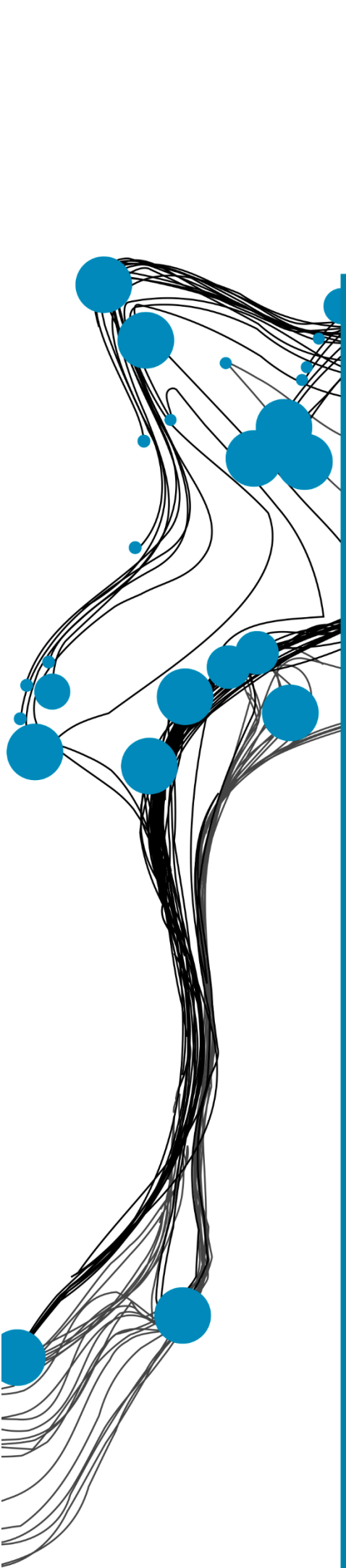


Augmenting the process of schema matching with machine learning-based Intelligence Amplification

Master thesis Industrial Engineering and Management

Specialization Production and Logistics Management

AUTHOR
T.H. Boerrigter
S1624024

EXAMINATION COMMITTEE
Prof. dr. M.E. Iacob
University of Twente

J.P.S. Piest MSc
University of Twente

EXTERNAL SUPERVISOR
L. Bekhuis
eMagiz Services b.v.

UNIVERSITY OF TWENTE.

eMagiz

Management Summary

In the integration of systems and applications, the Integration Platform as a Service (iPaaS) is a widely used environment. eMagiz offers such a platform, in which all data must be translated from and to a Canonical Data Model (CDM). This is done via a completely manual schema matching process, where the user defines that two data elements are related to ensure correct flows of data.

Repetition and low effort operations are common in the process of schema matching and are considered unwanted. Knowledge about the context of data schemas and similarities between schema elements is all in the heads of the users. To create a situation where a digital agent takes over parts of the schema matching process, a partial transition of knowledge to this agent is needed. This could speed up the process time-wise and alleviate the user from the more irrelevant tasks.

In this research, we establish a schema matcher based on the concept of Intelligence Amplification (IA), where the human-computer cooperation is stimulated to become $1+1 > 2$. In this cooperation tasks are divided between both parties based on their strengths and weaknesses, and both can augment one another in their tasks too. We want to construct a blueprint and a working prototype based, that shows its capabilities and scope of optimization based on the environment of eMagiz. The design should incorporate a learning aspect to evolve along with the practical schema matching cases and to take advantage of historical data.

We make use of the Action Design Research (ADR) methodology, which has its focus on investigating and developing an artifact in a business context. With this methodology, a practice-inspired research and theory-ingrained artifact are central, of which the latter emerges in cycles of reflection. Evaluation and expert knowledge are applied per cycle so the transition of knowledge and the accompanying artifact is smooth.

Systematic literature research showed us that there was no existing framework for the combination of schema matching and IA. Therefore we constructed our own out of three overlapping frameworks, and we configured it to be generic and reusable for other schema matching problems. Based on the systematic literature review and the context analysis, we created a solution design for a hybrid schema matcher. This hybrid matcher makes use of 6 different similarity measures, which each quantify a specific perspective on the similarity between schema elements. Some similarity measures are custom designed for this research context, such as the synonym similarity which makes use of a thesaurus. We present the construction of such a concept without the use of external services, that are often encountered in literature. For making schema matching predictions, two separate Deep Neural Networks (DNN) are used, one for entities and one for attributes to distinguish their individual behaviour. A constraint in prediction is that we apply a 1:1 matching cardinality, to limit matching complexity.

Parameter configuration experiments provided us with a set of configurations that yield the best average predicting behaviour on a grouped dataset of four different companies operating in the logistics sector. The average precision and recall are ~ 0.73 and ~ 0.54 for entities and ~ 0.65 and ~ 0.61 for attributes respectively. To put it in perspective, due to the 1:1 mapping cardinality a maximal possible average recall of only 0.644 and 0.826 is reachable for entities and attributes respectively. Results differ significantly per integration, indicating the complexity of the data set.

With the use of linear regression and polynomial regression, we found significant relationships between some integration characteristics and performance. These can be of use to the user in assessing whether to expect a beneficial contribution of the tool. Giving a concise prediction of the average performance based on a specific integration is not a straightforward task, since performance depends on many variables. It is up to the user to understand the different relationships and estimate the value of the IA solution.

The theoretical findings of this research resulted in a unique generic framework for schema

matching based on IA. This framework can be applied to all schema matching problems that aim for a learning solution. Also, we constructed a one-of-a-kind hybrid schema matcher design that can overcome language and abbreviation barriers without the use of external applications. With the provided algorithms, techniques can easily be replicated for another schema matching researches. The validated solution is proven to be really beneficial given certain conditions, outperforming the human. These conditions are made comprehensible by a logistic regression method that is unique to the field and valuable to both analysis and practical use. From a practical perspective, we provided eMagiz with a working prototype that is fully tested in a local environment. Also, relationships between integration characteristics and the expected performance of the solution are delineated for usability.

Preface

This thesis is the reference work of all activities undertaken and knowledge obtained in order to successfully complete my Master Thesis. It concludes the final part of my Master's degree in Industrial Engineering and Management at the University of Twente.

A result such as this report was not achieved in one day, and apparently required a winding road of 8 years. While my learning curve has fluctuated heavily over time, the enjoyment that came with being a student and all its accompanying situations and obligations, I would have never wanted to miss it. This period in my life has really helped me to explore the many possibilities and made me broaden my perspective on personal choices, social interactions and the working environment. Although motivation at the beginning of my study period was low, partially due to a lack of ambition, this certainly cannot be said for the final period. The last couple of courses and the Master thesis were one of the most interesting and challenging times, in which I really had the feeling I could add something, I knew what I was doing and I had the confidence to lead the projects I was working on. This transition is really valuable for me and would definitely help me in my working career that is about to start.

My master thesis is a product of a collaboration of knowledgeable persons, without them I could have never gotten to this result. First of all, I want to thank the people at eMagiz for giving me the opportunity to conduct a research such as this and for providing the necessary feedback when needed. Leo Bekhuis has been a great supervisor and listener, who steered me in the right direction and helped me keep my focus on the important things. The efforts of Samet Kaya to find the right thesis assignment for me and stick to the broader deadlines is also much appreciated.

The second group of persons I want to thank for their participation are Maria Iacob and Sebastian Piest, my supervisors from the University of Twente. They have both been equally involved in giving me feedback, sharing relevant topics that could be of value and making sure I finished my research within my own desired time frame.

The last and luckily largest group of people I am grateful for are my girlfriend, family, friends and my housemates from T.H.S.H. de Brakke Boomtor. Not only have they all been a great support during the times of my master thesis, but also in the other years that led to this moment. Without them, my time as a student would never have been a joyful one, so their involvement was and still is priceless.

Tom Boerrigter,
Enschede, August 16, 2021

Contents

List of Figures

List of Tables

1 Introduction

We start this thesis with an introduction to the origination of this research, to make it clear why we conduct it in the first place. For an associated motivation, we first sketch the scenario by discussing the company eMagiz and the concept of schema matching in Section 1.1. Section 1.2 completes the motivation by elaborating on the perceived problem and on the defined objectives that are set up together with eMagiz and its platform users. Adding to the context and direction of this research, Section 1.3 shortly describes what Intelligence Amplification is and why we apply it in this case. From Section 1.4 we turn towards the execution of the problem, of which the chosen methodology is explained first. The resulting research questions are stated in Section 1.5. Lastly, the structure of the thesis together with its methods can be found in Section 1.6.

1.1 The concept of schema matching and its relation with eMagiz

1.1.1 Schema matching definitions

Schema matching is described by (Bonifati & Eds, 2011, p. v) as "(...) the semi-automatic finding of semantic correspondences between elements of two schemas or two ontologies.". Its application is in many database uses such as integration of web data sources, XML message mapping and data warehouse loading (Do, 2006). The desire to automate the process of schema matching comes impracticalities that arise when schemas grow in size. When done by hand, it can take up much time and can be tedious or repetitive. Due to the often high semantic heterogeneities, schema matching is an inherent difficult task that has led to the development of many techniques and prototypes to semi-automatically solve the problem (Rahm, Do, & Maßmann, 2004).

A human that draws mappings roughly goes through a given process. Schema elements are interpreted, mappings are created on combinations that the human is confident with, remaining combinations are discussed with experts and lastly the result is evaluated to avoid mistakes. In Rahm (2011) a general workflow of a schema matcher is described, on which many prototypes are based (Figure 1.1). The machine receives input schemas of which all possible combinations have to be considered. For each of these combinations, the matcher computes a set of results that resemble the degree of similarity between schema elements. These results have to be interpreted by the matcher and turned into a *match* or *no-match* result. It is then the task of the human to evaluate these result mappings.

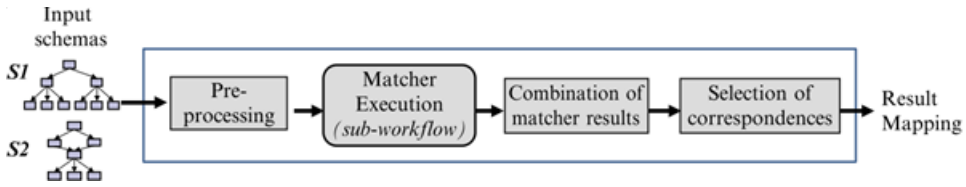


Figure 1.1: General workflow for pairwise schema matching, retrieved from Rahm (2011)

Before we continue on other subjects and dive deeper into the process of schema matching, it is useful to mention clarify the terminology that is often used. First, the terms schema **matching** and schema **mapping** are mentioned together quite often, but there is a subtle difference. A match is an association between individual structures in data schemas. Mappings are the products of this, and are an expression that describes how data of some specific format is related to data of another (Bonifati & Eds, 2011). Thus, a schema matcher relates to all the parts in the workflow of the figure above, whereas schema mapping only describes implementing the end result. The **schemas** that we describe in this research are all XML-schemas, which is a language to describe the structure of XML documents. A schema holds a collection of

metadata that consists of **entities** and **attributes**. An entity is a unique object that can be distinguished from other objects by its attributes. The attributes are the characteristics of the belonging entity, which are often not unique. In a schema matching problem there are always two schemas present, a **source schema** and a **destination schema**. Corresponding elements are also referred to as **source elements** and **destination elements**. The names depict the direction of the data flow within the mappings.

A single schema matching technique multiple combined ones are often referred to as **schema matcher**, schema matching tool or auto-matcher. This comprises the processes to acquire the mapping result which we explained earlier. An important part of the similarity assessment between schema elements are **similarity measures**. These represent a paradigm on similarity based on a certain element characteristic, such as subsequent letters or the data type. Schema matchers are defined by the similarity measures they use, and often multiple measures or techniques are combined. We call this a **hybrid matcher** and it is the most common approach for schema matching problems (Do, 2006). The techniques used are fixed and encompass different characteristics of the schema elements. In contrast to this, a **composite matcher** has a set of techniques from which it can choose based on the match task at hand.

1.1.2 Schema Mapping in eMagiz Enterprise iPaaS

The company that makes this research possible and serves as a validation case of our solution design is eMagiz. It is a small-sized IT-company, located in Enschede and specializes in integrating applications and systems for better management and automation of data flows. Originated from CAPE Groep, the main idea was to build an own solution modelled in Mendix to provide all integrations necessary in businesses.

The core product of eMagiz is its Enterprise integration Platform as a Service (iPaaS). It provides a low-code environment that can be easily understood by the user due to the integration by visualization. Most functionalities use drag and drop processes, derived from the underlying software of Mendix. There are three integration patterns provided for use case, which are messaging, API gateway and event streaming.

The customers of eMagiz's services are largely logistic, transportation and construction companies. However, not all companies directly work with the integration platform due to limited experience or know-how. A large part therefore decides to hire consultants from CAPE Groep to set up the desired integration. People that directly use the platform and interact with it are CAPE Groep IT consultants and IT specialists within other companies.

All customer system integrations are connected with a canonical data model (CDM) in the eMagiz platform. This means that all dataflows go through this CDM, which is custom built for each integration solution. Usually, a lot of different systems are present at the customer that all have their own structure of sending messages or data. These messages are sent to the CDM that only uses one standard structure, hence the name. To give an example, Figure 1.2 show two systems that are connected via the CDM in the platform of eMagiz. A translation must take place for the data to be interpreted correctly and sent to the right place. This is called schema mapping/matching and is a technique for connecting the corresponding entities and attributes between the system and the CDM. Figure 1.3 displays a snippet of mapped schema elements, which represents the translation of the order message coming from CAPE and going to the CDM (Figure 1.2). By sending the messages through to the correct customer destination system, the same process is used but the other way around. Schema matching in the platform is used often, is present at each project and can be a significant task in setting up an integration.

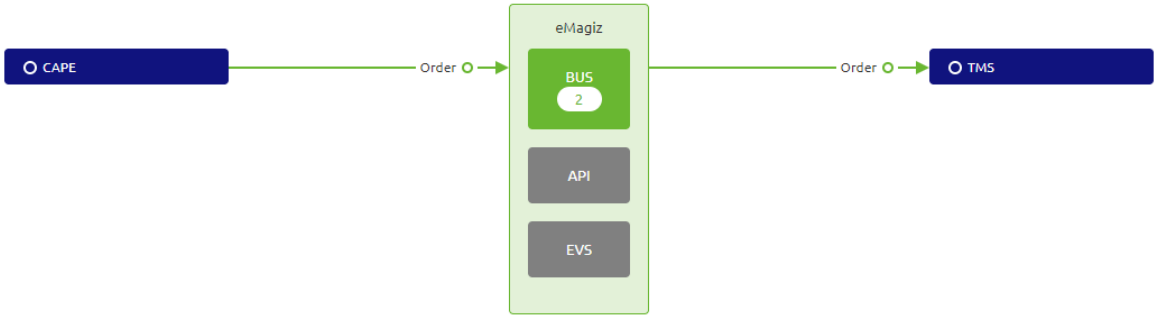


Figure 1.2: Example of two connected systems in the platform of eMagiz, linked by the CDM

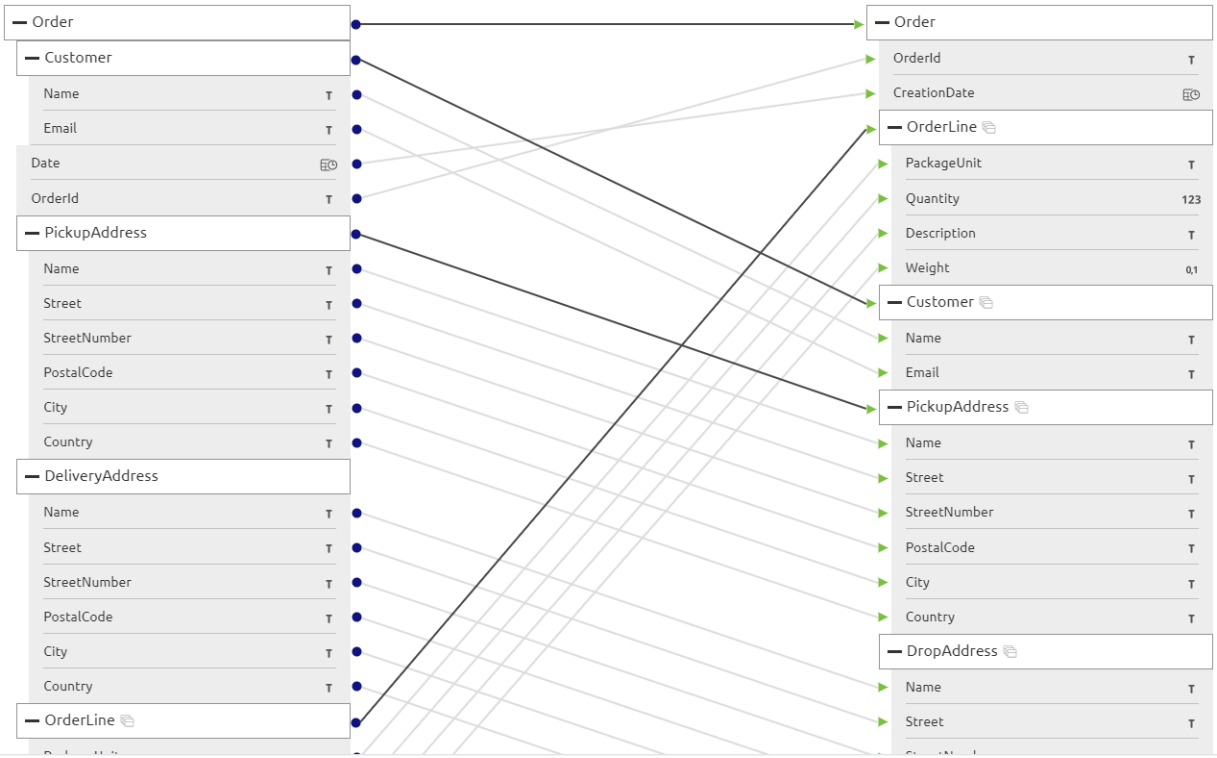


Figure 1.3: Snippet of mapped schema elements in the platform of eMagiz

1.2 Problem Statement and Research Objectives

The motivation for this research is practice-inspired and comes from eMagiz' desire to improve the user comfort when using their platform. Schema mapping in their integration context is a task that is typically executed and supervised by experienced consultants, of which the latter task is unlikely to be changed. Due to the occurrence of schema mapping situations where source and schema elements have high similarities, mappings are perceived as obvious choices that do not need any experience. Users have reported to find these situations repetitive and unchallenging, creating the feeling that their time is spend inefficiently. Thus, the perceived problem is at the consultants that feel they spend too much time on creating mappings in the schema mapping process. The core problem is that the knowledge needed for creating these mappings is solely in the heads of users. eMagiz therefore wants to transfer the knowledge and mapping logic to their platform in order to assist the user, relieve them from repetitive tasks and speed up the schema mapping process. Previously, this exact same problem is addressed in another master thesis research, but this did not lead to the desired result. Therefore, eMagiz

wants an improved version that is built upon the findings and lessons learned from the preceding work.

The single tangible research objective that comes from the initiative to assist the user in the schema mapping process in the iPaaS of eMagiz, is to provide a blueprint for a working schema mapping feature suited to their environment. This feature should be based on a collaboration between human and machine, which is named Intelligence Amplification (IA) and is discussed in the next section (1.3). Objectives that are at the base of this feature seek to obtain the knowledge needed for shaping a suitable solution. Formal knowledge from other researches and literature should give us the right information to construct our own design. The practical knowledge present at eMagiz helps us fitting in the artifact of this research with its characteristics into their platform. We want it to be configured at the best possible settings, substantiated by an exploration of the possibilities. Besides this, we want to be able to explain the performance behaviour of the artifact so the user is aware of what to expect. With this, we aim to get the user involved in a collaboration with the machine in order to speed up the schema mapping process.

1.3 Improving Schema Matching with Intelligence Amplification

The artifact that emerges as a product of this research can be described as a case of Intelligence Amplification. In IA, both the human and the AI agent form a symbiotic partnership, in which the human entity defines strategy, oversees the AI agent, and corrects its decisions when needed, and the AI agent executes routine tasks according to the strategy (Dobrkovic, Döppner, Iacob, & van Hillegersberg, 2018). This concept really fits well with the automation of schema matching in general and at eMagiz. The human in our case should always be able to oversee the predictions of the schema matcher, and can invoke the matcher to execute routine tasks at the moments the user thinks it would be beneficial. We use the concept of IA to give direction within our Literature Review in Chapter 3 and in the design of our solution in Chapter 4.

1.4 Research Methodology

Schema matching on its own and also in an iPaaS is part of, among others, the Information Systems (IS) discipline. As explained in 1.2, one of the objectives of this research is to deliver an artifact. Together with a solid description of the problem context, Design Science is the concept that comes into mind, as it centers around the design and investigation of artifacts on context (Wieringa, 2014). Design Science can serve as a guideline to setting up a sound research, when following a developed methodology. "Such a methodology might help IS researchers to produce and present high quality design science research in IS that is accepted as valuable, rigorous, and publishable in IS research outlets" (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007, p. 5). We take a look at the available methodologies in the Design Science field, and make a decision based on the properties that best fit our research objectives.

1.4.1 Related Methodologies

Wieringa (2014) has developed as an extensive guideline for practicing design science. It gives clear instructions for every major and minor detail that can play a role during a research. There is not much freedom for the researcher, but its explicitness can ensure a thorough performance.

The paper of Peffers et al. (2007) combines the processes of multiple IS researches and develops and presents a methodology that fills their indicated scientific gap. It is meant to be a general methodological guideline for effective DS research. "The Design Science Research Methodology should not be used as a rigid orthodoxy" (Peffers et al., 2007, p. 74), thus the interpretation should be taken loosely and is up to the researcher. They also state that a

researcher does not always find himself at the first activity, therefore there are many approaches centered around different purposes. The paper makes it possible to start at any activity, in contrast to Wieringa (2014).

Similar to the book of Wieringa (2014), Hevner, March, Park, and Ram (2004) proposes a guideline on how to conduct, evaluate and present design science research, although less extensive. Their focus is on brining together behavioral science and design science, as they are two complementary but distinct paradigms (Hevner et al., 2004).

Originated from the ideals of design research, Sein et al. (2011) indicates that current design science methods consider organizational intervention to be secondary. "Traditional design science does not fully recognize the role of organizational context in shaping the design as well as shaping the deployed artifact." (Sein et al., 2011, p. 38). Therefore Action Design Research (ADR) is developed, which stresses the cycle of concurrently building the IT artifact, intervening in the organization and evaluating it.

1.4.2 Action Design Research

Although we started off with the thought of embracing Design Science due to its focus on investigating and developing an artifact in a certain context, the ADR methodology relates much better to the origin of this research. The perceived problem at eMagiz is the catalyst for starting the research on schema matching in an iPaaS, whereas the university restrictions provide the contextual background.

Action Design Research is split up in four different stages to deal with two main challenges: (**1**) addressing a problem situation encountered in a specific organizational setting by intervening and evaluating; and (**2**) constructing and evaluating an IT artifact that addresses the class of problems typified by the encountered situation (Sein et al., 2011). These four stages are displayed in Figure 1.4 and elaborated next.

The first stage about the problem formulation and is triggered by the problem perceived in practice or anticipated by researchers. The two principles that are emphasized is that the research should be practice-inspired and the artifact theory-ingrained. Tasks that come with this stage are displayed in Appendix A Figure A.1. The second stage can be viewed as an iterative approach between *building*, *intervention* and *evaluation* (BIE) of the problem and the solution. Sein et al. (2011) defines a research design continuum delimited by an IT-dominant BIE and an organization-dominant BIE. We define our research to be at the IT-dominant side, since "this approach suits ADR efforts that emphasize creating an innovative technological design at the outset" (Sein et al., 2011, p. 42). The corresponding tasks of this stage can be found in Appendix A Figure A.2. The third phase is a continuous phase that parallels the first two stages, and is designed to be flexible with the changing situation during the research. Not often is the initial problem the only problem that needs to be solved, some might emerge along the way. The problem formulation and the company environment can have different focuses and desires, which should both be captured by the artifact. The tasks of this phase are displayed in Appendix A Figure A.3. The last phase is about pulling the findings and learning into a more general context, so that others can benefit from it in other similar problems. Tasks corresponding to this phase can be found in Appendix A Figure A.4.

