this technique did not report any results for the DBpedia:Cities, DBpedia:Albums and DBpedia:Movies evaluations.

	R-GCN	RDF2Vec	INK
BGS	82.1 (0.05)	86.9 (0.03)	**93.1 (0.00)**
AIFB	**96.1 (0.02)**	91.7 (0.00)	94.4 (0.00)
MUTAG	75.9 (0.02)	81.2 (0.02)	**82.4 (0.00)**
AM	89.3 *	88.3 *	**90.4 (0.00)**
DBpedia:Cities	/	77.9 (0.05)	**85.3 (0.00)**
DBpedia:Albums	/	74.4 (0.01)	**74.4 (0.00)**
DBpedia:Movies	/	80.6 (0.01)	**80.8 (0.00)**

classifier and embeddings generated using CBOW at depth 4. The best INK results required the usage of non-binary variant usage at depth 3 combined with an ExtraTree classifier.

- In the MUTAG dataset, the R-GCN classifier obtained the best results using one layer. RDF2Vec reported the best results when using a Random Forest classifier in combination with CBOW generated embeddings at depth 4. INK reported the overall best score, using an SVM classifier and embeddings taking into account the counts at depth 2.

- The AM dataset is already a large KG, and while using our setup, no results for the AM dataset could be reported for the R-GCN classifier. In the discussion section, we will rely on the results reported in the original research paper for this dataset [9]. Results were reported for the RDF2Vec classifier, resulting in the best score when using an SVM classifier with Skip-gram embeddings. However, creating embeddings at depth 4 was not possible using our setup for the RDF2Vec embeddings. Therefore, we will also rely on the RDF2Vec results obtained in [9]. The overall best score was obtained by INK at depth 1, using a logistic regression classifier in combination with the non-binary variant. At depth 4, it was impossible to generate results for the counts and numerical variants of INK, mainly because of the very large amount of generated features.

- INK with the numerical option set to true, in combination with a Nearest Neighbour classifier at depth 3 provided the best results for the DBpedia cities dataset. The best RDF2Vec results were obtained at depth 4 using the Skip-gram architecture in combination with the SVM classifier. No results for R-GCN could be generated.

- For the DBpedia albums tasks, INK and RDF2Vec hold similar results. The best INK results were obtained at depth 3, using the Logistic regression classifier and the non-binary INK variant. The best RDF2Vec results were also obtained at depth 3, using the CBOW architecture and SVM classifier.

- INK and RDF2Vec hold also similar results for the DBpedia movies task. The best INK results were obtained at depth 3, using the Logistic regression classifier and the count option set to true. The best RDF2Vec results were also obtained at depth 3, using the Skip-gram architecture and SVM classifier.

The second evaluation investigates each of these best results' performance parameters by analysing the time to create the embedding or internal representation, the time to train, and the time to make new predictions. These performance parameters are visualised for each benchmark dataset and for each of the best technique-classifier combinations and increasing depths in the second addendum section provided at the end of this chapter. A summary of these results is being provided in Table 5.5.

Table 5.5: Summary of the average total times (sum of dataset creation time, train time and test time in seconds) of R-GCN, RDF2Vec and our proposed approach on the seven benchmark datasets. The results are calculated based on the best results reported in our benchmark evaluation. The / in this table indicate the missing results for R-GCN, as this technique did not report any output for the AM, DBpedia:Cities, DBpedia:Albums and DBpedia:Movies analyses.

	R-GCN	RDF2Vec	INK
BGS	104 (2.10)	5.36 (0.04)	8.63 (0.13)
AIFB	4.42 (0.01)	848 (10.5)	18.9 (0.75)
MUTAG	5.58 (0.34)	25.8 (0.37)	41.2 (0.51)
AM	/	27.6 (0.09)	40.5 (0.43)
DBpedia:Cities	/	194 (10.3)	948 (10.9)
DBpedia:Albums	/	6,861 (487.8)	4,032 (22.1)
DBpedia:Movies	/	8,759 (363.6)	6,845 (11.8)

In the last benchmark evaluation, the used amount of memory to store either the INK representation, the embedded vectors and the R-GCN internal representation were compared for the best results obtained in Table 5.4 in function of the depth parameter. All individual results are provided in the third addendum section provided at the end of this chapter. Table 5.6 provides an overview of the obtained results.

5.7 Use Case: Supervised anomaly detection

The academic and industrial attention has focused on improving existing approaches for event processing solutions. Data from sensors and the metadata describing the manufacturing setup are combined in one big graph or KG in these processing units.

Table 5.6: Summary of the average dataset memory consumption (GB) of the R-GCN, RDF2Vec and our proposed approach on the seven benchmark datasets. The results are calculated based on the best results reported in our benchmark evaluation. The / in this table indicate the missing results for R-GCN, as this technique did not report any output for the AM, DBpedia:Cities, DBpedia:Albums and DBpedia:Movies analyses. As INK can offload some parts of data to disk, the memory consumption can be indicated higher than the available RAM. We marked these results with a * to indicate the total amount of RAM was used during this process and the values indicate the total size of the obtained datastructure.

	R-GCN	RDF2Vec	INK
BGS	0.27	0.01	0.35
AIFB	0.003	0.01	0.49
MUTAG	0.005	0.04	1.93
AM	/	0.13	1.12
DBpedia:Cities	/	0.06	18.4
DBpedia:Albums	/	0.21	202.1 *
DBpedia:Movies	/	0.33	444.6 *

Tasks such as anomaly detection and root cause analysis are not limited to the sensor data but must also deal with the contextual information or provided metadata. In this perspective, the ACM International Conference on Distributed and Event-based Systems (DEBS) started the 2017 Grand Challenge [52] to investigate the benefits of a KG in the detection of anomalies for sensor data streams generated by manufacturing equipment. The DEBS grand challenge organisers propose a specific anomaly detection method where the data produced by each sensor is clustered and the state transitions between the observed clusters are modelled as a Markov chain. Anomalies are detected as sequences of transitions that happen with a probability lower than a given threshold.

While this challenge also investigated the needs for how such a detection module should work in the context of a continuous streaming environment, several datasets dumps were provided. These datasets were intended to be used as a gold standard with which other unsupervised anomaly detection solutions could be compared. It was not, however, intended as the source for a supervised training procedure. The original task specified the metrics accuracy, latency, and throughput to evaluate any potentially viable solution. Accuracy is specified as the number of anomalies found by the detection module that were also found by the gold standard, while latency and throughput pertain to the benchmark streaming aspect. We will use one such dataset dump in a very different way than was the authors' original intention.

5.7.1 Dataset

The datasets we are referring to contain all the observations that are usually streamed. Furthermore, they also contain a record of all the gold standard anomalies, which we

can use to turn the problem into a supervised one. The dataset we are using in our evaluation contains 585 000 observations, with only 368 labelled anomalies[4].

```
debs:ObservationGroup_0 a I4.0:MoldingMachineObservationGroup .
debs:ObservationGroup_0 ssn:observationResultTime debs:Timestamp_0 .
debs:ObservationGroup_0 I4.0:machine WMetadata:Machine_59 .
debs:ObservationGroup_0 I4.0:observedCycle debs:Cycle_0 .
debs:ObservationGroup_0 I4.0#contains debs:Observation_0 .
debs:Cycle_0 a I4.0:Cycle .
debs:Cycle_0 IoTCore#valueLiteral "13" .
debs:Timestamp_0 a IoTCore:Timestamp .
debs:Timestamp_0 IoTCore:valueLiteral "2017-01-01T01:00:01+01:00" .
debs:Observation_0 a I4.0:MoldingMachineObservation .
debs:Observation_0 ssn:observationResult debs:Output_0 .
debs:Observation_0 ssn:observedProperty WMetadata:59_4 .
debs:Output_0 a ssn:SensorOutput .
debs:Output_0 ssn:hasValue debs:Value_0 .
debs:Value_0 a I4.0:NumberValue .
debs:Value_0 IoTCore#valueLiteral> 9433.11" .
```

Listing 5.4: Example of one DEBS Obervation

Every observation is associated with an observation group, which is itself associated with a cycle and a timestamp. The observation is then further associated with a particular output (a sensor output) which then has a value. An example of such an observation is visualised in Listing 5.4. The anomalies are described by a Markovian state indication number and the timestamps of the observation group they occurred. These two values are sufficient to uniquely identify an observation that might be considered an anomaly: Every timestamp belongs only to a single observation group, while every state is associated with only a single observation within the respective group. An example of such an anomaly description is provided in Listing 5.5.

```
debs:Anomaly_0 a Analytics:Anomaly .
debs:Anomaly_0 I4.0#machine WMetadata:Machine_59 .
debs:Anomaly_0 Analytics:abnormalDimension WMetadata:59_31 .
debs:Anomaly_0 Analytics:hasTimeStamp debs:Timestamp_24 .
```

Listing 5.5: Example of one DEBS Anomaly

Additional to these observations and anomaly descriptions, metadata by both an additional metadata description file and in the form of multiple ontologies was provided. As can be seen in the observation and anomaly description in Listings 5.4 and 5.5, the IOTCore, I4.0 and SSN ontologies are used for respectively basic IoT concepts such as timestamps and concepts on the observation dataset as well as a description about sensors and their observable properties. The additional metadata file, visualised in Listing 5.6 gives more details about the used machine and the different cycles or states in which they operate.

```
WMetadata:Machine_59 a WMetadata:MoldingMachine .
WMetadata:Machine_59 IoTCore:hasModel WMetadata:MachineModel_59 .
WMetadata:MachineModel_59 a I4.0:MachineType .
WMetadata:MachineModel_59 ssn:hasProperty WMetadata:59_5 .
WMetadata:59_5 a WMetadata:StatefulProperty .
```

[4]https://hobbitdata.informatik.uni-leipzig.de/StreamMachineLearning_1.0/

```
WMetadata:59_5 WMetadata:hasNumberOfClusters "3" .
```

Listing 5.6: Example of the DEBS metadata description

5.7.2 Evaluation setup

As the anomalies are indicated as abnormal observations, the whole problem of detecting the anomalies can be seen as a node classification task where the anomalies belong to one class and the ”normal” observations to the other one. Like our benchmark tests, we can acquire the neighbourhood of the observation and use it as features in a supervised classification model.

We compared the INK and RDF2Vec embedding technique for this classification task using two classifiers: the ExtraTree and RandomForrest classifier, each initialised with 100 estimators. For both RDF2Vec and INK, we generated embeddings for all observations at depth 1. No additional modules were used by our INK approach to generate additional features. The number of INK features was limited using a VarianceThreshold feature selector, removing all the low-variance features. The RDF2Vec results created 200 dimensional embedded vectors. An evaluation of R-GCN is omitted for this use case. The original case processed the observations in a streaming fashion. This requires the underlying KG to be updated every time a new event occurs. Both INK and RDF2Vec can handle these observations individually as the neighbourhood from that event can be built on the fly. R-GCN would require reloading the internal representation with every new observation. This unwanted behaviour limits the applicability of the R-GCN model for such a temporal graph use case.

A labelled dataset was generated with all anomalies belonging to the negative class. The Stratified 10-fold cross-validation approach was used to represent the results. Inside the cross-validation loop, the training set was balanced by under-sampling the majority class (normal observations). Instead of reporting the standard accuracy, both the balanced accuracy and micro F1 score are reported. As the KG contains in total 585 000 instances, it was infeasible to acquire results with the R-GCN implementation with the evaluation setup as described in Section 5.6.4.

5.7.3 Results

In Table 5.7 the results are listed for both the INK and RDF2Vec approach. From these results, it can be seen that INK is able to separate the anomalies from the normal observations. RDF2Vec focuses on the anomaly class, and a large number of normal observations were classified as anomalies. This also resulted in a low binary F1 score and a balanced accuracy close to 50%.

Table 5.7: Results supervised training on DEBS 2017 benchmark dataset

	Balanced Accuracy		Binary F1 score	
	Extra	Random	Extra	Random
INK	0.945 (0.01)	0.946 (0.01)	0.012 (0.00)	0.012 (0.00)
RDF2Vec	0.503 (0.01)	0.507 (0.01)	0.001 (0.00)	0.001 (0.00)

5.8 Discussion

As the INK representation is beneficial regarding the node classification task on KG, three main topics will be discussed. First, we describe the benchmark datasets' obtained results and how the increased performance should be interpreted. Next, we situate INK's interpretability and how even a standard classifier can help to achieve predictable results. At last, we discuss the results of the real-life use case.

5.8.1 INK compared to the state-of-the-art

In the overall comparison of Table 5.4, we see that the predictive accuracy of INK is high and outperforms the other techniques in 6 out of the 7 benchmark datasets. For the AIFB dataset, the R-GCN technique acquires a higher accuracy. The 3-class classification task performed for this AIFB dataset is imbalanced. All three classes or research groups also interact with each other, which results in more relationships being shared inside the node's neighbourhoods. R-GCN generalize better in this case, but the difference in predictive performance only indicates one additional correctly classified sample in the test set compared to INK. Better predictive performances can probably be acquired for both the RDF2Vec and INK embeddings by using additional hyper-parameter optimization of the used classifiers or using additional feature selection methods to reduce irrelevant features. The internal parameters of word2vec can also be further optimised within the RDF2Vec approach. These additional optimisations were not taken into account in this evaluation because the main goal was to align the whole evaluation setup with the ones performed in both the RDF2Vec and R-GCN previous works [9, 15]. This enables the authors to compare INK with more extended R-GCN results obtained by Schlichtkrull et al., where more resources were available, and results were obtained for the BGS, MUTAG, AIFB and AM benchmark datasets. Even in these cases, INK still has the best results for 6 out of the 7 benchmark datasets. The main reason why such high accuracy scores can be reached using a standard classifier is that the INK representation incorporates all the available semantic links directly and represents them in a way that is easily accessible by any ML model. When a large number of entities need to be classified as in the AM, the DBpedia albums and movies datasets, the closer the results of RDF2Vec and INK. This is expected behaviour, as better embeddings can be built by RDF2Vec when more information or walks are being provided. Our evaluation did not report any results

for the DBpedia datasets using R-GCN. The main reason for these lack of results is the internal specification of R-GCN to first load the whole graph into memory when creating the tensor representation. This is challenging for large graphs, but impossible for graphs with the size of DBpedia. As defined by the original R-GCN research team, it can be costly to train R-GCN on large graphs [53].

When comparing the best results for each dataset in the function of their time needed to 1) create the representation, 2) train the classifier and 3) the time to test, no clear winner can be indicated. The results depend highly on the used datasets, the used embedding technique, the used learning and used parameters as shown in the overview of Table 5.5. RDF2Vec needs, in general, some time to create the embedded vector. After the vector is created, the training and test times of RDF2Vec are relatively constant as the embedded vector is limited in size (during these evaluations, the vector dimension was fixed to 500). INK generates many features and these feature increases the model evaluations used during training and testing. The used additional modules even increase the number of features rapidly. Classifiers that do not scale in the number of features require a lot more time to train or make predictions. A large feature set can make INK become less attractive. Feature selection techniques, preferable unsupervised ones, are needed to select only those extracted features in the INK representation which are useful for the task at hand. The R-GCN time measurements depend highly on the characteristics of the datasets. The BGS datasets have many different relation types, which resulted in more time to create the internal representation. The internal R-GCN representation also obeyed the evaluation on larger graphs such as the DBpedia benchmarks As the code used to evaluate the R-GCN classifier was mainly designed to used specific hardware, such as a GPU, the time evaluations are limited to these simple tests. The authors prefer to fix the evaluation set up to ensure fair comparisons could be made.

As already indicated in Table 5.4, some results could not be obtained for the R-GCN and RDF2Vec approaches. All these missing results are due to memory issues. The memory used to create the embeddings is only a small indicator of why some results are missing. Again, the specific characteristics of the dataset are important. The memory consumption of INK is dependent on the number of nodes of interest, the depth parameter and the average number of object-predicate relationships for each node within the KG. Estimates for the required memory consumption can, therefore, be made upfront. As the best INK results for AM are already obtained at depth 1, the amount of memory consumed is lower than that of the best INK results for the MUTAG dataset. The INK approach uses more memory to create the embedding compared to the other techniques. INK stores these binary matrices in sparse representation. This sparse representation is needed as otherwise even more RAM should be needed. For very large representations, INK was also able to store parts of the dictionary representation on disk during the DBpedia evaluations. Of course, this storage mechanism results in large processing times. The embeddings created by RDF2Vec

and the memory required for the internal representation of the R-GCN classifier are constant, as the dimensions are fixed during these evaluations. However, during the creation of the RDF2Vec embeddings, more RAM is required to store the individual paths obtained to generate the embeddings. Similarly, the R-GCN approach could not build its internal representation for this AM and DBpedia datasets, as it would require enormous amounts of RAM to load the original graphs into memory. Another benefit of INK is that the binary representation is decoupled from the dictionary representation. The dictionary representation can be built once for certain depths, which enable researchers afterwards to experiment with the different INK extension modules without the need to query the KG again. As querying the neighbourhoods is time-consuming within INK but also for RDF2Vec, decoupling these two internal structures hold some advantages.

5.8.2 INK and Interpretability

The biggest advantage of INK results mainly in the fact that the generated embeddings remain interpretable. For every unique feature, the direct link with the original graph is still visible and it can be relevant to represent this information. To illustrate INK's interpretable aspects, the best INK result of the BGS benchmark dataset is represented using the Decision Tree classifier. Based on its implementation, the underlying used tree can be easily represented. Fig. 5.3 shows such a tree visualisation for the BGS dataset. In this tree, we see that the feature:

broader.hasLithologyComponent.hasLithology⟆RockName:DMTN

found at depth three for each node of interest is used to split the dataset into two parts. Either this value is positive (one in the INK matrix) or negative (zero in the INK matrix). This first test will check for each new node if this is the case or not. When this test succeeds, the path goes to the left, and when it does not, it goes to the right. For this simple example, the node of interest that does not have this textscRockName:DMTN value at depth 3 will be of the `GLACI` class. The Decision Tree classifier will inherently traverse through this representation, starting from the root tree node and visiting multiple intermediate tests before reaching the leaf classification nodes.

This example shows how a Decision Tree classifier can already deliver valuable insights into how the predicted class for each node can be explained. These visualisations are irrelevant when using the embeddings of the RDF2Vec approach as the values inside the embedded feature vector are meaningless. Other techniques to define the feature importance of a particular model can also help to deliver insights into the predictive behaviour [54]. A second possible way to show which features affect our model is to make use of SHapley Additive exPlanations or SHAP values [55]. SHAP is a game-theoretic approach to explain the output of any ML model. It connects optimal credit allocation with local explanations using the classical Shapley values from

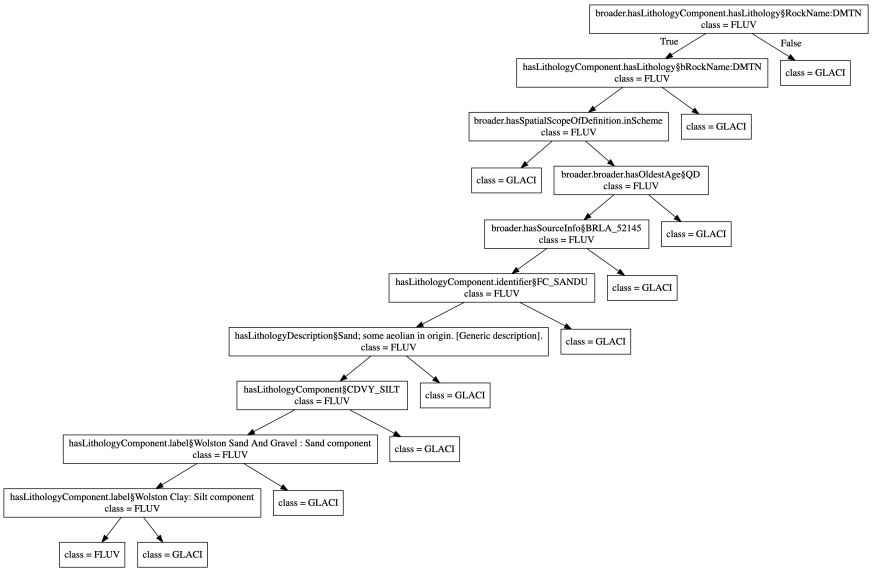


Figure 5.3: Example of how a Decision Tree classifier can be used in combination with the INK features.

game theory and their related extensions [55]. However, further research is ongoing to determine if the INK features provide enough insights to explain the predictions to a possible end-user using these SHAP values.

5.8.3 INK for supervised anomaly detection or other tasks

The supervised anomaly detection use case described in Section 5.7 shows the real benefit of INK compared to other embeddings techniques. As R-GCN could not load the whole graph into its internal representation and the RDF2Vec approach lacks to generate useful embedded vectors, INK delivers interesting results. The RDF2Vec results might improve by using additional hyper-parameter tuning of both the vector sizes and the word2vec internal parameters, but this will require regenerating the embeddings. INK delivers useful results without using any extension module or tuning additional model parameters. Further research can investigate whether INK is suitable for more unsupervised anomaly detection tasks on KGs.

Both the evaluation and supervised anomaly use case are within the field of node classification tasks. Both R-GCN and RDF2Vec were however designed and evaluated for the more general task of link prediction and even analysed for other similarity and classification tasks. This research can only claim the benefits of INK concerning the node clustering task. However, several aspects of INK such as unsupervised representation can make it possible to also evaluate the usefulness of INK in a broader context. The obtained representations for the three DBpedia benchmark datasets can

be used in a clustering task to verify if three broad clusters (cities, albums and movies) are available by using the INK embeddings. Also, both these three DBpedia tasks are originally defined as regression tasks, predicting the quality of life score, album and movie ratings respectively. INK could also provide useful analyses for such type of tasks. Further research is however needed to investigate these, more general KG tasks and state the benefits of INK in the context of link prediction.

5.9 Conclusion and Future work

In this research, we used a binary matrix to represent the nodes in a KG and performed node classification tasks on this representation. This approach's advantages are that both the predictive performance is improved and the preliminary interpretable aspect can be delivered compared to the current state-of-the-art techniques. Besides providing good predictive performance, the INK implementation can compete with the currently used node classification techniques on both time and memory consumption. However, one must still take into account the characteristics of the dataset. In a real-life use case, INK shows impressive results mainly because it considers the literal information inside the embedding, instead of reducing them into a lower vector space.

Future work will investigate the benefits of using this INK representation on numerous tasks, such as regression or node similarity. It can also be useful to perform INK on general graph tasks, e.g., to determine the similarity between two large (sub)graphs based on the nodes' neighbourhoods.

Reproducibility and code availability

The INK package and the code to perform this evaluation pipeline is provided on Github[5]. We also provide all the experimental results in the format of CSV files in this repository.

Acknowledgements

Bram Steenwinckel (1SA0219N) is funded by a strategic base research Grant of the Fund for Scientific Research Flanders (FWO). This research is part of the imec.ICON project PROTEGO (HBC.2019.2812), co-funded by imec, VLAIO, Televic, Amaron, Z-Plus and ML2Grow.

[5]https://github.com/IBCNServices/INK

Addendum: Predictive performance results

This section provides detailed performance results for all the defined datasets and classifiers in Section 5.6. For each classifier, the best mean accuracy over 5 runs is visualised in italic and the overall best result is highlighted in bold. The standard deviations for each of these 5 runs are represented between brackets. Results that did not finish using our setup are denoted as '/'.

Table 5.8: The accuracy scores and standard deviations for the described classifiers and embedding approached for the BGS dataset.

		BGS						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	89.7 (0.00)	68.3 (0.04)	73.8 (0.02)	72.4 (0.00)	77.9 (0.02)	75.2 (0.02)	65.5 (0.03)
	INK Counts	89.7 (0.00)	72.4 (0.03)	79.3 (0.00)	79.3 (0.00)	82.8 (0.00)	71.7 (0.02)	71.7 (0.02)
	INK Numerical	89.7 (0.00)	66.9 (0.02)	72.4 (0.03)	72.4 (0.00)	74.5 (0.05)	75.2 (0.03)	66.9 (0.05)
	INK Non binary	89.7 (0.00)	68.3 (0.03)	73.8 (0.02)	69.0 (0.00)	72.4 (0.00)	74.5 (0.03)	66.9 (0.03)
	RDF2Vec Skip-gram	54.5 (0.07)	64.8 (0.03)	56.6 (0.12)	65.5 (0.00)	65.5 (0.00)	62.8 (0.06)	65.5 (0.06)
	RDF2Vec CBOW	51.7 (0.05)	64.1 (0.06)	55.9 (0.10)	60.7 (0.07)	65.5 (0.00)	65.5 (0.03)	66.2 (0.03)
	R-GCN				*82.1 (0.05)*			
2	INK	89.7 (0.00)	79.3 (0.00)	89.7 (0.00)	86.2 (0.00)	*92.4 (0.02)*	84.8 (0.03)	80.0 (0.03)
	INK Counts	89.7 (0.00)	79.3 (0.00)	90.3 (0.02)	83.4 (0.02)	86.9 (0.02)	86.2 (0.00)	81.4 (0.03)
	INK Numerical	89.7 (0.00)	82.8 (0.00)	90.3 (0.02)	86.2 (0.00)	90.3 (0.02)	83.4 (0.03)	78.6 (0.02)
	INK Non binary	89.7 (0.00)	79.3 (0.00)	90.3 (0.02)	*86.2 (0.00)*	90.3 (0.02)	82.8 (0.02)	80.0 (0.03)
	RDF2Vec Skip-gram	70.3 (0.03)	85.5 (0.03)	71.7 (0.08)	82.1 (0.03)	82.1 (0.03)	*86.9 (0.03)*	*85.5 (0.03)*
	RDF2Vec CBOW	44.8 (0.05)	80.0 (0.07)	62.8 (0.04)	79.3 (0.02)	77.9 (0.02)	82.8 (0.04)	79.3 (0.02)
	R-GCN				76.6 (0.07)			
3	INK	89.7 (0.00)	84.8 (0.02)	91.7 (0.02)	80.0 (0.02)	81.4 (0.03)	82.8 (0.00)	82.8 (0.00)
	INK Counts	89.7 (0.00)	84.8 (0.02)	93.1 (0.00)	80.0 (0.02)	82.8 (0.03)	82.8 (0.00)	82.8 (0.00)
	INK Numerical	89.7 (0.00)	85.5 (0.02)	93.1 (0.00)	79.3 (0.00)	80.7 (0.02)	82.8 (0.00)	82.8 (0.00)
	INK Non binary	89.7 (0.00)	84.1 (0.02)	***93.1 (0.00)***	79.3 (0.00)	81.4 (0.03)	82.8 (0.00)	82.8 (0.00)
	RDF2Vec Skip-gram	84.1 (0.04)	77.9 (0.05)	72.4 (0.09)	83.4 (0.02)	83.4 (0.04)	84.1 (0.02)	84.8 (0.03)
	RDF2Vec CBOW	84.1 (0.03)	77.9 (0.04)	76.6 (0.06)	85.5 (0.02)	82.8 (0.02)	86.2 (0.05)	84.8 (0.02)
	R-GCN				77.9 (0.06)			
4	INK	93.1 (0.00)	86.2 (0.00)	87.6 (0.03)	80.0 (0.02)	79.3 (0.00)	82.8 (0.00)	82.8 (0.00)
	INK Counts	86.2 (0.00)	84.8 (0.02)	87.6 (0.02)	81.4 (0.02)	80.7 (0.02)	82.8 (0.00)	83.4 (0.02)
	INK Numerical	89.7 (0.00)	84.8 (0.02)	85.5 (0.02)	80.0 (0.02)	79.3 (0.00)	82.8 (0.00)	83.4 (0.02)
	INK Non binary	*93.1 (0.00)*	*86.2 (0.00)*	86.9 (0.02)	80.0 (0.02)	79.3 (0.00)	82.8 (0.00)	82.8 (0.00)
	RDF2Vec Skip-gram	82.8 (0.00)	77.2 (0.02)	77.9 (0.04)	83.4 (0.02)	82.8 (0.00)	84.1 (0.02)	84.8 (0.02)
	RDF2Vec CBOW	84.1 (0.02)	82.1 (0.04)	72.4 (0.09)	*86.2 (0.00)*	81.4 (0.03)	84.8 (0.02)	84.8 (0.02)
	R-GCN				78.6 (0.07)			

Table 5.9: The accuracy scores and standard deviations for the described classifiers and embedding approached for the AIFB dataset.