From this point on we apply the ADR methodology and indicate when we address principles. Tasks are mentioned when they are undertaken and finished.

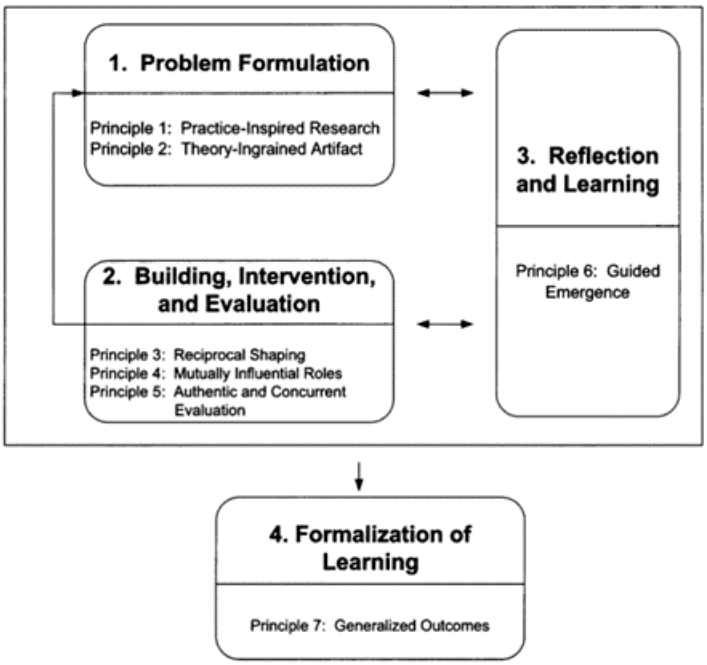


Figure 1.4: ADR Method: Stages and Principles, retrieved from Sein et al. (2011)

1.4.3 Literature review methodology

During the execution of our ADR defined research, we shape our artifact by theory. Applying what is already know is a useful decision that can save a lot of time, so we dedicate a part of our research to be filled in by a literature review. For such a task, a dedicated methodology is needed to " [...] summarise all existing information about some phenomenon in a thorough and unbiased manner" (Kitchenham & Charters, 2007, p. 7). A well known methodology is described by Kitchenham and Charters (2007). Their Systematic Literature Review (SLR) is appropriate to the needs of software engineering researchers, due to the general lack of presence of empirical research and data that characterizes the field. Guidelines for a review protocol are given, that is necessary to reduce the possibility of researcher bias. Without such a protocol, it may be possible that the selection of individual studies may be driven by the researchers expectations (Kitchenham & Charters, 2007).

1.5 Research Questions

With the research problem we want to provide an answer for the problem that as at the root of our research goal. As it is proposed in Verschuren and Doorewaard (2007), we first must know what knowledge is useful for reaching our research goal when formulating a main question. Therefore we repeat the goal of this research: Establish a man-machine relationship that benefits the schema matching process. The knowledge that must be obtained can be put in the following four aspects: 1. The exact tasks or processes that must be addressed, 2. The kind of technique that must be applied, 3. The degree of automation and 4. The performance of the schema matcher solution. Our main research question follows from the four different aspects, and is stated as:

MQ *How can the process of schema matching be augmented with machine learning techniques to provide highly accurate and time saving results, when integrating systems in an iPaaS?*

The research questions serve as guidance for answering the central research question. To get a thorough understanding of the context and on what has to be taken into account before presenting a solution, the following questions are proposed:

RQ 1 What are the challenging characteristics of a schema matching problem and how can we measure its performance?

RQ 2 What aspects define Intelligence Amplification in the schema matching context?

RQ 3 What automated learning techniques are suitable for schema matching?

RQ 4 What machine learning solutions have already been applied relevant to the research context?

RQ 5 What is an efficient Intelligence Amplification driven schema matcher design based on machine learning techniques?

RQ 6 What is the performance of the proposed solution design and when is it beneficial?

We discuss how and when these research questions are implemented and answered in the next section (1.6).

1.6 Thesis Structure

All upcoming parts needed to report the findings on our research questions including a thorough discussion and conclusion are structured in six chapters as follows:

2. Context Analysis elaborates on the aspects and issues that come with schema matching in general and in the integration platform of eMagiz. An additional important aspect is how performance is typically measured and what could be of use in the validation case. Together, this serves as an answer to RQ 2.

3. Literature Review discusses all the relevant theory that could be of use in shaping our solution. Together with Chapter 2 it describes Stage 1 Problem Formulation of the ADR methodology. We answer RQ 2, 3 and 4, from defining concepts in context to specific solutions that have already been implemented and validated.

4. Solution Design displays all features and underlying choices and algorithms that make up our solution design. In this chapter, RQ 5 is answered, which should yield us a schema matcher design satisfying our prerequisites.

5. Solution Results presents the results of the experiments designed to answer RQ 6. An overview of performance per parameter configuration, the behaviour per integration characteristics and the net benefit of the solution are all included. The second stage of the ADR, Building, Intervention and Evalution, is covered by Chapter 4 and 5.

7. Conclusions and Future Work completes the research by concluding all findings, together with a critical look on what has been done. It also gives recommendations for eMagiz and to where future research might be picked up, plus an overview of the contributions to theory and practice. It serves as the representation of ADR's Stage 4 Formalization of Learning.

2 Context Analysis

Before we dive any deeper in finding techniques that serve in a solution design, we need to have a clear view on the context that we are dealing with. This chapter first describes the issues that are characteristic to schema matching in section 2.1. Being familiar with the issues can be of great value as input for our literature review and to our solution design. Section 2.2 is about the various integration characteristics in the eMagiz iPaaS. The last Section 2.3 describes the conventional measures in schema matching to express performance and concludes which ones are useful to our research. With this, we have an answer to **RQ 1**: *What are the challenging characteristics of a schema matching problem and how can we measure its performance?*

2.1 Issues in Schema Matching

As it is shortly mentioned in Section 1.1, schema matching is an inherently complex task by typical high degrees of heterogeneities (Bonifati & Eds, 2011). Heterogeneities generally arise due to the fact that schemas are developed independently by different people with different purposes and perceptions of the situation (Do, 2006). There are various types of classifications all being described differently across papers, but most of them refer to ontology matching which entails irrelevant problems to this context. We therefore apply the types of heterogeneity proposed in Do (2006), since these refer problems closest to XML schema matching. The conflicts as they are called, can be divided into two classes: metadata-level conflicts and instance-level conflicts.

2.1.1 Metadata-level conflicts

A conceptual conflict is that "same names do not necessarily indicate the same semantics "(**1.a**) "and different names may in turn be used to represent the same real-world concept" (**1.b**) (Do, 2006, p. 8). It can be the case that attributes have the same name, but are related to a different entity. Some schema elements are broad concepts, that apply to multiple contexts. The conflict that different names may represent the same real-world concept, is also called a terminological heterogeneity in Euzenat and Shvaiko (2007). The book states that this may be caused by the use of different natural languages, technical sublanguages or the use of synonyms. Examples of conflicts **1.a**, **1.b** and those in upcoming paragraphs are given in Table 2.1.

"Element names may be encrypted or abbreviated so that they are only comprehensible to their creators" (**2**) (Do, 2006, p. 8). For sake of simplicity users can choose to make an abbreviation if the element name consists of multiple words or that it is simply too long. Most of the times additional abbreviated words are demarcated with a capital letter so that it is easy to spot them.

At schema level possible integrity constraints may not be specified, but in the programs accessing data it can be present and of influence for becoming a conflict (**3**) (Do, 2006). These constraints can be drawn up to make sure instance-level data stays in the desired format.

Schema element names may have different levels of details (**4**) (Do, 2006) which can be a result of practical choices or system characteristics. Elements with no detail can have different parts of data stored in the same place. If these parts are needed individually, it is necessary that they are split. Therefore differences in detail can often become a conflict.

2.1.2 Instance-level conflicts

Insights into the contents and meaning of schema elements can also be provided by instance data. Unfortunately, here we encounter three common conflicts, starting with different values that are employed to encode the same piece of information (**5**) (Do, 2006). Given some integrity constraints, it can cause serious conflicts. Integrity constraints can for example mean that instances can only take on certain values or that there is a maximum input length. Just like

Table 2.1: Metadata-level conflicts

Conflict number	Schema 1 element(s)	Schema 2 element(s)
1.a	*Name (company)*	*Name (person)*
1.b	*Paper*	*Article*
2	*Order Number*	*OrdNum*
3	*Password*	*Password (max 8 characters)*
4	*Address (street + nr.)*	*Address (street), House Number (nr.)*

the metadata-level conflicts, the example conflicts arising at instance-level are given in a table (Table 2.2).

Interpretation of values can differ per element by using different measurement units or string formats (**6**) (Do, 2006). The unit measures need conversion for it to work properly, and string formats can be countered by a split.

"Instance data may contain errors, such as misspellings, missing values, transposed or wrong values, duplicate records, etc." (**7**) (Do, 2006, p. 8). These errors cannot be easily be prevented, which causes the instance-level data to lose its value for providing insights.

In this research we focus on iPaaS, which is a suite of cloud services (Ebert, Weber, & Koruna, 2017). This means that most companies, and certainly eMagiz, do not posses or store the instance data that runs through their platform. Although it can give a great amount of extra information to provide additional services, there are a lot of barriers with storing. Privacy regulations are a major struggle when it comes to sharing, and companies are very hesitant in giving away their valuable data. Therefore, we do not include instance data advantages or conflicts in our solution.

Table 2.2: Instance-level conflicts

Conflict number	Schema 1 instance(s)	Schema 2 instance(s)
5	*"Female"*	*"F"*
6	*"$1000"*	*"€1000"*
7	*"Testt mesage"*	*"Test message"*

2.1.3 Match cardinalities

The match cardinalities determine how many relations are allowed considering source and destination elements. Set-oriented cases are one-to-many (1:n), many-to-one (n:1) or many-to-many (n:m) and a seperate and more common relationship is one-to-one (1:1) (Rahm & Bernstein, 2001). Cardinalities can be viewed as a constraint on mappings for all elements within a matching problem. 1:1 mapping is the most straightforward case that typically only displays the most confident match. In 1:n this technique is also applied, making these two the most used and investigated cardinalities. With the allowance of multiple mappings from an element, the complexity increases since the number of mappings related to an element can vary. Also taking into account the semantic diversity with n:1 and n:m, "(...) it is difficult, if not impossible, to automatically derive all correct mapping expressions for a given match problem." (Do, 2006, p. 38). Only very few efforts have been done on automatically deriving mappings on these problems.

2.1.4 Computing time issues

When predicting that two elements correspond to each other, it is to be expected that there are no better matching elements. This means that an element in one schema must be compared to all elements in the other schema, resulting in quadratic complexity (Do, 2006). With small integrations this should not be a problem, however it is something to be looked at when integrations grow. Furthermore, matching situations that have cardinality n:1 and no restrictions, may require computing 2^n comparisons which is intractable at high n (Gal, 2011).

2.2 Integration characteristics and issues at eMagiz

eMagiz offers their services to a wide variety of customers does not demand a certain way of notation or structuring of the integration schemas. Customers are spread across the Transport and Logistics, Building and Industry, Energy, Food and other sectors, which all apply different standards of notation and technical terms. There are initiatives that generalize the way of notation by creating a standard, such as the Open Trip Model, but only a part of the customers applies this standard and it is mostly sector specific. Together with all different commercial systems that are in use, a wide variety of notation is present that can be viewed as customer specific or even system specific. This causes similar terms to be used in different contexts and with different meaning.

Besides the customer specific standards that are applied across schemas, there are multiple characteristics that can define a schema matching problem in the eMagiz iPaaS. The first characteristic we mention comes with the different notation standards, where it is common to use English or Dutch. A different language can be a real barrier to a schema matcher since the semantics can be non-overlapping, which can only be overcome with a multi-language dictionary or thesaurus (Do, 2006). This could also be a good technique to interpret abbreviations or maybe expand them as a preprocessing step.

The structure and size of integration schemas are heavily dependent on the systems at use and the information to be transferred. There are integrations with only a single attribute and entity, but also with over a 1000 attributes. As discussed in the previous subsection (2.1.4), a higher number of schema elements makes the computation time grow exponentially and leaves more room for mistakes or disputable matching scenarios. The ratio of source and destination elements are also not constrained, although it is common to have roughly equal parts. Nevertheless, integrations that have for example 350 source and 20 destination elements, do not require all 350 elements to be mapped to the 20 destination elements. It really depends on the context and meaning of elements whether they should be considered for mapping at all. Besides the length of schemas, an extra characteristic can be the depth of entities. Entities can have multiple entities below or above them, all related to each other. There does not have to be a correspondence between the depths used at source or destination schemas. The depth can give away some information about element relations, but also can be totally non-functional.

A last characteristic of the eMagiz integration environment is that there is no constraint on matching cardinalities. It is really up to the matching problem whether a 1:1 or maybe n:m is needed.

2.3 Performance measure of Schema Matching

To validate the results of the solution design, we need measures that describe the quality of the situation and solution as it is. The aim of this section is to obtain relevant measures that are often used in similar studies, by which we can later assess our solution performance. Before we introduce several performance measures, we first sketch the situations that can occur regarding binary classification, when matching is done automatically. Figure 2.1 displays a field of matches, of which the real matches in the circle on the left and the automatically generated

matches are in the circle on the right. To obtain the real matches, the matching task first has to be manually solved, after which it can be used as a "gold standard" (Do, Melnik, & Rahm, 2003). The placed letters depict the following four situations:

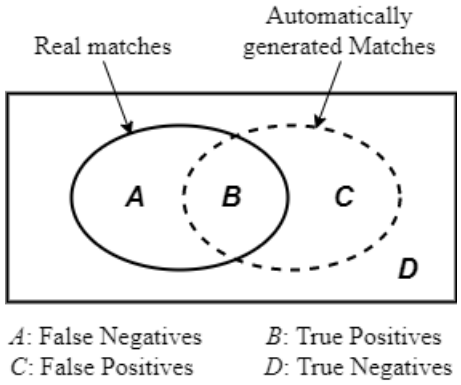


Figure 2.1: Visualization of matches

A: The match that needs to be generated is classified as negative by the auto-mapper, meaning that it is not regarded as a match. This means that the case is classified as False Negative, which is a situation that is unwanted but does not do much harm when integrating systems. It simply leaves the task to the user to notice and establish the match.

B: The real match corresponds with the automatically generated match, which is exactly what is to be aimed for. This situation is classified as True Positive and creates a correct mapping without human intervention needed.

C: An automatically generated match does not correspond to a real match and is thus classified as False Positive. This is the most unwanted situation, since creating this mapping requires human intervention to undo it and search for a proper match. The net result is that the mapping actually takes longer than without automation. Counterproductive behaviour of a solution design is labelled as a no-go by the users of the eMagiz platform.

D: This depicts the situations where no matches in reality are equally regarded by the auto-mapper, which is classified as True Negative. It always useful to have a correct estimation, but it does not do much to the actual goal of automating mappings since no mapping is created.

The principles of binary classification are used in measuring the quality of automatically generated matches. We use the labels proposed in Figure 2.1 for describing these measures. Three commonly used quality measures in Information Retrieval are *Precision*, *Recall* and *F-measures* (Van Rijsbergen, 1979). They are defined by Do and Rahm (2007) as follows:

$$Precision = \frac{|B|}{|B| + |C|} \tag{1}$$

(Reflects the share of real matches among all automatically generated matches.)

$$Recall = \frac{|B|}{|A| + |B|} \tag{2}$$

(Specifies the share of real matches that are found.)

$$F\text{-}measure = \frac{2 * |B|}{(|A| + |B|) + (|B| + |C|)} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

(Is the harmonic mean of *Precision* and *Recall*.)

A way to use *Precision* is to estimate the reliability of the match predictions (Anam, Kim, Kang, & Liu, 2015a). It can become 1 if false positives become zero, which is an ideal situation to strive for as it is explained in situation **C** above. Nonetheless, scoring high on *Precision* can be achieved at the expense of a poor *Recall* by returning only few but correct correspondences (Do, 2006). In this case overall performance is very low (Anam et al., 2015a). The interpretation of *Recall* is the percentage of real matches that is found (Do, 2006). When the false negatives become 0, the measure becomes 1 but this too can be achieved at the expense of others. By returning as many matches possible, *Recall* can be maximized but *Precision* is poor.

Neither *Precision* or *Recall* alone can accurately asses the match quality (Do et al., 2003). Hence it is necessary to consider both measures in a combined measure (Do, 2006), therefore the *F-measure* is considered. It is used for estimating match quality (Do et al., 2003). In the *F-measure* formulation above, *Recall* and *Precision* are weighted equally. Another formulation proposed in Do et al. (2003) introduces the use of α to shift importance. When $\alpha \to 1$, no importance is attached to *Recall*. When $\alpha \to 0$, no importance is attached to *Precision*. The formula is denoted as follows:

$$F\text{-}measure(\alpha) = \frac{|B|}{(1-\alpha) * |A| + |B| + \alpha * |C|} = \frac{Precision * Recall}{(1-\alpha) * Precision + \alpha * Recall} \tag{4}$$

Just like the *F-measure*, the *Overall* also is a measure that aggregates *Precision* and *Recall* (Euzenat & Shvaiko, 2007). *Overall* is specifically designed in the schema matching context and embodies the idea to quantify the post-match effort needed for adding False Negatives and removing True Positives (Do et al., 2003). Therefore the *Overall* is always lower than the *F-measure* and ranges between [-1, 1]. It is the ratio of the number of errors on the size of the expected matches (Euzenat & Shvaiko, 2007). The value of *Overall* can become negative if the number of false positives exceeds the number of the true positives (Do et al., 2003). This depicts that the proposed matching results are not worth the effort.

$$Overall = 1 - \frac{|A| + |C|}{|A| + |B|} = \frac{|B| - |C|}{|A| + |B|} = Recall * (2 - \frac{1}{Precision}) \tag{5}$$

Euzenat and Shvaiko (2007) proposes two less used measures called *Fallout* and *Miss*. The *Fallout* measures the percentage of false positives over the incorrect matches and is the opposite of the true negative rate in Formula 6. It measures the probability that an irrelevant match is discovered by the automated mapping tool (Bonifati & Eds, 2011). *Miss* measures the percentage of false negatives over the non generated matches and is the opposite of *Recall*: 1 - *Recall*. Both have an emphasis on the mistakes that are made.

$$Fallout = \frac{|C|}{|C| + |D|} \tag{6}$$

Of the proposed quality measures, we turn our focus to the *Precision*, *Recall*, *F-measure(α)*, *Overall* and *Fallout*. As described earlier, having the first two measures high is desired, but therefore an aggregation is needed in the form of the *F-measure(α)*. We use the variant that takes into account the α since true positives hurt our solution more than false negatives. The *Overall* measure is used due to its focus on measuring the effort required to fix the alignment, giving us more insight in the impact after generating matches. The last measure, *Fallout*, helps us identifying irrelevantly generated matches. *Miss* is left out for this research since it is the opposite of *Recall* and therefore already taken into account.

2.4 Conclusions RQ 1

RQ 1 What are the challenging characteristics of a schema matching problem and how can we measure its performance?

Schema matching is defined by the different heterogeneities that can be encountered, which make it an inherently difficult problem. The heterogeneities, or conflicts, can be classified into metadata-level conflicts and instance-level conflicts. As the names suggest, the conflict arises on the type of information is used, which in our validation case of the eMagiz iPaaS only metadata-level conflicts are considered due to the absence of instance data. Literature points out 4 different and common metadata-level conflicts which are about interpretation, notation practices, constraining or levels of detail. When applying this knowledge to the integrations of the eMagiz iPaaS, we find similar and specific conflicts. The notation practices are scattered by the use of different systems and standards, which is something that is not going to be solved in the near future. Also, a mixture of English and Dutch is used that can cause conflicts in both notation practices and interpretability. The level of detail, which includes splitting of strings or structuring entities and related attributes, also varies widely across different integrations. There is no correspondence between the depth and length source and destination schemas can have.

Cardinalities in schema matching can have a big impact on the complexity of the problem. Out of the four different cases, the 1:1 and 1:n are the most applied techniques because it requires the logic of only representing the most confident match. The other two cases, n:1 and n:m are said to be difficult or nearly impossible to derive all correct mapping expressions for a given match problem (Do, 2006). Very few efforts have been done on these cases, indicating the complexity that comes with it. In the eMagiz iPaaS there is no constraint on how many mappings a source or destination element can have related to them. It differs per integration whether one of the four cardinalities is applied.

The type of cardinality related to the problem and the length of source and destination schemas can have a very large influence on computing times. All combinations of schema elements have to be considered, which grows exponentially with schema sizes. At eMagiz, integrations do not have limits with respect to their schema size, and can vary from only a couple to over a thousand elements.

Predictions of a schema matcher are typically assessed on performance by the four different categories of a confusion matrix. The percentage of correct mappings out of all predicted mappings is defined as precision. This gives an indication of the share of false positive cases, which should be avoided most since these require negative user intervention. The recall displays the percentage of correct mappings found out of all mappings that should have been predicted. A harmonic mean of the two is given in the form of an f-measure, and is typically used for easy comparison. Likewise, the overall score is a dedicated measure for schema matching that quantifies the post-match effort needed for adding false negatives and removing true positives. While the first three have a range of {0, 1}, the overall score has a range of {-1, 1} with negative values indicating that the automatic matching procedure is not beneficial.

3 Literature Review

This chapter serves its purpose in describing and summarizing relevant researches with overlapping topics, to create a theory-ingrained base for our solution design. It is structured in three parts, that all cover a different individual research question. The chapter starts with defining a theoretical framework for IA in the schema matching context in Section 3.1. Section 3.2 describes the machine learning methods that are relevant in schema matching, which could possibly be fitted into the defined framework. The last part, Section 3.3, covers the SLR methodology for summarizing past researches in similar contexts. Per section, **RQ 2**, **3** and **4** are answered respectively.

3.1 A theoretical framework of Intelligence Amplification in the schema matching context

In this first part of this chapter, we use several definitions and frameworks published in literature to construct our own framework relevant to the context. We develop an answer to **RQ 2**: *What aspects define Intelligence Amplification in the schema matching context?*, with the use of the upcoming subsections. First, we touch upon the concept of IA (Section 3.1.1) and corresponding popular frameworks (Section 3.1.2). With this overview, we place IA in the context of schema matching in Section 3.1.3, which gives us an opportunity to transform the earlier found frameworks into one specific to schema matching.

3.1.1 Concept of Intelligence Amplification

The starting point of the now known concept of Intelligence Amplification (IA) was initiated by Ashby (1956). He reasons, by giving a few examples, that if the power of selection can be amplified, intellectual power as well can be amplified. This demarcates the bare foundation of Artificial Intelligence (AI), by stating that "what is commonly referred to as 'intellectual power' may be equivalent to 'power of appropriate selection'" Ashby (1956, p. 272). Licklider (1960) takes this thought further by placing it in the context of the then emerging electronic computers. He describes a vision, along with the prerequisites, that could realize a man-computer symbiosis. Estimations are made on what contributions men and equipment would make in such an anticipated symbiotic association. Based on the findings of this paper and related others, Engelbart (1962, p. ii) describes a "new and systematic approach to improving the intellectual effectiveness of the individual human being". The paper captures the process of augmenting human intellect by "increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems" Engelbart (1962, p. 1). Progression in the last decades has provided many new possibilities of the computer's role, which fulfilled the prerequisites of Licklider (1960). A more contemporary description of the subject is given by Pan (2016, p. 411) who states: "hybrid intelligence systems are formed by cooperation between computer and humans so as to form an augmented intelligence of '$1 + 1 > 2$'".

The concept of AI is very broad and boundaries are somewhat vague, but it contrasts with the form of enhancement used in IA (van Breemen, Farkas, & Sarbo, 2011). In AI, the machine is designed to mimic and replace the cognitive abilities of human intelligence, whereas IA seeks to establish a symbiotic relationship between a human and an intelligent agent (Dobrkovic et al., 2016). To make it more intertwined, when choosing a machine learning technique for the intelligent agent, we incorporate a part of AI in our IA solution.