		AIFB						
		NB	NN	DT	SVM	LR	Extra	Random
	INK	86.1 (0.00)	52.8 (0.00)	67.2 (0.04)	75.0 (0.00)	79.4 (0.02)	77.2 (0.01)	74.4 (0.04)
	INK Counts	83.3 (0.00)	55.0 (0.02)	63.9 (0.03)	75.0 (0.00)	77.8 (0.02)	79.4 (0.04)	71.1 (0.03)
	INK Numerical	86.1 (0.00)	51.1 (0.07)	67.2 (0.01)	75.0 (0.00)	80.0 (0.01)	77.8 (0.02)	76.7 (0.02)
1	INK Non binary	86.1 (0.00)	52.2 (0.01)	66.7 (0.03)	75.0 (0.00)	78.9 (0.02)	77.8 (0.03)	71.7 (0.02)
	RDF2Vec Skip-gram	56.7 (0.05)	51.7 (0.09)	48.9 (0.07)	57.2 (0.06)	60.6 (0.01)	56.7 (0.06)	57.2 (0.03)
	RDF2Vec CBOW	55.6 (0.05)	48.3 (0.04)	41.1 (0.04)	59.4 (0.03)	59.4 (0.02)	56.1 (0.08)	57.8 (0.04)
	R-GCN				95.6 (0.02)			
	INK	86.1 (0.00)	70.0 (0.02)	88.3 (0.04)	83.3 (0.00)	88.9 (0.00)	89.4 (0.01)	84.4 (0.02)
	INK Counts	86.1 (0.00)	56.7 (0.05)	87.8 (0.05)	79.4 (0.02)	81.7 (0.04)	80.6 (0.02)	79.4 (0.04)
	INK Numerical	86.1 (0.00)	67.2 (0.01)	*91.1 (0.04)*	81.1 (0.01)	87.2 (0.02)	89.4 (0.01)	83.3 (0.03)
2	INK Non binary	86.1 (0.00)	67.2 (0.02)	90.6 (0.02)	83.3 (0.00)	86.7 (0.01)	87.8 (0.02)	83.9 (0.01)
	RDF2Vec Skip-gram	70.6 (0.02)	78.9 (0.02)	71.1 (0.04)	82.2 (0.02)	82.8 (0.04)	81.7 (0.02)	80.6 (0.02)
	RDF2Vec CBOW	67.8 (0.02)	61.7 (0.02)	62.2 (0.08)	77.2 (0.04)	80.6 (0.04)	72.8 (0.04)	75.0 (0.04)
	R-GCN				***96.1 (0.02)***			
	INK	91.7 (0.00)	80.0 (0.01)	88.3 (0.04)	88.9 (0.00)	88.9 (0.00)	93.9 (0.01)	*92.2 (0.02)*
	INK Counts	91.7 (0.00)	78.9 (0.02)	85.0 (0.03)	80.6 (0.03)	86.1 (0.00)	91.1 (0.01)	90.0 (0.02)
	INK Numerical	91.7 (0.00)	83.3 (0.00)	87.2 (0.05)	*88.9 (0.00)*	88.9 (0.00)	93.3 (0.02)	91.1 (0.02)
3	INK Non binary	91.7 (0.00)	83.3 (0.00)	86.7 (0.02)	87.8 (0.02)	86.7 (0.01)	*94.4 (0.02)*	90.6 (0.02)
	RDF2Vec Skip-gram	88.9 (0.00)	*87.8 (0.02)*	76.1 (0.09)	88.3 (0.02)	90.6 (0.02)	91.1 (0.01)	91.1 (0.01)
	RDF2Vec CBOW	86.1 (0.00)	82.2 (0.02)	78.9 (0.05)	88.3 (0.01)	90.6 (0.02)	87.2 (0.02)	90.6 (0.02)
	R-GCN				95.0 (0.02)			
	INK	91.7 (0.00)	75.6 (0.04)	86.7 (0.02)	86.1 (0.00)	88.9 (0.00)	86.1 (0.00)	84.4 (0.02)
	INK Counts	91.7 (0.00)	76.7 (0.02)	82.8 (0.05)	80.6 (0.00)	86.7 (0.01)	86.7 (0.01)	83.9 (0.01)
4	INK Numerical	*94.4 (0.00)*	80.0 (0.01)	88.3 (0.02)	86.1 (0.00)	87.8 (0.02)	86.1 (0.00)	83.3 (0.00)
	INK Non binary	91.7 (0.00)	75.6 (0.04)	87.8 (0.04)	86.1 (0.00)	89.4 (0.01)	86.1 (0.00)	83.9 (0.01)
	RDF2Vec Skip-gram	88.9 (0.00)	85.0 (0.02)	71.1 (0.05)	88.9 (0.02)	89.4 (0.02)	91.1 (0.01)	89.4 (0.02)
	RDF2Vec CBOW	80.6 (0.00)	80.6 (0.00)	75.6 (0.07)	*88.9 (0.00)*	*91.7 (0.00)*	88.9 (0.02)	90.0 (0.02)
	R-GCN				93.3 (0.02)			

Table 5.10: The accuracy scores and standard deviations for the described classifiers and embedding approached for the MUTAG dataset.

		MUTAG						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	66.2 (0.00)	66.2 (0.00)	65.9 (0.03)	73.5 (0.00)	66.2 (0.00)	66.2 (0.00)	66.2 (0.00)
	INK Counts	69.1 (0.00)	66.2 (0.00)	57.4 (0.01)	75.0 (0.00)	66.2 (0.00)	66.2 (0.00)	66.2 (0.00)
	INK Numerical	66.2 (0.00)	66.2 (0.00)	67.1 (0.02)	73.5 (0.00)	66.2 (0.00)	66.2 (0.00)	66.2 (0.00)
	INK Non binary	66.2 (0.00)	66.2 (0.00)	66.2 (0.03)	74.4 (0.02)	66.2 (0.00)	66.2 (0.00)	66.2 (0.00)
	RDF2Vec Skip-gram	72.4 (0.01)	66.5 (0.07)	64.1 (0.04)	72.9 (0.02)	72.1 (0.02)	72.6 (0.06)	71.5 (0.03)
	RDF2Vec CBOW	67.4 (0.04)	67.1 (0.05)	65.9 (0.08)	74.1 (0.03)	75.0 (0.01)	72.6 (0.02)	72.6 (0.03)
	R-GCN				*75.9 (0.02)*			
2	INK	61.8 (0.00)	66.2 (0.00)	70.9 (0.04)	77.6 (0.01)	72.4 (0.02)	69.4 (0.02)	66.5 (0.01)
	INK Counts	61.8 (0.00)	66.2 (0.00)	73.5 (0.01)	***82.4 (0.00)***	75.0 (0.00)	71.8 (0.03)	68.5 (0.01)
	INK Numerical	63.2 (0.00)	63.8 (0.01)	66.2 (0.00)	77.1 (0.01)	78.8 (0.01)	72.1 (0.01)	69.1 (0.01)
	INK Non binary	67.6 (0.00)	66.2 (0.00)	*78.2 (0.01)*	73.5 (0.00)	72.6 (0.01)	70.3 (0.01)	69.1 (0.01)
	RDF2Vec Skip-gram	73.5 (0.01)	69.7 (0.04)	65.9 (0.05)	77.1 (0.04)	76.8 (0.01)	72.9 (0.04)	73.2 (0.03)
	RDF2Vec CBOW	74.4 (0.02)	68.5 (0.06)	62.1 (0.03)	74.1 (0.02)	77.6 (0.02)	72.1 (0.03)	73.5 (0.02)
	R-GCN				71.5 (0.07)			
3	INK	61.8 (0.00)	66.2 (0.00)	75.9 (0.02)	77.9 (0.00)	76.2 (0.03)	68.2 (0.02)	67.6 (0.02)
	INK Counts	64.7 (0.00)	66.2 (0.00)	73.5 (0.03)	80.9 (0.00)	76.5 (0.00)	75.3 (0.02)	70.0 (0.02)
	INK Numerical	63.2 (0.00)	63.2 (0.00)	65.3 (0.01)	77.9 (0.00)	77.6 (0.02)	73.2 (0.01)	72.1 (0.01)
	INK Non binary	67.6 (0.00)	66.2 (0.00)	78.2 (0.02)	77.9 (0.00)	74.1 (0.02)	70.6 (0.02)	68.5 (0.01)
	RDF2Vec Skip-gram	71.2 (0.02)	76.5 (0.05)	74.4 (0.05)	75.0 (0.01)	77.1 (0.03)	77.9 (0.03)	78.8 (0.03)
	RDF2Vec CBOW	*74.7 (0.01)*	76.8 (0.03)	68.2 (0.06)	76.5 (0.03)	78.8 (0.02)	76.8 (0.02)	78.2 (0.02)
	R-GCN				73.5 (0.05)			
4	INK	61.8 (0.00)	66.2 (0.00)	75.0 (0.02)	77.9 (0.00)	77.9 (0.00)	69.4 (0.01)	67.1 (0.02)
	INK Counts	66.2 (0.00)	66.2 (0.00)	71.2 (0.01)	76.5 (0.00)	76.5 (0.00)	73.8 (0.02)	72.1 (0.00)
	INK Numerical	63.2 (0.00)	64.7 (0.00)	66.2 (0.00)	77.9 (0.00)	77.6 (0.01)	71.8 (0.02)	73.2 (0.03)
	INK Non binary	67.6 (0.00)	66.2 (0.00)	78.2 (0.02)	77.9 (0.00)	77.6 (0.01)	70.6 (0.02)	70.0 (0.02)
	RDF2Vec Skip-gram	72.4 (0.01)	77.6 (0.04)	70.3 (0.05)	77.4 (0.02)	76.5 (0.06)	79.1 (0.02)	79.1 (0.03)
	RDF2Vec CBOW	73.2 (0.01)	81.2 (0.03)	72.6 (0.04)	75.6 (0.02)	80.9 (0.02)	80.0 (0.01)	*81.2 (0.02)*
	R-GCN				74.4 (0.03)			

Table 5.11: The accuracy scores and standard deviations for the described classifiers and embedding approached for the AM dataset.

		AM						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	79.8 (0.00)	72.6 (0.01)	83.8 (0.01)	84.3 (0.00)	89.4 (0.00)	83.5 (0.01)	83.2 (0.01)
	INK Counts	78.3 (0.00)	*74.7 (0.01)*	84.8 (0.00)	86.9 (0.00)	88.9 (0.00)	84.5 (0.01)	*83.8 (0.01)*
	INK Numerical	77.8 (0.00)	74.0 (0.00)	84.9 (0.01)	85.9 (0.00)	87.9 (0.00)	80.9 (0.00)	80.6 (0.01)
	INK Non binary	*84.3 (0.00)*	70.9 (0.02)	85.2 (0.01)	*87.9 (0.00)*	***90.4 (0.00)***	*85.8 (0.01)*	83.4 (0.01)
	RDF2Vec Skip-gram	17.1 (0.03)	33.6 (0.02)	27.5 (0.04)	44.8 (0.03)	40.5 (0.02)	39.2 (0.02)	38.2 (0.01)
	RDF2Vec CBOW	19.6 (0.05)	27.7 (0.02)	21.7 (0.03)	37.0 (0.02)	35.6 (0.01)	36.4 (0.01)	35.3 (0.01)
	R-GCN				/			
2	INK	78.3 (0.00)	68.8 (0.01)	83.6 (0.01)	80.8 (0.00)	86.4 (0.00)	80.3 (0.01)	79.4 (0.02)
	INK Counts	75.3 (0.00)	71.8 (0.01)	83.2 (0.01)	80.3 (0.00)	86.4 (0.00)	81.4 (0.01)	78.3 (0.01)
	INK Numerical	74.7 (0.00)	71.4 (0.00)	83.9 (0.01)	79.3 (0.00)	84.0 (0.00)	78.2 (0.00)	76.8 (0.00)
	INK Non binary	81.8 (0.00)	66.9 (0.01)	85.9 (0.00)	81.3 (0.00)	86.1 (0.01)	81.2 (0.01)	78.8 (0.01)
	RDF2Vec Skip-gram	73.5 (0.01)	69.7 (0.04)	65.9 (0.05)	77.1 (0.04)	76.8 (0.01)	72.9 (0.04)	73.2 (0.03)
	RDF2Vec CBOW	30.7 (0.00)	61.9 (0.01)	51.9 (0.03)	67.3 (0.02)	64.4 (0.03)	66.5 (0.02)	67.0 (0.01)
	R-GCN				/			
3	INK	78.8 (0.00)	66.3 (0.01)	85.4 (0.01)	81.8 (0.00)	84.9 (0.00)	78.1 (0.01)	76.7 (0.01)
	INK Counts	74.2 (0.00)	69.1 (0.01)	85.2 (0.02)	79.8 (0.00)	83.9 (0.00)	77.9 (0.02)	75.7 (0.02)
	INK Numerical	74.7 (0.00)	68.7 (0.00)	*87.2 (0.01)*	77.8 (0.00)	82.4 (0.01)	77.3 (0.01)	76.2 (0.01)
	INK Non binary	82.3 (0.00)	63.8 (0.01)	85.5 (0.01)	80.8 (0.00)	85.3 (0.00)	77.9 (0.01)	76.3 (0.01)
	RDF2Vec Skip-gram	34.5 (0.02)	70.2 (0.01)	48.6 (0.01)	77.0 (0.02)	76.1 (0.02)	68.4 (0.02)	69.8 (0.02)
	RDF2Vec CBOW	23.0 (0.01)	59.0 (0.01)	47.3 (0.01)	62.1 (0.02)	63.9 (0.02)	62.1 (0.01)	63.6 (0.01)
	R-GCN				/			
4	INK	79.3 (0.00)	67.6 (0.01)	85.2 (0.01)	81.8 (0.00)	85.8 (0.00)	76.3 (0.01)	75.2 (0.03)
	INK Counts							
	INK Numerical				/			
	INK Non binary	82.3 (0.00)	64.8 (0.01)	86.5 (0.00)	81.3 (0.00)	85.1 (0.00)	77.2 (0.01)	74.6 (0.00)
	RDF2Vec Skip-gram				/			
	RDF2Vec CBOW							
	R-GCN				/			

Table 5.12: The accuracy scores and standard deviations for the described classifiers and embedding approached for the DBpedia Cities dataset.

		DBpedia: Cities						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	64.7 (0.00)	67.6 (0.00)	68.4 (0.03)	73.5 (0.00)	67.6 (0.00)	75.0 (0.03)	71.3 (0.03)
	INK Counts	64.7 (0.00)	67.6 (0.00)	64.0 (0.04)	73.5 (0.00)	67.6 (0.00)	76.5 (0.03)	72.1 (0.03)
	INK Numerical	64.7 (0.00)	79.4 (0.00)	77.9 (0.05)	76.5 (0.00)	70.6 (0.00)	79.4 (0.02)	78.7 (0.01)
	INK Non binary	64.7 (0.00)	67.6 (0.00)	72.1 (0.02)	70.6 (0.00)	67.6 (0.00)	73.5 (0.02)	72.1 (0.04)
	RDF2Vec Skip-gram	64.0 (0.01)	54.4 (0.03)	52.2 (0.03)	61.8 (0.00)	62.5 (0.01)	52.2 (0.06)	52.2 (0.03)
	RDF2Vec CBOW	61.8 (0.00)	52.2 (0.07)	53.7 (0.05)	61.8 (0.00)	62.5 (0.01)	53.7 (0.05)	55.9 (0.10)
	R-GCN				/			
2	INK	67.6 (0.00)	73.5 (0.00)	69.9 (0.03)	73.5 (0.00)	79.4 (0.00)	71.3 (0.01)	76.5 (0.02)
	INK Counts	61.8 (0.00)	79.4 (0.00)	76.5 (0.02)	73.5 (0.00)	76.5 (0.00)	74.3 (0.01)	76.5 (0.04)
	INK Numerical	67.6 (0.00)	73.5 (0.00)	68.4 (0.04)	79.4 (0.00)	79.4 (0.00)	77.2 (0.04)	74.3 (0.03)
	INK Non binary	70.6 (0.00)	70.6 (0.00)	73.5 (0.02)	73.5 (0.00)	76.5 (0.00)	75.7 (0.03)	75.7 (0.03)
	RDF2Vec Skip-gram	60.3 (0.03)	62.5 (0.03)	54.4 (0.07)	66.2 (0.06)	68.4 (0.03)	70.6 (0.03)	66.9 (0.01)
	RDF2Vec CBOW	55.1 (0.01)	55.1 (0.05)	59.6 (0.05)	52.9 (0.00)	61.8 (0.05)	59.6 (0.03)	58.1 (0.04)
	R-GCN				/			
3	INK	73.5 (0.00)	67.6 (0.00)	75.7 (0.03)	82.4 (0.00)	79.4 (0.00)	81.6 (0.01)	82.4 (0.02)
	INK Counts	82.4 (0.00)	70.6 (0.00)	83.8 (0.03)	79.4 (0.00)	79.4 (0.00)	80.1 (0.01)	83.8 (0.02)
	INK Numerical	79.4 (0.00)	**85.3 (0.00)**	69.1 (0.04)	85.3 (0.00)	76.5 (0.00)	76.5 (0.00)	77.9 (0.02)
	INK Non binary	82.4 (0.00)	67.6 (0.00)	77.9 (0.03)	82.4 (0.00)	73.5 (0.00)	80.9 (0.02)	83.1 (0.01)
	RDF2Vec Skip-gram	68.4 (0.01)	69.1 (0.05)	64.7 (0.07)	74.3 (0.05)	72.1 (0.02)	73.5 (0.02)	73.5 (0.04)
	RDF2Vec CBOW	69.9 (0.04)	56.6 (0.04)	60.3 (0.05)	76.5 (0.02)	74.3 (0.06)	73.5 (0.03)	73.5 (0.06)
	R-GCN				/			
4	INK	76.5 (0.00)	67.6 (0.00)	75.0 (0.02)	76.5 (0.00)	79.4 (0.00)	79.4 (0.00)	85.3 (0.02)
	INK Counts	73.5 (0.00)	76.5 (0.00)	77.9 (0.03)	79.4 (0.00)	76.5 (0.00)	78.7 (0.01)	79.4 (0.02)
	INK Numerical	79.4 (0.00)	82.4 (0.00)	71.3 (0.04)	85.3 (0.00)	82.4 (0.00)	83.8 (0.02)	84.6 (0.01)
	INK Non binary	76.5 (0.00)	70.6 (0.00)	76.5 (0.03)	79.4 (0.00)	79.4 (0.00)	80.1 (0.01)	83.8 (0.02)
	RDF2Vec Skip-gram	68.4 (0.03)	65.4 (0.04)	61.8 (0.10)	*77.9 (0.05)*	73.5 (0.02)	69.1 (0.04)	68.4 (0.01)
	RDF2Vec CBOW	64.7 (0.05)	49.0 (0.09)	66.7 (0.06)	67.6 (0.03)	68.6 (0.06)	67.6 (0.00)	69.6 (0.07)
	R-GCN				/			

Table 5.13: The accuracy scores and standard deviations for the described classifiers and embedding approached for the DBpedia Albums dataset.

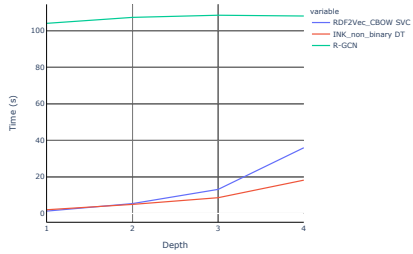
		DBpedia: Albums						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	54.0 (0.00)	54.6 (0.00)	54.1 (0.00)	56.9 (0.00)	54.0 (0.00)	55.4 (0.00)	55.7 (0.01)
	INK Counts	59.4 (0.00)	58.8 (0.00)	53.3 (0.01)	60.2 (0.00)	60.2 (0.00)	59.7 (0.01)	59.4 (0.00)
	INK Numerical	54.6 (0.00)	54.8 (0.00)	60.4 (0.01)	53.5 (0.00)	50.0 (0.00)	54.0 (0.01)	55.2 (0.01)
	INK Non binary	55.0 (0.00)	53.1 (0.00)	54.0 (0.00)	56.0 (0.00)	54.2 (0.00)	55.2 (0.00)	55.6 (0.00)
	RDF2Vec Skip-gram	51.9 (0.02)	50.8 (0.03)	50.6 (0.02)	53.1 (0.02)	50.5 (0.01)	51.1 (0.02)	51.1 (0.02)
	RDF2Vec CBOW	51.4 (0.01)	49.2 (0.01)	50.2 (0.02)	51.0 (0.01)	51.0 (0.02)	49.7 (0.03)	51.0 (0.03)
	R-GCN				/			
2	INK	67.3 (0.00)	66.0 (0.00)	64.4 (0.02)	73.5 (0.00)	70.6 (0.00)	71.5 (0.00)	71.9 (0.01)
	INK Counts	63.5 (0.00)	62.7 (0.00)	63.6 (0.01)	72.3 (0.00)	69.4 (0.00)	71.6 (0.01)	71.1 (0.01)
	INK Numerical	59.4 (0.00)	61.9 (0.00)	58.5 (0.01)	68.8 (0.00)	66.5 (0.00)	68.2 (0.01)	67.7 (0.01)
	INK Non binary	67.3 (0.00)	67.1 (0.00)	63.0 (0.01)	71.2 (0.00)	69.6 (0.00)	70.6 (0.01)	71.1 (0.01)
	RDF2Vec Skip-gram	69.1 (0.00)	70.1 (0.02)	63.7 (0.02)	72.6 (0.01)	69.9 (0.01)	71.8 (0.01)	71.7 (0.02)
	RDF2Vec CBOW	67.9 (0.01)	69.2 (0.01)	64.6 (0.02)	74.2 (0.01)	72.4 (0.01)	72.3 (0.01)	71.7 (0.01)
	R-GCN				/			
3	INK	66.2 (0.00)	64.6 (0.00)	68.6 (0.01)	72.5 (0.00)	74.4 (0.00)	71.8 (0.01)	71.7 (0.00)
	INK Counts	65.4 (0.00)	68.5 (0.00)	65.9 (0.01)	73.5 (0.00)	73.8 (0.00)	72.5 (0.01)	71.3 (0.01)
	INK Numerical	61.7 (0.00)	60.6 (0.00)	63.3 (0.01)	70.2 (0.00)	72.3 (0.00)	70.8 (0.01)	69.7 (0.01)
	INK Non binary	66.2 (0.00)	65.4 (0.00)	68.5 (0.02)	73.5 (0.00)	**74.4 (0.00)**	71.4 (0.01)	71.6 (0.00)
	RDF2Vec Skip-gram	70.4 (0.01)	74.3 (0.01)	64.1 (0.00)	74.3 (0.01)	71.0 (0.01)	73.5 (0.01)	73.6 (0.01)
	RDF2Vec CBOW	69.8 (0.01)	70.8 (0.01)	63.3 (0.01)	*74.4 (0.01)*	69.3 (0.02)	73.5 (0.01)	72.3 (0.01)
	R-GCN				/			
4	INK	64.8 (0.00)	66.7 (0.00)	65.7 (0.01)	73.1 (0.00)	73.1 (0.00)	74.3 (0.01)	73.1 (0.01)
	INK Counts	66.0 (0.00)	67.7 (0.00)	66.5 (0.02)	73.3 (0.00)	73.3 (0.00)	73.0 (0.01)	72.6 (0.01)
	INK Numerical				/			
	INK Non binary	65.2 (0.00)	65.0 (0.00)	65.5 (0.01)	74.0 (0.00)	73.3 (0.00)	73.3 (0.01)	72.6 (0.01)
	RDF2Vec Skip-gram	70.1 (0.01)	73.8 (0.03)	64.0 (0.01)	73.1 (0.00)	70.3 (0.00)	72.4 (0.02)	72.5 (0.01)
	RDF2Vec CBOW	68.9 (0.01)	68.2 (0.02)	61.0 (0.01)	71.7 (0.01)	71.1 (0.01)	72.2 (0.01)	70.7 (0.00)
	R-GCN				/			

Table 5.14: The accuracy scores and standard deviations for the described classifiers and embedding approached for the DBpedia Movies dataset.

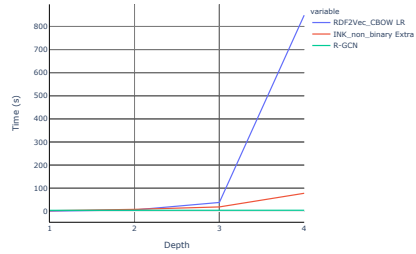
		DBpedia: Movies						
		NB	NN	DT	SVM	LR	Extra	Random
1	INK	64.2 (0.00)	64.2 (0.00)	65.8 (0.00)	67.8 (0.00)	65.8 (0.00)	68.2 (0.01)	70.4 (0.01)
	INK Counts	63.5 (0.00)	66.2 (0.00)	67.8 (0.01)	72.0 (0.00)	72.8 (0.00)	70.2 (0.01)	70.3 (0.01)
	INK Numerical	62.3 (0.00)	59.8 (0.00)	65.8 (0.00)	66.0 (0.00)	71.2 (0.00)	62.3 (0.02)	64.5 (0.00)
	INK Non binary	64.2 (0.00)	65.8 (0.00)	66.0 (0.00)	67.8 (0.00)	65.8 (0.00)	68.7 (0.01)	70.2 (0.01)
	RDF2Vec Skip-gram	56.0 (0.02)	54.2 (0.02)	52.8 (0.03)	61.3 (0.01)	59.5 (0.02)	55.8 (0.01)	55.9 (0.01)
	RDF2Vec CBOW	54.8 (0.00)	53.3 (0.02)	52.8 (0.02)	60.4 (0.01)	59.6 (0.01)	55.6 (0.03)	54.5 (0.02)
	R-GCN				/			
2	INK	69.2 (0.00)	66.8 (0.00)	74.5 (0.01)	77.8 (0.00)	79.2 (0.00)	79.3 (0.01)	80.4 (0.01)
	INK Counts	68.5 (0.00)	70.2 (0.00)	71.4 (0.00)	78.8 (0.00)	79.5 (0.00)	80.3 (0.01)	78.8 (0.01)
	INK Numerical	66.2 (0.00)	68.2 (0.00)	72.8 (0.01)	73.2 (0.00)	76.8 (0.00)	73.6 (0.01)	73.6 (0.01)
	INK Non binary	68.0 (0.00)	69.8 (0.00)	74.6 (0.00)	78.2 (0.00)	79.2 (0.00)	79.8 (0.01)	79.7 (0.01)
	RDF2Vec Skip-gram	72.8 (0.01)	69.8 (0.02)	67.4 (0.02)	79.3 (0.01)	78.5 (0.01)	76.6 (0.02)	76.9 (0.02)
	RDF2Vec CBOW	70.7 (0.00)	68.7 (0.01)	63.0 (0.04)	74.9 (0.02)	75.1 (0.02)	72.4 (0.02)	72.9 (0.01)
	R-GCN				/			
3	INK	67.8 (0.00)	66.0 (0.00)	72.4 (0.00)	77.0 (0.00)	79.2 (0.00)	78.9 (0.01)	78.2 (0.01)
	INK Counts	68.0 (0.00)	67.2 (0.00)	71.6 (0.01)	78.5 (0.00)	**80.8 (0.00)**	78.8 (0.01)	77.6 (0.00)
	INK Numerical	68.2 (0.00)	72.5 (0.00)	71.5 (0.01)	78.8 (0.00)	79.0 (0.00)	79.1 (0.01)	78.6 (0.01)
	INK Non binary	68.0 (0.00)	64.8 (0.00)	70.9 (0.02)	76.8 (0.00)	79.2 (0.00)	77.6 (0.01)	78.1 (0.01)
	RDF2Vec Skip-gram	73.6 (0.01)	72.4 (0.02)	65.4 (0.02)	*80.6 (0.01)*	79.8 (0.01)	77.3 (0.01)	78.1 (0.01)
	RDF2Vec CBOW	72.6 (0.01)	67.8 (0.01)	64.9 (0.02)	77.6 (0.01)	78.2 (0.01)	75.6 (0.01)	74.9 (0.01)
	R-GCN				/			
4	INK	64.0 (0.00)	68.0 (0.00)	68.0 (0.01)	78.2 (0.00)	79.5 (0.00)	77.9 (0.01)	77.9 (0.01)
	INK Counts	63.0 (0.00)	65.8 (0.00)	70.4 (0.01)	78.0 (0.00)	78.0 (0.00)	76.5 (0.00)	76.9 (0.01)
	INK Numerical				/			
	INK Non binary	63.7 (0.00)	68.0 (0.00)	66.9 (0.01)	77.8 (0.00)	79.2 (0.00)	77.8 (0.01)	77.6 (0.01)
	RDF2Vec Skip-gram				/			
	RDF2Vec CBOW	71.8 (0.01)	69.1 (0.02)	62.1 (0.03)	76.9 (0.01)	75.6 (0.01)	75.2 (0.00)	75.4 (0.02)
	R-GCN				/			

Addendum: Time measurements of best results

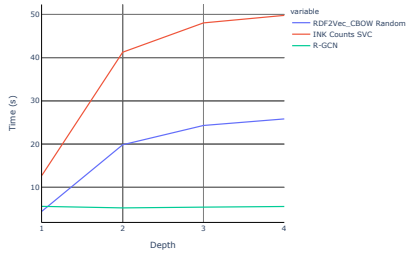
This section provides detailed information about the used time to generate the best results for all the defined datasets defined in Section 5.6. For each obtained results, the best mean time to 1) create the embedding, 2) train the classifier and 3) create the prediction over 5 runs is visualised in function of the depth. No graphs were added for techniques which did deliver useful results.