In literature oriented around the subject of IA, we encounter various different terms that partially or wholly describe the same concept. There is not yet a widely agreed description, which could distract the reader. For sake of simplicity, we persist in using Intelligence Amplification, but literature uses (among others) the following terms: Human-computer interaction,

Augmented artificial intelligence, Human-machine systems, Human-machine symbiosis and Intelligence augmentation.

3.1.2 Intelligence Amplification Frameworks

Engelbart (1962) has taken the role of a pioneer on the field of Intelligence Amplification, by constructing a framework that stimulates a way of looking at implications and possibilities through augmenting intellect. Being the first to do this, the framework remains very conceptual but makes useful distinctions that are later widely used. The paper tries to awaken the reader that augmentation of intelligence is already present and provides great potential. Xia and Maes (2013) revisits the framework of Engelbart (1962) since computers have progressed significantly over those 50 years. Their application goal is the personal wearable devices of the 21st century. Interpretation of the framework describes three steps for the design of an IA artifact:

- Step 1: Consider the desired state after augmentation
- Step 2: Identify the processes for the task
- Step 3: Identify how artifacts can change a process or the process hierarchy (Xia & Maes, 2013)

These steps can be seen as the flow in a research of this interest, but are still very broad and require multiple sub steps to provide answers.

Although not explicitly mentioning the term IA, the paper of Parasuraman et al. (2000) is closely related by describing levels of human interaction with automation. Importance of appropriate automation level selection is emphasized due to the impact that it can have on human activity and coordination demands on the human operator. The paper mentions four broad classes of functions that can individually be automated to a certain degree: information acquisition, information analysis, decision and action selection and action implementation. Information acquisition encompasses the sensing and registration of input data, where at high levels filtering is applied to only show relevant information to the operator (Parasuraman et al., 2000). The paper describes the second class, information analysis, by having a purpose of augmenting human operator perception and cognition. Lewis (1998) presents a more complex form called "information managers" that provide context-dependent summaries of data to the user (Parasuraman et al., 2000). Decision and action selection comprises selection among decision alternatives, of which the corresponding levels are displayed in Table 3.1. Although given as an example for decision and action selection, it serves as a generic model for all types of classes. The last class of action implementation refers to the execution of the chosen decision alternative. After proposing the model of different levels of automation and function classes, the paper contributes a flowchart to the automation implementation, that emphasizes primary and secondary evaluation criteria. These criteria help inform the designer what the impact of such an automation can have on both parties.

The paper of Dobrkovic et al. (2016) aims to provide an IA framework dedicated to a diverse application in decision making processes. The paper states that tasks requiring creativity are ideally destined for humans, and tasks marked by a high computational need ideally are suited for intelligent agents. This leads to a trade-off between high complexity, low workload problems and low complexity, high workload problems. To cope with situations that are all over this spectrum, a set of rules that can be interpreted as roles are proposed that build directly upon the vision of Licklider (1960). A visualization of these rules is given in Figure 3.1 to illustrate the hierarchical organization of the human-machine partnership. This description of the symbiosis sketches a clear task distinction easy for interpretation. However, it does not consider concepts such as the degree of automation as proposed in Parasuraman et al. (2000) or the complexity and capability of the AI. The paper makes a couple of key assumptions, leading to a strict

Table 3.1: Levels of Automation of Decision and Action Selection, retrieved from Parasuraman et al. (2000)

	LoA	Automation description
High	10	The computer decides everything, acts autonomously, ignoring the human
	9	Informs the human only if it, the computer, decides to
	8	Informs the human only if asked, or
	7	Executes automatically, then necessarily informs the human, and
	6	Allows the human a restricted time to veto before automatic execution, or
	5	Executes that suggestion if the human approves, or
	4	Suggests one alternative
	3	Narrows the selection down to a few, or
	2	The computer offers a complete set of decision/action alternatives, or
Low	1	The computer offers no assistance: human must take all decisions and actions

classification of tasks being assigned to either a human or an AI. Nevertheless, given certain AI designs, possible patterns invisible to humans can be discovered, blurring the distinction between complexity and creativity. In Engelbart (1962) this is called a composite process.

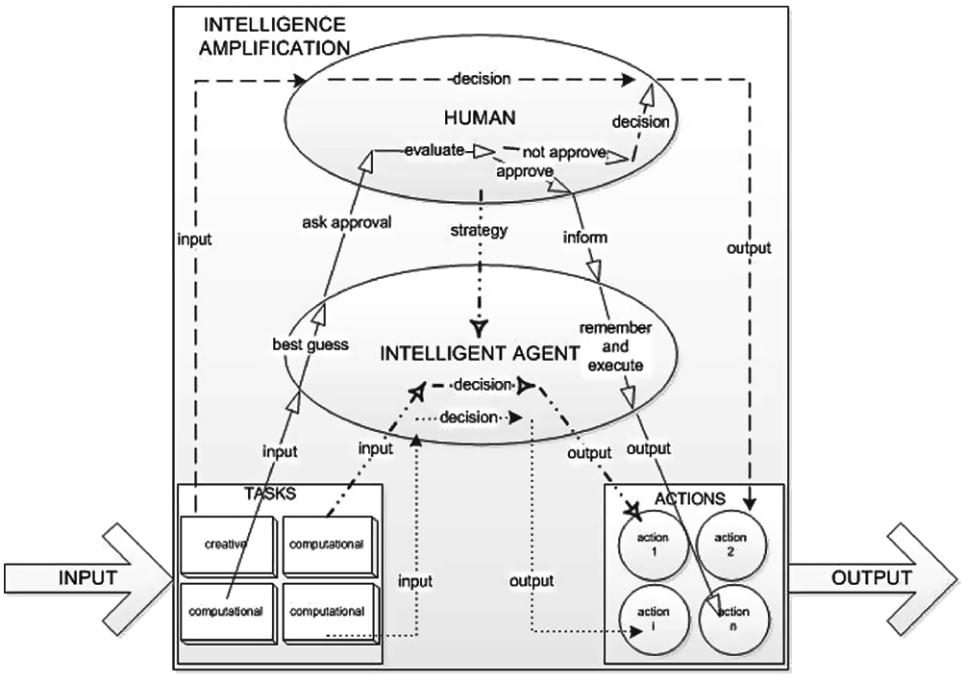


Figure 3.1: Hierarchical organization of the human-machine partnership depicting the information flow and the task division, retrieved from Dobrkovic et al. (2016)

In Zheng et al. (2017) a basic framework is proposed for human-in-the-loop (HITL) hybrid-augmented intelligence. "HITL hybrid-augmented intelligence is defined as an intelligent model that requires human interaction" (Zheng et al., 2017, p. 154) and follows all characteristics of an IA solution. The framework is constructed from the perspective of the limitations that come with machine learning methods. Figure 3.2 gives a representation of this basic framework, to which it should be noted that different systems should be constructed for different fields (Zheng et al., 2017). The paper considers the retrieved HITL hybrid-augmented intelligence as a loop of human intelligence transfering knowledge learning to artificial intelligence, which provides collaborative feedback to the human. The loops of feedback and tuning by the human are clearly visible in Figure 3.2 and contribute to the artificial intelligence. These imply a

learning aspect to the hybrid-augmented intelligence. This is something that is considered in the hierarchical organization of Dobrkovic et al. (2016) as well, but here 'training and validation' is not take into account. From a first sight comparison, the knowledge base also seems to be missing, and only part of this is backed by a set of strategic goals created by the human. Also the composing of new knowledge is missing, which is a feadback loop originating from output results. The knowledge base of Zheng et al. (2017) gives more intuitive visualization of which aspects account for a decision by the intelligent agent.

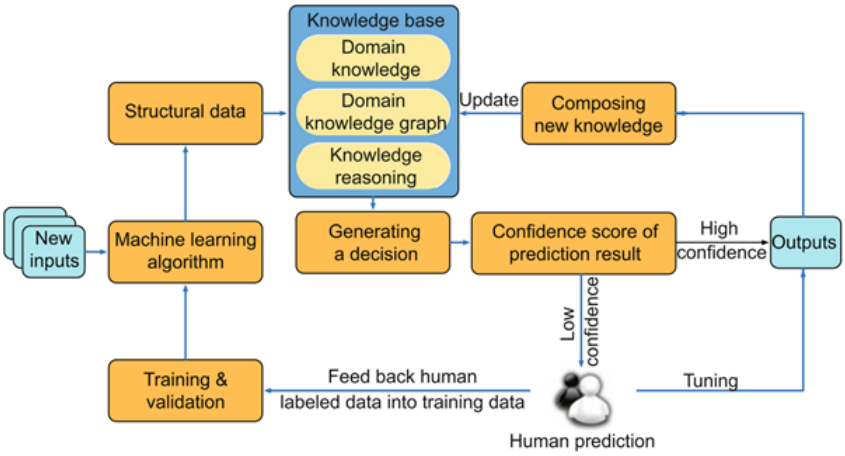


Figure 3.2: Basic framework of human-in-the-loop hybrid-augmented intelligence, retrieved from Zheng et al. (2017)

3.1.3 Combination of Intelligence Amplification and schema matching

So far we have seen various proposed IA frameworks that describe a human-machine symbiosis (Dobrkovic et al., 2016) (Zheng et al., 2017), emphasize the impact of implementation (Engelbart, 1962) (Xia & Maes, 2013) and describe the possible levels of automation in different stages (Parasuraman et al., 2000). In the quest to find an answer to "What aspects define Intelligence Amplification in the schema matching context?", we feel the need to develop a generic IA framework dedicated to schema matching. This framework can be used for any degree of automation and has a learning characteristic. In the next paragraph we elaborate on how we incorporated the discussed frameworks, resulting in the framework of Figure 3.3.

As discussed in Section 1.1, Rahm (2011) defines a general workflow for automatic, pairwise schema matching systems. We found that these four steps comply to the four-stage model of automated information processing described in Parasuraman et al. (2000). Combining the similarities to obtain a dedicated four-stage model for schema matching, we propose the following four stages:

1. *Interpretation*, a collection of the aspects necessary to make sense of the provided input schemas.

2. *Evaluation*, this stage takes the interpretation of the input schemas and looks at the similarity between selected options. By doing this, relations in various degrees are made visible between the starting element and candidate elements.

3. *Selection*, combines the results of the evaluation stage and selects the most promising match.

4. *Creation*, the last stage to finalize the process and to create the actual mapping that needs no further correction.

At all four stages, different levels of automation are possible, adjusted to the preferences of the user. Table B.1 in Appendix B displays the levels of automation applicable to our framework. The levels are equal to all stages, except for a conjugation of the stage name which can be interposed at the '(...)'. With all possible levels of automation, schema matching models that follow this same framework can still be very different or unique to work with and apply to many situations.

With only a description of the four stages and their automation levels, we have a very static model. What we aim for in this research is a solution that interacts with the user in order to create faster solutions and to learn, so it can stay up-to-date and transform along with the user context. Therefore, we made sure that framework of Zheng et al. (2017) fitted our stages nicely. There are two feedback loops securing a dynamic and learning characteristic to our framework. The first one originates from the creation phase, where the process of creating a mapping can provide useful information on the accuracy (or other performance measures) of the AI. Let us take a level 5 automation for example, where the model suggests creations of mappings to the human and waits for approval. With this approval the human can indirectly classify suggestions as True Positive, False Positive, etcetera. This feedback can then be used for training the interpretation algorithm and for validating the model. The dotted line back into the interpretation stage suggest that this flow is not necessarily executed each time a mapping is created, this depends on the choices of the user or creator. The second feedback loop is set in motion after the output is delivered. All output is perceived as correct by the user, creating knowledge for assessing future similarities between elements. Here we also encounter a dotted line, indicating that the flow is not instantaneously executed but only when new input is given. Both feedback loops contribute to the knowledge base, which represents a storage place of data. Here we deviate from the paper of Zheng et al. (2017), to more clearly indicate that data intended for new knowledge and for training and validation are stored for the same purpose: learning from knowledge.

The framework from Figure 3.3 leads to answering the three steps for the design of an IA artifact as described by Xia and Maes (2013). Based on the chosen levels of automation at the different stages, all lines to arrange a symbiotic man-machine relationship proposed by (Dobrkovic et al., 2016) are satisfied. Only in the case that stages are fully automated (level 9 or 10), the human cannot overrule the AI.

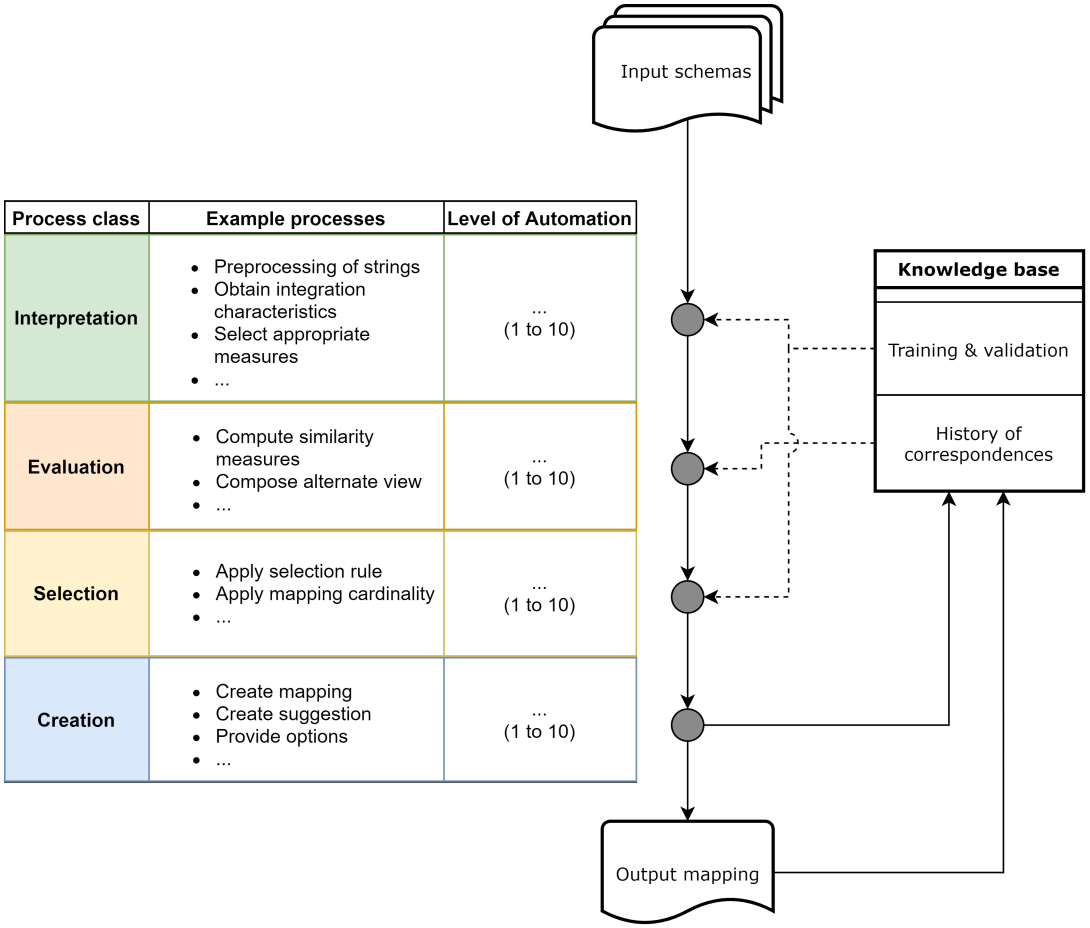


Figure 3.3: Generic IA schema matching framework

3.2 Machine learning and schema matching

In the book of Bonaccorso (2017, p. 9), machine learning is described as: "[..] an agent (which is a software entity that receives information from an environment, picks the best action to reach a specific goal, and observes the results of it) adopts a statistical learning approach, trying to determine the right probability distributions and use them to compute the action (value or decision) that is most likely to be succesful (with the least error)." There are broadly three classes of machine learning, which are described in the paragraphs below together with their relevance to schema matching. With this we answer **RQ 3**: *What machine learning techniques are suitable for schema matching?*

We want to deal with machine learning since schema matching is largely seen as a classification problem, based on stochasticity. Machine learning has a lot of overlap with statistics and statistical learning, although statistical learning focuses on models and their interpretability, which is less important in machine learning where mostly only results count (Groothuis-Oudshoorn, 2020). The learning aspect is desired for capturing human knowledge in a digital environment to use it any time.

Schema matching in the end simply comes down to selecting the appropriate destination element to the source element. This can be viewed as a classification problem or a decision making process. In relation to machine learning, schema matching requires some training data with included desired outputs, either from historical matches or from human input. Human intervention remains needed since schema matching cannot contain mistakes, and machines are never flawless. The number of variables used varies and is up to the user or problem context. The variables, often similarity measures, provide the base input on which the machine

learning techniques must recognize patterns. The size of the schema matching problem is heavily dependent on the context, which could make some techniques more suitable than others.

3.2.1 Machine learning paradigms

There are three main paradigms or categories in which a machine learning technique can be classified. Based on characteristics such as human intervention, input data and core tasks, we distinguish Supervised learning, Unsupervised learning and Reinforcement learning.

Supervised learning can be compared to a teacher or supervisor, that provides an agent with a precise measure of its error (Bonaccorso, 2017). The correct output is known for each training example, which is used to learn predicting and adjust when necessary (Mocanu, 2020b). This is done until a certain level of accuracy is reached. Supervised learning has a wide area of academic and industrial applications, which makes it the most used out of the three paradigms. Characteristic tasks are classification and regression, predicting an outcome as a discrete category or a continuous value respectively (Englebienne, 2019). Classification uses one input vector to predict one of a finite number of categories (Bishop, 2006). The continuous ouput of regression would be useful in cases such as predicting the temperature, pressure or tomorrow's stock prices. Schema matching can easily be described as a classification but hardly as a regression problem. Many of the solutions in literature and practice are in some form of supervised learning as we will discuss in Section 3.3.

In contrast to supervised learning, unsupervised learning lacks the presence of a supervisor and consequently absolute error measures. The application area is therefore different, the paradigm makes itself useful for discovering clusters, structures and similarities between sets of elements (Bonaccorso, 2017). Because correct outputs are not known in this case, as it is not part of the core tasks, the training data also does not include desired outputs. Application areas can for example be object segmentation (movies, songs, product, etc.) or automatic labeling. In the context of schema matching this can have very little relevance intuitively, although there are exceptional cases and methods imaginable but likewise irrelevant enough to discuss them here.

The last main paradigm, reinforcement learning, is based on reward and learns to maximize payoff (Mocanu, 2020b). This reward can be given by the user or through feedback provided by the environment. Learning is established in a combination of collected data and given rewards. The paradigm assumes that the world is Markovian, which describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event (Joy, n.d.). In schema matching this assumption does not hold, each element comes with its own individual situation. Matching of elements are not necessarily related to each other, although the correct mapping of one element could provide information for the mapping of another. Nevertheless, this is not a structural given thing so we cannot view the schema matching problem as a sequence of related choices. Reinforcement learning is best applied when multiple decisions need to be taken for short-sighted maximization of payoff, when the environment is not completely deterministic (Bonaccorso, 2017). Some concepts that benefit from reinforcement learning are decision making processes, strategic optimizations and on-line learning. Although in schema matching a model could definitely learn from user feedback, the environment is deterministic and cannot respond to the actions of the model. Similar to unsupervised learning, we could come up with an exceptional design that would be counter-intuitive to the ones proposed in literature. If you would see each element to be mapped as a decision that depends on the choices made at other elements, a state space would be created that grows exponentially and with large numbers. The rearrangement of a schema matching problem to be able to be processed by a reinforcement learning technique would be best described as cumbersome. Since there are no related papers written on the one-step prediction process of schema matching being solved with reinforcement learning, we do not address this paradigm as a relevant solution for schema matching.

3.2.2 Relevant techniques to schema matching

In the previous section we discussed several paradigms and their applications, to which we can conclude that schema matching is a classification problem that lends itself for supervised learning. The main requisite for supervised learning, a set of labeled samples, is logically present at the developers of an iPaaS and we simply want to classify cases into one of two discrete categories: 'match' or 'no match'. We therefore describe the most popular supervised learning classification techniques and discuss their application in this research.

- **Support Vector Machines**: From the shortcomings of linear classification models, Support Vector Machines (SVM) and Neural Networks as well are developed to deal with complex situations (Bishop, 2006). With the use of a kernel function, data can be analyzed in a higher dimension without needing to compute the entire higher dimensional state. In this higher dimension linear classification can be applied, which together with the kernel function is called the kernel trick. Strengths of an SVM is that the solution space is convex, so there are no local optima. It also scales relatively well to high dimensional data (Mocanu, 2020a). A weakness is that you need to choose a good kernel function and it does not provide probability estimates. The application of this classification technique is very wide, ranging from classifying images to outlier detection.

- **Decision Trees**: Decision Trees are an easy to interpret tool for classification, and are based on a set of rules applicable to the dataset. In this tree of rules, decision nodes define the various paths leading to the final classification. These decision nodes specify a test on a single variable. Although easy interpretation and fast performance are often the reasons to apply a decision tree, there are some weaknesses. Decision trees make hard splits based on a single variable, quickly resulting in a complex model when combinations of variables are regarded. With this, the pro of easy interpretation can be lost. (Bishop, 2006). The application of decision trees is very wide, can be quick, but is best when insight in the decision process is desired and the model is not complex. Unfortunately, this is not the case for this research, since the data possesses very different situations and can make the model grow very large. Also, there is no prerequisite that the model should be easy to interpret, the importance lies on correct outcomes.

- **Neural Network**: A neural network model is basically a nonlinear function from a set of input variables to a set of output variables, controlled by a vector of adjustable parameters (Bishop, 2006). A frequently used form of is a MultiLayer Perceptron (MLP), which is defined by a number of layers consisting of nodes that are all of the same type within each individual layer. Parameters of each node are updated after a specified number of training samples, which is called back-propagation. Interpretation of the model is poor compared to other techniques, and can be described as a 'black box'. On the other hand, it can handle great complexities and can be applied to a large variety of cases. The black box property of the model is no problem to our research case, and the complexity of the data set is no problem. In fact, discovering patterns in data that can be in different languages, knows abbreviations and specialist terms requires a powerful technique.

- **Naive Bayes**: The naive bayes classifier is built upon Bayes rule, plus having the naive assumption that the distributions of the input variables are independent (Bishop, 2006). The assumption is helpful in cases where the dimensionality is high and it can be applied if input vectors both contain discrete and continuous variables. Its performance is better than other models when training data is scarce, under the assumption that independence between variables holds. Nevertheless, if a categorical variable was not present in the training set, the model does not assign any probability to it. Just like the other classification techniques, application is very wide. Naive bayes classifiers know many implementations in text classification and lends itself well to our research case.

3.3 Solutions in practice

The last past decades more and more research has been done on the problem of schema matching, since the problem appears more frequently with the current speed of digitization. With the emergence of machine learning, it was directly used for research on this topic. This gives us much cases to learn from, to adapt to our situation or to improve upon. **RQ 4**: *What machine learning solutions have already been applied relevant to the research context?* is answered with the use of a Systematic Literature Review (SLR) as described by Kitchenham and Charters (2007). The research question describes our objective and demarcates the set of problems we are looking for.

3.3.1 Preceding research on schema matching in the eMagiz platform

The exploratory approach of the preceding research, Buis (2017) and Piest, Meertens, Buis, Iacob, and van Sinderen (2020), provides some useful findings and recommendations to our exact similar case. Their procedure is based on the approach of Rahm (2011), which consists of a pre-processing stage and a matching stage both executed by human and machine.

Their reference architecture of the pre-processing stage is designed for extension, but due to the scope only lower-casing the string is applied. It starts with indicating preferences of the human, which could be necessary for any later added pre-processing techniques. After lower-casing the strings, similarity measures are added to all possible candidate mappings. For assessing the similarity, the Levenshtein distance and JaroWinkler are implemented, which both fall in the category of string similarity measures.