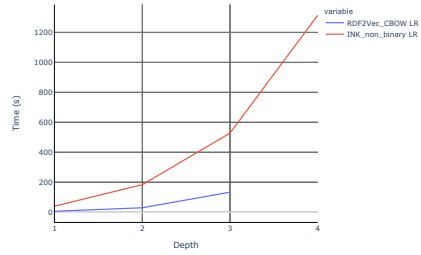
(a) BGS



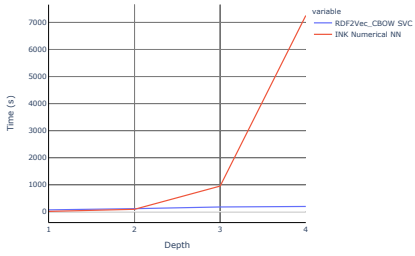
(b) AIFB



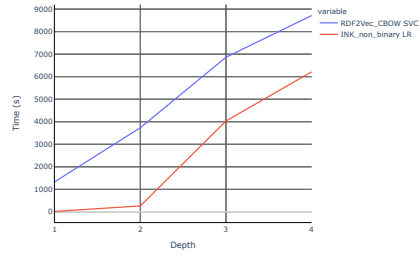
(c) MUTAG



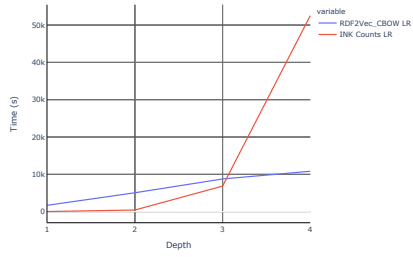
(d) AM



(e) DBpedia Cities



(f) DBpedia Albums

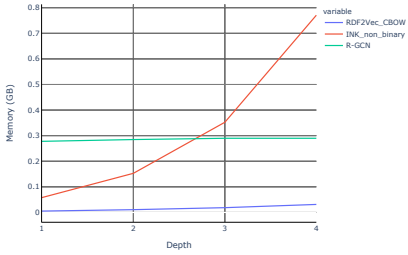


(g) DBpedia Movies

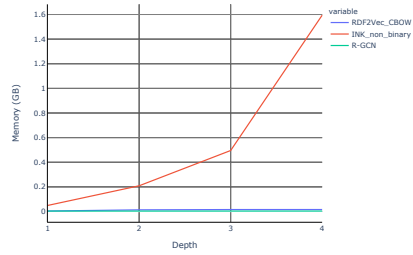
Figure 5.4: The average time measurements over 5 runs for all 7 benchmark datasets for the best results techniques and parameters as reported in Table 5.4. The time measurements are visualised in function of the depth.

Addendum: Memory consumption of best results

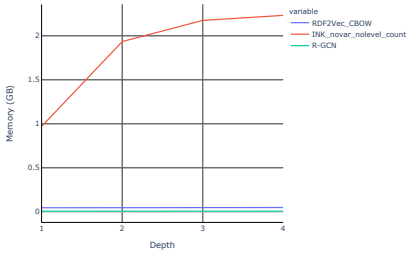
This section provides detailed information about the amount of memory used to generate the best results for all the defined datasets defined in Section 5.6. For each technique, the memory consumption of the internal representation is visualised in function of the depth. No graphs were added for techniques which do did deliver useful results.



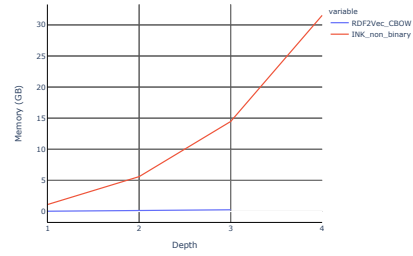
(a) BGS



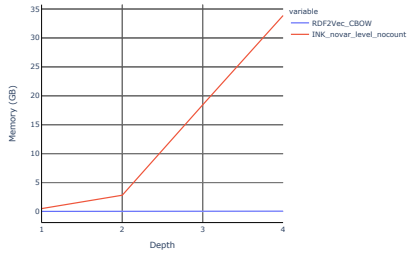
(b) AIFB



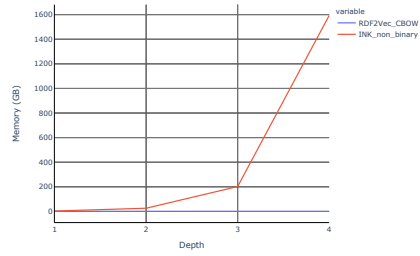
(c) MUTAG



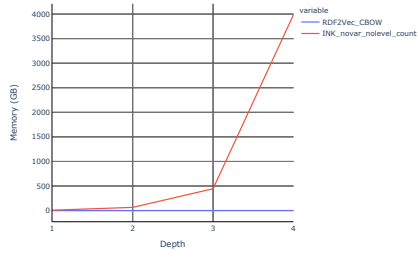
(d) AM



(e) DBpedia Cities



(f) DBpedia Albums



(g) DBpedia Movies

Figure 5.5: The average used memory in gigabytes over 5 runs for all 7 benchmark datasets, for the best results, techniques and parameters as reported in Table 5.4. The used memory is visualised in function of the depth.

References

[1] Z. Zhang, L. Cao, X. Chen, W. Tang, Z. Xu, and Y. Meng. *Representation Learning of Knowledge Graphs With Entity Attributes.* IEEE Access, 8:7435–7441, 2020.

[2] E. Miller. *An introduction to the resource description framework.* Bulletin of the American Society for Information Science and Technology, 25(1):15–19, 1998.

[3] L. Ehrlinger and W. Wöß. *Towards a Definition of Knowledge Graphs.* SEMANTiCS (Posters, Demos, SuCCESS), 48:1–4, 2016.

[4] D. Taniar and J. W. Rahayu. *Web Semantics & Ontology.* Igi Global, 2006.

[5] X. Wilcke, P. Bloem, and V. De Boer. *The knowledge graph as the default data model for learning on heterogeneous knowledge.* Data Science, 1(1-2):39–57, 12 2017. doi:10.3233/DS-170007.

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

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data.* In The semantic web, pages 722–735. Springer, 2007.

[8] D. Vrandečić and M. Krötzsch. *Wikidata: a free collaborative knowledgebase.* Communications of the ACM, 57(10):78–85, 2014.

[9] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. *Modeling relational data with graph convolutional networks.* In European Semantic Web Conference, pages 593–607. Springer, 2018.

[10] T. N. Kipf and M. Welling. *Semi-supervised classification with graph convolutional networks.* arXiv preprint arXiv:1609.02907, 2016.

[11] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. *Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks.* arXiv preprint arXiv:1909.01315, 2019.

[12] W. L. Hamilton, R. Ying, and J. Leskovec. *Representation learning on graphs: Methods and applications.* arXiv preprint arXiv:1709.05584, 2017.

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

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013.

[15] P. Ristoski, J. Rosati, T. Di Noia, R. De Leone, and H. Paulheim. *RDF2Vec: RDF graph embeddings and their applications.* Semantic Web, 10(4):721–752, 2019.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. *Distributed representations of words and phrases and their compositionality.* In Advances in neural information processing systems, pages 3111–3119, 2013.

[17] U. Lösch, S. Bloehdorn, and A. Rettinger. *Graph kernels for RDF data.* In Extended Semantic Web Conference, pages 134–148. Springer, 2012.

[18] P. Yanardag and S. Vishwanathan. *Deep graph kernels.* In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 1365–1374, 2015.

[19] D. K. Marzagao, T. D. Huynh, A. Helal, and L. Moreau. *Provenance Graph Kernel.* arXiv preprint arXiv:2010.10343, 2020.

[20] P. Ristoski, A. L. Gentile, A. Alba, D. Gruhl, and S. Welch. *Large-scale relation extraction from web documents and knowledge graphs with human-in-the-loop.* Journal of Web Semantics, 60:100546, 2020.

[21] M. M. Voit and H. Paulheim. *Bias in Knowledge Graphs–an Empirical Study with Movie Recommendation and Different Language Editions of DBpedia.* arXiv preprint arXiv:2105.00674, 2021.

[22] J. Portisch, M. Hladik, and H. Paulheim. *FinMatcher at FinSim-2: Hypernym Detection in the Financial Services Domain using Knowledge Graphs.* arXiv preprint arXiv:2103.01576, 2021.

[23] G. Vandewiele, B. Steenwinckel, P. Bonte, M. Weyns, H. Paulheim, P. Ristoski, F. De Turck, and F. Ongenae. *Walk Extraction Strategies for Node Embeddings with RDF2Vec in Knowledge Graphs.* arXiv preprint arXiv:2009.04404, 2020.

[24] M. Nickel, V. Tresp, and H.-P. Kriegel. *A three-way model for collective learning on multi-relational data.* In Icml, 2011.

[25] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. *Embedding entities and relations for learning and inference in knowledge bases.* arXiv preprint arXiv:1412.6575, 2014.

[26] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. *Complex Embeddings for Simple Link Prediction.* In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, page 2071–2080. JMLR.org, 2016.

[27] S. M. Kazemi and D. Poole. *Simple embedding for link prediction in knowledge graphs.* arXiv preprint arXiv:1802.04868, 2018.

[28] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. *Translating Embeddings for Modeling Multi-relational Data.* In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[29] Z. Wang, J. Zhang, J. Feng, and Z. Chen. *Knowledge graph embedding by translating on hyperplanes.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

[30] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. *Learning entity and relation embeddings for knowledge graph completion.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 29, 2015.

[31] H. Xiao, M. Huang, Y. Hao, and X. Zhu. *TransG: A generative mixture model for knowledge graph embedding.* arXiv preprint arXiv:1509.05488, 2015.

[32] Z. Tan, X. Zhao, Y. Fang, and W. Xiao. *GTrans: generic knowledge graph embedding via multi-state entities and dynamic relation spaces.* IEEE access, 6:8232–8244, 2018.

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

[34] S. Sabour, N. Frosst, and G. E. Hinton. *Dynamic routing between capsules.* arXiv preprint arXiv:1710.09829, 2017.

[35] T. Vu, T. D. Nguyen, D. Q. Nguyen, D. Phung, et al. *A capsule network-based embedding model for knowledge graph completion and search personalization.* In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2180–2189, 2019.

[36] B. Gunel. *Robust Relational Graph Convolutional Networks.* 2019.

[37] D. Neil, J. Briody, A. Lacoste, A. Sim, P. Creed, and A. Saffari. *Interpretable graph convolutional neural networks for inference on noisy knowledge graphs.* arXiv preprint arXiv:1812.00279, 2018.

[38] F. Baldassarre and H. Azizpour. *Explainability techniques for graph convolutional networks.* arXiv preprint arXiv:1905.13686, 2019.

[39] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta. *How to make latent factors interpretable by feeding Factorization machines with knowledge graphs.* ArXiv, abs/1909.05038, 2019.

[40] G. Vandewiele, B. Steenwinckel, F. Ongenae, and F. De Turck. *Inducing a decision tree with discriminative paths to classify entities in a knowledge graph.* In SEPDA2019, the 4th International Workshop on Semantics-Powered Data Mining and Analytics, pages 1–6, 2019.

[41] A. Zouaq and F. Martel. *What is the schema of your knowledge graph? leveraging knowledge graph embeddings and clustering for expressive taxonomy learning.* In Proceedings of The International Workshop on Semantic Big Data, pages 1–6, 2020.

[42] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu. *Modeling relation paths for representation learning of knowledge bases.* arXiv preprint arXiv:1506.00379, 2015.

[43] D. Krech. *RDFLib: A Python library for working with RDF*, 2006.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 12:2825–2830, 2011.

[45] G. Vandewiele, B. Steenwinckel, T. Agozzino, M. Weyns, P. Bonte, F. Ongenae, and F. D. Turck. *pyRDF2Vec: Python Implementation and Extension of RDF2Vec.* IDLab, 2020. Available from: https://github.com/IBCNServices/pyRDF2Vec.

[46] P. Ristoski, G. K. D. De Vries, and H. Paulheim. *A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web.* In International Semantic Web Conference, pages 186–194. Springer, 2016.

[47] V. De Boer, J. Wielemaker, J. Van Gent, M. Hildebrand, A. Isaac, J. Van Ossenbruggen, and G. Schreiber. *Supporting linked data production for cultural heritage institutes: the amsterdam museum case study.* In Extended Semantic Web Conference, pages 733–747. Springer, 2012.

[48] H. Paulheim. *Generating possible interpretations for statistics from linked open data.* In Extended Semantic Web Conference, pages 560–574. Springer, 2012.

[49] P. Ristoski, H. Paulheim, V. Svátek, and V. Zeman. *The Linked Data Mining Challenge 2016.* In (KNOW@ LOD/CoDeS)@ ESWC, 2016.

[50] P. Ristoski, H. Paulheim, V. Svátek, and V. Zeman. *The Linked Data Mining Challenge 2015.* In KNOW@ LOD, 2015.

[51] S. Union. *Stardog*, 2018.

[52] V. Gulisano, Z. Jerzak, R. Katerinenko, M. Strohbach, and H. Ziekow. *The DEBS 2017 Grand Challenge.* In Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems, DEBS '17, page 271–273, New York, NY, USA, 2017. Association for Computing Machinery. Available from: https://doi.org/10.1145/3093742.3096342, doi:10.1145/3093742.3096342.

[53] T. Thanapalasingam, L. van Berkel, P. Bloem, and P. Groth. *Relational Graph Convolutional Networks: A Closer Look*. arXiv preprint arXiv:2107.10015, 2021.

[54] S. Khalid, T. Khalil, and S. Nasreen. *A survey of feature selection and feature extraction techniques in machine learning*. In 2014 Science and Information Conference, pages 372–378. IEEE, 2014.

[55] S. M. Lundberg and S.-I. Lee. *A Unified Approach to Interpreting Model Predictions*. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4765–4774. Curran Associates, Inc., 2017. Available from: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

6

Data Analytics For Health and Connected Care: Ontology, Knowledge Graph and Applications

In this chapter, we discuss a large dataset consisting of both sensor measurements and user metadata within a real-life homecare setting. Accompanying this dataset, ontologies related to this healthcare setting and ontologies in the field of the data analytics domain were linked together to create the data analytics for healthcare and connected care (DAHCC) upper ontology. In this ontology, the result from healthcare-related AI models can be linked to both the original raw data and the user or sensor properties which are inputted into these AI models. The whole raw dataset is transformed into a KG based on the principles of this ontology and made available for further research. The dataset, KG and DAHCC ontology are used in the next chapter to analyse the influence of the time series sensor data in a KG for a real-life HAI healthcare application, how rules can be derived to still provide interpretable results and how KG embeddings can be used to feed the HAI pipeline.

⋆ ⋆ ⋆

B. Steenwinckel, M. De Brouwer, M. Stojchevska, J. Van Der Donckt, J. Nelis, J. Ruyssinck, J. van der Herten, K. Casier, J. Van Ooteghem, P. Crombez, F. De Turck, S. Van Hoecke, F. Ongenae

Published in the proceedings of the 16th EAI PervasiveHealth Conference, December 2022.

Abstract
Connected care applications are increasingly used to achieve a more objective, continuous and pervasive healthcare follow-up of (chronic) diseases. Within these applications, objective insights about the patients are collected by using Artificial Intelligence (AI) models on Internet of Things (IoT) devices in their homes and by using wearable devices to capture biomedical parameters. However, to enable easy re-use of AI algorithms and applications trained and designed on top of sensor data, it is important to uniformly describe the collected data and how this links to the health condition of the patient. In this chapter, we propose the DAHCC (Data Analytics For Health and Connected Care) ontology, dataset and Knowledge Graph (KG). The ontology allows capturing the metadata about the sensors and data, the different designed AI algorithms and the health insights they derive and how this can be correlated to the medical condition of the patients. To showcase the use of the ontology, a large dataset of 42 participants performing daily life activities in a smart home was collected and annotated with the DAHCC ontology into a KG. Three applications using this KG are also presented in this chapter as inspiration on how other connected care applications can utilize DAHCC. The ontology, KG and the applications are made publicly available at **https://dahcc.idlab.ugent.be**. DAHCC's goal is to integrate large care systems such that their outcomes can be visualised, interpreted and acted upon without increasing the burden of healthcare professionals who rely on such systems.

6.1 Introduction

From the perspective of healthcare professionals, our current healthcare system brings many challenges as well as opportunities for digital healthcare [1]. Each healthcare professional already takes care of multiple people as more and more in-person visits have to be reduced to an absolute minimum. Connected care solutions, including remote patient monitoring and secure communications between clinicians or caregivers and their patients, may rapidly become the first choice to provide care in a public health system [2].

First steps have already been taken to provide such connected care solutions. Smart cities are designed using smart sensors to track the well-being of their citizens [3], ambient intelligent homes where sensors track and derive information from their residents in a privacy-friendly manner are being deployed [4] and patients with specific healthcare-related problems are equipped with smart sensors, such as wearables, to track biometric properties [5].

Those care applications deliver insightful information for caregivers and health professionals and should eventually help them to work more effectively [6]. Artificial Intelligence (AI) applications, such as Human Activity Recognition (HAR), can be used to monitor the physical abilities of elderly [7]. The HAR outcome or prediction can be seen as new knowledge, which can be of interest to a healthcare professional [8].

However, monitoring and analysing all the data produced by intelligent sensors and care applications should not increase the burden and stress of those who rely on them [9]. Therefore, a uniform method is needed to both describe the connected care applications and associated data. By doing this, the derived knowledge from those care applications can be easily integrated into new smart applications and can be automatically reported using insightful dashboards. They can also be used in autonomous alerting tools to inform health care professionals when unwanted behaviour or situations of high risk occur.

In this paper, we present the Data Analytics for Health and Connected Care (DAHCC) Ontology, a method to semantically describe sensor data, human activities, smart applications, health actors, faulty or unwanted behaviour and link them all together. To facilitate the idea behind DAHCC and the creation of new connected care applications, an ambient intelligent dataset was created and semantically enriched into a Knowledge Graph (KG) with over 40 participants performing daily and nightly activities. Example applications show how new connected care applications can be developed based on this dataset. Both the used method (ontology), datasets, KG, as well as the used applications are made available online under an MIT license.

The remainder of this paper defines in Section 6.2 the different existing techniques to describe healthcare resources and applications uniformly. The creation of the DAHCC ontology based on the multiple reused industry standards and well-adopted ontologies is provided in Section 6.3. Section 6.4 describes the creation of the dataset and Section 6.5 details how it was transformed into a KG. The applications built upon this KG are described in Section 6.6.

6.2 Related Work

Monitoring of patients in either an ambient life setting or for general healthcare purposes covers a large research domain. The usage of IoT devices and wearables was crucial in the development of these applications [10]. In the last decade, healthcare ontologies became popular to incorporate and combine domain knowledge with these sensory devices. Table 6.1 summarises the field of connected care ontologies. Most of these ontologies are not publicly available to be reused in other applications or were designed for a specific use case within a connected care setting.

The ACCIO ontology [11] exploits and integrates the heterogeneous data by utilising a continuous care ontology for patient rooms of the future in a hospital setting. It was one of the first ontologies co-created with nurses, caregivers, patients, doctors and professionals working in the healthcare industry. The ontology is used to exploit and integrate heterogeneous data. The Patient-Centric for Telehealth ontology [12] was designed with an explicit focus on the personality traits of the patients to describe diseases, functioning and physiological measurement. It does, however, not integrate sensor-related information. The HealthIoT ontology [13] aims to rep-

Table 6.1: Overview of existing Healthcare-related ontologies

Ontology	Available	Reused Ontologies	Semantically describe
ACCIO [11]	Yes	SSN	Ambient patient rooms
Telehealth [12]	No	ICD, ICF, SNOMED-CT	Patient profile
HealthIoT [13]	No	NaN	Medical devices
SAREF4EHAW [14]	Yes	SAREF	Monitoring health actors
e-Health [15]	No	SSN	Device communication
IFO [16]	No	NaN	IoT health and fitness data
LHR [17]	No	SNOMED-CT, SSN	Health care data exchange
SHCO [18]	Yes	SAREF	Patient-doctor interactions
Do-Care [19]	No	FOAF, SOSA, ICNP	Nurse interactions
DAHCC	**Yes**	**SAREF, EEP, OWL Time**	**Connected care**

resent the semantic interoperability of the medical connected objects and their data. The ontology focuses only on the medical aspects of the devices and provides rules to analyse the detected vital signs. Those analyses can be delivered to a healthcare professional. SAREF4Health ontology [14] is an extension of SAREF, a framework for smart appliances references. The SAREF ontology describes how sensors and sensor data can be described and linked to each other. The SAREF4Health extension specifies eHealth/Ageing-well (EHAW) domain-related resources and defines how the sensor data relates to health actors. Care-specific needs or how new systems can provide additional information towards a connected care system are not provided in this SAREF4Health extension. The e-Health ontology [15] tries to reduce programmatically implementing the interpretation of the data sender and data receiver for each new healthcare device added into a system. This ontology lowers the efforts needed to extend a current healthcare setup in an ambient living context. The IoT fitness ontology (IFO) [16] presents a semantic data model useful to consistently represent health and fitness data from heterogeneous IoT sources and integrate them into semantic platforms to enable automatic reasoning by inference engines. This ontology does not take into account standards such as SSN or SAREF to describe the sensor and data. Linked Health Resource (LHR) ontology [17] integrates health data from different services as linked resources. The healthcare-IoT ontology [20] provides semantic interoperability among heterogeneous devices and users in the healthcare domain. The ontology models the exchange of health care data and home environment data. The smart healthcare ontology (SHCO) [18] define concepts for monitoring doctors and patients anytime, anywhere. SHCO is presented as a semantic model by extracting healthcare knowledge such as doctor-patient records, recommended diagnoses and treatment policies. This ontology only uses rather static non-sensory information. The Do-Care ontology [19] is modular and incorporates the International Classification for Nursing Practice (ICNP) ontology together with inference rules. The methodology is dynamic and adjustable to meet possible changes in the medication market, medical discoveries, and personal usersâ€™ profiles. Nurse interactions are the main focus of

this ontology.

Each of these ontologies describes a clear subpart within the connected care domain. Almost all ontologies already reused existing ontologies such as SSN or SAREF. Some of them are also made open-source and are available online for further reuse. However, an ontology which describes all concepts related to connected care, ambient living, patient care and their interaction with healthcare professionals is currently not available. Moreover, there doesn't exist to our knowledge an ontology which links AI models, with their predictions and model configuration, to the monitored context or situation of a healthcare actor. Such an ontology is of high need as more and more AI models are being used within healthcare. Those models generate new insights that, when semantically described, can deliver even more advanced input to healthcare professionals. In this paper, the DAHCC ontology is created to resolve this need.

6.3 DAHCC Ontology Creation

The DAHCC ontology describes real-world connected care entities (person, sensor, activities, etc.) and their interrelationships. It also models domain knowledge, e.g., performing human activities & profile data or location information where those sensors are installed. The design of DAHCC ontology was based on three types of information sources:

- Structured documents, e.g. the descriptions of the used sensors and monitoring systems in technical specifications & APIs, in-take documents used by caregivers, etc.
- An assessment of existing ontologies that could be reused.
- Some knowledge is only available from the stakeholders, who perform day-to-day assessment & handling of deviating situations. To derive this information, decision tree workshops were organised with nurses and caregivers as described in [21].

The structured documents contained mostly resident-specific, privacy-sensitive information. We kept those fields that are interesting for healthcare-related cases. SAREF Core[1], SAREF4BLDG[2], SAREF4EHAW[3], SAREF4WEAR[4], Execution Executor Procedure (EEP)[5] and the OWL Time ontology[6] were reused within DAHCC. The decision tree workshops resulted in a long list of care intervention reasons, and an indication of which data/information the caregivers used to assess alarming situations

[1]https://saref.etsi.org/core/v3.1.1/
[2]https://saref.etsi.org/saref4bldg/v1.1.2/
[3]https://saref.etsi.org/saref4ehaw/v1.1.1/
[4]https://saref.etsi.org/saref4wear/v1.1.1/
[5]https://iesnaola.github.io/EEP/index-en.html
[6]https://www.w3.org/TR/owl-time/

and how they handle them. The information inside these decision trees was used to define new concepts regarding patient monitoring within the DAHCC ontology.

6.3.1 DAHCC Ontology

The DAHCC ontology exists out of five sub ontologies which have links to each other. Each of these ontologies reuses one or more of the mentioned, already existing ontologies, combined with new concepts derived from the provided workshops [21]. These workshops brought together ontologists, healthcare providers and people who monitor patient calls. The results of these workshops are identified use cases by the participants, a long list of call reasons and resulting decision trees of how care is provided or how a person monitors patient calls. These decision trees were consolidated by the ontologist into two decision trees: one for the nurses and one for the caregivers. The information inside these decision trees was used to define new concepts regarding patient monitoring and handling of call operations within the DAHCC ontology. For the extensive documentation of the ontological concepts, we refer the interested reader to the ontology section at https://dahcc.idlab.ugent.be. The remainder of this section describes the 5 different sub ontologies in the DAHCC dataset.

6.3.1.1 Activity Recognition

The Activity Recognition sub ontology defines how the concepts of activities, performed by a saref:HealthActor, can be predicted using an activity recognition model. The ontology also describes how such a model can be defined, together with its configuration and input and output data. The ontology describes more in general lifestyles, routines and anomaly classes for, e.g., unwanted daily or nightly activities. The Activity Recognition ontology will be used to describe all the concepts related to both performed and predicted activities, together with their link to the models/techniques used to detect them, the performed routines and the lifestyle of the health actor. An overview of this ontology is provided in Figure 6.1.

6.3.1.2 Monitored Person

The Monitored Person sub ontology, as shown in Figure 6.2, describes the person itself who is monitored by all sensors and who performs the human activities and routines. It also describes the possible diseases, addictions, mental illnesses or allergies this monitored person can have. Medication and the current mental state of the monitored person are also defined as concepts within this ontology. This Monitored Person ontology has a strong link to the Activity Recognition ontology to link human activities.

6.3.1.3 Sensors and Actuators

This Sensors and Actuators sub ontology describes a numerous amount of sensors and actuators that produce data and can be equipped inside a household. Besides the sensors and the measurement properties, this ontology also defines where those sensors are placed and which appliance or rooms they analyse. This subontology is shown in Figure 6.3 and mainly extends the SAREF Core and SAREF4BLD ontologies with ambient life-specific concepts.