A complete overview of possibilities with their corresponding similarity measures are sent to the matching phase, consisting of a Neural Network constructed in Azure Machine Learning Studio. The choice of the machine learning technique is substantiated via experimentation with the most common and popular techniques. The software agent provides the tools for easy implementation of the user's design, although coming at a cost for usage per hour.

The validation of the model showed that the neural network improves the schema matching process time wise, but this heavily depended on the input. The use of different languages in schemas caused a poor performance, which could be encountered by certain pre-processing steps or different training of the neural network. The used similarity measures alone provided unsatisfactory results since these cover only a small part of what could be analyzed. Therefore, the use of hybrid matchers is recommended, that could for instance look at structural similarities. Entities were not taken into account in this research, only the matching of attributes. When incorporating these, it could provide useful information in matching both.

3.3.2 Systematic literature review on machine learning solutions in schema matching

The first two components of a review protocol as described by Kitchenham and Charters (2007), the background and research questions, are already explained in the section above. Next we define a search query that results in retrieval of a fairly complete set of primary studies, when applied to appropriate databases. Derived from the research question, the keywords *schema matching* and *machine learning* are a straightforward choice. Thereby, we include synonyms or subsets that are often used in articles, books and professional sources. Although relevance of unsupervised learning and reinforcement learning is questioned earlier, we still include these for getting a broader range of solutions. The part in our research question indicating solutions should be relevant to the context of this research is left out for now, since relevance is best assessed when reading the study itself. We do not want to reject apparent irrelevant studies upfront, wasting possible valuable information. An overview of the keywords that are inserted in our search query is given in Table 3.2.

Table 3.2: Search query keywords

Keywords	**Schema Matching**	**Machine Learning**
Synonyms	Schema Mapping	Supervised Learning
		Unsupervised Learning
		Reinforcement Learning
		Statistical Learning

Search query

For this SLR we use three scientific databases: Scopus, Web of Science and IEEE. These are large databases, recommended by the University of Twente and are able provide us with plenty trustful studies. The search query used can be simplified to the form of: ("Schema matching" OR "Schema mapping") AND ("Machine learning" OR "Supervised learning" OR "Unsupervised learning" OR "Reinforcement learning" OR "Statistical learning"). This gives us all possible combinations between the keywords we use. In Scopus, these keywords must be present in title, abstract or keywords. The query in Web of Science is applied for topic, which implies the same as title, abstract or keywords in Scopus. In IEEE a search was done in all metadata.

Selection procedure

Table 3.3 displays the upcoming procedure of selection relevant literature. When applying the search queries in the specified we start with a set of 156 studies, which we need to narrow down to a selection that can serve as an answer to our research question. We start with a set of inclusion criteria that can still be applied to the results in the three databases using filters. These criteria make sure that resulting studies have a certain academic standard and are written in a language that we can thoroughly understand. The inclusion of certain years of publication was not needed, all papers were published in a contemporary era where machine learning was present. The inclusion criteria are the following:

- Studies must be peer reviewed and written in English
- The study area must be related to Computer Science, Engineering or Information Systems
- Source type must be Journal Article, Conference Proceedings, Book Series or Book

Next, we remove all duplicate results, which is a common effect when using multiple databases. From the resulting unique studies, we read the title and abstract to give us an indication of the content and aim of the study. Based on this indication we discarded irrelevant studies, since the set of studies was too large for substantive assessment. The resulting set however was small enough for assessing the relevance based on reading through the whole study. Some studies could not be accessed so these were discarded too. After a quick read through of all remaining studies, we could make up the full relevance and keep the desired ones. Also, we were now able to draw up exclusion criteria, in order to make a last selection of the studies we want to assess and possibly use for this research. The applied exclusion criteria are:

- The paper makes use of data that is only relevant to a certain expertise or study area, irrelevant to ours
- Instance data is used for matching schemas
- No guideline, execution and results of the proposed solution are given
- Poor model performance

As a last task in our literature selection procedure, we look at the most referenced high regarded papers. The widespread ideas of the methods called GLUE, LSD, COMA++ and Similarity Flooding are taken into account because these are commonly used for comparison. After assessing their relevance, only the paper that proposes LSD could be included.

Table 3.3: Literature selection procedure

Activity	**Result**	**Number of Articles**
Execution of search query in Scopus, WoS and IEEE	+156	156
Apply inclusion criteria	-21	135
Remove duplicates	-31	104
Exclusion based on title and abstract	-62	42
Full text not available	-4	38
Not relevant after quick reading	-8	30
Apply exclusion criteria	-13	17
Related referenced papers	+1	18
Total number of references		**18**

Quality assessment

We want to get an overview of the different research purposes, methods and outcomes they produce, which is displayed in Table 3.4. The three research methods encountered are literature review (LR), observation (O) and experiment (E). For the produced output types, the categories are new theory (T), conceptual model (CM), artefact (A) and implemented artefact (IA).

What can be seen from this overview is that the majority of the references is similar structured in their purpose, method and output. This majority generally presents a new view on the schema matching context in the shape of an artefact. Rahm and Bernstein (2001) present a taxonomy and basically an overview of what is common in the field of schema matching. Some references are follow-ups, an idea presented in the first and developed further in the second. Because the second of the related references is contributing something new, these are kept in our systematic literature review.

Table 3.4: Quality assessment

	Reference	Research Purpose	Research method			Output			
			LR	O	E	T	CM	A	IA
1	Jeong et al. (2008)	"To this end, we present a supevised approach to measure semantic similarity between XML schema documents, and, more importantly, address a novel approach to augment reliably labeled training data from a given few labeled samples in a semi-supervised manner."	X		X	X	X	X	
2	Rahm and Bernstein (2001)	"We present a taxonomy that covers many of these existing approaches, and we describe the approaches in some detail."		X		X		X	
3	Marie et al. (2005)	"We present a boosting algorithm for schema matching with a unique ensembler feature, namely the ability to choose the schema matchers that participate in an ensemble."			X	X		X	X
4	Berlin and Motro (2002)	"In this paper we describe a system, called Automatch, that uses machine learning techniques to automate schema matching.			X	X	X	X	X
5	Anam et al. (2015b)	"In this research, we introduce a Knowledge-based Schema Matching System (KSMS) that matches schemas both at the element level and structure level and produces the final result."	X		X	X	X	X	X
6	Anam et al. (2010)	"This research proposes an incremental schema mapping method that employs Ripple-Down Rules (RDR) with the censored production rules (CPR)."	X		X	X		X	X
7	Mukherjee et al. (2020)	"[...] we propose a probabilistic framework for schema matching based on learning. [...]. We present a solution based on stochastic EM [...]."	X		X	X	X	X	X
8	Marie et al. (2008)	"We propose a model for schema matching, based on simple probabilistic principles. We then propose the use of machine learning in determining the best mapping and discuss its pros cons. Finally, we provide a thorough empirical analysis, using both real-world and synthetic data, to test the proposed technique."			X	X	X	X	X
9	Cheng et al. (2010)	"We exploited the observation that XML mappings have high degree of overlap, and proposed the block tree to store common parts of mappings. A fast method for constructing the block tree was proposed."	X		X	X	X	X	X
10	Berkovsky et al. (2005)	"We propose the use of machine learning algorithms to learn the optimal weight assignments, given a set of schema matchers. We also suggest the use of genetic algorithms to improve the process efficiency."	X						X
11	Oldham et al. (2005)	"We describe the architecture, implementation, and functionality of MWSAF as well as our machine learning approach to improve MWSAF in this paper."		X		X			X
12	Anam et al. (2015c)	"[...] we propose a hybrid approach, called Hybrid-RDR, which combines a machine learning algorithm with ripple-down rules (RDR), an incremental knowledge engineering approach."	X		X	X	X	X	X
13	Sahay et al. (2020)	"We explored existing methods [...] and proposed a hybrid approach framework that uses both provided data and the schema name to solve a one-o-one schema matching problem."	X		X	X	X		X
14	Schmidts et al. (2019)	"In this paper, we provide an approach based on concept name learning from known transformations to discover correspondences between two schemas."	X		X	X	X	X	X
15	Duchateau et al. (2009)	"In this paper, we propose YAM, which is actually not Yet Another Matcher."			X	X	X	X	X
16	Gal and Sagi (2010)	"In this paper we provide a thorough investigation of this research topic. We introduce a new heuristic, Schema Matcher Boosting (SMB)."	X		X	X		X	X
17	Rodrigues et al. (2015)	"In this paper we study the adoption of an active learning technique as an alternative to properly combine matchers in schema matching methods."	X		X	X	X	X	X
18	Doan et al. (2001)	"In this paper we describe the LSD (Learning Source Descriptions) system that uses and extends machine learning techniques to semi-automatically create semantic mappings."	X		X	X	X	X	X

Table 3.5: Data extraction

	Reference	Goal	Similarity measure	Technique
1	Jeong et al. (2008)	Measure semantic similarity between XML schema documents Augment training data in a semi-supervised manner	Semantic similarity	Neural Network-based Partial Least Squares
2	Rahm and Bernstein (2001)	Presenting a taxonomy that covers many existing methods Describing some in detail	Various	Various
3	Marie et al. (2005)	Choosing the right matchers for an ensemble	Various	Boosting
4	Berlin and Motro (2002)	Create an attribute dictionary based on probabilistic knowledge Automatch uses the attribute dictionary to create an optimal match	Various	Bayesian learning
5	Anam et al. (2015b)	Combining different matching algorithms using a hybrid approach Matching considering hierarchical structures	String similarity Text processing	Decision tree Similarity flooding
6	Anam et al. (2010)	Schema matching method that employs Ripple-Down Rules Performance is compared with machine learning techniques	String similarity Text processing	Censored Production Rules Ripple-Down Rules
7	Mukherjee et al. (2020)	Target-driven schema matching with the use of a knowledge graph	Token similarity based on knowledge	Knowledge graph. Stochastic Expectation Maximization
8	Marie et al. (2008)	Schema matching with a heuristic, based on a probabilistic model of matchers	Similarity matrix Measures up to user	Naïve Bayes heuristic
9	Cheng et al. (2010)	Managing and efficientely generating possible mappings	None	Block tree Probabilistic Twig Query
10	Berkovsky et al. (2005)	Measuring relative performance of schema matchers	None	Genetic search
11	Oldham et al. (2005)	Predicting the domain of a Web service based on training data	None	Naïve Bayes
12	Anam et al. (2015c)	Combining machine learning and knowledge engineering approaches	String similarity Text processing	Decision Tree Ripple-Down Rules
13	Sahay et al. (2020)	Finding out results of different machine learning methods for schema matching	Edit distance	Various
14	Schmidts et al. (2019)	Solving schema matching as a classification task with machine learning	Various, up to the user	Various
15	Duchateau et al. (2009)	Generating a dedicated matcher for a given schema matching scenario	Various	Decision Tree
16	Gal and Sagi (2010)	Tuning the ensemble selection process of schema matchers	Various	Boosting
17	Rodrigues et al. (2015)	Applying machine learning methods for schema matching	Various	Decision Tree
18	Doan et al. (2001)	Appying different learners for schema matching	String similarity Instance matching	Various

Data extraction

To answer our research question regarding the systematic literature review, data extraction is essential and displayed in Table 3.5. Here the used similarity measures and techniques are summarized as these make up for the greatest part of the solution. A comparison on performance is difficult to make, since not each paper does apply the same performance measures, different data sources are used relevant to the problem context and sometimes only relative comparisons are given. However, papers that show poor performance are already excluded during our selection procedure.

In Table 3.6 we summarize the references to fit in 5 categories of similarity measures. String similarities are most common and a lot of techniques that belong to this category are developed, such as Levenshtein distance, edit distance or Jaro-Winkler distance. In almost any case, several similarity measures are combined to make for a stronger estimation.

Jeong et al. (2008) provides a semantic similarity measure as a product of a Neural Network-based Partial Least Squares (NNPLS) method. This measure is built up from several user-chosen similarity measures related to string similarity, semantic information and structure. A prerequisite for semantic information based similarity measures is that a lexical knowledge resource (e.g. thesaurus) is absolutely required (Jeong et al., 2008).

There are multiple references that combine multiple measures since single techniques do not produce good performance for schema mapping (Anam et al., 2015b). String similarities, structural similarities, value type similarities and others can be used, which are then turned into e.g. a weighted combination (Marie et al., 2005) for a better prediction. All in all, various similarity measures together cover more facets of the schema matching problem.

Some references indicate that the use of similarity measures are really up to the situation of the schema matching problem and should be selected by the user. Others do not provide any at all, related to the type of output the paper produces. In case no artifact is implemented and tested, selecting a similarity measure for that research would not be necessary, and can better be left to the user of the schema matching problem.

Table 3.6: Similarity measures

Similarity measures	**Reference**
String similarity	5, 6, 7, 12, 13, 18
Semantic similarity	1
Various	2, 3, 4, 14, 15, 16, 17
Up to user	8, 14
None	9, 10, 11

Each research dealt with in this systematic literature review has applied one or more machine learning techniques, but application purposes can differ. A classification of these techniques is displayed in Table 3.7. Decision trees are a popular method that can be easily understood by the user and have a simplistic and flexible character (Rodrigues et al., 2015). Nevertheless, a single tree can quickly overfit the learning base causing the performance to decrease significantly with the increasing size of the tree (Anam et al., 2015b). Therefore, most researches use multiple trees, related to different matchers or domains. In Anam et al. (2015b) and Anam et al. (2010) the decision tree is made out of rules, and constantly extended when more knowledge is available.

A different technique, Bayesian Learning, is mostly implemented in the form of Naive Bayes and is reported to be a standard matcher for string similarity (Schmidts et al., 2019). This technique approaches the schema matching problem in a more probabilistic manner.

A couple of references use the machine learning technique in a different fashion. The one reference using a form of neural network, Jeong et al. (2008), uses it to construct a new semantic similarity measure as is discussed in an earlier section. Boosting is used in Marie et al. (2005) and Gal and Sagi (2010) in the context of schema matcher ensembles. A boosting algorithm

"[...] can strengthen weak classifiers to achieve arbitrarily high accuracy and has been shown to be effective in the construction of succesful classifiers." (Gal & Sagi, 2010).

The references of Cheng et al. (2010) and Mukherjee et al. (2020) use algorithms to establish a unique way of learning and capturing knowledge. The first presents a block tree to capture the commonalities among possible mappings. The second creates a knowledge graph based on stochastic Expectation Maximization (EM). This graph captures all relations between historical matched pairs.

Table 3.7: Techniques

Technique	**Reference**
Decision tree	5, 12, 15, 17
Bayesian learning	4, 8, 11
Neural network	1
Boosting	3, 16
Rules	6, 12
Algorithm	7, 9, 10
Various	2, 13, 14, 18

3.3.3 References with little relevance

In the previous section we created an overview of the solutions that are applied in the found literature. Their relevance to the topic is already shortly assessed by the selection procedure, but we have not shed light on the applicability to the context yet. Therefore we first elaborate on the papers that have little to none parts that we could apply. Subsequently, we describe the most relevant papers that can contribute to a possible solution design.

A very useful feature of researches found in a SLR is that multiple solution designs with corresponding experiments and results are proposed that could be of use to our own solution design. Unfortunately, there are researches that do not posses this feature and sometimes serve another purpose. The references of Rahm and Bernstein (2001), Berkovsky et al. (2005), Oldham et al. (2005) represent such cases. Rahm and Bernstein (2001) is formulated as a survey, describing some approaches in detail. This detail is still conceptual and refers to the related literature for an exact execution. In this SLR the paper proves to be useful for finding related papers, but does not provide explicit solution design information. Berkovsky et al. (2005) does not provide any description for implementation or design. Furthermore, together with Oldham et al. (2005) they both do not present any results, making it impossible to quantitatively assess their performance.

Closely related to the previous section, Marie et al. (2005) and Gal and Sagi (2010) miss out on their conceptual model description for a large part. They both present the same heuristic for boosting, but this covers only a small part of their solution. However opposing to the references of the previous section, they do describe their experiments and results in useful detail.

The research of Duchateau et al. (2009) presents a highly regarded framework that is further elaborated in Candidat (2009). Unfortunately, this paper is on operates on another level with its goal to generate a dedicated schema matcher. In our research, we are only interested in establishing matches with the use of a single schema matcher. However, just like Rahm and Bernstein (2001) it served as a useful overview for finding relevant researches that did not come up in our search protocol. In the same context of describing various schema matchers, (Marie et al., 2008) also describes solutions on a different level than is applied in this research.

Certain researches maintain prerequisites for their solution to work that can not be met in our context. Berlin and Motro (2002) proposes an attribute dictionary that works with Bayes Theorem. The dictionary is characterized by, among others, possible values and thus instance data. It can not be made clear what the impact of the instance data is in this solution and what

remains if this is left out. The research of Cheng et al. (2010) builds upon already computed possible matches, therefore not considering how these matches are constructed. However, the construction of matches is a core part of the solution to be created in the upcoming chapters. The last paper that also requires a prerequisite, is that of Rodrigues et al. (2015). Active learning is applied which only works when the user matches a couple of elements. This is used to indicate which decision trees should be used for the remaining elements. The order of steps in which this solution works is contradictory to the situation of our research, and would not prove to be useful in small schemas.

Although the research of Sahay et al. (2020) presents a decent solution method, the similarity measures used are not of great value to our context. Linguistic similarity is taken into account, but it is not in the shape that it supports different languages or entries defined with a different structure. Its goal is to match attributes that have similar looking names.

3.3.4 References with substantial relevance

In the article of Jeong et al. (2008), a method for constructing a Neural Network-based Partial Least Squares (NNPLS) similarity synthesis is proposed, along with various methods to augment a small training dataset for better training. Augmenting the training data is not of real relevance to our research, since a large amount is available. However, the proposed method that creates a synthesis out of multiple similarity measures has some benefits. It is explained earlier that more similarity measures better cover the various aspects of schemas. A combination of string and structure similarity measures is used, but the method leaves much flexibility to adapt it to the problem of the user. Multiple similarity measures usually require much computation time due to the high dimension of input variables, but NNPLS overcomes this problem by a PCA-like dimension reduction (Jeong et al., 2008). The paper does not display any standard performance scores, except for the correlation coefficient that tells the reader how precisely supervised labels match with predicted labels.

In the article of Anam et al. (2015b) schema matching is executed via a knowledge base that is updated by the user. First, text processing is applied before similarity measures related to string similarity are computed. They are then fed to a decision tree based on a single rule. When the user is not satisfied with the result, a new rule is added based on a knowledge acquisition process of Censored Production Rule (CPR) proposed in (Jung, Lee, & Cho, 2010). Simultaneous, structure level matchers based on Similarity Flooding (Do et al., 2003) evaluate structural similarity. An aggregation with the string similarities is constructed via various aggregation functions. This overview of different aggregation approaches and their impact create useful insights. The paper produces relatively good results, with only a small amount of training data. This is also the only way the results are validated, the impact of a large set of training data on the performance of the method and results is not investigated.

Mukherjee et al. (2020) makes use of historical matching data, but in such a way that a generalization of knowledge is applied to cover up for the need of excessive amounts training data. A learning knowledge graph is proposed that captures domain knowledge of concepts, terms and relationships between these. The matching process is subdivided into the 'matching inference problem' and the 'learning to match problem'. The first looks to assign the most likely values to latent variables, that help asses the alignment of entities and attributes. It is based on several self-constructed formulas. The second estimates the probabilities in the knowledge graph based on historical matching information and source-specific matching information. This problem is based on stochastic Expectation Maximization, included in an iterative learning algorithm. The paper's approach to assess similarity, whereas other papers use well known similarity measures, is quite original but lends itself poorly for using only a small part of the approach. The flexibility for implementation is lower than in the other papers described in this section, and would it be challenging construct a cooperation.

A much referenced approach is proposed by Doan et al. (2001), who describe the Learning

Source Descriptions (LSD) system. This composite matcher uses several base learners that make a prediction according their own orientation. A meta-learner uses these predictions to learn and assess the importance of each base learner to the outcome. Their base learners are mostly based on instance data, except for The Name Matcher which uses the elements tag name. The meta-learner uses a machine learning technique of stacking. This structure of base- and meta-learners is a common practice, although in hybrid matchers the base learners are usually similarity measures.

Besides tackling the schema matching problem with the LSD structure, Doan et al. (2001) extends machine learning techniques to incorporate integrity constraints and user feedback. Hard and soft constraints are applied after a prediction is made, and can change the preferred element to match to if it is not compliant to the constraints. These constraints are defined by the user at the beginning, but can also be added as user feedback if the user is not satisfied with the results. Both instance and element level constraints can be added and function as a quality assurance.

3.4 Conclusions RQ 2, 3 & 4

RQ 2 What aspects define Intelligence Amplification in the schema matching context?

The concept of Intelligence Amplification is around since the research of Ashby (1956), although sometimes under different names. A contemporary description can be formulated as: a hybrid intelligence system that is formed by a cooperation between computer and humans so as to form an augmented intelligence of '$1 + 1 > 2$' (Pan, 2016). Often confused with Artificial Intelligence, IA seeks to establish a relationship with the human where AI is designed to mimic and replace the cognitive abilities of human intelligence.

In the literature we found three detailed frameworks around the concept of IA that all had a different perspective. The work of Parasuraman et al. (2000) presents a set of Levels of Automation, that depicts what to expect from human and machine at a certain stage of the design. These levels can be set at four different classes of functions that apply to most IA situations: information acquisition, information analysis, decision and action selection, and action implementation.The framework proposed in (Dobrkovic et al., 2018) is dedicated to a diverse application in decision making processes and makes a distinction between creative and computational tasks. Due to assumptions, tasks are assigned to either a human or an AI. The last framework of Zheng et al. (2017) proposes a framework quite similar to the one of Dobrkovic et al. (2018), but it also incorporates feedback loops.

In literature we did not found any generic IA frameworks applied to schema matching. Besides this, the frameworks we described could all benefit from each other by merging it to a single framework. The configurability of Parasuraman et al. (2000), the process flows of Dobrkovic et al. (2018) and the feedback loops of Zheng et al. (2017) are unique and fit well within a learning IA schema matching context. Based on the four different schema matching stages of Rahm and Bernstein (2001), we proposed our own generic learning IA schema matching framework that includes the unique characteristics of the three described papers. This framework can be used to design a schema matching solution customized to a certain situation, and represents the global aspects of IA in the schema matching context.

RQ 3 What automated learning techniques are suitable for schema matching?

Schema matching is largely seen as a classification problem, based on stochasticity. This goes really well with machine learning, where multiple techniques are designed for such problems. Machine learning has a lot of overlap with statistics and statistical learning, but it has very few focus on the interpretability of the underlying model and most of the times only the results

count. When viewing schema matching as a machine learning case, a selection of the appropriate destination element to the source element at hand must be made. Input variables are often represented by similarity values, of which the number of variables depend on the matching problem. The solution should find patterns in these input variables, and is trained by a set of historical matches.

In machine learning, three main paradigms or categories of techniques are present: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, the correct output is known for each training sample to learn predicting and adjust when necessary. It is the most used paradigm and has its applications in a very wide area of classification and regression problems. As discussed earlier, schema matching can be seen as a classification problem, but hardly as a regression problem. Supervised learning classification techniques could be of great use to a schema matching problem. In contrast to supervised learning, unsupervised learning lacks the presence of correct output during training. As a result, the application area is different and focused on discovering clusters and structures in sets of elements. It therefore does not match well with schema matching. The last paradigm, reinforcement learning is based on reward and learns to maximize payoff. This reward can be given by the user or through feedback provided by the environment. Applications of this technique are in decision making processes, strategic optimizations and on-line learning. Schema matching can possibly be solved by a reinforcement learning technique, although these techniques thrive at other purposes.

Considering the supervised learning classification techniques, we reviewed the most popular four that could be of relevance to schema matching. A Support Vector Machine (SVM) maps data in a higher dimension without needing to compute the entire higher dimension state. It has a convex solution space, meaning that there are no local optima. If a good kernel function is obtained, which is not an easy task, it can scale relatively well to high dimensional data. Decision trees is a technique that is easy for interpretation and has fast performance. With larger problems, the decision tree quickly looses its interpretability. In this research and end artifact, there is no focus on interpretability. Another technique, neural network models, can handle great complexities and have a large application area. It is based on a network of nodes, that produce output based on a function and adjustable weights. Often, this technique is referred to as a black box, but this is no problem to our research. The last technique discussed is Naive Bayes, and is built upon Bayes rule and the naive assumption that the distribution of the input variables are independent. If training data is scarce, performance is often better than other techniques. Nevertheless, if a categorical variable was not present in the training set, the model does not assign any probability to it.

RQ 4 What machine learning solutions have already been applied relevant to the research context?

The preceding research on the same case by Buis (2017) has an exploratory approach and provides us some recommendations and insights. Their design featured two similarity measures, but because of the outcomes a hybrid setup was recommended to overcome language differences and abbreviations. A neural network was used constructed in Azure Machine Learning, which performed better than other supervised learning classification techniques. The end result was not as fast as desired, left out matching of entities and was beneficial in a small set of scenarios. Taking these conclusions into consideration was strongly recommended to come up with an improved version of a schema matcher suitable to the iPaaS of eMagiz.

In an SLR, the scientific databases Scopus, Web of Science and IEEE were used to obtain literature on machine learning solutions in schema matching. The studies had to be written and peer reviewed in English, must be related to Computer Science, Engineering or Information Systems, and must be of a journal article, conference proceedings, book series or book type.

Papers were left out if they used data irrelevant to our case, instance data was used, no guideline or results were given and if they had poor model performance. This resulted in a total of 18 references, that were assessed on content.

Out of the carefully read references, a selection of four seemed substantially relevant to our research. Jeong et al. (2008) proposes an interesting synthesis based on similarity measures that could benefit a solution design taking many similarity measures into account. This serves the use of a multitude of similarity measures, as recommended by Buis (2017). The article of Anam et al. (2015b) provides a useful validated overview of aggregation approaches. Their approach is also promising in results, but it makes little use of the historical data. Mukherjee et al. (2020) proposes an original take on the schema matching problem, that would mean an all-the-way implementation since it does not fully lend itself for using only parts of it. Out of historical data, a knowledge graph is constructed which is an approach we did not encounter before. Lastly, Doan et al. (2001) describe their LSD system which is based on base- and meta-learners. Unfortunately they use a lot of instance data base learners, which makes it difficult to assess their performance. Another part of the paper addresses integrity constraints and user feedback implementation, which could limit bad results and serves as a quality assurance.

4 Solution Design

This chapter gives answer to **RQ 5**: *What is an efficient Intelligence Amplification driven auto-matcher design?*. We therefore start by elaborating on our solution structure and the decisions that are underpin its design in Section 4.1. The subsequent section, 4.2, discusses aspects about the data that is used for this research, considering the steps that need to be taken for tables in a database to become input for a deep neural network. After the structure and data preparation are established, the next step is to discuss the interpretation of the similarity measures and deep neural network properties in Section 4.3 and 4.4 respectively. This chapter closes with an insight in the *Building, Intervention and Evaluation* phase as described in Section 1.4 on our chosen ADR methodology. The ideas, user experiences and knowledge of the practitioners about the emerging artifact are discussed in Section 4.5.

4.1 Solution structure

4.1.1 Similarity measures

For the choice of measures that assess similarity between all combinations of schema elements, we turn towards a hybrid approach. To repeat: single similarity measures do not produce good performance for schema mapping (Jeong et al., 2008). A great part of the references discussed and displayed in Table 3.6 use multiple similarity measures to cover the multiple facets of their problem. Since we think there is information left behind when, for example, we only look at string similarity and disregard the structure the elements are in, we want to cover the most distinct characteristics. Also, previous research conducted by Buis (2017) suggested a hybrid approach based on the results that came from a pure string similarity approach. However, there is a boundary to adding more and more similarity measures. The more is added, the more computational time is needed and the less new information is gained. In this solution structure we use three different approaches as described in Do (2006) and Rahm and Bernstein (2001): a linguistic, a constraint-based and a structure-level approach. These are the ones that are the most apparent characteristics of our matching problem and also the ones that are feasible, since we cannot make use of instance-based matching for example. For the linguistic approach, we make use of multiple similarity measures. This class of similarity measures offers more interesting perspectives than others, which are widely combined and applied in most researches from our SLR. The similarity measures that we implement and present in our initial solution structure are explained in more detail in Section 4.3.

4.1.2 Supervised learning classification technique

Based on the findings of Section 3.2 considering supervised learning classification techniques, Support Vector Machines, Neural Networks or Naive Bayes could all find its application in this research. Nevertheless, one could be more appropriate than the other in terms of complexity, scalability or handling. Taking into consideration the experiment of Buis (2017) where multiple techniques were analyzed on performance in the same context, the neural network was chosen as a favorite. It should be noted that the input is slightly different than intended in this research. We conclude from our findings of Chapter 3 that a deep neural network is appropriate in this context to find structures in our input variables and to predict a *match* or *no match*.

4.1.3 Solution structure based on IA framework

When designing the solution structure, we make use of our generic IA schema matching framework (Figure 3.3) to secure a desired structure and define the levels of automation. We base the processes in these classes on the common habits in the schema matching literature and on the preferences of eMagiz. These processes are displayed in Figure 4.1. In the interpretation phase,

it is necessary to preprocess each element to a single format since different formats are present in the data we use for this research. For evaluation, we compute different similarity scores that we use as input for our Deep Neural Networks. This structure is a common practice and the most intuitive. Following up the evaluation class, the selection class applies a 1:1 mapping cardinality policy. We use this to lower the complexity of the schema matching problems. In the last class, we have a final list of matches that need to be validated in order to get a classification of correct or incorrect.

All process classes of the framework require to have a level of automation assigned to them, which we base on the possibilities and process flows of an iPaaS as eMagiz. Figure 4.1 displays these levels, of which their implementations have the following meaning: Classes *Interpretation*, *Evalution* and *Selection* all have a level of 10, meaning that the human is not involved and cannot intervene. The users and the process of schema mapping are solely availed by the correct suggestions that a schema matching solution can provide, as we further discuss in Section 4.5. Helping the computer with interpretation and evaluation are costly actions time-wise and is something that only complicates the process of schema mapping. The selection class is on a level 10 as well since we apply one-to-one mapping, which we elaborate on in further paragraph. It means that the most probable mapping is suggested by the computer, and all other possibilities are not being considered in the presence of a human. The last class, *Creation*, is set on a LoA of 5. This means that a suggestion of the mapping is given, but the human still has the control to approve or remove it. This makes sense since there is always a chance that a suggested mapping is false positive, something that is not tolerable to be accepted autonomously.

Process class	Used processes	Level of Automation
Interpretation	• Preprocessing element names to string format	10
Evaluation	• Compute similarity measures • Predict confidence of matches	10
Selection	• Apply 1:1 mapping cardinality	10
Creation	• Present list of matches • Validate matches	5

Figure 4.1: Processes and Levels of Automation (LoA) within solution design

The process flow of our model stays similar to the framework, where we do not return to past classes and just follow the presented flow. This method is seen in many of the relevant references acquired in our systematic literature review, such as Jeong et al. (2008), Doan et al. (2001) and Anam et al. (2015b). In Figure 4.2 we display the process flow that makes up our solution design. Indicated in grey are the external sources that provide information to the process steps. The feedback loops are discussed in Section 4.1.4 due to the constraints that come with the platform. In Appendix C a complete and detailed flow of processes is displayed. An elaboration on receiving schemas, preprocessing, similarity scores and predicting matches is given in Section 4.2.1, Section 4.2.3, Section 4.3 and Section 4.4 respectively.

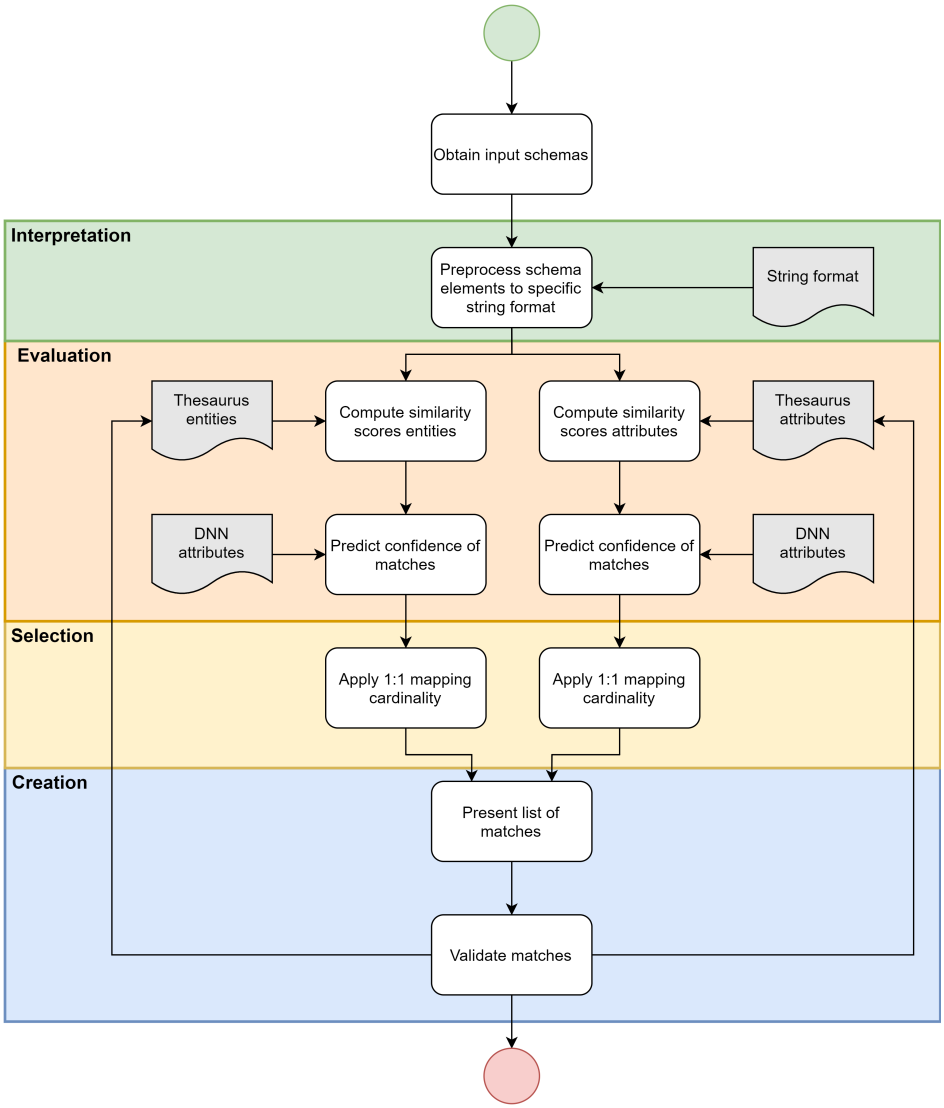


Figure 4.2: Solution design process flow

4.1.4 Feedback loops and knowledge base

The position of schema mapping in the platform of eMagiz has its implications on the feedback loops as proposed in the generic IA framework for schema mapping. We therefore discuss the three places a feedback loop can reenter the general solution process of Figure 4.2. These three places are based on our proposed framework of Figure 3.3.

- The first feedback loop enters the interpretation process class and can aid or improve the interpretation of schema elements. In Section 4.2.3 we discuss the necessary preprocessing steps for our used data, and conclude that in here no learning is needed. Therefore we do not make use of this feedback loop. Preprocessing follows a number of steps that apply to all data of eMagiz. If schema elements were to be replaced by synonyms for example, learning could be useful.

- The feedback loop into the evaluation process class serves in our solution structure to support the synonym similarity measure of Subsection 4.3.4. It can look up schema elements in a thesaurus based on a knowledge base of all relational information within eMagiz.

- The last feedback loop is used to train both DNN's, which are active in the selection process class. The information requested from the knowledge base is only used in the training process.

The feedback loops that flow from the creation process class is not used in this solution design. Due to user experience preferences, direct feedback to the knowledge base is not desired. Mappings are created in the so-called Design phase, which has the option to correct created mappings and change source or destination schemas. It means that created mappings in the Design phase are not definitive and should not be directly used as feedback. This means that the other feedback loop that flows into the knowledge base is deployed in another phase of eMagiz, called Create. At the end of this phase, mappings can be called definitive and are therefore checked and classified as correct. It is then ready to be added to the knowledge base.

The knowledge base consists of three objects, the database of eMagiz and two thesauri. The first has information stored about all integrations that are created in the platform, which can be used for training the DNN's. The latter two are thesauri for entities and attributes of which the creation and use are discussed in Section 4.3.4.

4.1.5 Mapping cardinality

Out of the existing match cardinalities as described in 2.1.3, we choose to stick to 1:1 mappings. Set-oriented cases such as n:1 and n:m have proven to be difficult, if not impossible, to automatically derive all correct mappings (Do, 2006) and more work must be done to explore the needed sophisticated criteria (Rahm & Bernstein, 2001). With the idea in mind that we value quality over quantity, i.e. precision over recall, we keep limit ourselves to 1:1 where simply the most confident match is returned.

4.1.6 Model environment

For validation of our solution design, a set-up is developed that is able to cope with the environment of eMagiz iPaaS, which is Mendix. This software is not designed to create and execute a neural network, so we make use of the modelling language of Python. Python offers a wide selection of reviewed libraries covering similarity measures and supervised learning methods. It also gives us a lot of flexibility in creating a tailor-made solution, is understood by many people and is free of charge. We construct a Python script that handles all computational issues, so Mendix only needs to share source and destination schemas and receives the list of auto generated matches. To communicate with each other, we set up an API that handles POST requests. A POST request sends data to a web server to create or update a resource. In our case, Mendix sends schema information to our Python script which is made a web server. Our web server then sends back the information Mendix needs for constructing auto generated matches. This is the fastest and most conventional method in combination with Mendix, and excludes dependencies on other parties.

4.2 Data in- and output shape

In this section we describe the information data must posses for training and testing the neural network and when the auto-mapper is in production. The training and testing data should represent the production environment as good as possible. Since integrations over the past 10 years have been saved, we do not have to worry about a lack of data. However, we should not use all integrations available at eMagiz for several reasons. Among others, Kavzoglu (2009) describes that a small number of training samples are often not sufficient to derive characteristics of the dataset, but a large number of training samples can result in overfitting the data. Besides, larger training samples mean longer training times. He notes that a larger number of training data is always favoured over a smaller number. A rule of thumb proposed in Kavzoglu (2001) is that a

randomly selected sample set should be between $30 \times N_i \times (N_i+1)$ and $60 \times N_i \times (N_i+1)$, where N_i represents the number of input nodes. To reduce complexity, we start by using integrations from companies that are in the same, and roughly operate with overlapping technical terms. A large part of the customers of eMagiz operates in the logistics sector, which we choose for our training and test set. The merged training data consists of integrations from Kien Logistics Management, Neele-VAT Logistics, PostNL and DHB Logistiek. All used integrations are indicated to be in production and thus secured to be thoroughly validated.

All information that comes with source and destination schemas in schema mapping, except for instance data, are saved in the database of eMagiz. From this, we only need the essential information that can be used as input for our solution structure. Figure 4.3 shows a simple mapping case, of which we can derive certain information. Below, all mapping information that we use in this research is described. The structure and retrieval of this essential data is discussed in the Subsection 4.2.1.

Figure 4.3: Example of a mapping

- **Direction**: The direction of an integration can be either 'IN' or 'OUT', meaning that data is going from the CDM into a system of the customer or that data is going out of the system of the customer to into the CDM respectively. We make use of this for constructing and working with thesauri, which we explain in Subsection 4.3.4.
- **Integration ID**: The ID gives us the possibility to select only one integration. We want to map attributes or entities within the same integration.
- **Entity name**: Entity names are used for similarity comparison, and belong to the attribute depicted in the *Attribute name* and *Attribute ID* columns.
- **Entity ID**: The entity name is related to the ID, which gives us the ability to distinguish similar named entities.
- **Attribute name**: Similar to the *Entity name* column, this is used for similarity comparison. Each row consists of a unique attribute name and ID, since entities can have multiple attributes both not the other way around.
- **Attribute ID**: Used for distinguishing attributes with similar names.
- **Data type**: This indicates the data type of the attribute, and is used for comparing data type similarity (Subsection 4.3.5).

4.2.1 In- and output information

We first describe the information that is used as input to train and test our models on. Two separate retrievals of data are made, one for mapping attributes and the other for mapping entities. Table 4.1 gives a representative example of the structure that is retrieved from the database for mapping attributes. This example shows only the source or destination schema information. A separate retrieval containing the missing source or destination schema is also added on the same row so the correct mapping can be deducted. While mapping attributes all information is used, but for mapping entities the columns *Attribute name*, *Attribute ID* and *Data type* are dropped because they hold no useful information. A more extensive construction of the retrievals is displayed in Appendix D.

Table 4.1: Data structure example

	Direction	Integration ID	Entity name	Entity ID	Attribute name	Attribute ID	Data type
1	IN	287104476	Customer	329325227	Name	331014572	String
...	...	...	...	...	...	...	...

The output information has only a single purpose: tell which attributes or entities should be mapped. Therefore, only the attribute or entity ID's are sent back, with each single row containing only the two ID's that need to be mapped.

4.2.2 Balancing training data

Source and destination schemas differ heavily in size per integration and sometimes even within an integration. When an integration is expanded in size, the number of possible combinations to assess grows exponentially while the number of correct mappings generally grows linearly. To give an indication, when picking random integrations from our dataset up to 1000 samples, we derive a *match*-to-*no-match* ratio of 52:948 or approximately 1:18. According to Haibo and Garcia (2009, p. 1265), "... any data set that exhibits an unequal distribution between its classes can be considered imbalanced". If we were to leave this imbalance as it is, there is a high chance that the DNN converges into a local optima, where it classifies each combination as *no-match*. Although this method provides us quite good performance metrics based on our random samples, it still is not able to detect matches. To tackle this problem, there are two popular ways to straighten up the imbalance: Sampling and Cost-Sensitive Learning. Within sampling, random oversampling and undersampling are common techniques, where in the first data is added to cover the imbalance and in the latter data is removed for the same purpose (Haibo & Garcia, 2009). Cost-Sensitive Learning applies specified costs to wrong predictions of a certain class, which can be implemented in the loss function of a neural network.

Although both techniques roughly have the same effect, we apply undersampling for our dataset. Finding the right costs in cost-sensitive learning is not an obvious task. Also, since we have an abundance of data, we can always find enough samples for both classes to train on.

4.2.3 Preprocessing strings

The columns *Entity name* and *Attribute name* contain strings that describe the corresponding data, and are used for measuring similarity. Across companies and system providers, different notation styles are applied. Therefore, entities or attributes with similar meaning can be described differently, which is a problem that can affect the quality of similarity measures that assess string similarity. To avoid the occurrence of this problem, preprocessing is applied to standardize descriptions without losing its meaning.

In the context of eMagiz, common characters that have no purpose other than separating words are '__', '_' and '-'. These are removed and replaced by a white space. Next, parts of strings were separated if they began with a uppercase letter which is directly followed by a lowercase letter. This excludes the separation of abbreviations which are mostly all written in uppercase and should be kept together. Now all separated parts of strings should represent a word, and are then capitalized. The final procedure is to concatenate all parts into a single string, of which each word is depicted by an uppercase letter and each abbreviation by two or more consecutive uppercase letters. The procedure is displayed below in Algorithm 1.

Algorithm 1: Preprocessing

Data: Set of all sources S, Set of all destinations D
Result: Set P of all preprocessed sources p, Set T of all preprocessed destinations t
for *each s in S* **do**
    p = **replace**(s) '__', '_', '-' with ' '
    p = **split**(p) on ' '
    p = **capitalize**(p)
    p = **concatenate**(p)
for *each d in D* **do**
    t = **replace**(d) '__', '_', '-' with ' '
    t = **split**(t) on ' '
    t = **capitalize**(t)
    t = **concatenate**(t)

4.3 Similarity measures

In this subsection we elaborate on all 6 similarity measures that we use in our solution design, provided with their related algorithm.

4.3.1 Levenshtein distance

In the category of linguistic approaches, we first describe the Edit Distance based similarity measure: The Normalized Levenshtein Distance. In short, "The Levenshtein distance is the minimum number of *insertions*, *deletions* and *substitutions* of characters required to trasnform one string into the other" (Euzenat & Shvaiko, 2007, p. 91). There are many popular variations on this metric such as the Damarau-Levenshtein distance and the Jaro-Winkler distance. These both have a slight emphasis on transposition of (adjacent) characters which make them suited for detecting typos (Euzenat & Shvaiko, 2007), a phenomenon we do not have to deal with in this research. We use the normalized variation of the Levenshtein distance since this gives a relative perspective of the edit distance, which is useful if words can differ greatly in length. Short words can never result in a high score but long words easily do, thereby losing the interpretation of the score if not normalized.

The normalized Levenshtein distance is provided by the strsimpy library for Python. A score of 1 is summed each time an insertion, deletion or substitution takes place. It is normalized by dividing the total by the length of the longest string. A score is computed for each possible combination of a set of attributes or entities belonging to the integration that is to be mapped. This procedure is displayed in Algorithm 2.

Algorithm 2: Normalized Levenshtein Distance

Data: Set of all sources S, Set of all destinations D
Result: Table of Normalized Levenshtein Distances
Initialize table $x_{S \times D}$
for *each s in S* **do**
    for *each d in D* **do**
        x_{sd} = NormalizedLevenshteinDistance(S_s, D_d)

4.3.2 Cosine similarity

The Cosine similarity is a token-based distance that is used to measure overlap between words and sentences (Bonifati & Eds, 2011). A string is considered as a multiset of words of a size picked by the user, and can be applied to evaluating contraction words such as *CustomerOrder* to *Customer* and *Order* (Euzenat & Shvaiko, 2007). The algorithm used for this similarity measures the cosine of the angles made by two vectors. An angle of 0° depicts complete similarity, and an angle of 90° of two vectors relative to each other means no similarity. The two vectors in this case are the elements to be examined. The Jaccard index and Euclidean Distance can be used for somewhat similar cases, but have different characteristics. The Jaccard index only looks at unique sets of words or tokens and therefore does not cope with repetition. The Euclidean Distance calculates the shortest distance between vectors, but is less relevant when magnitudes of vectors do not matter such as in working with text data represented by word counts. The Cosine similarity is very often used in information retrieval (Euzenat & Shvaiko, 2007) and therefore we leave out other options.

First all source and destination elements are preprocessed by Algorithm 1. The strsimpy library in Python provides an implementation of the Cosine similarity, and functions according to a parameter that characterizes the length of the vectors. For our purpose we divided elements into vectors of size 2, since this is the smallest perceived element name. The vectors of the source and destination are compared to each other with the following formula: $similarity = cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$, where A and B are source and destination elements respectively. Algorithm 3 displays the procedure that we use.

Algorithm 3: Cosine similarity

```
Data: Set of all sources S, Set of all destinations D, Parameter P
Result: Table of Cosine similarity scores
Initialize table x_{S×D}
PreProcessing(S)
PreProcessing(D)
for each s in S do
    for each d in D do
        p_s = DivideIntoWordsOfLengthP(S_s)
        p_d = DivideIntoWordsOfLengthP(D_d)
        x_sd = CosineSimilarity(p_s, p_d)
```

4.3.3 N-Gram similarity

When we take a look at string equality, assessing the similarity of substrings is quite useful in cases where one is a substring of the other. The length of the substrings to be assessed is chosen by the user, and cannot be longer than the shortest string present. Common used techniques are the Hamming distance and the N-Gram similarity. On the one hand, the Hamming distance counts the number of positions in which two strings differ. On the other hand, the N-Gram similarity focusses on the similarity of common *n-grams* i.e. strings of n characters (Euzenat & Shvaiko, 2007). We choose to implement the N-Gram similarity since the Hamming distance has much overlap with edit distance based techniques, which we already implement.

The choice for the n is set on 2, based on the smallest perceived element name. Elements are first preprocessed with the use of Algorithm 1. Similar to the previous two similarity measures, the strsimpy libary in Python also provides an N-Gram procedure. We compute the similarity for all possible combinations and save this for later use. These steps are displayed in Algorithm 4 below.