6.3.1.4 Sensors And Wearables

Similar to the Sensors and Actuators ontology, the Sensors and Wearables ontology (Figure 6.4) describes the wearables and sensors that can be attached to or near a monitored person. This sub ontology extends the SAREF4WEAR ontology and adds specific connected care concepts to it. Wearables range from medical devices using simple near-field communication or Bluetooth connections to send medical parameters to a cloud environment, as well to more sophisticated smartwatches to track residents inside a building 24/7.

6.3.1.5 Caregiver

The Caregiver sub ontology defines the link between caregivers and monitored persons. It uses the SAREF4EHAW concepts to assume the possible relation between health actors. Additionally, it also describes how the required care can propagate to the responsible entity or caregiver. It also defines the Care Provisioning activity. An overview of this ontology is provided in Figure 6.5.

6.4 DAHCC Dataset

To show how the DAHCC Ontology can be used to annotate healthcare data and how this is beneficial for the creation of connected care applications and extracting knowledge, a large ambient intelligent data collection campaign at the HomeLab of the IDLab research group of Ghent University was performed. The HomeLab is an actual standalone house offering a unique residential test environment for IoT services and smart living. A wide range of IoT technologies are deployed, and the set-up allows to add new devices using technical corridors, hollow floors and ceilings[7].

In total, 42 participants were invited to the Homelab and were asked to either perform their daily or nightly routines. Both the Homelab and the participants were equipped with a large number of sensors which try to capture the participant's activities. The data collected for each participant, as well as the daily life annotated activities, are made available at https://dahcc.idlab.ugent.be/dataset.html.

[7] https://www.ugent.be/ea/idlab/en/research/research-infrastructure/homelab.htm

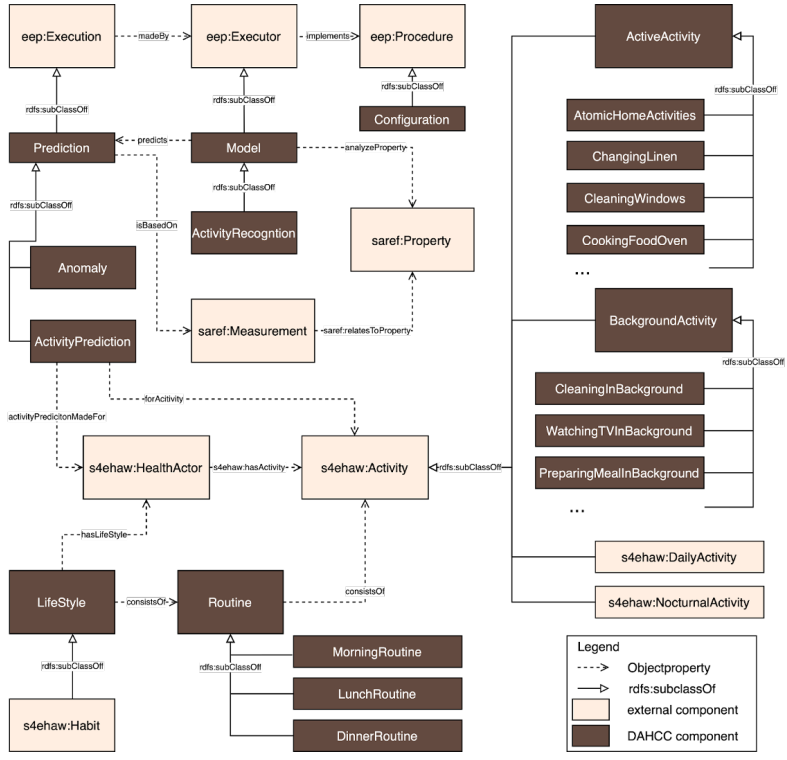


Figure 6.1: Overview of the DAHCC Activity Recognition sub ontology

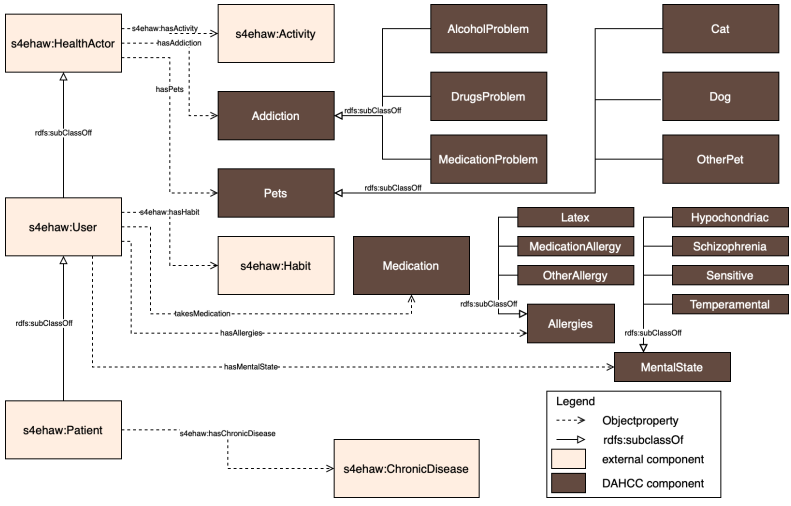


Figure 6.2: Overview of the DAHCC Monitored Person sub ontology

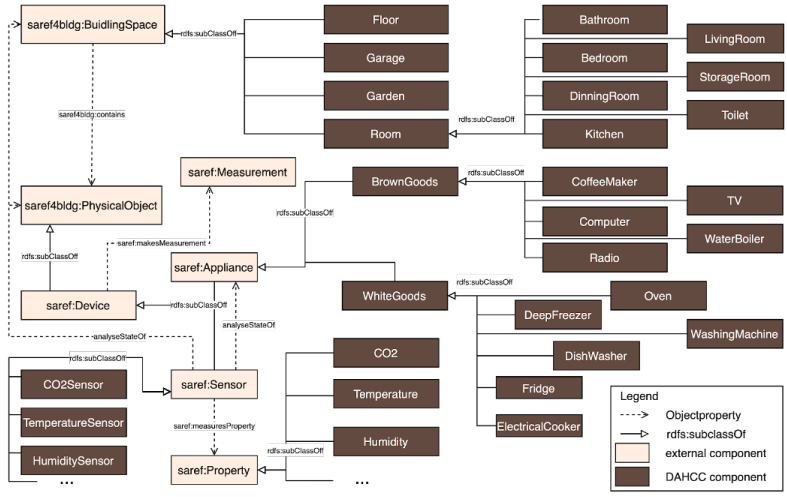


Figure 6.3: Overview of the DAHCC Sensors and Actuators sub ontology

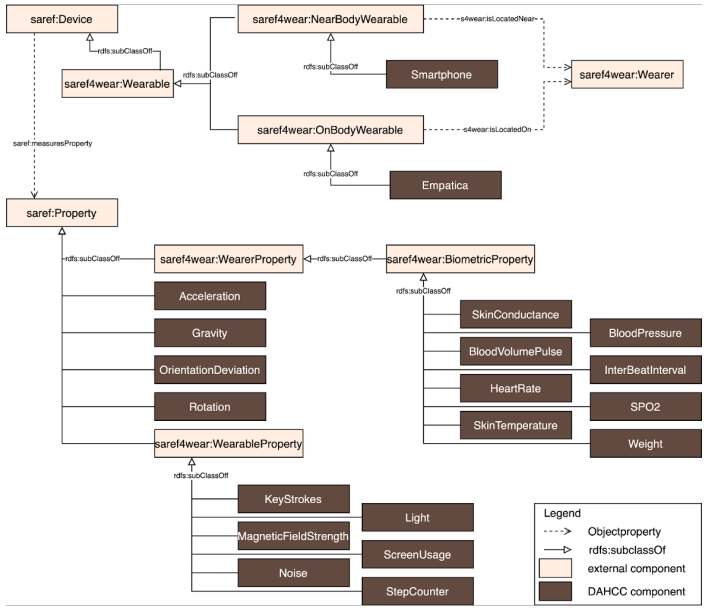


Figure 6.4: Overview of the DAHCC Sensors and Wearables sub ontology

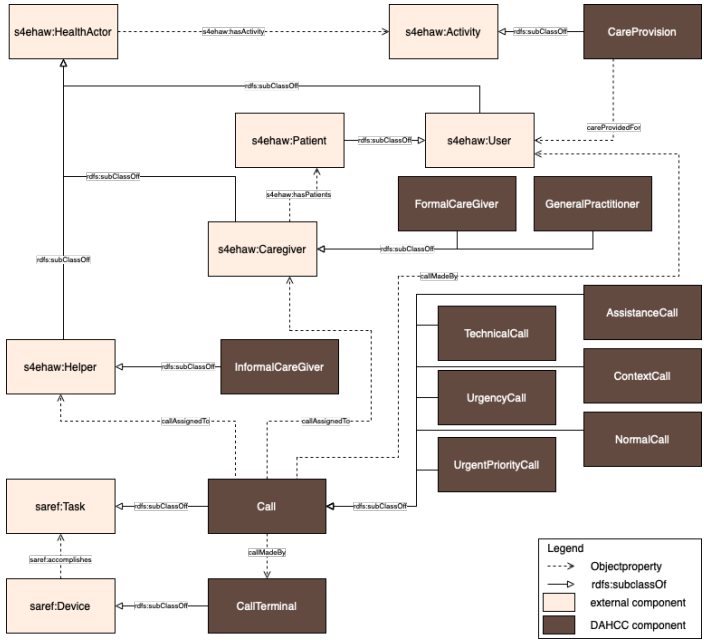


Figure 6.5: Overview of the DAHCC Caregivers sub ontology

6.4.1 Data collection setup

To derive the daily and nightly activities, a data platform was designed to capture and store the sensor readings and participant annotations. An overview of this platform is provided in Figure 6.6. The rooms within the Homelab were also equipped with many contextual sensors. People Counters and the AQURA Indoor localization system of Televic Healthcare[8] were used to derive the indoor location of the participants. Door contact sensors were installed on every door, window and cabinets in the kitchen the grab their open and close states. Velbus sensors[9] were used to control and monitor the lights, indoor temperature, opening or closing of the blinds and the energy consumption of all appliances. A specifically designed water running sensor was used to detect when water was being used from the faucets in the bathroom and kitchen. At last, several rooms were equipped with a CO2, Humidity and Loudness sensor. All these Homelab sensors were integrated using the DYAMAND platform [22]. DYAMAND collects the data from these sensors and provides it in a JSON format to the data lake.

A web application was designed that allows participants to (a) enter the activities

[8]https://www.televic-healthcare.com/en/solutions/wireless-nurse-call-systems/wireless-localisation-tags
[9]https://www.velbus.eu

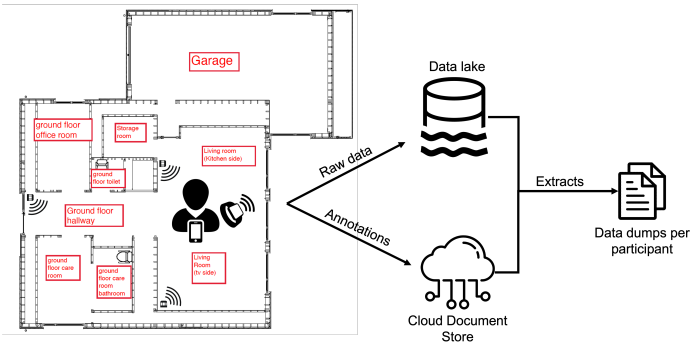


Figure 6.6: Overview of the DAHCC Homelab data collection campaign. Data captured from different sensors and wearables was sent to the data lake. Data dumps for each participant with sensor data and annotating labels are generated and made available.

they perform as part of a routine, and (b) indicate when they start/stop a specific activity. The app was used during the whole data collection to allow participants to annotate the actions they are performing. Every time a participant interacted with the application (when a start, end or cancel button was pressed), a timestamp together with the performed action was sent to a log file on a cloud document store. This log file was later on analysed to derive the annotations.

Together with this annotation app, each participant was asked to install two additional smartphone applications: (a) the streaming application to collect data from a wearable (Empatica E4[10]) and smartphone sensors and send them to the data lake, and (b) the Sleep as Android application[11] (to track sleep during the night protocol). Besides the wearable device, the blood pressure, body weight, body temperature and spO2 biomedical parameters were measured at the start of the day if the participant gave their approval inside the informed consent.

6.4.2 Collected data

In total, 31 "day in life" participants and 12 "night" participants enrolled in this data collection campaign and annotated, on average, 70.7 activities using the mobile annotation app. On average, more than 1 gigabyte of data was collected for every participant. Before making the gathered data publicly available, the data samples were anonymized by erasing the date information within the timestamps of the sensor values (the original times were kept). Also, the participant numbers were randomised, such that participant 1 isn't the first participant who gathered data within this data collection campaign.

[10] https://www.empatica.com
[11] https://sleep.urbandroid.org

6.5 DAHCC KG

The described dataset in Section 6.4 was transformed based on the DAHCC ontology into a KG linking all information together. To create this KG, we performed to following steps:

- In a first step, we mapped the Homelab floor plan and location of all sensors and actuators based on the DAHCC Sensor and Actuators sub ontology. We also semantically defined all the major appliances and provided the link to those sensors that measure their energy consumption. The semantic representation of the Homelab is made available as an additional resource[12].

- Secondly, we also mapped the used Empatica E4 wearable and other biomedical devices using the Sensors and Wearables sub ontology. Again, we made this resource available for future reference.

- In a third step, all the data from the data collection participants were transformed into a semantic representation using a Python script. This script maps each sample to a participant-specific URI identifier and links it to the concept described in the Sensors and Actuators and Sensors and Wearables sub ontologies.

- A similar Python script was created to map the participants' annotations onto the concepts of the Activity Recognition sub ontology.

For each participant, the output of these scripts generated NTRIPLE files, which were combined and gzipped to reduce the KG file size. Those individual KG files, per participant, were also made available at https://dahcc.idlab.ugent.be/dataset.html.

6.6 DAHCC Semantic Applications

Inspired by the available DAHCC KG, we created three applications which can be used to derive new knowledge from or infer useful results for healthcare professionals. A first application defines how the available domain knowledge within the DAHCC ontology can be used to generate more advanced events using reasoning. A second application describes how semantic rule mining based on the inferred knowledge can be performed. A third application shows how such a rule can be incorporated into a semantic stream reasoning engine. Implementations and examples on how to use each of these applications are provided in a GitHub repository[13].

[12]https://github.com/predict-idlab/DAHCC-Sources
[13]https://github.com/predict-idlab/DAHCC-Sources/tree/main/Applications

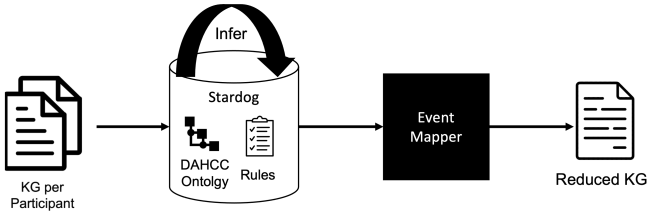


Figure 6.7: Overview of the Inferred Knowledge generation. This generator was adapted from https://www.stardog.com/labs/blog/stream-reasoning-with-stardog/

6.6.1 Semantic higher-order events generation

The DAHCC KG defines the sensor data and corresponding metric information. Due to all the available knowledge, we can reduce this KG by transforming groups of sensor observations into more relevant events happening inside the Homelab. Most sensors measured the state of an appliance or object within a certain room, e.g. the Velbus energy sensors measure the energy consumption of the cooking top in the kitchen. Instead of storing only the data observations in a semantic format, the usages of appliances and objects were also inferred based on the sensor values inside the KG.

In this approach, the semantic observation samples are loaded within a Stardog[14] database together with the DAHCC ontology and some generic, predefined rules. Reasoning-on-query is then executed on these observation samples within the Stardog database to get the inferred and derived events. For each defined rule, a specific event is created when inferred (e.g. the Start command rule creates an on state event for a specific appliance).

Next to the action and corresponding appliance or physical object related to the occurring action, both the time when this event happens and the participant who executes the action are stored. The annotated activities, the inferred events and sensor observations were all combined within a new KG which now only contains events instead of a large number of observations.

Within this application, the DAHCC ontology is used to deliver additional context information to enrich the raw data. Since the specific Homelab appliances, rooms and sensors are described by such DAHCC components, rules could be designed to combine the data and metadata to generate new insights. Those defined rules are due to the use of the DAHCC ontology generically and are easy to interpret.

The creation of higher-order events is needed in a healthcare setting to reduce the number of raw data samples that have to be monitored by a healthcare professional. Due to the available ontology and the metadata described by this ontology, simplifications of the data can be provided to create more interpretable information about what is going on in a smart home.

[14]https://www.stardog.com

6.6.2 Deriving rules for shower events

The previous application generates more advanced events to monitor the behaviour of a person in an ambient house setting. Additionally, it would also be beneficial to detect the lifestyle activities performed by these people as those activities define the current behaviour and state of the person. Based on the available data and metadata, and to ensure the detection method is interpretable, a semantic rule mining application was designed to derive lifestyle rules based on the semantic events generated by the previous application. The INK rule miner [23] was used to derive task-specific rules for one group of activities (e.g. shower events) compared to all other events. This rule miner is based on the INK representation to embed the KG but performs a task-specific rule mining operation based on Bayesian rule set [24]. The task here is to find semantic rules which discriminate against one type of activity as best as possible regarding precision (defining how many predicted rule outcomes were relevant) and recall (defining how many relevant triggered rule outcomes were retrieved). Table 6.2 shows examples of relevant rules found for the Shower, Toilet and Washing Hands activities. For all tasks, a highly imbalanced set of labels has been provided and the INK rule mining parameters were set to mine the most precise rules. Therefore, the recall scores are significantly lower than the precision scores for the obtained rules.

Table 6.2: Results of the performed rule mining operation on the dataset

Event	Rule	Precision	Recall
Shower	hasEvent.kitchen.Temperature >21.75 and hasEvent.bathroom.Loudness_mean >46.83 and NOT hasEvent.personInÂ§kitchen	0.9411	0.6153
Toilet	hasEvent.LightSwitchOnInÂ§toilet1 and NOT hasEvent.personInÂ§kitchen	0.8175	0.5734
WashingHands	hasEvent.LightSwitchOnInÂ§toilet1 and hasEvent.usingÂ§waterpump	0.8600	0.3385

The DAHCC ontology delivers in this application the additional knowledge to mine more insightful rules. Without the DAHCC ontology, rules will be less generic and it will be harder to mine rules based on related data observations (e.g. observations from similar sensors or observations made within the same room). Mining rules automatically is also needed to create an interpretable, but adaptive healthcare monitoring system where a human only has to verify but not create the rules.

6.6.3 Semantic stream reasoning

Mining rules based on semantic events deliver useful insights into the daily life pattern of a person. Rule-based systems are frequently used as connected care applications as they are both reliable and interpretable. We created a semantic stream reasoning application to show how DAHCC can be used for such a technique.

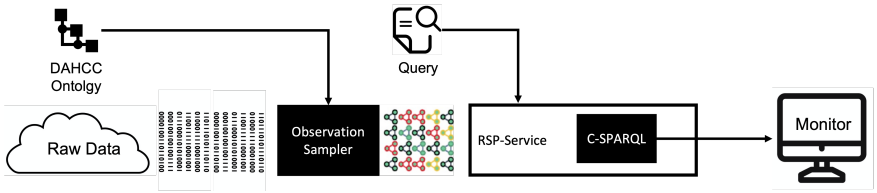


Figure 6.8: Overview of the semantic stream reasoning setup

An overview of this semantic stream Reasoning unit is provided in Figure 6.8. First, the raw data samples are mapped in a semantic format one by one. Next, those semantic observations are fed to a C-SPARQL [25] engine. C-SPARQL is used here in combination with an RSP-Service[15]. This combination of RSP and C-SPARQL makes it possible to dynamically load query rules to detect the lifestyle activities of a person. In the last step, the query is performed on a window of obtained semantic events and when a result can be inferred, this semantic result is sent to a monitoring application.

An instantiated application where we try to find shower activities within a semantic data stream is made available. Again the DAHCC components described the semantic observations, on which the shower query rule is defined. Multiple sensor values to determine the location of the person and the humidity sensor values within the bathroom are combined in order to generate a rule-based prediction. The whole setup and the defined rule-based predictions are also semantically described using the DAHCC ontology.

In the healthcare sector, semantic stream reasoning can be used to monitor a patient with specific care needs. Cases exist where based a smart room adapts to the needs of a patient suffering a concussion [26]. Using the DAHCC ontology different alert levels and priorities can be automatically defined and monitored based on the patient's care needs and the available sensors and actuators inside a smart home.

6.7 Discussion & Conclusion

In this work, we presented DAHCC, a combined resource which provides healthcare knowledge in numerous settings using a maintained ontology and a large dataset to build and create semantic connected care applications. All the resources, the resource creation files and example semantic connected care applications files are made open source.

The applicability of the ontology is evaluated by transforming the dataset into a semantic format, resulting in a KG. The construction of these KG files was created using a script because all the original raw dataset files had a similar structure. Tech-

[15] https://github.com/streamreasoning/rsp-services

niques exist to provide a more user-friendly and standardized way to transform such data files into a semantic representation. The applicability and overview of these existing techniques, their benefits and drawbacks, were left out of scope in this research.

The open-sourced KG is based on all raw sensor input from a smart lab environment. Therefore, the KG shows only one part of the available DAHCC ontology. The goal of the DAHCC ontology is that additional instances such as designed artificial intelligence models or patient-specific information are also made available in such a KG. The design of such an artificial intelligence model is kept to a minimum and is part of future work. Patient-specific information was not incorporated in the dataset as anonymization of the participants was required to make it publicly available.

At last, the ontology itself was built by taking into account the input from different people within the healthcare domain. While the current setting was mainly focused on how ambient living and connected care systems can be combined, the ontology itself is defined and constructed to be further extended or adapted when new information or new use cases become available. Making the ontology open-source makes it possible for other researchers to further extend it.

Resource Availability Statement: The DAHCC ontology, datasets and KG are available online from https://dahcc.idlab.ugent.be. The source code for the applications is available on Github at https://github.com/predict-idlab/DAHCC-Sources.

Acknowledgement: Bram Steenwinckel (1SA0219N) is funded by a strategic base research Grant of the Fund for Scientific Research Flanders (FWO). This research is part of the imec.ICON project PROTEGO (HBC.2019.2812), co-funded by imec, VLAIO, Televic, Amaron, Z-Plus and ML2Grow.

References

[1] T. Davenport and R. Kalakota. *The potential for artificial intelligence in healthcare*. Future healthcare journal, 6(2):94, 2019.

[2] T. Schinköthe, M. R. Gabri, M. Mitterer, P. Gouveia, V. Heinemann, N. Harbeck, M. Subklewe, et al. *A web-and app-based connected care solution for COVID-19 in-and outpatient care: qualitative study and application development*. JMIR public health and surveillance, 6(2):e19033, 2020.

[3] J. Hofman, V. P. La Manna, and J. Muylaert. *Measuring and Modeling Air Quality in Smart Cities*, 2021.

[4] F. De Backere, F. Ongenae, F. Van Den Abeele, J. Nelis, P. Bonte, E. Clement, M. Philpott, J. Hoebeke, S. Verstichel, A. Ackaert, et al. *Towards a social and context-aware multi-sensor fall detection and risk assessment platform*. Computers in biology and medicine, 64:307–320, 2015.

[5] M. De Brouwer, N. Vandenbussche, B. Steenwinckel, M. Stojchevska, J. Van Der Donckt, V. Degraeve, J. Vaneessen, F. De Turck, B. Volckaert, P. Boon, et al. *mBrain: towards the continuous follow-up and headache classification of primary headache disorder patients*. BMC medical informatics and decision making, 22(1):1–34, 2022.

[6] S. Y. Y. Tun, S. Madanian, and F. Mirza. *Internet of things (IoT) applications for elderly care: a reflective review*. Aging clinical and experimental research, 33(4):855–867, 2021.

[7] A. Ganesan, A. Paul, and H. Seo. *Elderly People Activity Recognition in Smart Grid Monitoring Environment*. Mathematical Problems in Engineering, 2022, 2022.

[8] A. Aldahiri, B. Alrashed, and W. Hussain. *Trends in using IoT with machine learning in health prediction system*. Forecasting, 3(1):181–206, 2021.

[9] L.-C. Chiang, W.-C. Chen, Y.-T. Dai, and Y.-L. Ho. *The effectiveness of telehealth care on caregiver burden, mastery of stress, and family function among family caregivers of heart failure patients: a quasi-experimental study*. International journal of nursing studies, 49(10):1230–1242, 2012.

[10] S. D. Mamdiwar, Z. Shakruwala, U. Chadha, K. Srinivasan, C.-Y. Chang, et al. *Recent advances on IoT-assisted wearable sensor systems for healthcare monitoring.* Biosensors, 11(10):372, 2021.

[11] Ongenae, Femke and Bleumes, Lizzy and Sulmon, Nicky and Verstraete, Mathijs and Van Gils, Mieke and Jacobs, An and De Zutter, Saar and Verhoeve, Piet and Ackaert, Ann and De Turck, Filip. *Participatory design of a continuous care ontology*

: towards a user-driven ontology engineering methodology. In KEOD 2011: proceedings of the int. conf. on knowledge engineering and ontology development, pages 81–90. INSTICC, 2011.

[12] D. B. Jørgensen, K. Hallenborg, and Y. Demazeau. *Patient centric ontology for telehealth domain*. In int. conf. On Smart homes and health Telematics, pages 244–255. Springer, 2015.

[13] A. Rhayem, M. B. A. Mhiri, and F. Gargouri. *HealthIoT ontology for data semantic representation and interpretation obtained from medical connected objects*. In 14th int. conf. on computer systems and applications, pages 1470–1477. IEEE, 2017.

[14] J. Moreira, L. F. Pires, M. van Sinderen, L. Daniele, and M. Girod-Genet. *SAREF4health: Towards IoT standard-based ontology-driven cardiac e-health systems*. Applied ontology, 15(3):385–410, 2020.

[15] W. Jin and D. H. Kim. *Design and implementation of e-health system based on semantic sensor network using IETF YANG*. Sensors, 18(2):629, 2018.

[16] R. Reda, F. Piccinini, and A. Carbonaro. *Towards consistent data representation in the IoT healthcare landscape*. In Proceedings of the 2018 int. conf. on Digital Health, pages 5–10, 2018.

[17] C. Peng and P. Goswami. *Meaningful integration of data from heterogeneous health services and home environment based on ontology*. Sensors, 19(8):1747, 2019.

[18] S. Tiwari and A. Abraham. *Semantic assessment of smart healthcare ontology*. International Journal of Web Information Systems, 2020.

[19] H. B. Elhadj, F. Sallabi, A. Henaien, L. Chaari, K. Shuaib, and M. Al Thawadi. *DoCare: A dynamic ontology reasoning based healthcare monitoring system*. Future Generation Computer Systems, 118:417–431, 2021.

[20] T. Sondes, H. B. Elhadj, and L. Chaari. *An ontology-based healthcare monitoring system in the internet of things*. In 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pages 319–324. IEEE, 2019.

[21] F. Ongenae, P. Duysburgh, N. Sulmon, M. Verstraete, L. Bleumers, S. De Zutter, S. Verstichel, A. Ackaert, A. Jacobs, and F. De Turck. *An ontology co-design method for the co-creation of a continuous care ontology*. Applied Ontology, 9(1):27–64, 2014.

[22] J. Nelis, T. Verschueren, D. Verslype, and C. Develder. *Dyamand: dynamic, adaptive management of networks and devices*. In 37th Annual IEEE Conference on Local Computer Networks, pages 192–195. IEEE, 2012.

[23] B. Steenwinckel, B. Pieter, F. De Turck, and O. Femke. *INK: Knowledge graph representation for efficient and performant rule mining*. In preparation for: Semantic Web Journal, 2022.

[24] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. *A bayesian framework for learning rule sets for interpretable classification*. The Journal of Machine Learning Research, 18(1):2357–2393, 2017.