Algorithm 4: N-Gram similarity

Data: Set of all sources S, Set of all destinations D, Parameter P
Result: Table of N-Gram similarity scores
Initialize table $x_{S \times D}$
PreProcessing(S)
PreProcessing(D)
for *each s in S* **do**
  for *each d in D* **do**
    x_{sd} = NGramSimilarity(p_s, p_d)

4.3.4 Synonym similarity

With synonym similarity, we indicate the similarity in a couple of concepts of which a thesaurus can create insight. "A thesaurus is a kind of lexicon to which some relational information has been added." (Euzenat & Shvaiko, 2007, p. 99). The purpose of a thesaurus is to resolve any abbreviations, homonyms, synonyms or language differences (Lee, Yang, Hsu, & Yang, 2002). Resolving in this case is not some form of preprocessing, but indicating the relation that certain words may contain the same meaning. There are multiple online thesauri such as WordNet that are widely applied. However, the integrations that are used within the iPaaS of eMagiz are highly customer specific and can differ in languages, whereas most thesauri know only general terms and are made up out of one language. We apply a self-constructed thesaurus which probably provides much more information about all sector-specific abbreviations and the used languages of English and Dutch.

The creation of our thesaurus is a rather simple method. With a database request of all established mappings we already have access to all relational information. Within Python we create a dictionary of which all element names used by the CDM are displayed as keys. The element names of the CDM are chosen by platform users to be as generic as possible, making the thesaurus a lot smaller. The other option is to use a key for each element name that has ever been used in eMagiz, but this results in a file that is more than 3 times larger, which is unfavourable in storing and computation terms. We create two separate files, a thesaurus for entities and a thesaurus for entities, due to the different behaviour and relations both can have.

In assigning synonym similarity scores, we make use of a variation (Algorithm 5) of the algorithm presented by Lee et al. (2002) called *OntologySim*. Source and destination elements that are equal get assigned a value of 1. Destination elements that occur as a synonym of the source element in our thesaurus get assigned a value of 0.8. This holds true for the other way around too, when a source element is a synonym of a destination element. For all other combinations, a value of 0 is assigned by initializing the table x_{sd} as a table of zeros.

Algorithm 5: Synonym similarity

Data: Set of all sources S, Set of all destinations D, Thesaurus
Result: Table of Synonym similarity scores
Initialize table of zeros $x_{S \times D}$
for *each s in S* **do**
  for *each d in D* **do**
    if $s = d$ **then**
      $x_{sd} = 1$
      continue
    if *s in Thesaurus* **then**
      if *d in Thesaurus[s]* **then**
        $x_{sd} = 0.8$
        continue
    if *d in Thesaurus* **then**
      if *s in Thesaurus* **then**
        $x_{sd} = 0.8$

4.3.5 Data type similarity

The information available of both the source and destination schemas that constrain the values that pass through them, can very well be used for determining similarity (Do, 2006). Constraints such as allowable values, value ranges, cardinality etc. are often present with the data. In this research we make use of the data type constraint, since it is a piece of information that is required to be specified with each attribute.

The usual practice of assessing datatype similarity is by constructing a compatibility table, specifying the degree of compatibility between a set of predefined data types (Do & Rahm, 2002)(Do, 2006). One can imagine that certain scenarios are not feasible, such as connecting a *Decimal* with a *DateTime* data type, which will lead to unconventional or incorrect data transformations. The set of data types is displayed as the column and row indices of Table 4.2, derived from the eMagiz platform.

For constructing the data type compatibility table, we establish an estimation based on a large sample of integrations. These integrations originate from the dataset as described in Section 4.2, and they represent a fair part of the integrations that are well established and operational. The compatibility scores are a row-normalized representation of the occurrences, rounded to 2 decimals for better readability (Table 4.2). When assigning similarity scores for the neural network, the exact same values are used. In Algorithm 6 the procedure of assigning similarity scores to all possible matches is displayed.

Table 4.2: Data type compatibility

From\To	**String**	**Integer**	**Decimal**	**Boolean**	**DateTime**	**Enumeration**
String	1	0	0	0	0.01	0.01
Integer	0.83	1	0.01	0.01	0.01	0.01
Decimal	0.43	0.04	1	0.01	0	0.01
Boolean	0.17	0	0	1	0	0.05
DateTime	0.28	0.01	0.01	0	1	0
Enumeration	0.85	0	0	0.01	0	1

Algorithm 6: Data type similarity

Data: Set of all data type integrations I, Set of possible data types J, Set of all sources S, Set of all destinations D
Result: Table of Data type compatibility, Table of Data type similarity scores
Initialize table $x_{S \times D}$
Initialize table $c_{J \times J}$
for *each i in I* **do**
  $c_{i_0 i_1}$ += 1
c = Normalize(c)
for *each s in S* **do**
  for *each d in D* **do**
    $x_{sd} = c(j_s, j_d)$

4.3.6 Structure similarity

Where the previously described similarity measures only consider same-level elements or information, structure-level approaches look at relationships between elements that appear together in a structure (Do, 2006). When comparing attributes, the corresponding entities may possess crucial information, which also holds true the other way around.

Approaches considering the name path of an attribute or entity are described by Lee et al. (2002) and Do and Rahm (2002) and both follow the same principle. Their respective *Path Context Coefficient* and *NamePath* make use of earlier computed similarity measures based on attribute or entity names, which we have in use too. An attribute name is extended with the related entity name preceding it. As an example, *Customer* becomes *Order.Customer*, thereby displaying its path in a single string. Then, these strings can be compared for measuring similarity by making use of one of the linguistic approaches (Levenshtein, Cosine, N-Gram, Synonym). This procedure can work both ways, assessing attributes on their entities or assessing entities on their attributes. The latter needs a weighted average in case the entity is described by multiple attributes.

For our application, we do not compute the linguistic similarity measures again for the name path, but make use of the already acquired measures and take a weighted average. This saves computing time and does not make a difference in the final result. Also, we can take four different perspectives on the similarity, which makes it more precise than the one or two similarity measures proposed in Lee et al. (2002) and Do and Rahm (2002). We reason that with calculating the similarity measures mentioned in previous sections for the entities and attributes, and then combining these two in a weighted average, gives virtually the same results as doing so for a contracted name path. A small difference in some measures could occur when lengths of words differ significantly, but this would not make an impact due to the combination of multiple measures. We make use of a depth of 2 given the complexity, making sure that all directly connected entities and attributes are assessed. This is displayed in Algorithm 7.

Algorithm 7: Structure similarity

Data: Set of attribute similarity scores A, Set of entity similarity scores E, Set of all sources S, Set of all destinations D
Result: Table of Structure similarity scores
Initialize table $x_{S \times D}$
for *each s in S* **do**
  for *each d in D* **do**
    $x_{sd} = \frac{1}{2} \frac{\sum_{i=1}^{A} a_{isd}}{|A|} + \frac{1}{2} \frac{\sum_{i=1}^{E} e_{isd}}{|E|}$

4.4 Deep Neural Network properties

With the choice of a Deep Neural Network and 6 unique similarity measures, there comes a result of two separate DNN's since the input vector is different for entities and attributes (Table 4.3). We discussed earlier in Section 4.3.4 and 4.3.5 that we make use of two separate thesauri and that datatypes are not applicable to entities. Although it is a necessary choice, capturing the different behaviours and characteristics of entities and attributes is better off this way by assigning each their own DNN.

Table 4.3: Input variables comparison

Type of approach	**Input variable**	**DNN Entities**	**DNN Attributes**
Linguistic approach	Levenshtein distance	X	X
	Cosine similarity	X	X
	N-gram similarity	X	X
	Synonym similarity	X	X
Constraint-based approach	Datatype similarity	-	X
Structure level approach	Structure similarity	X	X

There are no clear rules on how to configure a model for a given problem Brownlee (2018), that is why we devote an experiment in Section 5.2 to obtaining suitable parameters. These parameters include the number of hidden layers, nodes, epochs and the dropout rate. However, we can already specify certain properties that are specific to our schema matching problem. These properties can be found in Table 4.4 and are elaborated in the upcoming paragraphs.

Each layer in a DNN consists of a number of nodes, that have their own weights and activation function. The combination of the input and weights inside an activation function determine the output of the node. The input layer of our entity and attribute DNN consist of 5 and 6 nodes respectively, because it must match the number of input variables. Since we use binary classification, predict *match* (1) or *no-match* (0), the output layer consists of 1 node with a sigmoid activation function (Brownlee, 2018). The input and hidden layers make use of a activation function called rectified linear activation unit, ReLU for short. This has become the default activation function when developing most types of neural networks (Brownlee, 2018), because of the principle that models are easier to optimize if their behavior is closer to linear (Goodfellow, Bengio, & Courville, 2016).

A loss function calculates how far the predictions are from the correct output. The lower the loss, the better the predictions. A Binary Cross-Entropy Loss function is the default loss function for binary classification problems (Brownlee, 2018), and works with a sigmoid output activation function. There are other popular functions such as Hinge Loss, but this one better suits the use of support vector machines and problems where targets are in the range of {-1, 1}. The Keras deep learning library in Python has a built-in Binary Cross-Entropy Loss function, so we do not bother the mathematics. The focal point is to minimize the calculated loss during the training of both our DNN's.

The use of an optimizer tells us what learning rates and learning rate schedules are used during training. A learning rate allows the weights of each node to be adapted by a certain rate according to the samples that have just passed the nodes. When using adaptive learning rates, weights can be set quickly into the right direction by using a relatively high learning rate in the beginning of training, and convergence to a local optimum can be achieved by using a relatively low learning rate in the end. The Keras deep learning libary provides three popular variations of stochastic gradient descent with adaptive learning rates, of which there is no single best algorithm (Brownlee, 2018). In this research we apply Adaptive Momentum Estimation (Adam).

Table 4.4: Initial properties of both DNN's

Properties	**Deep Neural Network**
Input layer	5 nodes / 6 nodes (Entity / Attribute)
Output layer	1 node
Activation function input and hidden layers	ReLU
Activation function output layer	Sigmoid
Loss function	Binary Cross-Entropy Loss
Optimizer	Adam

4.5 Validation with user

Together with two experienced eMagiz users working as a consultant at CAPE Groep, we discussed the solution design according to the IT-dominant BIE of the Action Design Research methodology (Sein et al., 2011). Some ideas that could be of use in the solution design were discussed on usefulness. For example, displaying suggestions of mappings on which the DNN has a lower confidence could be a feature for guiding the user or getting feedback from the user if he/she accepts or declines the suggestion. This feedback could flow through the first feedback loop into the knowledge base, as discussed in Subsection 4.1.4. After discussing the idea, it seems too distracting and focus can better be on the confident mappings. According to the consultants, the auto-matcher should be a tool for quick support to establish the easy mappings or give some starting hints in complex integrations.

Performance of the auto-matcher must be kept within reasonable time limits. The consultants mentioned that willingness to use the tool declines when predicting matches takes longer. Besides the computing speed, calling the tool should be fast as well. The preferences on this topic are noted and given to the employees responsible for the user experience of the platform.

Overall, the feedback on the solution design was positive without large remarks. According to them, there are at this stage no large gaps in using schema information or in the functionality.

4.6 Conclusions RQ 5

RQ 5 What is an efficient Intelligence Amplification driven schema matcher design based on machine learning techniques?

Based on analysis of the context, a review of relevant literature and recommendations of preceding research, a hybrid approach is chosen when it comes to similarity measures. We include three commonly used approaches: a linguistic, a constraint-based and a structure-level approach. The similarity measures related to these approaches do not consider instance data and all cover a different perspective on the possible matching combinations. The Levenhstein distance, Cosine similarity and N-gram similarity are all linguistic approaches and cover the edit distance, similarity of parts of words and successive letters respectively. We construct a synonym similarity to built a bridge between abbreviations and different languages, which is also a linguistic approach. The data type similarity is constraint-based and the structure similarity is evidently a structure-level approach. The inclusion of multiple linguistic approaches is due to the relatively many perspectives on this item and the availability of a wide range of validated algorithms.

The predictions made by our hybrid schema matcher are a product of a Deep Neural Network (DNN), due to the complexity it can handle, the scalability of the technique and the recommendations of preceding research. A deeper meaning of this DNN is given in following sections.

Taking into regard our proposed generic schema matching framework, we apply Levels of Automation of 10, 10, 10 and 5 to the Interpretation, Evaluation, Selection and Creation classes.

Choices of these levels are based on the environment of the eMagiz iPaaS and user preferences. A user should not be bothered with extra tasks other than assessing matches. This means that the user only has influence on the last class, Creation. Our solution design does not make use of the proposed framework feedback loop on the interpretation of elements, since in this design this is a fixed process. The feedback loop into the evaluation class is used to update the thesauri that are at the base of computing synonym similarity. By constantly updating the thesaurus with all historical matches, it transforms along with all integrations. The last feedback loop, into the selection class is used to train the DNN's with information from the knowledge base. Outgoing feedback loops are not used in this solution design, due to user experience preferences. A mapping is only user-validated when the total integration has passed the Create phase in the eMagiz iPaaS. This means that user feedback right after the creation of mappings is useless, because these mappings are not definitive and can be changed on or multiple times. The knowledge based, as described in our framework, consists of the eMagiz database and the two thesauri.

Out of the four different cardinalities a matching problem can have, we choose to only apply 1:1 because of the difficulty that comes with the other options. We therefore choose quality over quantity and return only the most confident prediction.

The schema matcher design is made possible with the software of Mendix and Python. Mendix is the software eMagiz has its integrations in and only accounts for making a POST request and displaying matches. Python does all other processes that belong to our schema matcher, and is chosen because it is flexible, easy to understand, free and compatible with Mendix.

A data set of four companies operational in the logistics sector is chosen, to reduce complexity of different technical terms and meanings. This sector represents a large part of the customers of eMagiz. The data is preprocessed when it is used in our model, transforming it to a single notation style so it can be easier assessed on similarity without losing its meaning. Because our data is heavily imbalanced, with the *match*-to-*no-match* ratio of approximately 1:100 and 1:1000 for entities and attributes respectively, undersampling is applied for our training data. Since we have plenty of samples to train on, undersampling corrects the ratio to 1:1 and makes it easier for the model to predict both categories.

The Levenshtein distance, Cosine similarity and N-Gram similarity are implemented according to the standard formulation, with the use of the strsimpy library for Python. For assessing synonym similarity, two thesauri are created that contain all previously mapped element names. It is saved in a separate json file, one for entities and one for attributes. A score of 1 is assigned if element names are equal, a score of 0.8 if they are represented in the thesaurus or else a 0. The thesauri thus contain a mixture of English, Dutch and abbreviations. The data type similarity is assessed on a tailored compatibility table, which should represent how conventional the combination of source and destination element data types is. We base the structure similarity on a principle described in Lee et al. (2002) and Do and Rahm (2002), and make use of the already acquired measures so the algorithm does not have to calculate the same things multiple times. The algorithm only accounts for an entity-attribute relation depth of two out of complexity perspective.

We choose to construct two separate deep neural networks (DNN), one for entities and one for attributes. This is due to the number of input variables, which are different for both (5 and 6 respectively). The datatype similarity measure is not applicable to entities, since entities are not described by a data type. Both DNN's make use of a standard activation function (ReLU) in the hidden layers, a sigmoid activation function for the output layer and a Binary Cross-Entropy Loss function, which is the default for a problem as ours. From the Keras deep learning library, we apply a popular stochastic gradient descent algorithm with adaptive learning rates called: Adaptive Momentum Estimation (Adam). The number of input nodes is equal to the number of input variables related to the element type, and there is a single output node that predicts

either 1 or 0 (*match* or *no-match*). Every number of hidden layers and nodes in these layers are deducted from the parameter configuration experiment in the next chapter.

5 Solution Results

This chapter explores the results of our solution design by answering **RQ 5**: *What is the performance of the proposed solution design and when is it beneficial?* We investigate this in three subsequent parts. Section 5.2 seeks to find the optimal settings of our model and tests this on the complete data set to obtain the average performance of our obtained solution. To be able to explain the variability behind this average performance, Section 5.3 describes a classification of performance based on integration characteristics. In the last Section (5.4) we assess how beneficial our Intelligence Amplification solution is given the different integrations present at eMagiz.

5.1 Data properties

The data that we use for validating our solution design is related to four companies operating in the logistics sector, as explained in Section 4.2. All integrations of these companies are in operation which means that they are thoroughly checked and represent real life cases. In Table 5.1 an overview of the properties is given that represent the data we use for training, validating and testing.

The number of integrations that come from the four companies with mapped entities and mapped attributes are 272 and 284 respectively. These integrations consist of a source and destination schema, that can widely vary in size. To create a better understanding of the variability in schema size in the data set, we display the lengths of source and destination schemas in a histogram for both entity and attribute data Appendix E Figure E.1 and E.2. The number of validated entity and attribute mappings are 2,085 and 9,345, which is a far smaller number than the possible combinations that do not have to be mapped: 226,807 and 9,580,385 respectively. This should give a feeling on the data imbalance that is discussed earlier.

An important thing to keep in mind is that the data set consists of all cardinality types, whereas we only apply 1:1 mapping. This means that our solution chooses the match with the highest confidence, discarding all other matches that must be mapped. As a result, these discarded matches are classified as false negatives and directly lead to a lower recall. With 1:1 mapping it is not possible on this data set to achieve a recall of 1.0, therefore we want to know what the maximum possible score is. If we keep all first occurrences of the source and destination elements of the mapping samples based on their ID, we have a total of 1,343 and 7,717 mapping samples for entities and attributes respectively. Taking the average over the complete data set, a maximum recall of $\frac{1,343}{2,085} = 0.644$ for entities and $\frac{7,717}{9,345} = 0.826$ for attributes is possible. This does not imply that individual integrations cannot achieve a higher recall.

Table 5.1: Properties of entity and attribute data

Characteristics	**Entity data**	**Attribute data**
Number of integrations	272	284
Number of mapping samples	2,085	9,345
Number of mapping samples (1:1)	1,343	7,717
Number of non-mapping samples	226,807	9,580,385
Maximum average recall	0.644	0.826
Schema language	English, Dutch	English, Dutch

5.2 Behaviour analysis of Deep Neural Network parameter configurations

The first of our experiments serves as a foundation for the other two and is based upon the structure as described in Section 4.4. The aim of this experiments is to describe the behaviour of the average performance on different parameter settings. With this, we can make a conclusion

on what configuration suits best to the eMagiz platform and what performance we can expect of our solution design. We use the chosen optimal parameter configuration as the standard configuration on which we conduct the experiments of Section **??** and 5.3.

5.2.1 Experiment settings

The parameters that we vary are the number of hidden layers, number of nodes per hidden layer, the number of training set samples and the confidence threshold. An overview of the used values is given in 5.2 and is elaborated in the next sections.

Table 5.2: Parameter values

Parameters	**Values DNN entities**	**Values DNN attributes**
Number of hidden layers	1, 2	1, 2
Number of nodes per hidden layer	3, 5, 10, 15, 20	4, 6, 12, 18, 24
Training set samples	900, 1400, 1800	1250, 1900, 2500
Validation set samples (*match*)	350	500
Confidence threshold	0.80, 0.85, 0.90, 0.95, 0.975	0.80, 0.85, 0.90, 0.95, 0.975

Model size

The number of layers and nodes make up the capacity of the DNN, and determine whether a model will underfit or overfit the data (Brownlee, 2018). For the number of layers, Goodfellow et al. (2016) states that a network with one hidden layer can approximate any given measurable function, provided that the network is given enough hidden number of nodes. Nevertheless, it is computationally more efficient to not have a large number of nodes in a single layer and increasing the number of layers can increase the capacity of the model. However, the more hidden layers, the more chance of the model not being able to learn the function. Therefore we also want to observe the performance of 2 hidden layers. We take for the number of nodes a multiplication of the input variables and apply this to all hidden layers. To add to this, we also take a smaller number than the number of input variables to also explore this area.

Training and validation set size

As discussed earlier, the rule of thumb proposed by Kavzoglu (2009) advises the number of training samples to be between $30 \times N_i \times (N_i + 1)$ and $60 \times N_i \times (N_i + 1)$, where N_i represents the number of input nodes. Since we have plenty of data available, we want to explore the full range of advised training samples. The range for the entity DNN is characterized by 5 input nodes and results in 900 and 1800 samples as its limits. For the attribute DNN this is approximately 1250 and 2500, because it has 6 input nodes. We take one extra value for both DNN's that lies somewhere in the middle of this range for a better performance perspective (Table 5.2).

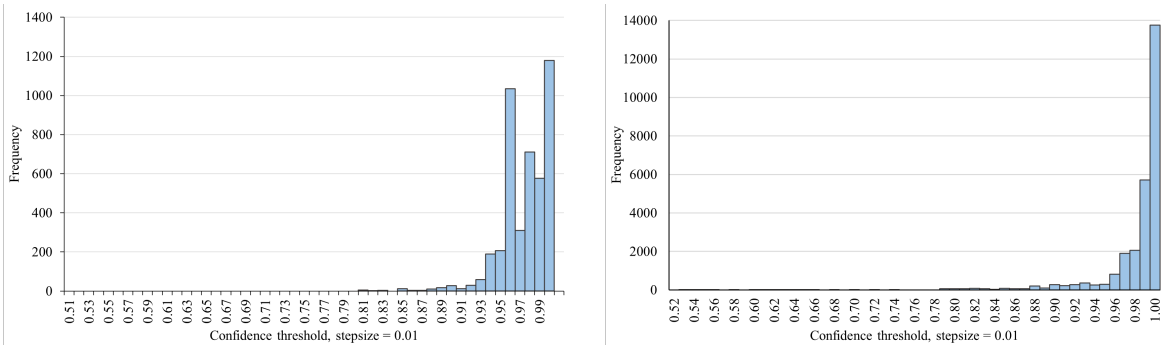
The number of validation samples follow a different method, since we do not have any constraints regarding the dataset size. We take a fairly common 80%/20% ratio of training and validation set samples, but for the validation set the samples represent the *match* samples and not the total number of samples we validate on. This is because the training set is balanced to become 50% *match* and 50% *no-match* samples, whereas the integrations the model is validated on have a strong imbalance that is dominated by *no-match* samples. The value of the *match* samples in our validation set is chosen to be at least 20% for all number of training set samples. We use the complete set of samples that belong to integrations because it resembles the eMagiz environment better. The possible difference in outcomes due to the sample size is covered by replicating all experiments 5 times, each with a different random seed. By doing this, we train and validate on 5 different sets of samples, whereby it is important to note that the validation set does not contain samples from the training set and vice versa. From the outcomes of these

5 different seeds we can take an average over each experiment and use that as representative performance.

We emphasize that training and validating on random samples and integrations still does not give us the performance of the complete dataset. Therefore, in Subsection 5.2.4 we train the model based on the optimal parameter settings and test these settings on the complete dataset.

Confidence threshold

The last parameter that we vary is not integrated in our DNN, but has a direct influence on the predictions it makes. A prediction of the DNN is a confidence score between 0 and 1, based on the input variables, whether the combination to be predicted is a *match*. To get a binary classification, we must round the confidence score based on a certain threshold. We can see in Figure 5.1 that a trained model has its confidence scores distributed closely to the range limits. Everything below 0.50 is not taken into account, because these values did not result in any frequencies except around 0.00, which caused a spike that would make the histogram unreadable due to the imbalance of *match* and *no-match*. It can be observed that both the DNN's have a similar confidence distribution and differences could probably be accounted due to the number of samples. Based on this figure we use a set of relatively high values as given in Table 5.2 for both the entity and attribute DNN.



(a) Histogram threshold values on DNN entities, number of layers = 1, number of nodes = 5, training samples = 1900

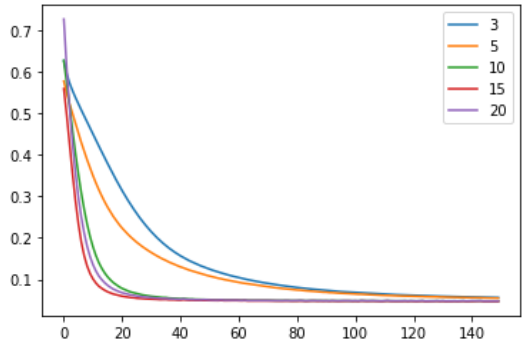
(b) Histogram threshold values on DNN attributes, number of layers = 1, number of nodes = 6, training samples = 2500

Figure 5.1: Confidence distribution for DNN entities and DNN attributes

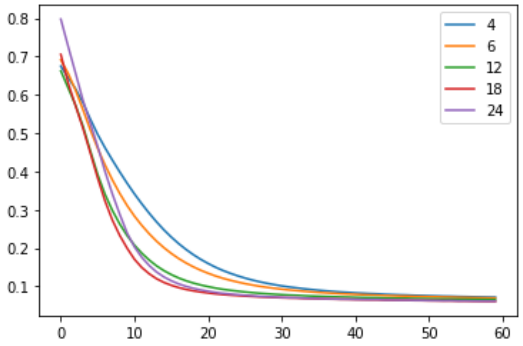
Bach size and number of epochs

Two important features of training a DNN that we need to set are the number of repetitions of the training process (*epochs*) and the number of samples to train on before updating the model parameters (*batch size*). By repetition, model parameters receive better training and can be better adjusted. A batch of a certain size is put through the DNN and parameters are adjusted, repeated as many times as the number of epochs. These two values are closely related to the learning rate, but since we set this a schema produced by the Adam optimizer, we only need to find a value for the batch size and number of epochs. This is typically done by plotting the training loss and continue to the point the loss gradient comes converges. A typical value for the batch size is 32 and the number of epochs is 100. The number of epochs can be set to a large value and almost ignored since after convergence nothing will change (Brownlee, 2018). We pivoted with the batch size and number of epochs for both the entity and attribute DNN, and aimed for all settings to converge. This is done at the smallest number of training samples and layers, since a model fed by more samples and more layers is able to learn the problem more quickly (Brownlee, 2018). The results are the direct plots from the Spyder software in Figure 5.2. For training the entity DNN, we find a batch size of 16 and the number of epochs of 150 sufficient to have convergence at all model settings. For training the attribute DNN, this is 32

and 60 respectively.



(a) Training error entities on the number of epochs, batch size = 16, epochs = 150, training samples = 900, hidden layers = 1



(b) Training error attributes on the number of epochs, batch size = 32, epochs = 60, training samples = 1250, hidden layers = 1

Figure 5.2: Training error for DNN entities and DNN attributes

5.2.2 Results model size and training samples attributes

Due to computational issues, a full factorial experimental setup is too costly time wise. We therefore compare the performance of both DNN's on the combinations of hidden layers, number of nodes and number of training samples, all with the same confidence threshold of 0.8. We take the optimal settings found as the DNN settings for later experiments.

DNN entities

We display the average of f-measures ($\alpha = 0.5$) of our attribute experiment settings in Figure 5.3 to get a feeling of both precision and recall. The results are quite equal, except for the 2 layered models trained by 900 and 1400 samples. These averages are lower because in some replications the parameters were not trained well enough to predict outcomes that were confident enough to exceed the threshold. It classified no matches at all for certain integrations, resulting in a precision and recall of 0. What can also be observed, although only by a little, is that having more samples results in a higher average f-measure. This is caused by the average recall being higher in those cases (Figure 5.5), because the average precision shows almost no difference in Figure 5.4.

Before we make a decision on choosing the optimal DNN size and number of training samples, we first want to explore the ascending performance trend of the training samples. With these results, it is no given thing that 2500 is the optimum, since it could be that an even larger number of training samples results in a better performance. Therefore, we include an extra experiment setting of 2300 samples. Fortunately, performance of this experiment setting is lower than 1800 samples, indicating an optimum around 1800 samples. The comparison of 1800 and 2300 samples is displayed in Appendix E Figure E.3. With this additional overview of parameter settings, we choose a setting of 2 layers, 10 nodes and 1800 training samples for the entity DNN. This settings scores as one of the best at average precision and the best at average recall. It must be said that score differences are quite small, so favoring this setting over others will not make a drastic difference.

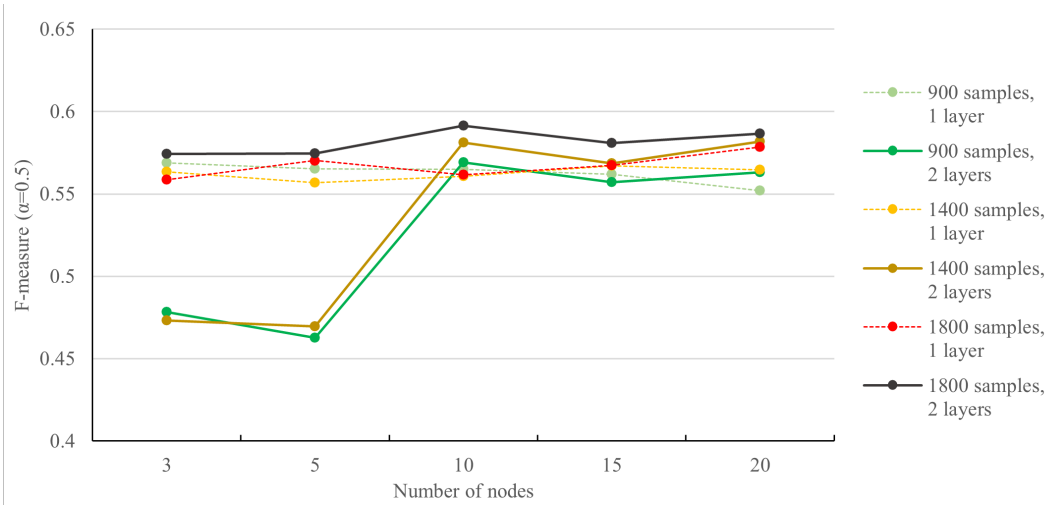


Figure 5.3: DNN entities: Overview of the average f-measure on all samples, layers and nodes combinations. Confidence threshold = 0.8

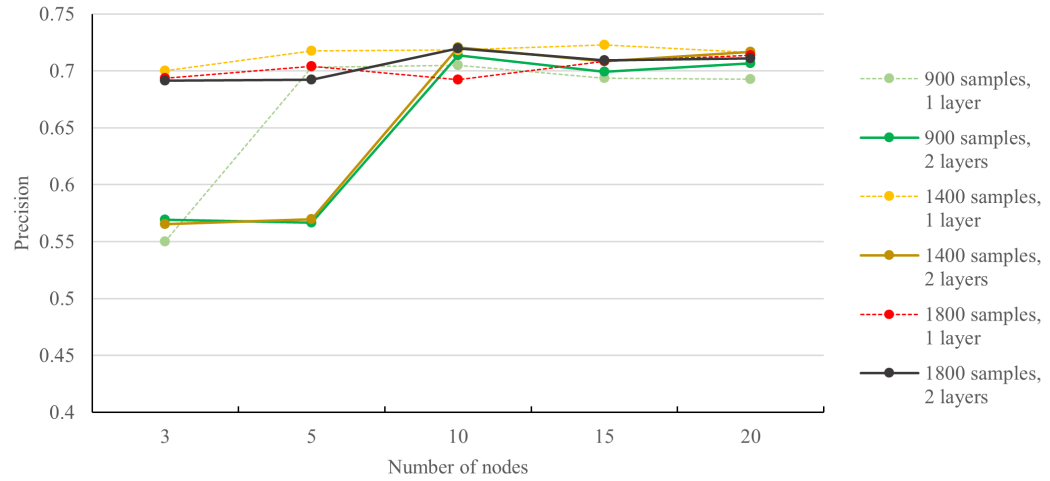


Figure 5.4: DNN entities: Overview of the average precision on all samples, layers and nodes combinations. Confidence threshold = 0.8

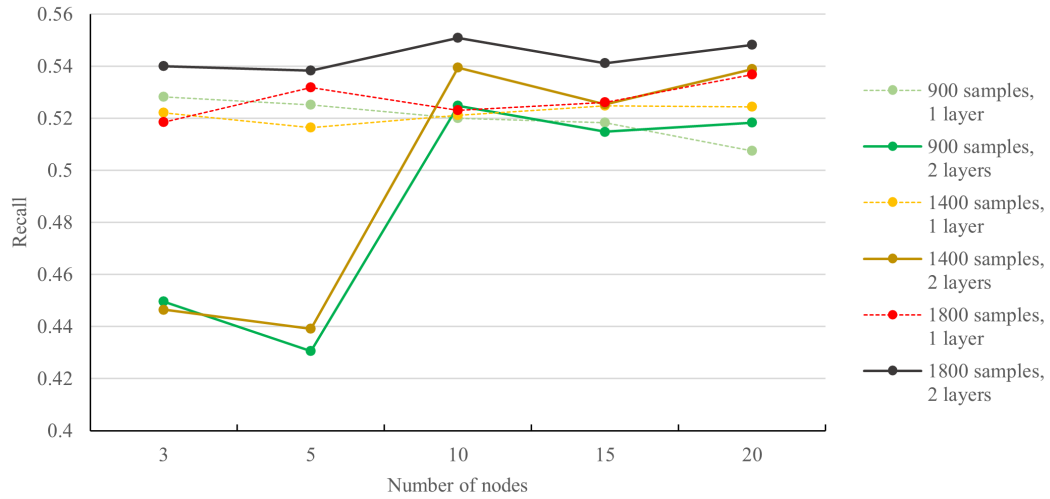


Figure 5.5: DNN entities: Overview of the average recall on all samples, layers and nodes combinations. Confidence threshold = 0.8

DNN attributes

For the deep neural network of attributes we follow the same approach and display the average of f-measures ($\alpha = 0.5$) of our attribute experiment settings in Figure 5.6 to get a feeling of both precision and recall. It can be observed that the DNN's with 2 layers score generally higher than the 1 layered DNN with equal number of training samples. With a larger number of nodes in each layer, the average f-measure seems to converge. The difference with the smaller number of nodes can be a product of the DNN's size (number of hidden layers $\times$ number of nodes). This substantiates that the DNN's of small sizes score worse and that results gradually converge when the DNN size exceeds a certain magnitude. The precision and recall that are at the base of the f-measure show both a similar behaviour in Figure 5.7 and 5.8. Convergence is especially present at the average precision, making it redundant to evaluate DNN's of even larger sizes. When taking into consideration the number of training samples, we see that the largest setting of 2500 samples also scores the highest.

To quickly get a better understanding of the fluctuating f-measure behaviour between the settings, we computed the standard deviations of the f-measures of the validation set. Nevertheless, all the outcomes are close to each other and do not hold much value in making a decision on the optimal parameter settings. Figure E.5 in Appendix E displays these values. An elaboration on what causes this standard deviation is the main topic of the experiment of Section 5.3.

Like the performance behaviour of the entity DNN, we see a better performance at a higher number of samples, and ask ourselves if this trend continues. In Appendix E Figure E.4, a comparison of 1800 samples and 2300 samples on all DNN size settings is given. This comparison too shows that the upper limit of the rule of thumb from Kavzoglu (2009) is closest to the optimum. With this overview of settings, we choose a setting of 1 layer, 18 nodes and 2500 training samples for the attribute DNN, since it scores best on precision and recall.

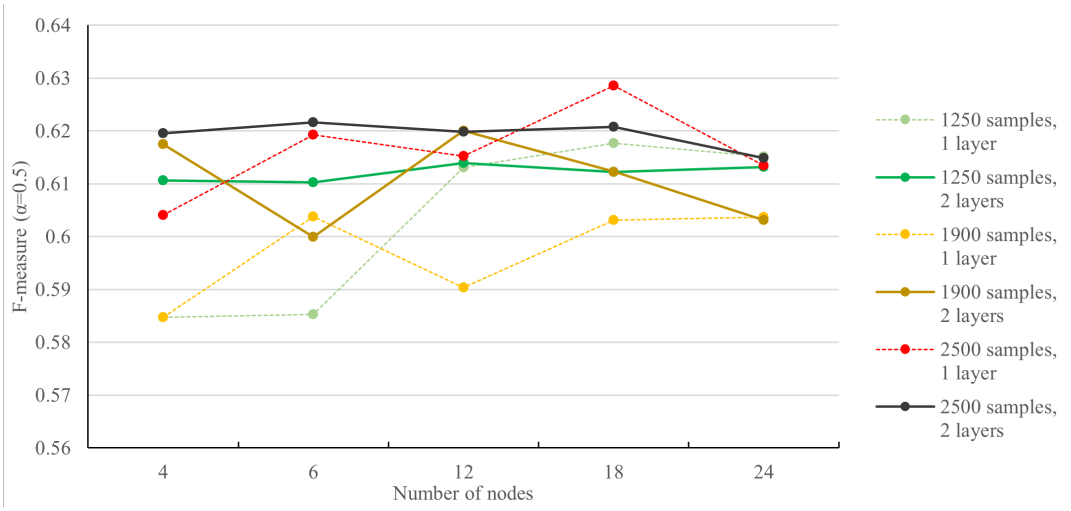


Figure 5.6: DNN attributes: Overview of the average f-measure on all samples, layers and nodes combinations. Confidence threshold = 0.8

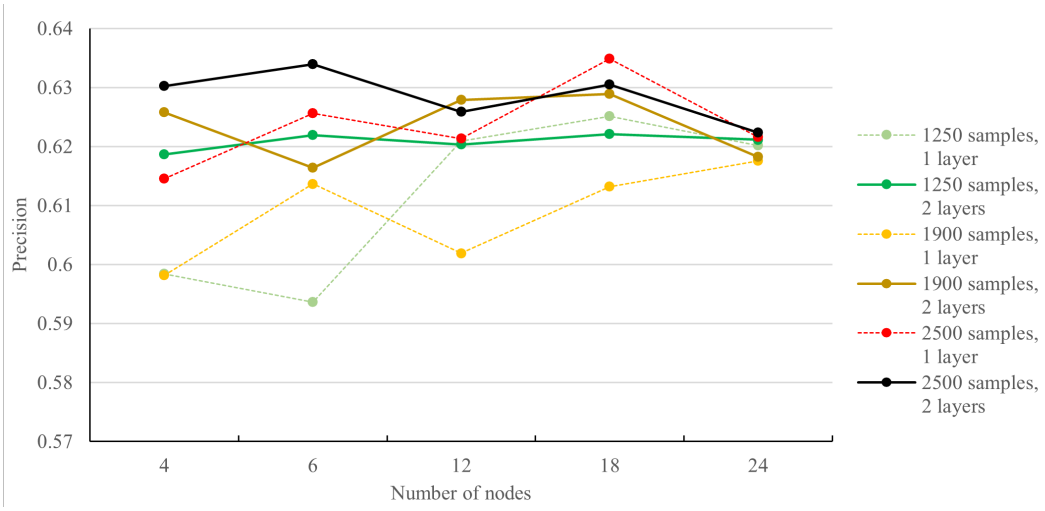


Figure 5.7: DNN attributes: Overview of the average precision on all samples, layers and nodes combinations. Confidence threshold = 0.8

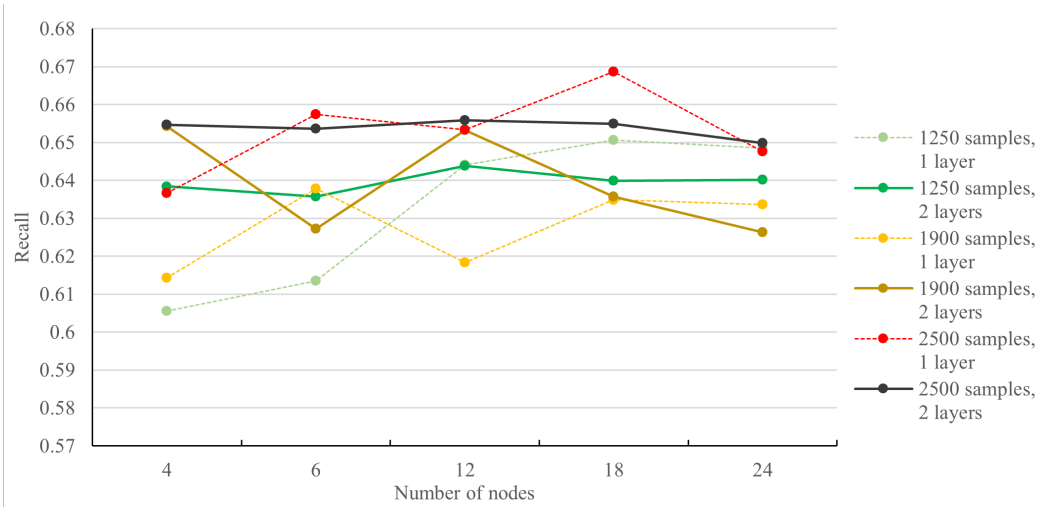


Figure 5.8: DNN attributes: Overview of the average recall on all samples, layers and nodes combinations. Confidence threshold = 0.8

5.2.3 Results threshold values

Assessing the influence of a threshold value on the prediction performance originates from the thought that predictions that show higher confidence should be more accurate and should lead to a higher precision. To check if this hypothesis is true, we take our chosen settings and compute the performance measures, of which we display the f-measure, precision and recall. Figure 5.9 displays the behaviour of the entity DNN, Figure 5.10 does the same but for the attribute DNN. Clearly visible is the decline of the average recall when the confidence threshold increases. This makes sense since there are correctly predicted matches without a confidence close to 1, which are increasingly excluded and classified as *no-match*. The average precision in both figures do not seem to have a trend. The precision does not constantly increase with the threshold value, so our hypothesis is not true. We can conclude that the more confident predictions do not necessarily hold more correct matches. Based on the average f-measure we can conclude that a threshold value of 0.8 and 0.85 for the entities and attributes respectively results in the best combination of precision and recall.

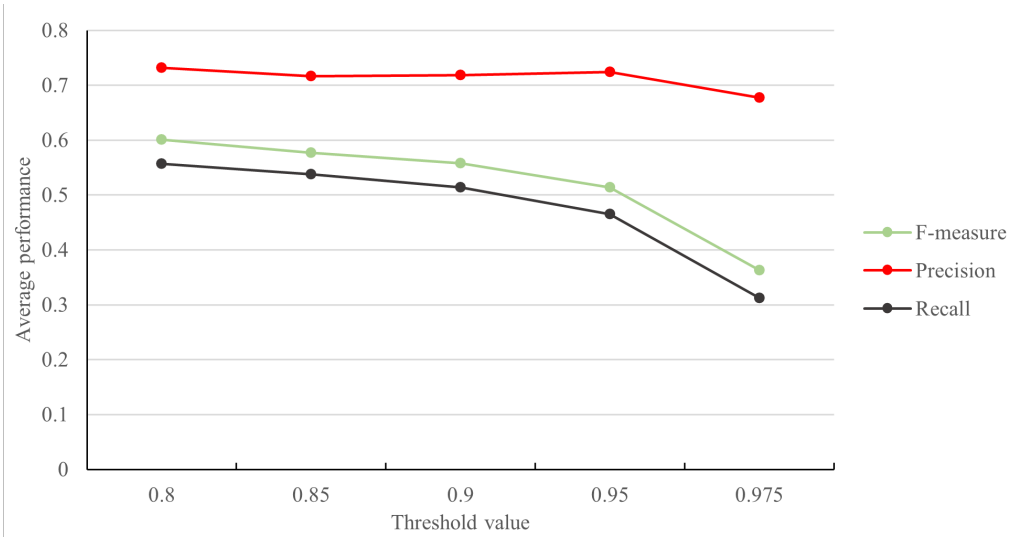


Figure 5.9: DNN Entities: Average performance on all confidence thresholds. 2 layers, 10 nodes, 1800 training samples

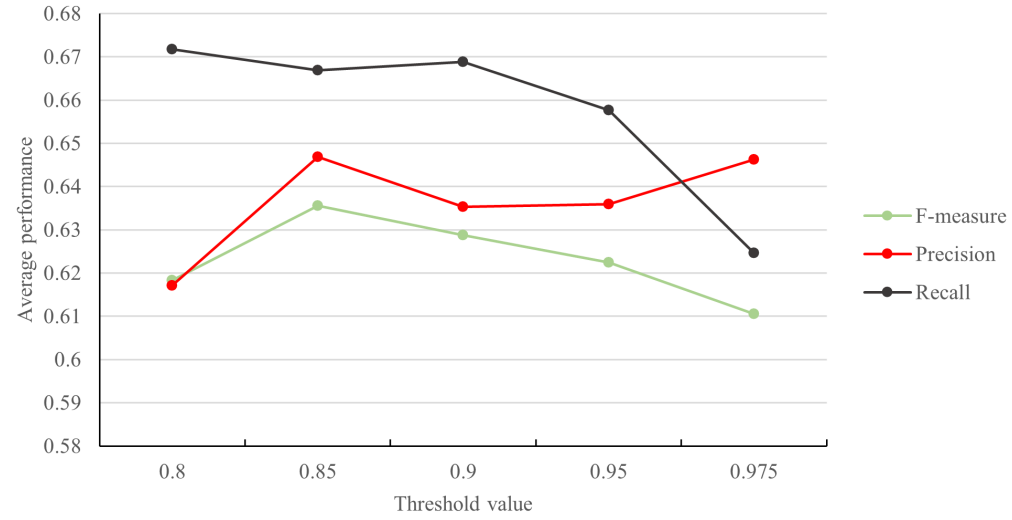


Figure 5.10: DNN Attributes: Average performance on all confidence thresholds. 1 layer, 18 nodes, 2500 training samples

5.2.4 Testing optimal settings on complete data set

Now that we have validated the model configurations and chosen the optimal settings, we can test these settings on the complete data set. Just as with the validation we take 5 replications but this time make sure that each integration is at least once used for training and all integrations are approximately evenly tested on. Meanwhile, we prohibit integrations to be both in the training and test set at the same replication. We then can take the average of each integration over all replications that the integration was included in the test set. An average performance over the whole data set can then be obtained by averaging all integration averages. If we would have averaged all integrations per replication and than averaged these averages, an incorrectly calculated average performance would be displayed. The size of the test set depends on the number of integrations the training set needs to get to a certain number of positive mapping samples, and therefore varies per replication. Taking the average over a variable size would give more weight to the integrations in a smaller test set, therefore creating an unequal calculation.

Average performance DNN entities

Training and testing with 5 replications according to the method explained at the beginning of this subsection, resulted in each integration being included in the test set at least four times. If integrations were included a fifth time, these last performance measures were discarded for even comparison.

When displaying the average f-measures of all entity integrations based on frequencies, we observe a scattered distribution with a large spike at 1.00. To put this spike into perspective, 47 out of the 272 integrations have this score, which is approximately 0.173%. Appendix E Figures E.6 and E.7 show the average precision and recall, of which the share of precision scores of 1 is substantial with 109 out of 272 (approximately 0.401%). The average performance on our chosen measures over all integrations is given in Table 5.3. Note that the maximal average recall to be reached is 0.644, so the score of 0.53909 is not bad at all with only a difference of 0.10491. The average overall scores are plotted in Figure 5.12, which shows that 212 out of 272 integrations (77,9%) are beneficial in terms of human effort according to Euzenat and Shvaiko (2007). Although we concluded earlier that the fallout is not useful in our schema matching situation, we do plot the average over the whole dataset to put into perspective the performance on the negative samples.