[25] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. *C-SPARQL: SPARQL for continuous querying*. In Proceedings of the 18th int. conf. on World wide web, pages 1061–1062, 2009.

[26] M. De Brouwer, F. Ongenae, P. Bonte, and F. De Turck. *Towards a cascading reasoning framework to support responsive ambient-intelligent healthcare interventions*. Sensors, 18(10):3514, 2018.

7

TALK: Tracking Activities by Linking Knowledge

The dataset, ontology and KG of Chapter 6 can be used in HAI application to detect and determine human activities in a homecare setting. Here, the sensor values of the smart home devices provide data that can be used to determine these human activities, but these sensor values have no meaning without the accompanying metadata of what they output or where they are placed. In this chapter, the time series observations of multiple sensors and the provided metadata are aggregated together in one KG. Events nodes inside this KG are used to represent the state of both the user and their household for a predefined amount of time. These events are also linked to each other such that the predecessor of an event is also an event that happened in the past. By creating this link, the original time series aspect is kept within this KG. An HAI activity detection algorithm is developed on top of this KG event representation to classify the events or nodes of interest to which activity a user is performing in a smart home environment. The interpretable KG embedding techniques of chapter 5 are fed to a traditional ML algorithm to create such an HAI detector. This chapter investigates research question 5: "Can time series and metadata efficiently be combined in a KG for HAI applications without losing explainability?" and validates hypothesis 5: "An HAI model relying on a KG linking time series data with domain knowledge is at least 10% more precise than single-model ML-based alternatives while maintaining explainability.".

⋆ ⋆ ⋆

B. Steenwinckel, M. De Brouwer, M. Stojchevska, F. De Turck, S. Van Hoecke, and F. Ongenae,

Submitted to Elsevier Journal of Engineering Applications of Artificial Intelligence, August 2022.

Abstract
Dependable and accurate monitoring of elderly at home becomes crucial to limit both the costs and human efforts of following up elderly for establishing a healthy care system. Human Activity Recognition (HAR) tools, based on sensors installed in smart homes, will become an important tool to provide useful information to the caregiver when something happens in the house of an elderly and care is required. The current available detection tools either exist out of interpretable knowledge-driven techniques or scalable data-driven ones. In this chapter, a hybrid methodology that combines both approaches is designed and evaluated to Track Activities by Linking Knowledge (TALK). Both sensor data and their link to the relevant domain knowledge about where those sensors are installed, the performed activities that occur, and how the household is constructed, are generalised in a specific knowledge graph (KG) structure to represent continuous events. The interpretable knowledge graph embedding technique INK is then used to transform these events inside the KG to a tabular format, which can be used by any traditional machine learning classifier to create a HAR tool. The TALK methodology is evaluated on two HAR datasets and shows (a) that TALK outperforms both traditional automated data-driven as well as knowledge-driven techniques in terms of predictive performance, and (b) how TALK can be easily used in a more out of lab environment. All these results and the interpretable aspects show that TALK can become an important tool to monitor elderly in their homes efficiently, effectively and with less intrusive techniques.

7.1 Introduction

With the current ageing population, our care needs are shifting from acute to chronic care where people are living longer with one or more chronic diseases [1]. Such a chronic disease requires more complex care, requiring an estimated increase between 3.6% and 4.4% of elderly requiring beds in residential care centres by 2030 in Europe [2].

To uphold the rather optimistic scenario of "only" 3.6% beds, care delivery should become more transmural and be facilitated at home and in service flats. By doing this, residential care can be reserved for those with severe care needs. Therefore, to maintain a sustainable healthcare model, the accessibility of homecare should increase from 5.8% of the european population to 8% in 2030 [2].

To facilitate this shift to homecare, dependable & accurate monitoring and follow-up of the elderly at home is crucial. Today, elderly are already increasingly equipped with Personal Alarm Systems (PAS) & monitoring devices (lifestyle monitoring, medical sensors, localization, etc.) [3]. These devices generate alarms that are forwarded to a call centre operator who is responsible for assessing the priority and context of the call and delegate it to appropriate caregivers. Whereas such generated alarms were previously efficiently handled in a hospital or nursing home due to direct access to

the patient, they lead to a number of problems in the context of homecare, such as the inability to quickly assess the priority and validity of the alarms [4]. As such, precious time is often lost trying to reach the elderly. In the cases that the elderly can be contacted through e.g phone, they are often unable to communicate their situation clearly. Also false alarms, which account for more than one-third of the calls, cause a huge amount of lost time for caregivers [5].

As more and more households are equipped with smart Internet of Things (IoT) sensors, the context of what is happening when a personal alarm is generated can now be captured and analysed automatically to provide better care [6]. To deliver objective information to both the operator and caregiver when such an alarm is generated, the available monitoring devices in an ambient living setting can provide useful insights through Human Activity Recognition (HAR) models that help to analyse the sensor signals without the need of a nurse or operator.

More concrete, an operator could be alarmed that a resident needs some care. Based on the HAR results the operator could identify that certain daily routines, such as eating breakfast and showering were not performed. This information can already indicate more specialised care will be required and that, in this case, the chance of a false alarm will be rather low.

However, the currently available HAR models either focus on the data generated by the monitoring devices or use so-called domain knowledge to derive the human activity [7]. A combination of both data- and knowledge-driven techniques are rather sparse and are mostly limited by advanced rule-based systems [8]. Moreover those combined approaches rarely take into account all domain related knowledge, to incorporate information about the sensor placements, the different rooms inside the house or the possible human activities that can occur inside those rooms. Combining both the available knowledge about a household and monitoring device together with the generated data could not only be used to learn the detection of human activities, they could also provide more explainable results towards the nurse and operators and let them verify whether the predictions of such a model can be trusted.

In this work, we present such a combined HAR model to Track Activities by Linking Knowledge (TALK). The TALK methodology transforms all the gathered data in the context of a smart house together with the available domain knowledge into a Knowledge Graph (KG). The data is grouped into events, which represent nodes within our KG. On those event nodes, data observations from different devices are linked together with the additional knowledge of, e.g., where those devices are placed within the house and what they are actually measuring. The KG embedding technique Instancce Neighboring using Knowledge (INK) is then used to generate interpretable KG embeddings for each of these events. The result of INK is later fed to a Machine Learning (ML) classifier to predict the corresponding human activity associated with these events. An evaluation of TALK is performed based on the Data Analytics for

Health and Connected Care (DAHCC) dataset[1], which contains gathered data of more than 5 different monitoring devices for 30 participants performing daily life activities in a home [9]. The obtained results show that the TALK methodology is indeed effective while still being interpretable. The contributions of the paper are therefore summarized as follows:

- We design and present the TALK approach that combines time series events and activity meta information together in a unified KG, which is ideally suited as input for ML methods.
- We designed a novel activity recognition technique based on hybrid AI, which combines both raw sensor data with metadata about the environment in one unified and generic approach.
- We show that by using our own interpretable KG embedding technique INK in the hybrid AI method, an activity recognition technique can be achieved that is interpretable and can thus deliver insights on why a particular activity was recognized by the AI based on all input in the KG.
- Based on both our own Open Dataset, as well as an external benchmark dataset, we showcase that the presented hybrid AI method outperforms the state-of-the-art activity recognition algorithms, both in terms of prediction accuracy, as well as in terms of interpretability of the results.

The remainder of this paper is structured as follows: Section 7.2 provides an overview of the relevant HAR studies and how they relate to the problem discussed above. The description of the TALK methodology on this DAHCC dataset is described in Section 7.3. Section 7.4 describes the open DAHCC ontology and datasets on which TALK is evaluated. Section 7.5 described the evaluation and obtained results. These results are discussed in Section 7.6. At last, future work and a conclusion is provided in Section 7.2.

7.2 Related work

HAR algorithms and models for smart-home environments can be classified in the area of pattern recognition. Two broad fields of research exist in literature [10]: data-driven and knowledge-driven approaches. On the one hand, data-driven approaches rely on gathered data from sensors and actuators about the behavior of the users to create an Artificial Intelligence (AI) model to recognize human activity. On the other hand, expert knowledge and common-sense rules are used in the knowledge-driven field. They use prior knowledge, the modelling information of the domain and logical reasoning to infer human activity. The following two subsections further elaborate on the state-of-the-art within both fields and provide their advantages and drawbacks.

[1] https://dahcc.idlab.ugent.be

7.2.1 Data-driven HAR

Data-driven HAR models are differentiated between their generative and discriminative capabilities [11]. Generative models use probabilistic analysis models such as Markov models and Bayesian networks to define the activity input or data space. Such a generative model takes into account the inhabitant's preferences and tunes the models according to this information. The drawback of this approach is its rather static nature, non-evolving and tailored to the provided data. In contrast, the discriminative approach maps the obtained inputs to the activity outputs, usually provided as ground-truth labels by the users, e.g. by annotating activities or analysing video images of the user's activities. Machine Learning (ML) is such a discriminative approach in this field. Within ML, as well as in data-driven HAR, both supervised and unsupervised learning methods exist.

In previous research, decision trees [12], conditional random fields [13], support vector machines [14], naive bayes classifiers [15] and Multi-Layer Perceptrons [16] are used to detect and classify human activities. While some models outperform others, the specific use case setting or the difference in amount of gathered data to train the models make it difficult to define a clear winning prediction model for HAR.

All data-driven HAR model have the advantage of probabilistic modelling. It can handle uncertainty or provide a probabilistic outcome for all learned activities when a new observation or set of observations needs to be analysed [7]. Such ML models can also handle noisy, uncertain and incomplete data. To learn these models, no upfront domain knowledge is required.

The drawback of all these data-driven HAR techniques is that both the generative and discriminative approach requires a large amount of data. The need for data is also reflected in the cold-start problem of these methods. A large amount of data should be available upfront to learn and train the models before predictions can be made or adapted to a more personalised setting. In the case of a supervised training approach, even a large amount of clean and correctly labelled data is needed. Another problem with data-driven HAR approaches is that they are explicitly tailored to the given dataset and domain [17]. Therefore, new models and even new data collection campaigns are needed when HAR has to be performed in a new environment, with different sensors and with different activity labels.

Another drawback of this field is the less interpretable predictions generated by a data-driven model. Most of the time, an operator still has to correlate in many cases the sensor values and interpret the results to understand why a certain prediction was made.

7.2.2 Knowledge-driven HAR

Knowledge-driven HAR methods exploit the activity and sensor knowledge modelling and use logical reasoning to perform activity recognition. The general procedure of a

knowledge-driven approach can be summarised in 3 steps [17]:

1. Explicitly define and describe all possible activities within the domain using a knowledge representation formalism.

2. Aggregate and transform the sensor data into logical, interpretable terms and formulas.

3. Perform logical reasoning to extract a minimal set of rules (models) which could explain the activities based on a set of observations.

The knowledge structure is modelled and represented through, e.g., schemes, rules, or networks. Knowledge-driven HAR is further divided in three sub-approaches: mining knowledge from web resources, where textual descriptions of human activities are translated into concepts and actions that can be processed by an inference engine [18], logic-based approaches [19], and, the more recently adopted, ontology-based approaches. A well-known logic based HAR approach is finite automate or finite state machines [20]. In this technique, activities are defined as states and rules are constructed to go from one state to another. These state transitions depend upon the provided input symbols, such as discrete sensor values. Finite automata are especially tailored to a specific task and context. When the context of the task changes, a new automaton has to be designed by a human expert to make it adaptable to this new case. The ontology-based approaches do not depend on algorithmic choices and are, therefore, preferred over the other methods in the last decade. Hooda et al. [21] proposed a an overview of ontology-based HAR and also constructed sensor and activity ontologies for explicit domain modelling to infer human activities. Ontological representations use assertion axioms learned from data or defined by the user to make these inferences of the activities [22].

Knowledge-driven techniques have the advantages to represent and model the activities as most complete as possible to overcome the activity diversity and provide an explanation why a certain prediction was made. However, the limitations of these approaches are the complete domain knowledge requirements to build activities models and the weakness in handling uncertainty and adaptability to changes and new settings or activities [17]. They need domain experts to design knowledge and rules and new rules can break or bypass the previous rules.

7.2.3 The need for a hybrid approach

While both separate approaches have their shortcomings, both the knowledge-driven and data-driven HAR solution can also be combined to resolve multiple of the above-mentioned issues and obtain better, interpretable results.

First steps were already taken to incorporate data-driven learning capabilities into knowledge-driven approaches to address the aforementioned problems of activity

modelling [17]. The process consists of three key phases. In the first phase the initial knowledge-driven models are created through ontological engineering by leveraging domain knowledge and heuristics. This solves the so-called cold-start when not enough data is available to create data-driven detectors. The ontological engineering method can now be applied on a small amount of data, and can be seen as a new automatic procedure to get more reliable labels for a data-driven model. The usage of user-feedback can help to correct and adapt faulty or missed predictions in this case. In the third phase, the classification results from the second phase are analysed to discover new activities and create data-driven HAR models. These new learnt activity patterns are in turn used to update and extend the knowledge-driven models. Once the first phase completes, the remaining two-phase process can iterate many rounds to incrementally evolve the models, leading to a complete, accurate and up-to-date HAR. While this form of a hybrid approach overcomes all shortcomings, it also implies multiple systems have to be designed to work together. This hybrid AI architecture has already been efficiently implemented in a predictive maintenance domain [23] and is translated to a HAR setting. In these HAR cases, either ontological activity concepts are used to fix inconsistencies in the outcome of a ML classifier [24] or a knowledge-driven reasoning step is performed to detect a first set of activities, which can later improve this initial knowledge-driven activity model [25]. Most of these techniques are dependent on the environmental context and in many cases, two or more models have to be maintained when applied in a real-time, streaming context.

The recent advances in knowledge engineering offers also the possibility for a new type of hybrid approach using a KG. Here, both the sensor data and contextual metadata are combined in one graph, which links the domain knowledge with the sensor or input observations. When all information is available, so-called KG embeddings can be used to transform the more graphical representation of all the data into a representation that can be used as input in a ML model [26, 27]. When the embedding procedure can be guaranteed to generate interpretable embeddings, the outcome of the generated models can also provide interpretable predictions. This combination of incorporating both the sensor data while providing interpretable results is crucial to let these HAR models operate in a healthcare setting. Techniques exist which can also take into account a KG as input [28, 29]. But to our knowledge, we are the first to evaluate and propose a hybrid approach for HAR, which takes a KG as input and is still able to provide interpretable results that have not been reported upon before. Here, less individual knowledge- and data-driven systems have to be designed and combined to generate a new solution.

7.3 TALK methodology

The TALK hybrid approach presented in this paper consists of 3 main steps. First, the sensor data, activity information and existing contextual information must be com-

bined in one data structure. To link all this information together, a KG is being used, backed by an ontology to clearly define the relationship between the activities and the sensor data. Second, we create KG embeddings for those nodes of interest which hold activity information. At last, these node embeddings are fed to an ML classifier together with the corresponding labelled information to train and make activity predictions. An overview of this approach is visualised in Figure 7.1. This section further describes these three steps in detail.

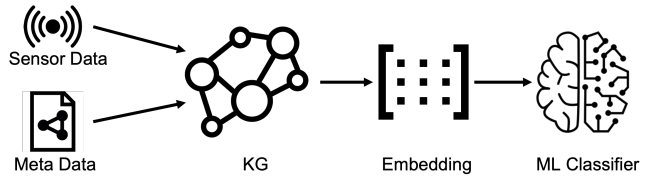


Figure 7.1: Overview of the TALK approach to create a Hybrid AI HAR detection tool.

7.3.1 TALK KG

The KG structure used within the TALK methodology had two requirements:

- Data and metadata should be linked together such that relevant information regarding a performed activity can be found in a limited number of hops.
- As activities have a temporal aspect, the KG should also keep such a temporal structure. It should be possible to hop from the current obtained information to the previously seen data.

Figure 7.2 shows how the TALK KG is designed to meet those requirements. Event nodes are generated which aggregate observations for a certain amount of time (x seconds, x minutes, ... depending on the use case at hand). To this event node, observations can be linked. Instead of linking the observations directly to this event node, additional sub nodes are created to aggregate those observations related to the same concept together. Here, in the domain of lifestyle and activity detection, both the rooms inside the residents' house and the residents themselves can be in a certain state for a certain amount of time. The sensor observations in these rooms or the observations from the wearable devices attached to the resident are linked to these states when they occurred during the time this particular state was captured (e.g. when they occurred within the time range of the state). This link relationship is based on which sensor created the observation, together with the provided meta information of where this sensor is installed (e.g., which room, attached to which user etc.).

In a less abstract sense, the TALK approach will group the sensor observations based on both a certain time interval (state) and the location where this sensor orig-

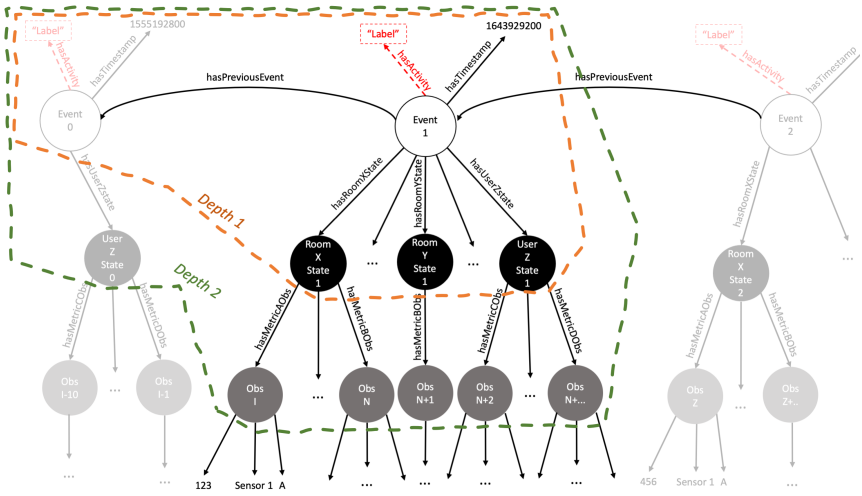


Figure 7.2: Representation of the KG within TALK. Sensor observations are linked to events using contextual state nodes. The event nodes are linked to each other using the "hasPreviousEvent" relationship. The neighborhood of the Event 1 node for depth 1 (orange) and depth 2 (green) is also highlighted in this Figure.

inated from. One of these locations can be the body of the user (e.g. for wearable devices).

Events are also linked to each other using the "hasPreviousEvent" relationship to enable efficiently hopping to an event back in time. Events that belong to a certain performed activity can also incorporate the "hasActivity" label information.

7.3.2 TALK INK embedding

The KG combines efficiently both the data and metadata of a performed activity. Traditional ML models are unable to deal directly with graph-based input. As such, if such a ML model wants to detect activities based on this information automatically, the KG should be represented as a vector. A large amount of so-called KG embedding techniques exist, which transform the whole KG or particular nodes within a KG to such a vector representation [27], such as TransE and RDF2Vec. All those techniques however have the drawback of transforming the interpretable KG into uninterpretable embeddings, which result in predictions being made with a ML classifier which are hard to relate back to the originally provided information. Moreover, the transformation always leads to a loss of information. Techniques exist, i.e. graph neural networks, which directly take the KG as input and make use of Deep Learning (DL) to implicitly learn an embedding and simultaneously accomplish the classification task [30]. However, these techniques do not scale towards large graphs, and whenever the KG changes (e.g. new nodes or edges being added), a new model has

to be trained. These techniques require a large amount of data to be trained properly. Therefore, we designed a novel embedding technique called INK [31], which is optimal for usage within TALK as INK embeds the KG in an interpretable 2D matrix and is not dependent upon the ML model that takes this 2D matrix as input. To generate such a 2D matrix, INK queries the neighborhood of a node of interest and transforms the information within this neighborhood into features. As an example, INK will embed the "Event 1" node in Figure 7.2 as follows. In a first step, INK gathers the neighborhood of this event node. A neighborhood of a certain node is defined by all the nodes that can be reached starting from the node of interest (here the "Event 1" node). To gather those nodes, INK traverses paths following the direction of the edges starting from the node of interest towards all nodes that can be reached. As this neighborhood can be very large, we usually limit the search depth by a parameter value. This neighborhood depth indicates the number of edges that can be taken starting from the node of interest towards the nodes within the neighborhood. In our example, a neighborhood depth of 1 will contain the nearby nodes of our "Event 1" node that can be reached following the connected outgoing relationship edges. This is shown in Figure 7.2, where the neighborhood of Event 1 at depth 1 is surrounded in orange. These are all the room and user state nodes, the timestamp, the activity label and the previous "Event 0" node.

After INK acquires the neighborhood of the node of interest, it transforms the relevant information in this neighborhood into a dictionary format. The dictionary key is defined by the edge relationship. The value is the list of nodes related to this relationship as a relationship can occur multiple times starting from a node of interest reaching different nodes (e.g. multiple room X states linked to an event node). In our example a hasRoomXState→[RoomXState1] key-value pair will be available in this dictionary, together with all other pairs found at neighborhood depth 1 as shown in Table 7.1. When creating these key-value pairs for a neighborhood depth larger than

Table 7.1: Dictionary representation created by INK for the Event 1 node in Figure 7.2.

Key	Value
hasRoomXState	[RoomXState1]
hasRoomYState	[RoomYState1]
hasUserZState	[User2State1]
..	..
hasRoomXState.hasMetricAObs	[Obs1]
hasRoomXState.hasMetricBObs	[ObsN]
..	..

1, INK concatenates the relationship edges together and neglects the intermediate nodes as this information is made available within our dictionary when creating key-value pairs at a lower depth. INK would create the following dictionary entry for a neighborhood depth 2: hasRoomXState.hasMetricAobs→[Obs1]. In our example

Figure 7.2, the neighborhood depth 2 is visualized in green. One can see that a minimal depth parameter of 3 is required to capture the sensor observation values (3 edges have to be traversed to reach this sensor information). If the sensor values of the previous event are also of interest, a depth parameter value of 4 is required.

A neighborhood dictionary is made for every node that is of interest. In our example, INK would create this dictionary for every event node for which an activity label is provided. To transform all these dictionaries in a 2D matrix, we take as an index the according node of interest and create column features by the concatenation of the relationship key and the value in the list according to this key within the dictionary. Both the keys and a combination of keys and values are provided in this 2D matrix. The creation of the key-value combination is repeated for every value within the dictionary value list. An example of such a 2D Matrix for our example is provided in Table 7.2. In our example of the "Event 1" node, this specific event node is defined as an index entry, and hasRoomXState$RoomXState1 is a generated column feature from the "Event 1" dictionary. The "$" sign is used as concatenation character, and indicates where the relationship string ends. To indicate whether this feature can be found within our index node of interest, we provide a binary indicator in the according cell.

Table 7.2: Example of a depth 3 INK two dimensional representation for the three event nodes in example Figure 7.2. INK can both combine real values with binary indicators to indicate the relational information when available.

	hasRoomXState	hasRoomXState$RoomXState1	hasRoomXState.hasMetricAObs.hasValue	...	hasTimestamp
Event 0	0	0	Nan	...	Value
Event 1	1	1	123	...	Value
Event 2	1	0	456	...	Value

When more and more nodes of interest transform their dictionaries within this 2D matrix representation, the more similar information that can be found in these neighborhoods will be mapped on the same feature columns. This is visualized in Table 7.2 where an example 2D matrix representation is shown for the three event nodes in our example of Figure 7.2. The nodes "Event 1" and "Event 2" both have "hasRoomXState" information as shown in Figure 7.2 and the first column of Table 2 while the "Event 0" node doesn't provide this information.

INK has the option to neglect certain relationships, such that this information is not being used during the creation of the INK embedding. In the context of HAR, the "hasActivity" relationship was neglected by INK such that the labeled information was not incorporated in the embedding itself as this would introduce a label leakage during the training and evaluation process of a ML classifier. INK also has the ability to avoid transforming numerical values into separated columns. In the third column of Table 7.2, we see for our example nodes that their raw sensor values are not transformed into separated binary column indicators, but that they are provided as is.

7.3.3 TALK classifier

The INK embedding can be seen as a traditional feature matrix, where for each event node, features are constructed which hold both sensor and contextual information. The HAR labels accompanied with these events can be queried from the original KG based on the event's unique identifier. This combination of a feature set and an according label set can be provided to any supervised ML classifier.

7.4 DAHCC Ontology and Datasets

To provide a link between sensors and observations together with the human activities being predicted by an AI model, the Data Analytics for Health and Connected Care (DAHCC) ontology [9] is used to describe this.

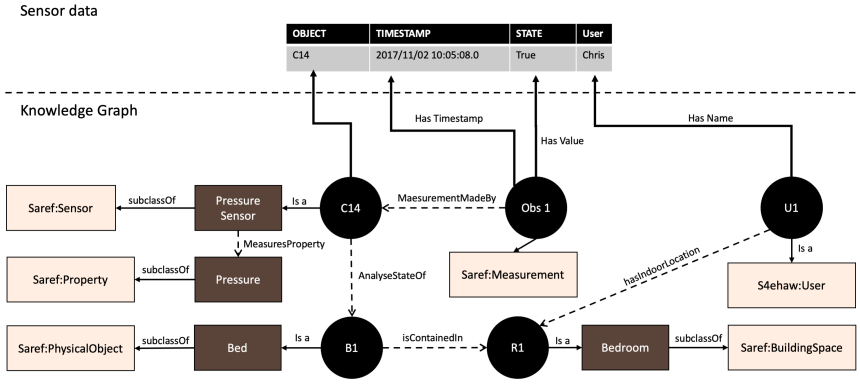


Figure 7.3: Semantic enrichment of a sensor observation using the DAHCC components. Additional domain knowledge about the use case can also be linked. In this example the sensor data of a pressure sensor, measuring the pressure of a bed inside a bedroom is being enriched. The user responsible for these sensor values is also mapped within this sub graph. Black round circles represent the instantiated nodes in our KG. All squared boxes represent ontological concepts either from the DAHCC ontology or from external ontologies).

The DAHCC ontology consists of 4 sub ontologies, ranging from human activities to sensor observations for both wearable and ambient living. These ontologies are based upon the SAREF standards to describe sensors and their observations, buildings and physical objects as well as how these concepts relate to health actors and patients. The DAHCC ontology also describes the concepts related to ML models based on the Execution-Executor-Procedure (EEP) ontology. An example of how the observation data of a sensor can be enriched with this ontology is shown in Figure 7.3. The data of a single sample is mapped to an observation node in our KG and this node is linked to the corresponding sensor responsible for generating such observations. The sensor itself analyses the state of a certain object, which is located at a certain location (in the

example of Figure 7.3, a pressure sensor analyses the state of the bed, which is located in the bedroom. This bedroom can be located at a certain floor in a certain house). Similarly, we can define the user in our KG and define e.g. its indoor location.

The semantically enriched observation using the DAHCC ontology holds enough information to transform the data and metadata into the TALK KG as described in Section 7.3.

To evaluate the TALK methodology and to show the advantage of combining data and metadata together in one KG, we used two HAR life style datasets:

- UCAmI Cup dataset [32]: A HAR dataset to track activities of daily living generated in the UJAmI Smart Lab, University of Jaén. The dataset was chosen for the first edition of the UCAmI Cup and represents 246 activities performed over a period of ten days carried out by a single inhabitant. The dataset includes four data sources: (i) event streams from 30 binary sensors, (ii) intelligent floor location data, (iii) proximity data between a smart watch worn by the inhabitant and 15 Bluetooth Low Energy beacons, and (iv) acceleration of the smart watch. Activity labels were provided for every 30 seconds. An overview of this dataset is provided in Table 7.3. As this dataset was also part of a competition, a clear train-test split was also provided. The UCAmI Cup dataset was semantically enriched using the DAHCC concepts in order to evaluate the TALK methodology for this paper. This UCAmI TALK KG is also made available in our repository[2]

Table 7.3: Summary overview of UCAmI cup dataset.

Source	Raw Data	Details	Description
Acceleration	X, Y and Z axis	acceleration of inhabitant measured at 32hz	
Intelligent floor	Boolean contact	Indoor location tiles	Location is 2D space
Proximity	Object, RSSI	Book, TV controller, Door entrance, Medicine box, Cupboards, Fridge, Garbage can, Wardrobe, Drawer, Tap, Toothbrush, Laundry basket	Location of inhabitant near these objects
Binary Sensors	Object, State	Door open, TV, Motion sensors, Dishwasher, Drawer state, Water boiler, Microwave, Tap, Tank, Bed, Kitchen faucet, Sofa pressure	Usage of objects
Activity	Category + label	Shower, Brush Teeth, Use toilet, Get dressed, Take medicine, Dinner, Lunch, Breakfast, Take snack, Prepare breakfast, Prepare dinner, Prepare Lunch, Go home, Leave home, Visit lab, Sleep, Relax on sofa, Play videogame, Read book, Watch TV, Work at table, Do dishes, Put washing machine on, Take out trash, Throw waste in bin	Activities performed by a single user

- DAHCC dataset [9][3]: Ambient living situation where a lot of non-invasive sensors are installed on two floors at the HomeLab of imec. 30 different participants performed daily life activities and sensor data from various sources was

[2]https://github.com/predict-idlab/TALK/tree/main/UCAmI
[3]https://dahcc.idlab.ugent.be/dataset.html

captured. Participants were also equipped with smartphone and wearables to analyse their smartphone usage, indoor location and some biomedical parameters, e.g. skin conductance and heart rate variability. An overview of this dataset is given in Table 7.4. Together with this dataset, all metadata related to the imec HomeLab, the sensor installations and performed activities are semantically enriched using the DAHCC ontology. This DAHCC TALK KG is also made available in our repository[4]. Labelled activities were provided by the participant using a smartphone application. They indicated the start and stop times every time a human activity was performed. The average number of activities registered per participant is 70.7.

Table 7.4: Summary overview of DAHCC dataset.

Source	Raw Data	Details	Description
Wearable	X, Y and Z axis Acceleration X, Y and Z axis Gyroscope Blood Volume Pulse (BVP) Galvanic Skin Response (GSR) Skin temperature	Inhabitant specific parameters	Empatica E4 was used as wearable device
Netatmo	Various values within a specific room	Rooms: Kitchen, Master bedroom, Bathroom, Toilet	Room temperature, Room CO2, Room humidity, Room loudness
EnOcean	Object state	Door contact sensor, cabinet contact sensor	Measure open/close state of doors/drawers/cabinets
Steinel	People presence People count	Rooms: Living room, kitchen, hallways, master bedroom	Detects and counts the number of people within a certain room
Velbus	Various values within a specific room	Available in all rooms	Measures the energy consumption of each wall socket, the energy consumption of the major appliances, indoor temperature within a room, state of the windows, state of the blinds, state of the lights, state of the motion detectors
Aquara	Location	Proximity based indoor localisation detection	Indoor localisation system of Televicc Healthcare
Activity	Label	RoomTransition, Toileting, Organizing,Working, WashingHands,DrinkPreparation, WatchingTVActively, UsingMobilePhone, PreparingMeal,EatingMeal, GettingDressed, UsingComputer, BrushingTeeth, DoorWalkThrough, Sleeping, WakingUp,Serving, ObjectUse, SocialInteraction, GettingReadyToSleep, Walking,Drinking, Showering, ShavingBrushingHair, TakingMedication, SocialMedia, EatingSnack, PreparingSnacks, Dishwashing, Exercising, Wandering, Cleaning, Cosmetics	Activities performed by a 42 users

Although both datasets contain different sensors and different household layouts, the obtained TALK KGs are quite similar to each other. Both the DAHCC TALK KG and UCAmI Cup TALK KG describe observations related to the state of an appliance/physical object within a room or building space of the smart labs.

7.5 Evaluation and Results

For both semantically enriched datasets, INK embeddings were generated for all nodes containing an associated activity label. The labels were excluded from the KGs when creating the embeddings to avoid labelled information getting incorporated. For the UCAmI Cup dataset, event nodes were embedded for every 30 seconds, as the labelled information was originally provided for every 30 seconds. The DAHCC dataset

[4]https://github.com/predict-idlab/TALK/tree/main/DAHCC

didn't have activity labels being partitioned every x seconds. Therefore, events are created every 30 seconds, and we compare the activity begin and end timestamp to assign the corresponding label(s).

Only a single activity at a time was performed during the UCAmI Cup dataset. In the DAHCC dataset, multiple activities can occur at the same time event (e.g. eating a meal while watching TV). Analyses were performed combining these activities together (e.g. eatingMealWhileWatchingTv).. However, this resulted in too sparse labels and training a model on these sparse labels created a non generalizable solution. Therefore, only the most dominant activity, which was the activity which occurs the most in the overall dataset, was kept (here eating meals). As some activities were only performed by a single participant or by a small group of participants, only activities occurring more than one hour in total, over all participants, in the dataset (which means for labels provided every 30 seconds, that a specific label should occur more than 120 times in the dataset to be considered). This was done to ensure enough labelled events could be provided during the training phase for each activity group. The activities who did not meet these criteria were labelled in one, general class: "Other". In total, an evaluation on 11 activities was performed: DrinkPreparation, Eating, Organizing, PreparingMeal, Showering, Toileting, UsingMobilePhone, Walking, WatchingTVActively, Working and Other.

For the UCAmI Cup TALK KG and DAHCC TALK KG, INK embeddings till depth 11 were generated. As the events in both KGs are obtained for every 30 seconds, the events of interest in both datasets take into account all the past events in the last 5 minutes. This means that the ML model trained upon these INK embeddings will have to decide which activity is performed based on the last 5 minutes of available data. To analyse the influence of taking into account previous events, a comparison was made using INK embeddings till depth 3 (so, without taking into account previous events) from the UCAmI Cup TALK KG

A clear training and test set was provided for the UCAmI Cup dataset. The train set contained 7 days of continuous sensor data of one person and according labelled activities. The test set contained 3 days of sensor data from the same person, obtained directly after the 7 days in the training set. the TALK approach is evaluated according to this provided split. The generated INK embeddings were provided to an Extra-tree classifier with 1000 estimators. This classifier was chosen based on previous experiments of INK on defined benchmark datasets [31]. Class weights were calculated based on the labels in the training set using the following formula to cover the imbalance in the dataset:

$$\frac{\text{number of samples in training set}}{\text{number of classes} * \text{Count of number of occurrences of each label}}.$$

The DAHCC dataset did not contain such a predefined split and also had a lot more samples and activities to predict. A participant leave-one-out cross validation evaluation was performed to show the benefits of TALK to predict activities for an unseen

DAHCC participant. The generated INK embeddings were provided to an Multiclass Catboost model as more categorical data was provided in this dataset. To avoid overfitting, the Catboost number of iterations are evaluated against a validation set. This validation set is created using a group shuffle split on the original train samples. Again class weights were provided to cover the imbalance in the dataset following the same formula described above.

All evaluations were performed on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz processor with 32 cores and 128gb RAM. For both evaluations, results are provided in the form of the accuracy metric, the weighted F1 score and confusion matrices. All experiment code was made available on our repository[5].

7.5.1 UCAmI Cup results

As originally indicated by UCAmI Cup competition, the accuracy and F1 results were measured on the hold-out test set are provided in Table 7.5. A test was performed for both INK embeddings at depth 3 and depth 11.

Table 7.5: TALK accuracy and weighted F1 score results for the UCAmI cup test set

Method	Accuracy	Weighted F1 score
TALK depth 3 with Extra-tree classifier	61.54%	0.6749
TALK depth 11 with Extra-tree classifier	76.44%	0.7744

The normalised confusion matrix for each predicted activity in the test set using the INK embeddings at depth 11 is shown in Figure 7.4

Our classifier has difficulties to predict when a visitor is at the door of the lab. This activity is confused with entering the lab as both actions are closely related to each other. The model also has difficulties distinguishing breakfast from preparing breakfast and waking up. Other activities which were difficult to classify are putting waste in the bin and washing dishes.

7.5.2 DAHCC results

All leave-one-user-out obtained predictions are averaged together for the DAHCC dataset and are visualised in Table 7.6. This summary table shows the precision, recall and F1-score for each predicted class that occurred more than 120 times in the dataset as described above. The total level indicates how many of these labels could be found in the dataset. The total accuracy score is calculated based on the following formula:

$$\text{accuracy} = \frac{\sum_C \frac{\text{True Positive C}+\text{True Negative C}}{TotalC}}{\text{Amount of classes}}$$

[5]https://github.com/predict-idlab/TALK

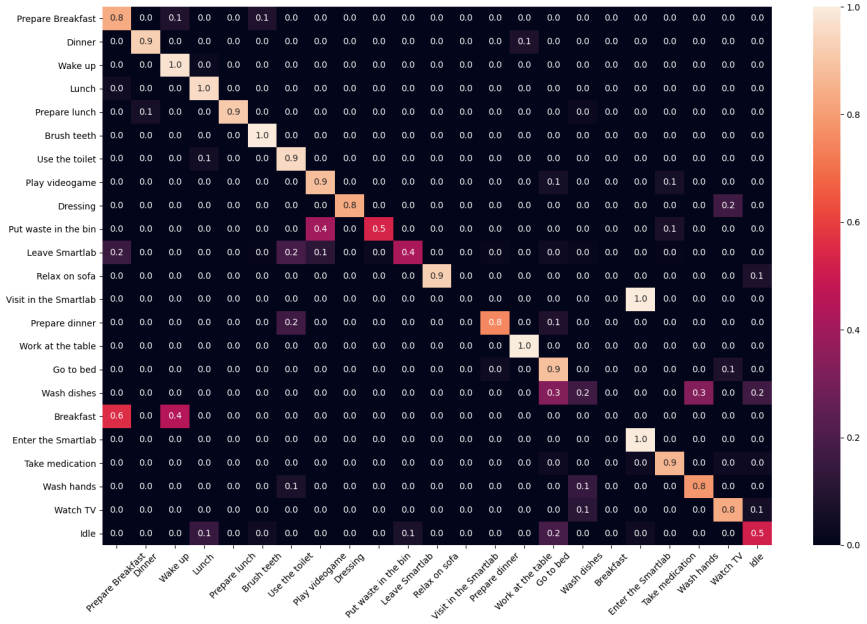


Figure 7.4: Normalised confusion matrix for all test set predictions of the UCAml dataset, using the TALK approach.

With C one of the 11 classes. The true positives and negatives can be calculated based on the precision ($\frac{\text{True Positives}}{\text{True Positives+False Positives}}$), the recall ($\frac{\text{True Positives}}{\text{True Positives+False Negatives}}$) and the fact that Total amount of samples per class = True Positives + True Negatives + False Positives + False Negatives. The Macro average score of the precision, recall and F1-score can be calculated by the sum of all individual class results divided by the amount of classes. The weighted average is calculated similarly, but it multiplies the individual scores by the portion of actual occurrences of the class in the dataset before summing all these results and dividing it by the total number of classes.

The normalised confusion matrix for each predicted activity in the test set is shown in Figure 7.5

7.6 Discussion

In this section, both the predictive performance of the TALK methodology and its interpretability are discussed.

Table 7.6: Summary overview of the leave-one-user-out DAHCC evaluation. Precision, recall, F1-score and total values are provided for both individual classes, as accuracy and the macro and weighted averages for the whole evaluation set.

	precision	recall	F1-score	Total
DrinkPreparation	0.15	0.45	0.23	363
Eating	0.37	0.44	0.40	2428
Organizing	0.39	0.39	0.39	1247
Other	0.61	0.43	0.50	2892
PreparingMeal	0.83	0.71	0.77	2026
Showering	0.71	0.81	0.76	454
Toileting	0.64	0.78	0.70	685
UsingMobilePhone	0.30	0.40	0.34	753
Walking	0.58	0.85	0.69	1039
WatchingTVActively	0.56	0.60	0.58	1013
Working	0.86	0.78	0.82	11238
accuracy			0.66	24138
macro avg	0.54	0.60	0.56	24138
weighted avg	0.69	0.66	0.67	24138

7.6.1 TALK compared to other approaches

By evaluating TALK on the UCAmI Cup dataset, we are able to compare the obtained results of Table 7.5 to other solutions generated in the past. Table 7.7 shows the predictive performance of TALK against previous UCAmI Cup competitors. These results show that our TALK approach outperformed all traditional ML models (e.g., Random Forests, Neural Networks and Naive Bayes classifiers). It also performed better than the Multi-input Temporal ensemble, which is a Deep Learning (DL) technique that fuses several sensor inputs together and makes predictions for a large number of windows (here 30s, 15s, 10s, 6s and 5s windows). Predictions for each of these windows are later on combined to decide which activity happened in the last 30s. The different results in Table 7.5 also show the influence of using the information of previous events in this evaluation. The results using INK embeddings at depth 11 are significantly higher than when using the INK embeddings without incorporating past events (at depth 3).

Our model does perform worse than the Finite Automata model. However, this approach is especially designed to work with the given competition data. The Finite Automata approach is tailored to the tasks and context (e.g. the smart lab), making them not directly adaptable towards other use cases. The evaluation of new data by this approach has to be performed offline, which makes it hard to make these automata operational in a real-time setting. Finite Automata also takes into account

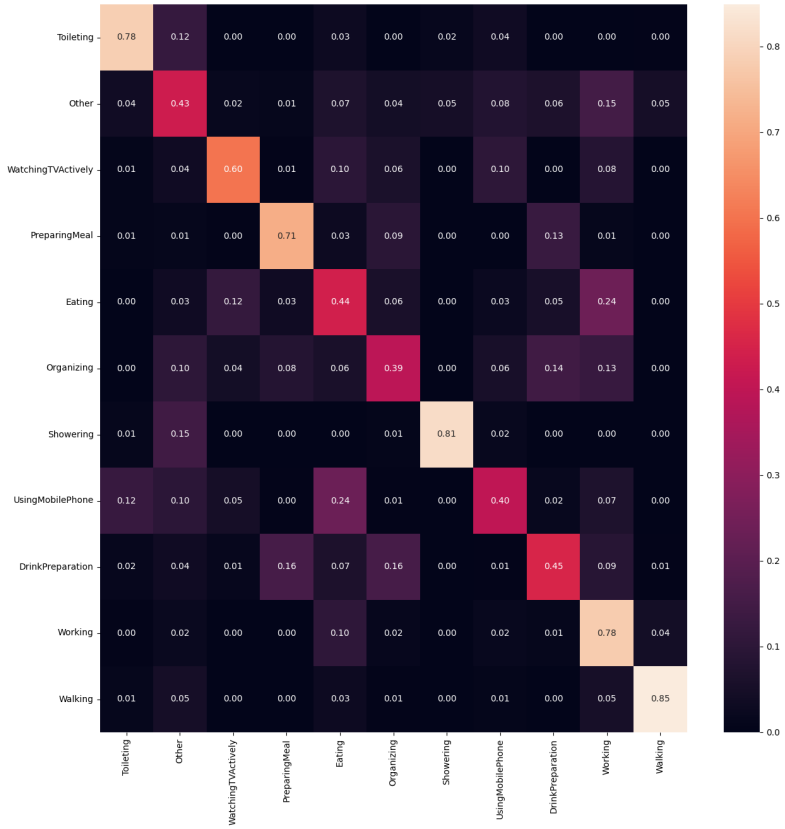


Figure 7.5: Normalised confusion matrix for all leave-one-participant-out evaluation of the DAHCC dataset, using the TALK approach.

the previously performed activity and uses probabilistic reasoning to determine which activity comes next. Our TALK approach does not take into account these previously performed activities.

Other data-driven research exists that achieves more comparable results as our TALK approach, but in these approaches the original UCAmI Cup activity labels were modified (some labels were aggregated together to boost the performance and making it a more easy classification problem) [33]. During our evaluations of the used models in Table 7.7 the original UCAmI Cup dataset was used as is, without any modification to compare with the created competition models.

TALK can be used in different scenarios as shown in the DAHCC evaluation. Both DAHCC and UCAmI Cup evaluations are, however, hard to compare to each other. The UCAmI Cup tries to make predictions for the next couple of days, for a single user, while the DAHCC evaluates one day of lifestyle activities for a new,

Table 7.7: Summary of the results obtained by other UCAmI cup participants.

Method	Accuracy
Markov Model + NN [34]	45%
Random Forest [35]	47%
Neural Network [36]	60.10%
Naive Bayes Classifier [15]	60.50%
Multi-input Temporal Ensemble [37]	73%
TALK (with INK depth 11 embeddings)	**76.44%**
Finite Automata [20]	90.65%

unseen participant.

In Table 7.6 and within the confusion matrix of Figure 7.5, most DAHCC activities were also predicted correctly by our TALK approach. However, some activities have a rather low prediction outcome. As the DAHCC dataset is captured in a free living environment, giving an accurate representation of real life activities, it can happen that different activities are performed in similar conditions. This is clearly the case for the activities: "Working" and "Eating", which were, in the context of the DAHCC dataset, occurred in the same place and as almost all participants just took their lunch while working. Also more general activities like "Organizing" can be performed at any time in every room, and therefore conflicts with many other performed activities. In the context of our use case regarding enriching the personal call systems of elderly, the most important activities like going to the toilet, preparing meals, showering and going out for a walk can be detected by the TALK approach and will deliver useful information to the operator which has to decide the appropriate action.

As stated in the description of the evaluation setup (Section 7.5), one general class "Other" was created to combine all labels that do not occur more than 120 times in the DAHCC dataset. This set of "Other" activities is quite diverse, and in combination with the ML classifier. which takes into account the class weights, the results of this class are rather low. More of these event samples will probably improve the "Other"'s class predictability. One could evaluate this whole setup without taking into account any of these activities that occur less than 120 times (removing them instead of relabelling them to one class). This would, however, reduce the applicability of such a model in a real-life, streaming context where these lower activities do occur and will then be mapped on one of the provided classes. By creating the "Other" class, we do already have the possibility to see the model's performance in those cases.

7.6.2 TALK's Interpretability

The TALK approach uses the INK embedding to represent the obtained KG into a tabular format. A wide range of KG embedding techniques however exist. In the evaluations of Section 7.5, INK already showed that it can handle both categorical data

(in the format of binary vectors) as well numerical values. These numerical values frequently occur in the context of sensor observations, which justifies the usage of INK in this context.

Table 7.8: INK task-specific rule mining precision and recall results on the DAHCC dataset.

Rule	Prec.	Rec.
Prev.Prev.phone.MagnetometerX.MinValue < -541.9 and Prev.Prev.Kitchen.humidity.MaxValue <= 62.5 and Living.Tv§off and Prev.Prev.Wearable.AccelerationZ.MeanValue > 31.62 =>Working	0.89	0.74
Prev.Prev.phone.GravityY.MaxValue <= 0.0017 and Prev.phone.LocationLatitude.MaxValue > 51.012 and Prev.Kitchen.Peopledetected.MeanValue <= 0.84 and Kitchen.EnvironmentWaterrunning and floorKitchen.Loudness.MaxValue > 44.5 =>EatingMeal	0.67	0.27
Prev.Localisation.location§living and Prev.phone.AccelerationY.MeanValue > -6.59 and Kitchen.Window§closed and Living.Tv§on =>WatchingTVActively	0.86	0.68
Prev.Living.PeoplePresence.MinValue <= 0.5 and Prev.Toilet.Light.MinValue <= 988.5 and Toilet.Light.MeanValue > 494.25 =>Toileting	1.0	0.65

INK also keeps a level of interpretability, similar to the interpretability levels of the original KG. The created INK column features still have a human interpretable aspect and can be analysed to see which features, or nodes and edges within our original KG had an effect during the classification of events. To show this benefit, besides the INK representation, the INK implementation also contains semantic rule mining modules[6] and is able to mine task-specific rules given a set of positive and negative samples [38]. An experiment was performed where for each of the 12 selected classes in the DAHCC dataset, a task-specific semantic rule miner was trained using INK. As a positive set, we used all positive samples for one class, while all other samples not from this class were used as negative evidence. A summary of the some found rules in combination with their predictive performance is provided in Table 7.8. They Show that several values regarding the phone, humidity level in the kitchen and the current off state of the television have a high impact on the fact that someone is working or not. Also the fact that water is being taken from the kitchen faucet and the loudness

[6]https://github.com/IBCNServices/INK/tree/master/ink/miner

value increases in the kitchen indicates whether or not someone is eating a meal. The last two rules indicate whether a person is watching TV or going to the toilet. For the last rule, one can see that the fact that the toilet light changes in a previous event regarding the current event is a crucial aspect in the detection of this particular activity.

The whole approach shows that the used TALK approach in combination with INK can create an interpretable tool to track activities in a smart home environment.

7.7 Conclusion

In this work, TALK, a new hybrid AI approach to track human activities using linked knowledge is proposed and evaluated in detail. The results showed that both a high predictive performance and the ability to adapt to different use cases within this domain can be delivered by this new methodology. The TALK approach is competitive with knowledge-driven approaches by providing interpretable outcomes in the form of simple interpretable rules. While still can incorporate new information and learn from those cases such as the data-driven variants.

As future work directions we see additional resources and even made predictions to be linked back to the TALK KG to provide even more information to embed. The TALK approach could take the previous predictions into account by adding an additional relationship to each event. The INK embedding would then also generate a new feature column based on this information. Similarly, predictions from other ML models could also be incorporated in the TALK KG. Another research direction can also extend the TALK approach towards other domains, which also uses a combination of domain knowledge and sensor data to predict event-related outcomes.

Reproducibility

The created TALK KGs, the used INK embeddings, the files to create those KGs and embeddings, and the full evaluation pipelines are all made available on our Github repository[7]. INK is also made available on another Github repository[8]. The DAHCC ontology and dataset is also made available open-source[9]

Acknowledgements

Bram Steenwinckel (1SA0219N) is funded by a strategic base research grant of the Fund for Scientific Research Flanders (FWO). This research was partly funded by

[7]https://github.com/predict-idlab/TALK
[8]https://github.com/IBCNServices/INK
[9]https://dahcc.idlab.ugent.be

the imec.ICON project PROTEGO (HBC.2019.2812), co-funded by imec, VLAIO, Televic, Amaron, Z-Plus and ML2Grow.

References

[1] N. D. Witte, J. Campens, L. De Donder, E. Dierckx, S. Rammelaere, and D. Verté. *Kwetsbaarheid bij Thuiswonende Ouderen*, May 2019. Available from: https://sociaal.net/achtergrond/oudere-mensen-blijven-langer-thuis-wonen/.

[2] S. Spasova, R. Baeten, B. Vanhercke, et al. *Challenges in long-term care in Europe*. Eurohealth, 24(4):7–12, 2018.

[3] J. C. Hou, Q. Wang, B. K. AlShebli, L. Ball, S. Birge, M. Caccamo, C.-F. Cheah, E. Gilbert, C. A. Gunter, E. Gunter, et al. *Pas: A wireless-enabled, sensor-integrated personal assistance system for independent and assisted living*. In 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007), pages 64–75. IEEE, 2007.

[4] R. Stokke et al. *The personal emergency response system as a technology innovation in primary health care services: an integrative review*. Journal of medical Internet research, 18(7):e5727, 2016.

[5] A. Taylor, H. Pizey, C. Whittet, D. Hammond, and S. Milne. *Analogue to Digital Telecare: Findings and Themes from a User-Centred Study to Help People Live in the Community Safely*. In 34th British HCI Conference 34, pages 203–213, 2021.

[6] T. H. Jo, J. H. Ma, and S. H. Cha. *Elderly perception on the internet of things-based integrated smart-home system*. Sensors, 21(4):1284, 2021.

[7] D. Bouchabou, S. M. Nguyen, C. Lohr, B. LeDuc, and I. Kanellos. *A Survey of Human Activity Recognition in Smart Homes Based on IoT Sensors Algorithms: Taxonomies, Challenges, and Opportunities with Deep Learning*. Sensors, 21(18), 2021. doi:10.3390/s21186037.

[8] M. De Brouwer, B. Steenwinckel, Z. Fang, M. Stojchevska, P. Bonte, F. De Turck, S. Van Hoecke, and F. Ongenae. *Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for IoT data streams with DIVIDE*. Semantic Web, (under review)(Preprint), 2022.

[9] B. Steenwinckel, M. De Brouwer, M. Stojchevska, J. Van Der Donckt, J. Nelis, J. Ruyssinck, J. van der Herten, K. Casier, J. Van Ooteghem, P. Crombez, F. De Turck, S. Van Hoecke, and F. Ongenae. *Data Analytics For Health and Connected Care: Ontology, Knowledge Graph and Applications*. In International Conference on Pervasive Computing Technologies for Healthcare, page (in preparation). Springer, 2023.

[10] D. Bouchabou, C. Lohr, I. Kanellos, and S. M. Nguyen. *Human Activity Recognition (HAR) in Smart Homes*. arXiv preprint arXiv:2112.11232, 2021.

[11] C. Goyal. *Deep understanding of discriminative and generative models*, Jul 2021. Available from: https://www.analyticsvidhya.com/blog/2021/07/deep-understanding-of-discriminative-and-generative-models-in-machine-learning/.

[12] K. Maswadi, N. A. Ghani, S. Hamid, and M. B. Rasheed. *Human activity classification using Decision Tree and Naive Bayes classifiers*. Multimedia Tools and Applications, 80(14):21709–21726, 2021.

[13] M. H. Siddiqi, M. Alruwaili, A. Ali, S. Alanazi, and F. Zeshan. *Human activity recognition using Gaussian mixture hidden conditional random fields*. Computational Intelligence and Neuroscience, 2019, 2019.

[14] A. E. Minarno, W. A. Kusuma, and H. Wibowo. *Performance comparisson activity recognition using logistic regression and support vector machine*. In 2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS), pages 19–24. IEEE, 2020.

[15] A. R. Jiménez and F. Seco. *Multi-event Naive Bayes classifier for activity recognition in the UCAmI Cup*. Multidisciplinary Digital Publishing Institute Proceedings, 2(19):1264, 2018.

[16] M. SEDKY, C. HOWARD, T. Alshammari, and N. Alshammari. *Evaluating machine learning techniques for activity classification in smart home environments*. International Journal of Information Systems and Computer Sciences, 12(2):48–54, 2018.

[17] L. Chen and C. D. Nugent. *Human activity recognition and behaviour analysis*. Springer, 2019.

[18] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson. *Mining models of human activities from the web*. In Proceedings of the 13th international conference on World Wide Web, pages 573–582, 2004.

[19] A. Patel and J. Shah. *Sensor-based activity recognition in the context of ambient assisted living systems: A review*. Journal of Ambient Intelligence and Smart Environments, 11(4):301–322, 2019.

[20] S. Salomón and C. Tîrnăucă. *Human activity recognition through weighted finite automata*. Multidisciplinary Digital Publishing Institute Proceedings, 2(19):1263, 2018.

[21] D. Hooda and R. Rani. *Ontology driven human activity recognition in heterogeneous sensor measurements*. Journal of Ambient Intelligence and Humanized Computing, 11(12):5947–5960, 2020.

[22] P. Foudeh and N. Salim. *An Ontology-Based, Fully Probabilistic, Scalable Method for Human Activity Recognition*. arXiv preprint arXiv:2109.02902, 2021.

[23] B. Steenwinckel, D. De Paepe, S. V. Hautte, P. Heyvaert, M. Bentefrit, P. Moens, A. Dimou, B. Van Den Bossche, F. De Turck, S. Van Hoecke, et al. *FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning.* Future Generation Computer Systems, 116:30–48, 2021.

[24] R. Mojarad, F. Attal, A. Chibani, S. R. Fiorini, and Y. Amirat. *Hybrid approach for human activity recognition by ubiquitous robots*. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5660–5665. IEEE, 2018.

[25] A. S. A. Sukor, A. Zakaria, N. A. Rahim, L. M. Kamarudin, R. Setchi, and H. Nishizaki. *A hybrid approach of knowledge-driven and data-driven reasoning for activity recognition in smart homes*. Journal of Intelligent & Fuzzy Systems, 36(5):4177–4188, 2019.

[26] S. Choudhary, T. Luthra, A. Mittal, and R. Singh. *A survey of knowledge graph embedding and their applications.* arXiv preprint arXiv:2107.07842, 2021.

[27] M. Wang, L. Qiu, and X. Wang. *A survey on knowledge graph embeddings for link prediction*. Symmetry, 13(3):485, 2021.

[28] R. Mondal, D. Mukherjee, P. K. Singh, V. Bhateja, and R. Sarkar. *A New Framework for Smartphone Sensor-Based Human Activity Recognition Using Graph Neural Network*. IEEE Sensors Journal, 21(10):11461–11468, 2021. doi:10.1109/JSEN.2020.3015726.

[29] T. Ahmad, L. Jin, X. Zhang, S. Lai, G. Tang, and L. Lin. *Graph Convolutional Neural Network for Human Action Recognition: A Comprehensive Survey*. IEEE Transactions on Artificial Intelligence, 2(2):128–145, 2021. doi:10.1109/TAI.2021.3076974.

[30] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling. *Modeling relational data with graph convolutional networks.* In European semantic web conference, pages 593–607. Springer, 2018.

[31] B. Steenwinckel, G. Vandewiele, M. Weyns, T. Agozzino, F. D. Turck, and F. Ongenae. *INK: knowledge graph embeddings for node classification.* Data Mining and Knowledge Discovery, 36(2):620–667, 2022.

[32] M. Espinilla, J. Medina, and C. Nugent. *UCAmI Cup. Analyzing the UJA Human Activity Recognition Dataset of Activities of Daily Living.* Proceedings, 2(19), 2018. doi:10.3390/proceedings2191267.

[33] N. Irvine, C. Nugent, S. Zhang, H. Wang, and W. W. Y. NG. *Neural Network Ensembles for Sensor-Based Human Activity Recognition Within Smart Environments*. Sensors, 20(1), 2020. doi:10.3390/s20010216.

[34] P. Lago and S. Inoue. *A hybrid model using hidden Markov chain and logic model for daily living activity recognition.* Multidisciplinary Digital Publishing Institute Proceedings, 2(19):1266, 2018.

[35] M. A. Razzaq, I. Cleland, C. Nugent, and S. Lee. *Multimodal sensor data fusion for activity recognition using filtered classifier.* Multidisciplinary digital publishing institute proceedings, 2(19):1262, 2018.

[36] J. D. Cerón, D. M. López, and B. M. Eskofier. *Human activity recognition using binary sensors, BLE beacons, an intelligent floor and acceleration data: A machine learning approach*. Multidisciplinary Digital Publishing Institute Proceedings, 2(19):1265, 2018.

[37] F. Pegoraro. *Multimodal Sensor Data Fusion for Activity Recognition using Deep Neural Network Ensembles.* Master's thesis, Università degli Studi di Firenze, 2018.

[38] B. Steenwinckel, F. De Turck, and F. Ongenae. *INK: Knowledge graph representation for efficient and performant rule mining.* Semantic Web, (under review)(Preprint), 2023.

8
Conclusion

"I'm going back to the start."

– Coldplay, The Scientist (2002)

Both data-driven and knowledge-driven AI have their shortcomings when predictions about a given task have to be made in time-critical domains, such as the predictive maintenance and eHealth field. The rise of HAI to combine both data and knowledge in one application aimed to resolve these issues, but the first created HAI applications could still not cover the adaptiveness of the mentioned fields and still had issues incorporating complex data types, such as time series data. Moreover, previously stated drawbacks such as the difficulties to capture and expose knowledge within such applications remain.

This dissertation proposed several solutions to these shortcomings. First, an automated manner is created to format structured documents into a KG to minimise the need for an ontology expert to adapt and maintain a specific KG. Second, an optimal methodology was proposed to combine both data- and knowledge-driven systems into a common architecture based around a KG for continuous exchange of insights and enriching the knowledge base. To allow for the HAI applications to continuously adapt accurately and efficiently, a new semantic rule mining approach was developed based on an explainable KG representation that was devised within this dissertation, namely INK. This same INK KG representation was further on used as KG embedding to enable knowledge in an ML task to its fullest. Finally, a KG was constructed to take into account time-dependent data. Both rules and ML models based on the

KG representation were created to show the benefits of such a KG representation and why a KG is not limited to static data in a time-critical domain.

8.1 Review of the Research Questions and hypotheses

Section 1.4 proposed five research questions, their outcome can now be evaluated. To resolve the lack of adaptability in a knowledge-driven approach, I proposed research question 1:

Can existing, structured documents reduce the need for manually creating knowledge in HAI?

Previous existing solutions were limited to either ontology experts creating the domain ontology based on the input of the domain experts or were limited to tools and training sessions created by ontology experts to let the domain experts learn how to create an ontology and accompanying KG. In our new proposed solutions in Chapter 2 and the appendices, existing FMEA documents formatted as Excel documents were used as input to generate new ontological concepts. FTA trees, also provided by the domain experts, were used to link these concepts together and to a more general upper ontology. The existence of these upper ontologies, where concepts within a general domain are linked together, is of high importance for multiple application domains. In this dissertation, an upper ontology describing anomalies and their causes was created and used in two predictive maintenance applications. We also showed how an explainable embedding can help to transform structured documents into linked knowledge. The combination of this linked knowledge and an adaptive expert-driven knowledge incorporation methodology made it possible to let experts update and adapt the KG without the need for an ontology expert to define, validate or maintain this whole process. This allows validating hypothesis 1: The design of a methodology that can automatically extract knowledge from structured documents and link this knowledge to existing ontologies will lead to an adaptive and continuously evolving KG, without the need for an ontology expert to make these adaptations.

A methodology to let experts incorporate efficiently and iteratively their knowledge inside a KG only provides one aspect of making HAI applications adaptable. Both the data- and knowledge-driven components should be combined in one architecture such that the knowledge-driven parts can change to new unseen cases and the data-driven parts can provide explainable results. Moreover, the outcomes of both approaches would ideally be combined in one structure such that both data- and knowledge-driven parts can update their internal mechanisms based on the found insights. To research the existence and applicability of such a new architecture, I proposed research question 2:

"Can an architecture be devised that uses a KG to empower an entire HAI application and increase the explainability of its outcomes?"

In Chapter 3, an HAI architecture was proposed to combine both data- and knowledge-driven components which relied on the expert knowledge provided in one central KG. Three phases were used to show how the data and knowledge propagated through this architecture. First, the designed architecture deployed a data-driven ML model, making predictions based on volatile sensor data, and a knowledge-driven rule-based expert system. Second, based on the information in the KG, the ML predictions could be made explainable and based on provided user feedback, predictions were combined, labelled and stored in the KG. At last, semantic rule mining was used to correlate the occurrence of the ML-based predictions and the available knowledge inside the KG. The generated rules were applied in future iterations to explain the made predictions. This whole architecture was evaluated on a predictive maintenance use case in the railway domain. By using this architecture, we were able to create a more precise HAI model, as the generated rules were used to reduce the number of false alarms by providing a clear explanation based on the available knowledge. It also provided context to the ML predictions, as a clear link between the prediction and the metadata was provided inside the KG. The semantic rule mining tool, but also the approach in the previous paragraph made this HAI model adaptive towards new unseen cases. This whole HAI methodology allows validating hypothesis 2: A KG-central approach for both the data and knowledge-driven components makes HAI outcomes explainable compared to its data-driven variants, and the HAI approach adaptable compared to its knowledge-driven variants.

Semantic rule mining steered our HAI application to a next level of adaptiveness. The currently available miners had, however, some limitations regarding the type of rules that could be mined and the learning rate to adapt HAI applications to their fullest. Therefore, I proposed research question 3.

"Can we optimise rule mining techniques for HAI applications by representing the KG in one data structure such that new insights and rules can be derived more accurately and efficiently?"

The current state-of-the-art semantic rule minders, DL-Learner and AMIE+ are designed to mine rules for either descriptive or prescriptive reasoning tasks. In Chapter 4, a new semantic rule miner was designed based on INK, which proposes a new intuitive internal KG representation. This novel rule miner can mine rules for both cases. For mining rules for a prescriptive case, INK outperformed DL-Learner in more than 75% of the provided benchmark datasets and the rules were mined more

than 15 times faster. For the benchmark dataset where we obtained both results for INK and DL-Learner, INK outperformed DL-Learner's accuracy by 14.25% on average. For these benchmarks, INK obtained these results on average in 81.74 seconds while DL-Learner required on average 879.04 seconds. INK mines prescriptive rules, therefore, 89.24% faster than DL-Learner. When mining rules in a deductive setting, the number of mined rules almost doubled. INK minded on average 7700 rules over all 5 descriptive benchmark datasets. In contrast, AMIE+ mined 4208 rules on average. This is an increase of almost 55%. The mined INK descriptive rules were also relevant, as the average confidence level of all the mined descriptive INK rules did not decrease. The internal representation of INK within this rule mining evaluation allows to validate hypothesis 3: A KG rule mining technique which can mine rules for both descriptive and prescriptive reasoning cases, is in general more than 10% accurate and requires less than 50% of the time to generate prescriptive rules while being able to generate 5% more rules in a descriptive setting on well-stated benchmark datasets and compared to the current state-of-the-art techniques.

An HAI application where all data and metadata are managed within a central KG must also be able to provide this information to an ML model. To investigate this incorporation, I proposed research question 4.

"Can knowledge be incorporated in an ML model without losing the inherently explainable characteristics of the provided knowledge?"

The previously designed INK representation to mine semantic rules can also be directly provided as embedding to an ML model, as researched in Chapter 5. Based on a set of nodes of interest, an explainable INK embedding was provided to various, commonly used ML classifiers. The goal of the examined task is a classic, but frequently occurring ML problem: predicting to which class a given node belongs. Both an evaluation on popular benchmark datasets and on a predictive maintenance use case was performed. The INK embedding outperformed two state-of-the-art techniques regarding the predictive performance on more than 85% of the benchmark datasets. Over the seven proposed benchmark datasets, INK had an average accuracy of 85.8%. RDF2Vec obtained an average accuracy of 83% on the same seven benchmark datasets. This is an increase in predictive performance of more than 2%. When comparing INK on the limited set of 4 benchmark datasets where also R-GCN results could be provided, INK (90% average accuracy) also outperformed R-GCN (85% average accuracy). The fact that INK can handle other different types of literal data, such as numerical values, almost doubled the balanced accuracy for a predictive maintenance node classification use case. When combined with a white-box ML classifier, such as a decision tree, the different explainable parts on which the decision is based can also be visualised. The ability to use INK in an ML setting allows to validate

hypothesis 4: A KG embedding technique keeping the inner explainable characteristics of the KG outperforms other KG embedding techniques with more than 2% in predictive performance on well-stated KG classification benchmark datasets.

To further enlarge the application range of HAI applications, it is necessary that the underlying KG can also incorporate time-dependent data because of both the examined time-critical domains, i.e. predictive maintenance and eHealth. To investigate this issue, I proposed research question 5:

"Can time series and metadata efficiently be combined in a KG for HAI applications without losing explainability?"

This dissertation proposed a new methodology to define time events within a KG within Chapter 7. The events themselves can contain sensor observations, grouped together based on a certain contextual parameter, such as location, and these events can be linked to each other based on a "hasPrevious" relationship. Semantic rule mining or KG embedding techniques can be applied to these event nodes, which can be further on used in ML classifiers to make predictions based on the occurred event. This whole methodology is tested and evaluated for an eHealth application where daily-life activities have to be tracked throughout the day. Chapter 6 described the large data collection effort that was set up to accomplish a real-life dataset of time series data. This dataset was also accompanied by an upper ontology for data analytics in healthcare, specially created to be used in a wide variety of eHealth applications. The KG HAI solution proposed in Chapter 7 outperformed all single ML models and is competitive with more complex ensemble techniques in a simplified homecare use case. Comparing the balanced accuracy of ML solutions like Markov Models (45%), Random Forest (47%), Neural Nets (60.1%) and Naive Bayes classifiers (60.5%), our approach (76.6%) was at least 16% more accurate. Besides more powerful knowledge-driven techniques, this whole methodology is also adaptive to other contexts, such as the more advanced homecare case described in Chapter 6. Since the methodology of Chapter 7 also incorporates the explainable KG representation INK, explainable results can be provided for all predicted activities. The incorporation of these events in a KG made it possible to validate hypothesis 5: An HAI model relying on a KG linking time series data with domain knowledge is at least 10% more precise than single-model ML-based alternatives while maintaining explainability.

8.2 Value and Impact on the HAI Domain

Different challenges were provided in Chapter 1, defining what is needed to go beyond the existing solution in time-critical applications that are rich in available complex domain expertise. In this section, we explain how all the research performed in this

dissertation impacts this HAI domain.

HAI applications are and will always be dependent on the knowledge provided by experts within the field to further improve or explain why a certain decision is made [1]. The ability to incorporate this knowledge based on given structured documents made it possible to generate HAI applications which can evolve over time, using the newly updated expertise. This adaptiveness did not improve the HAI solutions alone, it also resolved a drawback of the knowledge-driven techniques, as they are not further dependent on static, predefined information alone.

The adaptiveness of HAI applications is even further improved by the combination of data-driven output predictions with the provided data and knowledge. In a predictive maintenance use case, this HAI setup was able to derive new insights from the provided predictions and could further reduce the false positive rate. In the eHealth setting, the incorporation of predictions could be seen as new linked features in a KG embedding, reducing the complexity of the graph as the data accompanied by such a prediction can be neglected in these cases. The ability to incorporate the predictions of various ML models also creates an explainable, linked ensemble of newly generated knowledge.

Bringing the adaptiveness even to a higher level, HAI applications could incorporate semantic rule mining to automatically derive new insights on the provided KG. As the semantic reasoner can be either descriptive or prescriptive, it is a good idea to create a system to generate semantic rules for both paradigms [2]. Another requirement is related to the performance of such a semantic rule mining system. Based on the time-critical domain, rules must be generated on a large amount of data and within a certain time. The creation of INK resolved these two problems and automated the generation of new facts for the HAI domain.

The central usage of a KG resulted in the need for ML models to also incorporate knowledge within their systems. The current state-of-the-art approaches all had one limitation: the explainable aspects of the KG were completely or partially removed when the KG was used to make ML-based predictions [3]. The designed INK representation resolved this issue, as it can be incorporated within an ML model while holding the earlier required explainable aspects used to mine rules. Within these two application domains, INK positioned itself as both a performant semantic rule miner and a very competitive KG embedding technique.

The combination of automated knowledge incorporation and a semantic rule mining module made it possible for an HAI application to adapt the knowledge-driven parts to more dynamic settings. However, the data itself in the application field of a predictive maintenance and eHealth use case also has a dynamic character [4, 5]. Time series are frequently used to represent the data of sensor streams. Providing the KG with a central role in HAI applications, resulted in the need to define such time-dependent data efficiently or at least be able to evolve the metadata related to events inside a KG. The ability to generate events, link data observations to these events and

link these events together both showed that knowledge-driven techniques can start to use temporal reasoning within an HAI context and those previous events, predictions and obtained knowledge can have an influence on ML predictions.

The different research components, integrated in one system and defining a central role of the KG, results in HAI applications which can incrementally evolve over time and perform in various configurations beyond the currently available chain of knowledge and data-driven solutions.

8.3 Open Challenge and Future Directions

To further improve the provided research, this section highlights the four most important open challenges and discusses future work directions to resolve them.

8.3.1 Going beyond structured documents

The availability of structured documents enabled the HAI applications used in this dissertation to adapt and incorporate knowledge in an automated way. But not all knowledge within a certain expert domain is provided in these structured documents. Extraction of knowledge out of systems, provided code, management systems and even databases can further improve the automated incorporation of new insights into a domain-specific KG [6]. To maintain and link all these new concepts together, there is a clear need for expert-driven tools to transform information and knowledge. The current availability of mapping languages such as RML [7] can help to deliver this transformation. Such tools and the ability to let experts easily provide and update their so-called tacit knowledge in a KG are of uttermost importance for the further applicability of HAI systems in the future.

8.3.2 Unsupervised ML

All the provided use cases within this dissertation were based on the availability of labelled data or on an expert which could provide feedback on the generated predictions. Many applications within the predictive maintenance domain are related to anomaly detection and providing the right cause for the occurred anomaly [8]. In many of these situations, no clear label is provided to either train or verify a made classification. Data-driven techniques exist to deal with these kinds of unsupervised learning cases [9]. If we want the KG to play a central role within such unsupervised cases, similarity measures will be required to define if a certain node of subgraph within the KG holds anomalous behaviour or not. A similarity measure based on the explainable aspects of INK would be ideal, such that the reason why two nodes are dissimilar can be easily interpreted.

8.3.3 Temporal semantic rule mining

The incorporation of time-dependent data within this dissertation was tested on a repetitive and sequential time series dataset. In many HAI applications, it would not always be possible to chunk the data so easily into aggregated events. Using such events in a KG for HAI applications will need additional adaptations as well. One interesting field is incremental KG embeddings [10]. Here, the goal is to create an embedding of interest from the obtained sensor data. As new knowledge can be provided on the fly in an HAI application, embedding approaches such as INK should also be able to incorporate this information without the need to recreate the whole embedding from start. Another interesting field is temporal semantic rule mining. Here, the goal is to find a semantic rule given the available knowledge, but also to provide a time frame or a specific indication of which events have to happen in the past before the current event can be predicted [11]. Temporal rule mining applications were somewhat limited in the past, as only rather simple event-driven time series input rules could be generated. The generation of these event KGs could open a whole new research direction, towards generated temporal rules which can be applied in more general semantic stream reasoning use cases.

8.3.4 Interpretability

During this dissertation, the research components were designed to keep the explainable aspects of the KG. This explainability is needed to let the experts within the domain decide upon the usefulness of an HAI prediction. However, the interpretable aspects are not tested within this dissertation. Research can still be conducted to investigate how the INK embedding or the INK-generated rules can be visualised to the expert [12, 13]. Highlighting several nodes within the graph or transforming the generated rules back to queries such that they can be applied directly on the KG are only two possible solutions. Evaluations with experts within the HAI application domain will be needed to evaluate how interpretable results can be provided.

References

[1] A. Geurts, R. Gutknecht, P. Warnke, A. Goetheer, E. Schirrmeister, B. Bakker, and S. Meissner. *New perspectives for data-supported foresight: The hybrid AI-expert approach*. Futures & Foresight Science, 4(1):e99, 2022.

[2] P. Hitzler, F. Bianchi, M. Ebrahimi, and M. K. Sarker. *Neural-symbolic integration and the semantic web*. Semantic Web, 11(1):3–11, 2020.

[3] M. Palmonari and P. Minervini. *Knowledge graph embeddings and explainable AI*. Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges, 47:49, 2020.

[4] G. Aceto, V. Persico, and A. Pescapé. *Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0*. Journal of Industrial Information Integration, 18:100129, 2020.

[5] P. P. Jayaraman, A. R. M. Forkan, A. Morshed, P. D. Haghighi, and Y.-B. Kang. *Healthcare 4.0: A review of frontiers in digital health*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(2):e1350, 2020.

[6] B. Bozic, J. K. Sasikumar, and T. Matthews. *KnowText: Auto-generated Knowledge Graphs for custom domain applications*. In The 23rd International Conference on Information Integration and Web Intelligence, pages 350–358, 2021.

[7] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. *RML: a generic language for integrated RDF mappings of heterogeneous data*. In Ldow, 2014.

[8] P. Tanuska, L. Spendla, M. Kebisek, R. Duris, and M. Stremy. *Smart anomaly detection and prediction for assembly process maintenance in compliance with industry 4.0*. Sensors, 21(7):2376, 2021.

[9] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly detection: A survey*. ACM computing surveys (CSUR), 41(3):1–58, 2009.

[10] A. Daruna, M. Gupta, M. Sridharan, and S. Chernova. *Continual learning of knowledge graph embeddings*. IEEE Robotics and Automation Letters, 6(2):1128–1135, 2021.

[11] A. Segura-Delgado, M. J. Gacto, R. Alcalá, and J. Alcalá-Fdez. *Temporal association rule mining: An overview considering the time variable as an integral or implied component*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(4):e1367, 2020.

[12] P. N. Srinivasu, N. Sandhya, R. H. Jhaveri, and R. Raut. *From Blackbox to Explainable AI in Healthcare: Existing Tools and Case Studies.* Mobile Information Systems, 2022, 2022.

[13] S. Vollert, M. Atzmueller, and A. Theissler. *Interpretable Machine Learning: A brief survey from the predictive maintenance perspective*. In 2021 26th IEEE international conference on emerging technologies and factory automation (ETFA), pages 01–08. IEEE, 2021.

A

Automated Extraction of Rules and Knowledge from Risk Analyses: a ventilation unit demo

The new paradigm proposed in Chapter 2 resolved the adaptability issues by constructing the necessary domain knowledge from structured documents. This made it possible for experts to adapt the domain ontology and accompany KG by providing new information in the structured documents. This whole paradigm was also tested and evaluated using a ventilation use case. This appendix describes this use case and the parading was applied in this context. This appendix further investigates research question 1: "Can existing, structured documents reduce the need for manually creating knowledge in HAI?" and validates hypothesis 1: "The design of a methodology that can automatically extract knowledge from structured documents and link this knowledge to existing ontologies will lead to an adaptive and continuously evolving KG, without the need for an ontology expert to make these adaptations.".

⋆ ⋆ ⋆

B. Steenwinckel, P. Heyvaert, D. De Paepe, O. Janssens, S. Vanden Hautte, A. Dimou, F. De Turck, S. Van Hoecke and F. Ongenae

Published in the Proceedings of the International Semantic Web Conference (ISWC) Posters & Demonstrations, October 2018.

Abstract
Assessing upfront the risks, causes and effects of failures, is an important aspect of system manufacturing. Nowadays, this analysis is performed by a large number of people with different expertise and captured in various documents. To enable semantic unification and easy operationalization of these risk analyses, this appendix demonstrates an approach to automatically map the captured information into an ontology and accompanying rules. The approach is demonstrated with a use case to identify anomalies and their causes within a ventilation unit.

A.1 Introduction

Analysing upfront the risks and their potential causes are of high importance to let manufactures predict & prevent system failures. However, defining unwanted behaviour, i.e. anomalies, and pinpointing its cause requires much expert knowledge. Experts possess the contextual background information needed to model the anomalies & effects, using existing risk analysis tools, such as Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) [1].

This process can be time-consuming when applied thoroughly. Due to the large number of people involved, who each have expertise on other parts of the system, ambiguities and inconsistencies are common. To resolve this, ontology-based approaches have been proposed [2, 3] to structure the risk analyses, impose a common vocabulary and semantically link the different components, faults and causes. However, most system experts are not familiar with ontology design, which makes these approaches difficult to implement and maintain.

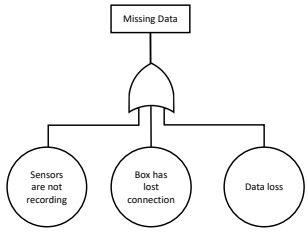
This appendix demonstrates an approach to automatically extract the knowledge captured in FMEA tables and FTA trees into domain-specific ontologies and rules. These can be used to easily spot and analyze the ambiguities, duplicates and inconsistencies in a structured format. Moreover, the ontology and rules support root cause analysis and automated anomaly detection. The proposed approach enables incremental knowledge incorporation by multiple domain experts while enabling some primary anomaly analysis.

A.2 Mapping risk analyses to ontologies & rules

A FMEA analysis is performed per system component to get a full view of all the possible failures that can occur, their effects and their possible causes. This results in an FMEA table, as visualized in Figure A.1a. In contrast, a FTA allows to identify the link between a particular (sensor) observation and a possible failure, as shown in Figure A.1b. Our mapping approach allows to easily combine the two, by offering methods that automatically translate the FMEA tables and FTA trees to ontologies

Component	Function	Failure Mode	Failure Effect	I	Failure Cause	O	Control Method	D	RPN
CO2 sensor	Instrument for the measurement of carbon dioxide gas	Values for specific sensor are set to 0	Missing data	2	Sensors aren't recording	4	Inform the user via the app.	5	40
				2	Box has lost connection	5	Inform the user via the app.	3	30
				2	Data loss	3		2	12

(a)



(b)

Figure A.1: Example of a (a) FMEA and (b) FTA

and accompanying rules that correlate the different observations, system components, faults and possible causes. Data generated by the systems can then be semantically annotated using these ontologies, while a reasoner can be used to interpret these observations and automatically derive the correlated anomalies and faults with the generated rules.

The full mapping approach is given in Figure A.2. It consists of two main parts, domain knowledge transformation and rule generation.

The **domain knowledge transformation** process is shown on the bottom of Figure A.2. Entries from FMEA tables can be mapped to RDF using a mapping language, resulting in a domain-specific ontology with risk analysis information. As such, anomaly knowledge can be extracted from the FMEA, and the causes of these anomalies can be derived by following the semantic links.

To enable this, an upper ontology, called Folio[1], was designed. It correlates all application-independent concepts that occur within FMEA tables and anomaly detection methods, such as `FailureCause`, `FailureEffect`, `Criticality` and `DetectionMethod`. The Semantic Sensor Network (SSN) ontology[2] is used to describe the various system components. Relationships were defined in Folio to correlate the SSN concepts with possible failures and effects.

To translate the informal information inside the FMEA table, usually in the CSV format, to ontological concepts, i.e. RDF, the mapping language RML [4] is used. To create the mappings, i.e. RML rules, the graphical user interface RMLEditor was used. The RML rules maps the different columns in the FMEA table on concepts of the Folio and SSN ontologies. Afterwards, the RMLMapper, a tool to execute RML rules, is used to generate the domain-specific ontologies for the mapped FMEA tables. The mappings ensure that for each cell in the FMEA table, a new concept is created in the ontology, which is a subconcept of the concept on which the column

[1] Folio ontology: https://github.com/IBCNServices/Folio-Ontology/blob/master/Folio.owl
[2] SSN ontology: https://www.w3.org/TR/vocab-ssn/

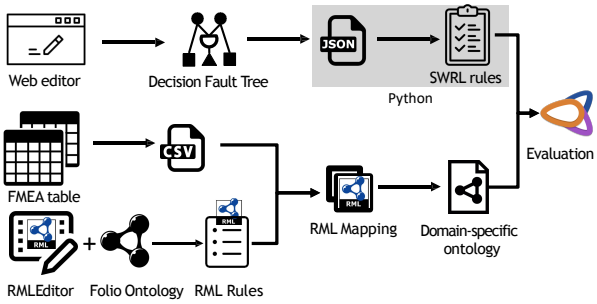


Figure A.2: Overview of the full mapping approach

is mapped according to the rules. For example, if we consider the 4th cell on the first row of Figure A.1a, the RMLMapper will create a new concept `Missingdata` in the ontology, which has as superclass the `FailureEffect` class.

As such, these mappings[3] can be re-used to translate any FMEA table that is created according to the standard FMEA structure. If a new column is added, a new rule can easily be created to map this column to the Folio ontology by using the RMLEditor.

The **rule generation** process is visualized on the top of Figure A.2. Observations outputted by the system will be the main entry point for further analyses of unwanted behaviour. This analysis can be expressed as a FTA tree that determines how new observations should be interpreted in the context of failures and effects. Classical fault trees do not allow complex analysis of observations. Therefore, our approach also supports an extension, namely decision fault trees (DFTs), which allow tests on the edges of the tree in order to allow for more complex interpretation and combination of the system observation values. A user interface was designed to built such DFTs. In this editor, descriptions of the observation and failure nodes can be given. These different node concepts should align with the concepts defined in the FMEA. Tests describing the relations between these observations and failures can be added or adapted. To transfer these tests to decisions, a script[4] was constructed that derives Semantic Web Rule Language (SWRL) rules from DFTs, formatted as JSON files.

The ontologies and rules generated by using the full translation approach can be incorporated in a knowledge-based monitoring system to continuously identify anomalies and their causes. When new (sensor) observations are generated by the system, they can be semantically annotated using the domain-specific ontologies generated by mapping the FMEA tables. A reasoner can then process the generated SWRL rules and links defined in the ontologies to determine whether failures are occurring and what their possible causes are.

[3]RML rules: https://github.com/IBCNServices/Folio-Ontology/blob/master/mapping.rml.ttl
[4]Script: https://github.com/IBCNServices/Folio-Ontology/blob/master/swr_builder.py

A.3 Demonstrator: a ventilation use case

The Healthbox 3.0 is a ventilation unit for residential buildings, offered by Renson Ventilation NV, that controls the airflow 24/7 to enhance indoor air quality. It extracts polluted air from different rooms in a house. One central fan is responsible for creating the requested airflow. The Healthbox is equipped with a series of sensors of which the observations can be used to detect possible failures. When for instance the requested airflow could not be achieved, the valves could be malfunctioning, or the fan speed can be too low. The described approach was used to automatically generate ontologies and rules that capture the knowledge of the Renson experts for the detection of ventilation anomalies and their causes.

Renson performed a FMEA analysis for each component of the Healthbox to get a full view on all the possible failures, their effects and causes. The demo will show how the designed RML rules, see previous section, can be used to easily map the contents of this FMEA to a Ventilation ontology, which is an extension of the SSN en Folio ontologies. The user will also be able to add new rows and column to the FMEA. It will be demonstrated how the RMLEditor can be used to map the new columns to the Folio ontology in a user-friendly manner. It will be shown that the new rows also automatically get transformed to the Ventilation Ontology without requiring to adapt the mappings.

The designed tree editor was used by experts to construct DFTs that link the system observations to possible failures of the Healthbox. The example DFT describes a possibly malfunctioning CO_2 sensor because the measured values are not in the acceptable range provided by the Renson experts. This sensor is important for the correct functioning of the valves, enabling the unit to ventilate when needed. It will be shown that this tree editor automatically exports JSON files that can be translated to SWRL rules by using the script detailed in the previous section. The user will be able to adapt or construct DFTs during the demonstration with the user-friendly editor. It will be shown that the script is also able to translate these new or adapted DFTs to SWRL rules.

The demonstration ends by combining the Ventilation ontology and SWRL rules in Protégé with some generated sensor observation instances. The reasoner is executed to illustrate that the rules extract the correct failures from these observations and link them to the possible causes through the ontology. The whole demo is also available on https://youtu.be/S3pe47Sn2Qs.

Acknowledgment: This research is part of the imec ICON project Dyversify, co-funded by imec, VLAIO, Renson Ventilation NV, Televic Rail & Cumul.io.

References

[1] J. Peeters and et al. *Improving failure analysis efficiency by combining FTA and FMEA in a recursive manner*. Reliability engineering & system safety, 172:36–44, 2018.

[2] Z. Rehman and et al. *An Ontology to Support Semantic Management of FMEA Knowledge*. International Journal of Computers, Communications & Control, 11(4), 2016.

[3] A. Venceslau and et al. *Ontology for computer-aided fault tree synthesis*. In Emerging Technology and Factory Automation (ETFA), 2014 IEEE, pages 1–4. IEEE, 2014.

[4] P. Heyvaert and et al. *RMLEditor: a graph-based mapping editor for linked data mappings*. In International Semantic Web Conference, pages 709–723. Springer, 2016.

B

MAGIC: Mining an Augmented Graph using INK, starting from a CSV

In Chapter 2 a whole transformation mechanism was created to transform the information in a structured document into a KG. Such a transformation approach is not the only option to link the textual knowledge within such structured documents. A large number of concepts are already defined online, provided as linked data and building on the fundamentals of the semantic web. One can also try to map textual descriptions inside the structured document to already defined concepts inside these linked data resources, such as e.g. DBPedia. Several of those so-called semantic annotators are available nowadays, but in this appendix, we show that the interpretable knowledge graph representation INK can also be used to find and match the concepts within a structured file with the corresponding textual description. The INK representation also has an additional advantage over the currently available techniques: due to the interpretable embedded representation, the structured files could also be augmented with new information that is linked to the annotated concepts when the INK embedding was created. This appendix further investigates research question 1: "Can existing, structured documents reduce the need for manually creating knowledge in HAI?" and validates hypothesis 1: " The design of a methodology that can automatically extract knowledge from structured documents and link this knowledge to existing ontologies will lead to an adaptive and continuously evolving KG, without the need for an ontology expert to make these adaptations.".

⋆ ⋆ ⋆

B. Steenwinckel, F. De Turck, F. Ongenae

Published in the Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th ISWC conference, online, November 2021.

Abstract
A large portion of structured data does not yet reap the benefits of the Semantic Web. Therefore, The "Tabular Data to Knowledge Graph Matching" competition at ISWC tries to bridge this gap by evaluating and promoting the creation of such semantic annotations tools. Besides annotating data semantically, the system should also be able to further augment the datasets based on the provided annotations. In this appendix, we propose a system that is capable of both annotating and augmenting a dataset by using the interpretable embedding technique INK. The "Tabular Data to Knowledge Graph Matching" competition was used to evaluate the proposed annotation capabilities of our proposed system.

B.1 Introduction & Challenge Description

A large portion of existing or newly produced structured data is not semantically annotated. Solutions exist to enrich raw structured data semantically based on the textual descriptions within these structured files [1–6].

The "Tabular Data to Knowledge Graph Matching" competition that has been hosted for several years at the International Semantic Web Conference (ISWC) [7, 8] stimulates the creation and optimization of these semantic annotation systems. These systems try to extract semantic annotations from a given Knowledge Graph (KG): a collection of interlinked descriptions of entities, both human and machine-readable. DBpedia [9] and Wikidata [10] are two such KGs.

Given a Comma-Separated Values (CSV) file, most of the time, three different challenges have to be tackled in the "Tabular Data to Knowledge Graph Matching" competition, as visualized in Figure B.1: (i) the automated assignment of a DBpedia or Wikidata column type (Column-Type Annotation (CTA)), (ii) a DBpedia or Wikidata entity has to be assigned to the different cells (Cell-Entity Annotation (CEA)), and (iii) relations between different columns have to be inferred, when possible (Columns-Property Annotation (CPA)). In our example Figure B.1, the CEA task would have to assign the *dbr:Coldplay* or wiki:Q45188 to text description of Coldplay in Col0. For the CTA task, Col3 must be assigned to *dbo:PersonFunction* or *wiki:Q66715801*. The relation between Col1 and Col2 for the CPA task must be defined as *dbp:birthDate* or *wiki:P569* in a semantic annotation system. The competition consists of different

rounds, with structured files from different domains. No ground truth labels are provided upfront, which means only unsupervised learning methods can be used.

Most of the existing solutions are based on external lookup methods and infer the column and property types afterwards by either clustering the entities based on an uninterpretable embedded vector or by using entity-specific scoring procedures [2, 3]. These annotators are denoted as semantic annotation platforms and provide a link to existing KGs but are currently not able to augment an annotated, structured file with new information available in those KGs. When for example, semantic annotations are provided for all entities within our example in Figure 1, the current techniques are not able to augment this table with additional linked data. Here, as an example, the inception dates for each entity in Col0 could be added to further enrich the current structured file. Semantic data augmentation tools, which can add new information based on tabular data, exist today [11]. Semantic data augmentation tools which can both define semantic annotations and also provide and new information based on semantic annotations are of high interest nowadays as more and more machine learning (ML) techniques try to use a graphical representation as input and tasks such as node classification are becoming popular [12]. The current nature of the existing semantic annotators, such as the uninterpretable embedding characteristics makes them not directly suitable for such an augmentation preprocessing step.

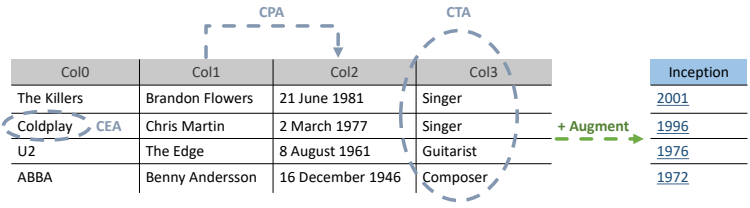


Figure B.1: The three different subchallenges of the competition are explained in a simple example. In the CEA task, a system tries to define each individual cell entity. Besides these 3 sub-challenges, a more general data augmentation task can also be interesting to perform.

Therefore, in this appendix, we describe MAGIC: a data mining tool to augment a structured file with data residing in a KG. In order to mine this augmented data, MAGIC will make use of INK [13], a fully interpretable embedding technique capturing all information from the neighbourhood of a node within our graph. Based on that interpretable embedding, the MAGIC platform can easily perform the CEA, CTA and CPA tasks all at once compared to its competitors and can provide additional linked data defined in those interpretable embeddings. By performing all tasks at once and storing the obtained INK embedding, the number of external calls to entity lookup services can be reduced. The fact that INK can also query KGs stored locally reduces the calls needed to external SPARQL endpoints even further.

We have organised the rest of our appendix as follows: In Section B.2, we give a general introduction to our interpretable embedding technique INK. Next, Section B.3 describes how INK is being used in our semantic annotation tool MAGIC. Implementation details are provided in Section B.4. Section B.5 shows the results obtained during the "Tabular Data to Knowledge Graph Matching" competition. To illustrate the additional benefits of both the interpretable embedding technique INK in combination with the MAGIC platform, we demonstrate how an existing structured file can be augmented with additional information within a KG in Section B.6. Finally, Section B.7 concludes this appendix and shows some additional future research directions.

B.2 INK: Instance Neighbouring by using Knowledge

Various techniques exist to transform information residing in KGs to a more appropriate format for an ML model, such as RDF2Vec [14]. INK is such a technique that builds node embeddings by transforming the neighbourhood of the node within a KG into an interpretable structured format. An example of the INK node extraction approach is provided in Figure B.2. Here, the goal is to build an INK embedding for the Coldplay node. INK will first query the neighbourhood until a predefined depth. If we define the depth parameter K to be one, only the nodes within the direct neighbours (visualized in grey) will be visited. INK iteratively extracts neighbourhood information and store these neighbourhoods efficiently (visualized in Figure B.2 on the right). The predicates in the neighbourhood of depth one are concatenated with their corresponding objects as values (concatenation here performed using the special character §). To add the neighbourhoods of depths > 1 (shown by the orange and green coloured edges within our example KG), INK concatenates all the relations on a path from the root node to the object node together, without providing detailed information about all intermediate nodes on that path. This intermediate information is still available due to the extraction at the lower neighbourhood's depths.

All this extracted information is stored in a dictionary value with as key the root node. When extractions are provided for multiple of these root nodes, a 2D matrix or data frame can be constructed defining which of these extracted (relation, object) pairs occur in the root nodes. Accompanied with descriptive labels for each of such a root node, this two-dimensional representation can be used to perform some more general machine learning tasks, such as node classification. More information about INK and the performance of this technique on several node classification tasks can be found in previous work [13].

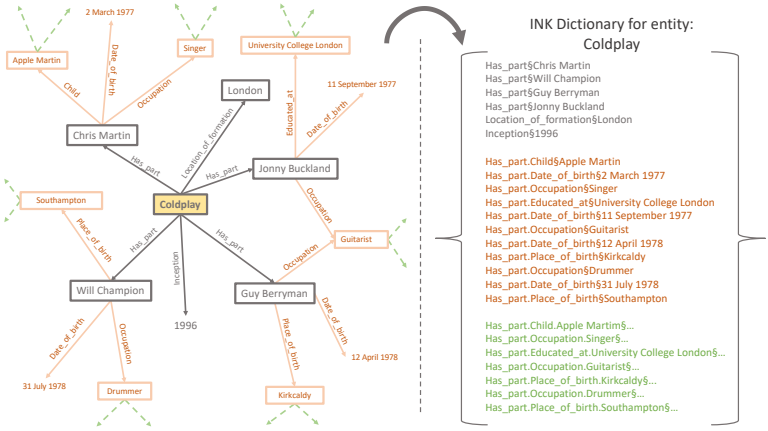


Figure B.2: Example of the INK dictionary representation (right), extracted from several neighbourhoods of the Coldplay node within a larger KG (left).

B.3 System Description

The interpretable embedding approach as discussed in Section B.2 is used in a system called MAGIC to annotate structured files. An overview of the MAGIC system is provided in Figure B.3. In general, three modules can be derived: A preprocessing module to select the appropriate annotation candidates, a processing module based on INK to get the interpretable embedding and select the best matches and at last, a post-processing module to offload the obtained information to the corresponding competition tasks. The rest of the section describes, in detail, the eight different steps to process the structured input file.

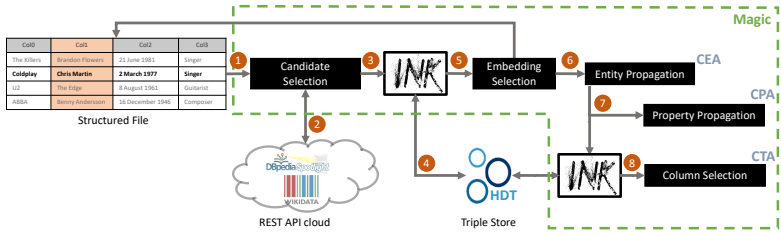


Figure B.3: Overall MAGIC architecture to define semantic annotations.

B.3.1 Defining the Major Column in a Structured File

In most cases, one single column holds multiple relationships to other columns within a table, for which annotation should be provided. This column can be seen as the

major column. In our example of Figure B.1, Col1 is our major column as both the information of Col0, Col2 and Col3 are derived from the information in this column. In many cases, this major column is known upfront and can be provided as additional input. During the "Tabular Data to Knowledge Graph Matching" competition, the major column was derived from the CPA target file when available. If such a target file was not provided, the MAGIC approach was rerun for every available column.

B.3.2 External Entity Lookup

Each cell description of only the major column is then provided to the more general MAGIC annotator. A pool of possible annotation candidates is generated from these cell descriptions. For the competition tasks which requires DBpedia annotations, the DBpedia Spotlight [15] service was used to generate possible candidates. When the annotations had to be linked to Wikidata entries, requests were sent to the Wikidata API[1] to search for entities using labels and aliases (wbsearchentities).

B.3.3 Candidate Selection

The external entity lookup services are not restricted, which means all possible candidates matching the provided cell description are returned. Based on multiple predefined characteristics, candidates can be prefiltered. The candidate selection step reduces, in this perspective, the number of matches obtained from the external API's. In this first version of the MAGIC system, we neglected any intelligent filtering of candidates.

B.3.4 Generating embeddings with HDT backend

For the selected candidate annotations, an INK embedding of depth 2 was generated as discussed in Section B.2. To limit the number of external SPARQL requests, INK extracted the neighbourhood information for these embeddings from a local compact data structure called (Header, Dictionary, Triples) or HDT [16]. HDT is a binary serialization format for RDF that keeps big datasets compressed to save space while maintaining search and browse operations without prior decompression. This makes it an ideal format for storing and sharing RDF datasets on the Web. An HDT-encoded dataset is composed of three logical components (Header, Dictionary, and Triples), carefully designed to address RDF peculiarities.

- Header: The Header holds metadata describing an HDT semantic dataset using plain RDF. It acts as an entry point and shows the key properties of the content even before retrieving the whole dataset.

[1] https://www.wikidata.org/w/api.php

- Dictionary: The Dictionary is a catalogue comprising all the different terms used in the dataset, such as URIs, literals and blank nodes. A unique identifier (ID) is assigned to each term, enabling triples to be represented as tuples of three IDs, which reference their respective subject/predicate/object term from the dictionary.

- Triples: As stated before, the RDF triples can now be seen as tuples of three IDs. Therefore, the Triples section models the graph of relationships among the dataset terms. By understanding the typical properties of RDF graphs, we can come up with more efficient ways of representing this information, both to reduce the overall size, but also to provide efficient search operations.

Popular and well-known libraries (like RDFLib [17]) provide additional mechanisms on top of this HDT data structure to translate general SPARQL queries to the underlying data sources. HDT datasets can be generated with a script but both Wikidata and DBpedia HDT versions were already made available by the HDT community. Those were used during this competition[2].

B.3.5 Selecting the best candidate embedding

At this point, interpretable embeddings of depth 2 were generated for multiple candidates, originating from a single cell within the major column of our structured file. Within these interpretable embeddings, a next function will search for matching information residing in the other cells on the same row from which the candidate embeddings were generated. In our example, the text description of Chris Martin can return two Wikidata entities: one is the Coldplay singer (Q712860), the other candidate is an American football player (Q519982). The INK embedding generated for the Coldplay singer will contain the information residing in the other cells of that same row (the is_part, birth date and occupation relation). As an embedding of depth 2 is provided, the labels of these associated nodes are incorporated in the embedding. The best candidate or embedding can be easily selected by counting the number of times additional information residing in other cells from the same row, is also available in the generated embedding of the candidate. In our example, the Chris Martin Coldplay singer embedding will have a matching count of 3, while the other Chris Martin has a zero match count. Therefore, the Chris Martin Coldplay singer annotation is selected as the best candidate in this situation.

B.3.6 Filling Additional Cells based on Selected Candidates

Selecting the best candidate according to the embedding also provides all information to the cells within the same row of the original structured file. Instead of performing

[2]https://www.rdfhdt.org/datasets/

an entity lookup for those cells, the annotations are directly derived from the major column embeddings. This eventually reduces both the costs of performed SPARQL lookups and external API calls. The annotations are all stored within a dictionary and output for the CEA task.

B.3.7 Defining Relationships Between Cells

Similarly, the embedding also provides the information going from the major column cell to the neighbouring cells within the same row. This information is directly derived from the provided embedding by selecting the object or data property relationships from the INK embedding. The combination of a relationship between two cells is kept within a dictionary. After the procedure to provide annotations for all cells is finished, the relationships between all those two cells are counted and the relationship with the maximal count is returned for the CPA task.

B.3.8 Defining column types based on additional type embeddings

The previous process is iterated for each cell within our major column. After all those annotations are provided, the column type can be derived from all cells containing annotated values. For all cells, an INK embedding of depth 1 is generated and the rdf:type for DBpedia and Wikidata P31 relationships are kept to determine the column type annotations. Again, the annotation with the highest count value is kept and returned for the CTA task.

B.4 Implementation

Both MAGIC[3] and INK[4] are implemented in Python and are made available on Github for future research. INK's implementation details can be found in the original appendix. The MAGIC code is designed to perform evaluations for both DBpedia and Wikidata tasks but can be adapted to any other task.

- The code to select the major column is currently left outside MAGIC. This makes it possible for users to provide either this major column by itself or to write specific code to determine this column within a structured file.
- The code to search for entity candidates is abstracted. This ensures that future approaches can integrate other entity search API's (such as the DBpedia lookup service) without redesigning the internal MAGIC code. Also, additional pre-processing steps, such as translations, spell checks, etc. can be added to this component in the future.

[3]https://github.com/IBCNServices/MAGIC
[4]https://github.com/IBCNServices/INK

- INK abstracts which data source it uses to generate the embeddings. In this version of the system, the embeddings were generated using an HDT backend and an HDT INK connector was made based on RDFlib to perform this task. In future projects, other already existing connectors (such as the INK Stardog[5] connector to connect to a triple store) can be used or created when needed.

B.5 Evaluation Results

The "Tabular Data to Knowledge Graph Matching" competition of 2021 consisted of three rounds in which multiple structured CSV files had to be annotated for either one, two or all CEA, CPA and CTA tasks. The different rounds consisted of multiple datasets from various domains where either DBpedia or Wikidata annotations were requested.

In total, four different metrics were used to evaluate the system. On the one hand, we had the F_1-score, which is a harmonic mean of the precision and recall of our system:

$$\begin{aligned} \text{precision} &= \frac{\#\text{correct annotations}}{\#\text{annotations made}} \\ \text{recall} &= \frac{\#\text{correct annotations}}{|\text{target cells}|} \\ F_1 &= \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{B.1}$$

During the second round, new metrics were used for the CTA challenge. Perfect annotation were encouraged, and at same time one of its ancestors (okay annotation) were also evaluated. Thus we calculate Approximate Precision (APrecision), Approximate Recall (ARecall), and Approximate F1 Score (AF1).

$$\begin{aligned} \text{APrecision} &= \frac{\sum_{a \in all \text{ annotations}} g(a)}{\text{all annotations \#}} \\ \text{ARecall} &= \frac{\sum_{\text{col} \in \text{ all target columns}} (\text{max\_annotation\_score(col)})}{\text{all target columns \#}} \\ AF1 &= \frac{2\times \text{ APrecision} \times \text{ ARecall}}{\text{APrecision} + \text{ARecall}} \end{aligned} \tag{B.2}$$

with # denotes the number, g(a) returns the full score 1.0 if a is a perfect annotation, returns $0.8^{d(a)}$ if a is an ancestor of the perfect annotation and its depth to the perfect annotation d(a) is not larger than 5, returns $0.7^{d(a)}$ if a is a descendent of the perfect annotation and its depth to the perfect annotation d(a) is not larger than 3 and returns 0 otherwise. $max\_annotation\_score(col)$ returns g(a) if col has an annotation a, and 0 if col has no annotation.

All evaluations were performed on the same 32 core Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz cluster node with 125 gigabyte RAM. The Wikdata HDT datasource of 3 march 2020 was used for all Wikidata related tasks. The October 2016

[5]https://www.stardog.com

English DBpedia HDT datasource was used for all DBpedia related tasks. The results of our approach are summarized in Table B.1 respectively.

Table B.1: Results obtained by the magic annotation system within the "Tabular Data to Knowledge Graph Matching" competition. / indicate that no evaluation mechanism was provided to evaluate these results. DNE (Did Not Execute) represents the tasks for which MAGIC did not provide any useful results.

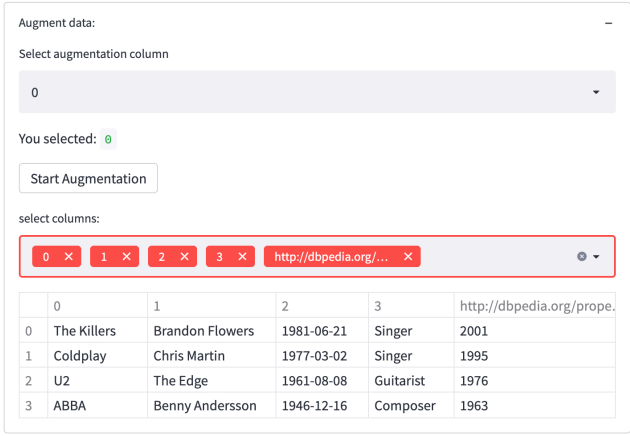
	CEA		CTA		CPA	
	F1	Precision	F1	Precision	F1	Precision
Round 1						
DBpedia tables	0.184	0.506	0.159	0.628	/	/
Wikidata tables	DNE	DNE	DNE	DNE	/	/
Round 2						
BioTable Wikidata	0.837	0.838	0.916	0.916	0.838	0.888
HardTable Wikidata	0.836	0.947	0.757	0.681	0.865	0.954
Round 3						
BioDivTab Wikidata	0.100	0.253	0.142	0.192	/	/
HardTableR3 Wikidata	0.641	0.721	0.687	0.687	0.788	0.936
GitTables	/	/	DNE	DNE	/	/

The MAGIC system has competitive results when a clear CPA task has been defined. When such a task and corresponding links between the columns within our structured files is not defined, the system has more difficulties selecting the correct entity descriptions. For some datasets, our system extracted too many candidate embeddings to evaluate. This resulted in memory issues and loss of information within the corresponding tasks. The evaluations of these datasets (in particular, the Wikidata tables and GitTables) are therefore not provided in this table.

B.6 Data Augmentation

Besides the provided annotations for the CEA, CPA and CTA task within the "Tabular Data to Knowledge Graph Matching" competition, the MAGIC framework holds an additional advantage compared to its competitors. The INK embeddings generated in the magic system are interpretable and used to match the nearby cells within the same row. Additionally, information originating provided from a new relationship, which holds for all cells within a column can be added when it is available from the interpretable embedding. When e.g., we annotated all cells within our structured file in Figure B.1. Combining all INK embeddings of Col0 will reveal additional information regarding the Bands listed in this file. One such additional relationship could be the Inception year, which is information that can easily be added as a new column to our original dataset. MAGIC can automate this process, revealing new possibilities to

extend the original dataset with new information.



	0	1	2	3	http://dbpedia.org/prope.
0	The Killers	Brandon Flowers	1981-06-21	Singer	2001
1	Coldplay	Chris Martin	1977-03-02	Singer	1995
2	U2	The Edge	1961-08-08	Guitarist	1976
3	ABBA	Benny Andersson	1946-12-16	Composer	1963

Figure B.4: MAGIC GUI application to augment annotated, structured files.

To make these benefits even more tangible, an additional GUI application has been developed that displays both the annotation and augmentation parts. An example of the GUI is visualized in Figure B.4. A video about this GUI application, using the basic example of Figure B.1, is also made availabe[6]

B.7 Conclusion and Future Work

In this appendix, a system to annotate and augment a structured file with semantic knowledge is being proposed. This system shows the benefits of combining the interpretable embedding technique INK with a semantic annotation tool. Future work can now focus on both the preprocessing and post-processing functionalities to improve the generated annotations. Currently, matches are provided on exact string comparisons, without taking any malformed or misspelt text into account. The generated embeddings are also not used to detect wrongly annotated cells. Simple outlier detection or clustering tools based on the generated embedding of a single column can already help to filter those wrong annotations. At last, a thorough evaluation is needed of how this system can help to augment existing datasets and how this augmented data can help in more broad ML tasks.

[6]https://www.youtube.com/watch?v=ZhTKxcTBZNE

References

[1] B. Steenwinckel, G. Vandewiele, F. De Turck, and F. Ongenae. *Csv2kg: Transforming tabular data into semantic knowledge.* SemTab, ISWC Challenge, 2019.

[2] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, and H. Takeda. *MTab4Wikidata at SemTab 2020: Tabular Data Annotation with Wikidata.* In SemTab@ ISWC, pages 86–95, 2020.

[3] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin, and R. Troncy. *DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular Data.* In SemTab@ ISWC, pages 27–39, 2020.

[4] N. Abdelmageed and S. Schindler. *JenTab: Matching Tabular Data to Knowledge Graphs.* In SemTab@ ISWC, pages 40–49, 2020.

[5] M. Cremaschi, R. Avogadro, A. Barazzetti, and D. Chieregato. *MantisTable SE: an Efficient Approach for the Semantic Table Interpretation.* In SemTab@ ISWC, pages 75–85, 2020.

[6] R. Azzi and G. Diallo. *AMALGAM: making tabular dataset explicit with knowledge graph.* In SemTab@ ISWC, pages 9–16, 2020.

[7] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, and V. Cutrona. *Results of semtab 2020.* In CEUR Workshop Proceedings, volume 2775, pages 1–8, 2020.

[8] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. *Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems.* In European Semantic Web Conference, pages 514–530. Springer, 2020.

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data.* In The semantic web, pages 722–735. Springer, 2007.

[10] D. Vrandečić and M. Krötzsch. *Wikidata: a free collaborative knowledgebase.* Communications of the ACM, 57(10):78–85, 2014.

[11] S. Galhotra, U. Khurana, O. Hassanzadeh, K. Srinivas, H. Samulowitz, and M. Qi. *Automated feature enhancement for predictive modeling using external knowledge.* In 2019 International Conference on Data Mining Workshops (ICDMW), pages 1094–1097. IEEE, 2019.

[12] B. Steenwinckel, G. Vandewiele, P. Bonte, M. Weyns, H. Paulheim, P. Ristoski, F. D. Turck, and F. Ongenae. *Walk Extraction Strategies for Node Embeddings with RDF2Vec in Knowledge Graphs.* In International Conference on Database and Expert Systems Applications, pages 70–80. Springer, 2021.

[13] B. Steenwinckel, G. Vandewiele, M. Weyns, T. Agozzino, F. De Turck, and F. Ongenae. *INK: knowledge graph embeddings for node classification.* Data Mining and Knowledge Discovery, page in production, 2021.

[14] P. Ristoski and H. Paulheim. *Rdf2vec: Rdf graph embeddings for data mining.* In International Semantic Web Conference, pages 498–514. Springer, 2016.

[15] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. *DBpedia spotlight: shedding light on the web of documents.* In Proceedings of the 7th international conference on semantic systems, pages 1–8. ACM, 2011.

[16] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. *Binary RDF representation for publication and exchange (HDT).* Journal of Web Semantics, 19:22–41, 2013.

[17] D. Krech. *Rdflib: A python library for working with rdf.* Online https://github.com/RDFLib/rdflib, 2006.