Table 5.3: Average of all integrations on all performance measures

Performance measure	**DNN Entities**	**DNN Attributes**
Precision	0.72651	0.65206
Recall	0.53909	0.61465
F-measure ($\alpha = 0.5$)	0.58163	0.61171
Overall	0.25900	-0.003557
Fallout	0.01978	0.00411

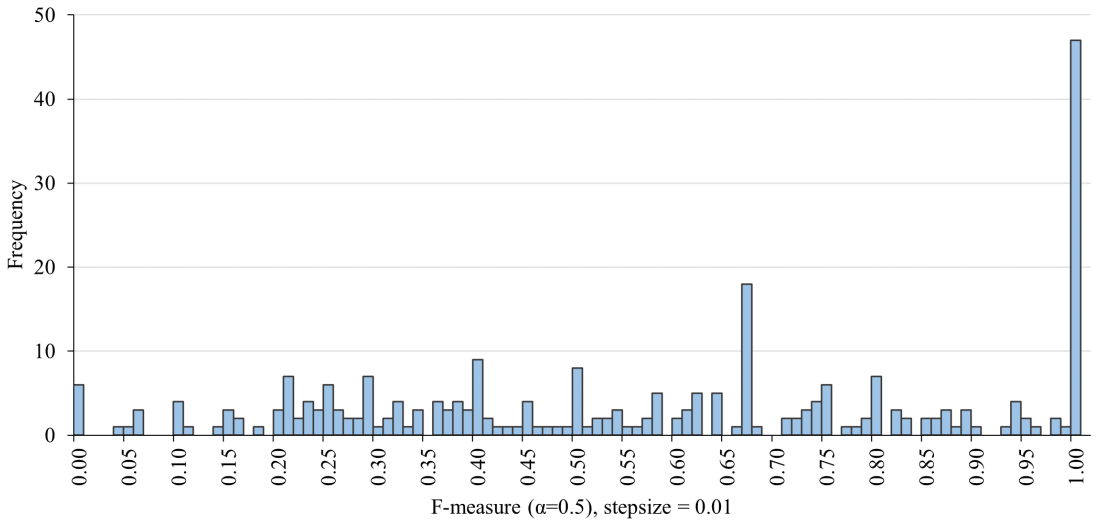


Figure 5.11: DNN entities: Frequency of average f-measure on all integrations

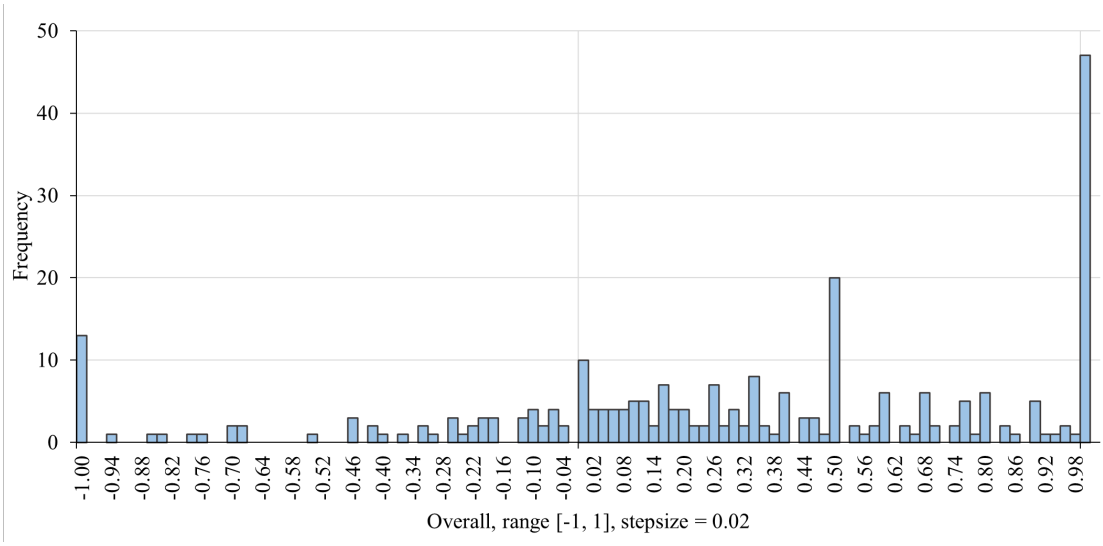


Figure 5.12: DNN entities: Frequency of average overall score on all integrations

Average performance DNN attributes

The average performance expressed by the f-measure ($\alpha = 0.5$) is displayed in Figure 5.13. A spike of 35 can be observed at an f-measure of 1.00, while the other frequencies are scattered across all scores. There is no distribution visible other than random if the integrations with an f-measure of 1.00 are left out. It is therefore impossible, just like the with the entity DNN, to predict a good approximate performance of the schema matcher based on the information we used so far. In Section 5.3 we aim to find underlying reasons for this performance behaviour, so we can estimate performance of an integration more precise. The frequencies of the average precision and recall on all integrations do not hold any extra valuable information and are therefore attached in Appendix E Figure E.8 and Figure E.9 respectively. The difference with the maximal attainable recall score of 0.826 is 0.21135. According to the average overall score, 196 out of 284 (0.69%) are labeled as beneficial, which is a little less than the entity DNN's performance.

All averages of performance measures are displayed in Table 5.3. If we compare both DNN's, the one for attributes scores lower on all measures, except for the fallout. A reason for this behaviour could be related to the generally greater number of combinations possible at matching attributes. It comes with predicting a lot more negative combinations, which also result in a higher chance of making more mistakes. When the number of negative samples are far bigger than the positive samples, these mistakes count heavy at precision and recall but not at fallout.

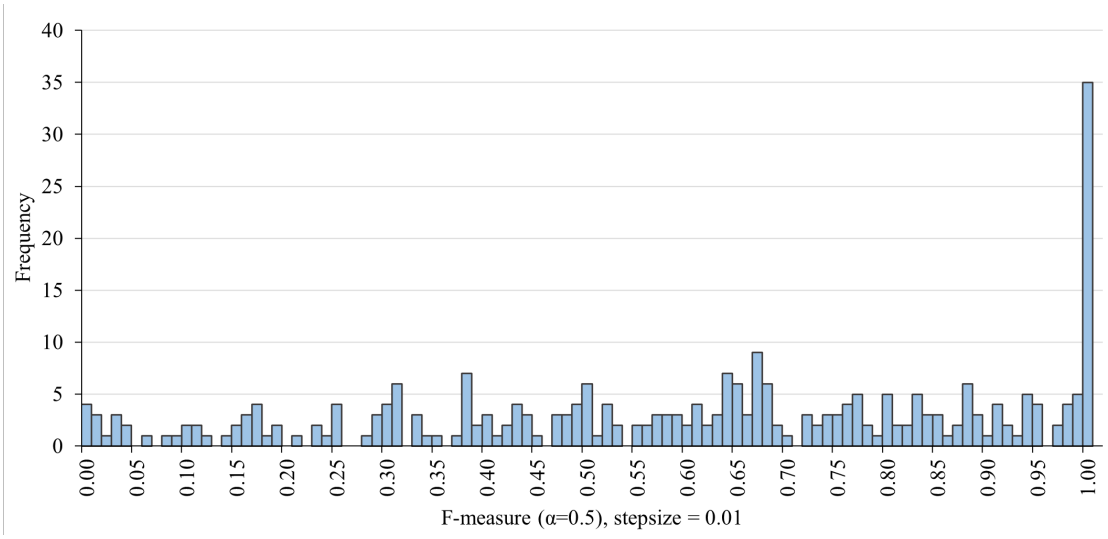


Figure 5.13: DNN attributes: Frequency of average f-measure on all integrations

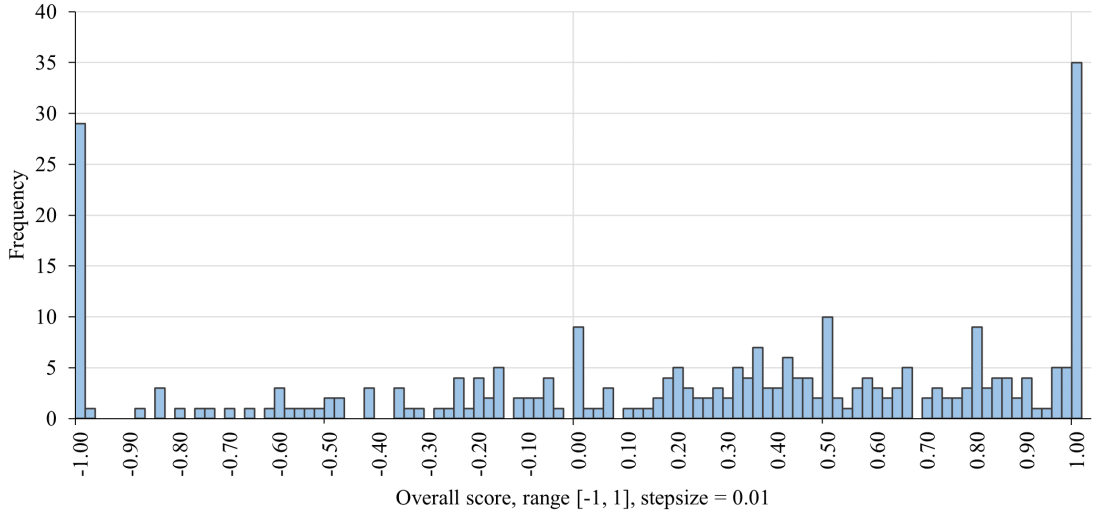


Figure 5.14: DNN attributes: Frequency of average overall score on all integrations

5.3 Influence of integration characteristics on performance

In the previous section we obtained average performances, and observed that the underlying integration performances are widely spread. We thus cannot make any specific claims on expected performance or why the performance has such an unpredictable behaviour. Therefore, we aim to provide insights into the relationship between performance measures and the characteristics of individual or groups of integrations. This aids the practical user-friendliness of the schema matcher, reveals parts of the prediction behaviour and serves as a base for assessing the benefits of the model in Section 5.4.

5.3.1 Predictor variables

Each integration is distinguishable from others by several characteristics, which typically have its effects on the types of heterogeneities to deal with. For example, a characteristic can be that source and destination schemas are in different languages, resulting in a language heterogeneity. To see what the influence of the integration characteristics on the performance are, we make use of linear regression. It is a widely used statistical learning method and is a useful tool for predicting a quantitative response (James, Witten, Hastie, & Tibshirani, 2013). Multiple questions can be answered with this method such as; *Is there a relationship between variable x and y?*, *How strong is the relationship between variable x and y?* and *How accurately can we predict future performances?*.

Using linear regression in our research context, we apply the terms response variable and predictor variables to the performance measure and integration characteristics respectively. While we already established our response variables (precision, recall, etc.), the predictor variables largely have yet to be established.

Literature does not provide us with reviewed methods on constructing predictor variables in schema matching, so we base our variables on the traits we could identify an integration with. These can be roughly subdivided into the categories: *size*, *cardinality implications* and *use of language*. In the category *size*, integrations can be differentiated by: number of source schema elements, number of destination schema elements, the ratio between the previous two, the number of matching combinations possible and the depth of schemas. The last characteristic, how many levels of entities and attributes are related to each other, is something that we do not investigate due to complexity reasons. The category *cardinality implications* is a product of our 1:1 cardinality constraint, and it's quality is a bit harder to estimate without predefined knowledge. The average cardinality shows how many mappings are related to a single schema element on average. A higher average should imply a lower recall, since the model can only produce a single mapping per element. The average number of (almost) equal matches depicts the matches that are discarded because only the most confident match is mapped. These matches share almost equal names, but sometimes differ in a related entity name. When a lot of repetitive element names are present, this average is higher. The last category *use of language* includes two binary characteristics that explain whether multiple languages are used or not, and that a constant use of abbreviations is used or not. All categories and their characteristics are displayed in Table 5.4 along with their the name we use for the predictor variable.

5.3.2 Method of analysis

To assess the relationship between response and predictor variables, we follow a certain process in order to make a distinction between variables based on relevancy and to give useful insights. We apply this to precision and recall of both DNN's, because we think there can be different behaviour which can also give indications for other performance measures.

First, multiple linear regression is conducted on all eight predictor variables, to find their significance on the response variable. Second, we assess the significant predictor variables individually to check whether we can say something about its individual influence on the precision.

Table 5.4: Structured overview of predictor variables

Category	Characteristic	Predictor variable name
Size	Number of source schema elements	Source
	Number of destination schema elements	Destination
	Ratio between source and destination schema length	Ratio
	Possible matching combinations	Combinations
Cardinality implications	Average cardinality	Cardinality
	Average number of (almost) equal matches	Loss
Use of language	Use of multiple languages	Language
	Constant use of abbreviations	Abbreviations

Third, we look for a better individual fit by applying polynomial regression. At last, we quantify our findings in order to contribute to practical usability.

Multiple linear regression fits a linear regression line based on all predictor variables. The difference with applying simple linear regression to all predictor variables individually is that in the latter approach the influences of other predictor variables are ignored, and may result to very misleading estimates of the individual effects (James et al., 2013). The interpretation of the coefficients with multiple linear regression is the average effect on the response variable of a unit increase in a single predictor variable, holding all other predictors fixed.

If we want to create non-complex visualizations of the relationship between predictor and response for better practical interpretation, simple linear regression is applied first. This method is linear regression with only a single predictor variable, and tests the significance of the relationship without taking into consideration all other predictor variables. It could well be that this regression outcomes is different from the multiple linear regression. If there is significance in the multiple linear regression but not in the simple linear regression, only the combination of this predictor variable with others could be of significance. Therefore, it is harder to find and visualize its relationship. On the other hand, if there is significance at the simple linear regression but not at the multiple linear regression, a relationship could have been discovered that makes no sense in the presence of other predictor variables. This is why we only consider predictors that are significant in both simple and multiple linear regression. Significance codes of 0.05 and less are taken into account, which is standard (James et al., 2013).

After a significant relation has been established, we can look for higher orders of regression to check whether there is a better fit than a linear line. For this we assess the significance of polynomial regression of a single predictor variable, and check with the anova method if the model improvement is also significant. If so, the model of higher order can be accepted and interpreted (James et al., 2013).

5.3.3 Relationship between precision and predictor variables

DNN entities

We start our method of analysis by applying multiple linear regression on the performance data from our DNN entities, with the use of RStudio. The average precision is taken as the response variable and all predictor variables are included. This results in the following overview of Figure 5.15. In this figure, the 'estimate' per predictor variable is the coefficient of the linear regression line, of which the intercept is the starting point of this line. The other information is not important to our results, except for the significance displayed in the different codes and the residual standard error. When the p-value drops below a certain value specific to the regression, a degree of significance is given that tells if there is a strong relation between the predictor and response variable. Each significance indicated by a * symbol should be taken into account. The residual standard error is minimized by the multiple linear regression, and the lower it is the better the fit.

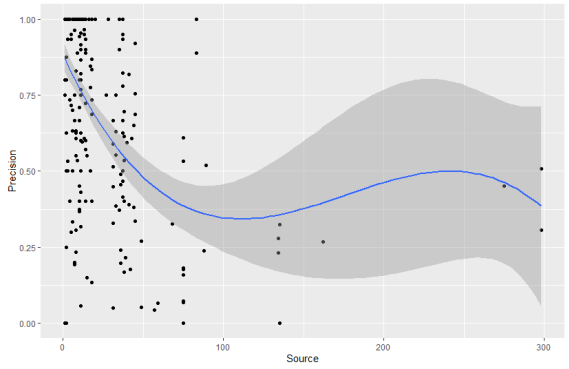
We take into account the predictor variables Source, Destination, Ratio, Cardinality, Loss and Language for our response variable Precision on the entity data.

```
Call:
lm(formula = Precision ~ Source + Destination + Ratio + Combinations +
    Cardinality + Loss + Abbreviations + Language, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.92761 -0.10429  0.09051  0.12671  0.49783

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    8.923e-01  2.278e-02  39.176  < 2e-16 ***
Source        -1.689e-03  5.485e-04  -3.079  0.00230 **
Destination   -1.505e-03  5.516e-04  -2.728  0.00681 **
Ratio          5.481e-03  1.316e-03   4.164 4.24e-05 ***
Combinations   5.687e-06  4.834e-06   1.176  0.24047
Cardinality    2.169e-02  6.606e-03   3.283  0.00116 **
Loss          -3.549e-02  5.209e-03  -6.812 6.52e-11 ***
Abbreviations  3.490e-02  5.565e-02   0.627  0.53117
Language      -1.305e-01  3.042e-02  -4.289 2.53e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2306 on 263 degrees of freedom
Multiple R-squared:  0.4473,    Adjusted R-squared:  0.4305
F-statistic: 26.61 on 8 and 263 DF,  p-value: < 2.2e-16
```

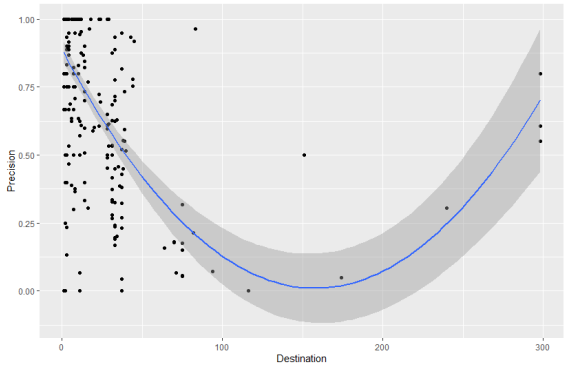
Figure 5.15: DNN entities: Multiple linear regression with precision as the response variable

In Appendix F Figure F.1 all individual simple linear regressions are displayed. It can be seen that the predictor variables Ratio and Cardinality are not significant anymore, while the other four are significant. Ratio and Cardinality are significant when taking into account other predictor variables, but we cannot display a proper relationship between precision and a single of these in a graph. We therefore apply polynomial regression to the other four predictor variables, and increase the polynomial until the new coefficient is not significant anymore. Then, the last polynomial is chosen where the anova method indicates if the order increase resulted in a significantly better line.

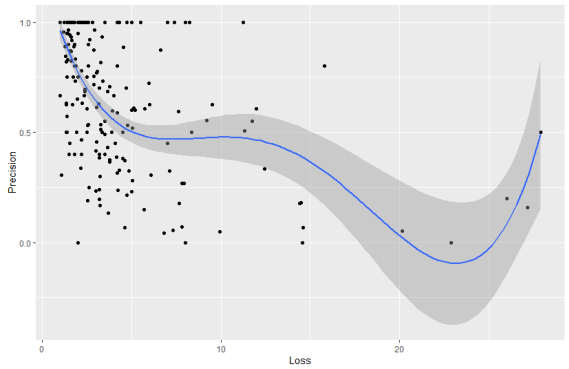
We display the data and their corresponding polynomial regression line in Figure 5.16. It can be seen at predictor variables Source, Destination and Loss, that the confidence interval around the blue regression line, indicated in grey, widens. The start of these lines are pretty accurate, but when the x axis increases, less data points are available and the confidence interval gets larger. We could say that roughly after a third of the x axis, interpretation of the data is not relevant anymore and the degree of polynomial does not matter anymore. A steep descending pattern is visible starting at 0, which tells us with high confidence that the precision is generally decreasing when the x is increased. The binary predictor variable Language can only take on a linear line, which shows that precision on average is higher when schemas are in the same language.



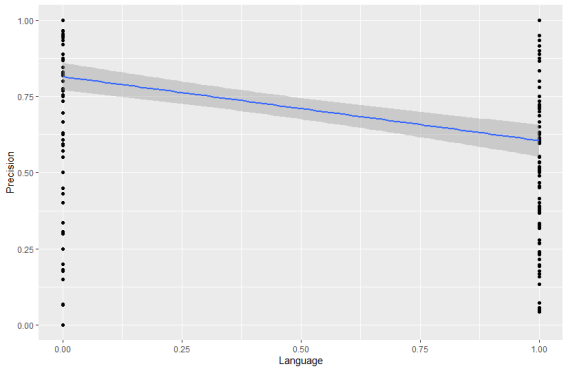
(a) 3rd order polynomial regression entity data, source on precision



(b) 2nd order polynomial regression entity data, destination on precision



(c) 4th order polynomial regression entity data, loss on precision



(d) 1st order polynomial regression entity data, language on precision

Figure 5.16: DNN entity: Polynomial linear regression of all significant predictor variables on precision

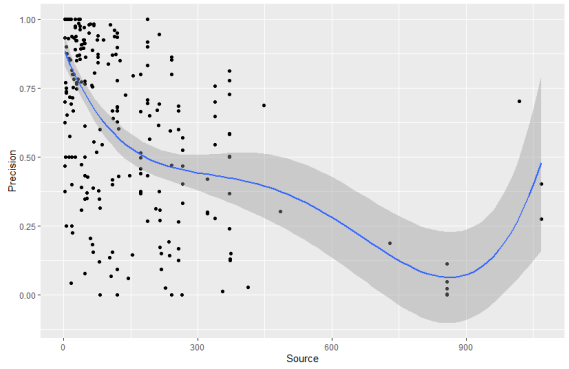
DNN attributes

We assess the relationship between the predictor variables and precision on the attribute data in the same way as described in the previous section about the entity data. The multiple linear regression outcomes are can be seen in Figure 5.17. A different outcome can be observed when compared to the entity data, only Source, Destination and Language are significant.

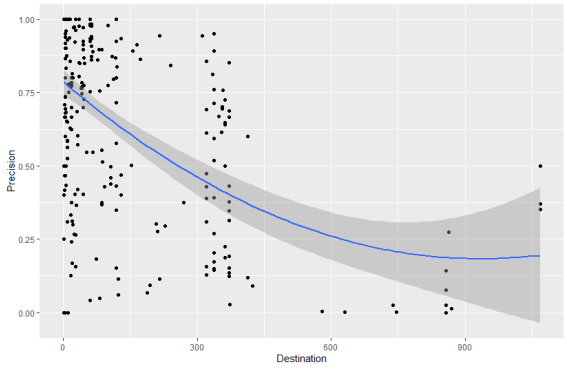
The three predictors that were significant in the multiple linear regression, also showed to be significant in the simple linear regression displayed in Appendix F Figure F.2. Therefore, we do a polynomial regression on all three predictor variables of which the results are given in Figure 5.18. Similarly as we observed at the entity data, we see an increasing confidence interval at the non-binary variables Source and Destination. A quick decline in precision can be expected when the size of schemas is increasing from 0 to somewhere around 100. After this, it is harder to give an exact indication. The precision of the language predictor decreases if different languages are used in schemas.

```
Call:
lm(formula = Precision ~ Source + Destination + Ratio + Combinations +
    Cardinality + Loss + Abbreviations + Language, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.78312 -0.16198  0.06097  0.12557  0.72974

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    9.679e-01  4.132e-02  23.423  < 2e-16 ***
Source        -6.515e-04  1.291e-04  -5.047 8.17e-07 ***
Destination   -5.079e-04  1.065e-04  -4.771 2.98e-06 ***
Ratio         -1.520e-03  1.064e-03  -1.428 0.154410
Combinations   3.290e-07  2.834e-07   1.161 0.246814
Cardinality   -5.790e-02  3.120e-02  -1.856 0.064559 .
Loss          -2.162e-03  1.480e-03  -1.461 0.145238
Abbreviations -3.114e-02  4.732e-02  -0.658 0.511047
Language      -1.163e-01  3.259e-02  -3.568 0.000425 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2326 on 275 degrees of freedom
Multiple R-squared:  0.4761,    Adjusted R-squared:  0.4608
F-statistic: 31.24 on 8 and 275 DF,  p-value: < 2.2e-16
```

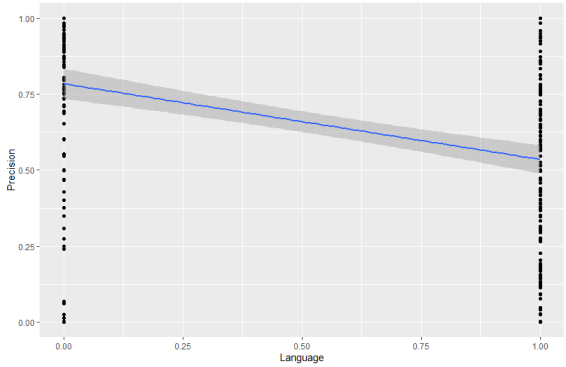
Figure 5.17: DNN attributes: Multiple linear regression with precision as the response variable



(a) 4th order polynomial regression attribute data, source on precision



(b) 2nd order polynomial regression attribute data, destination on precision



(c) 1st order polynomial regression attribute data, language on precision

Figure 5.18: DNN attribute: Polynomial linear regression of all significant predictor variables on precision

Comparison between found precision relationships

A correspondence in entity and attribute data is that the predictor variables Source, Destination and Languages have a significant impact on the precision. Smaller schemas and schemas in a single language seem to perform better on average. A rapid decline of performance can be seen in schema lengths, and also an increase in confidence intervals. The scarcity of data points when the x-axis takes on large values is present in all plots except for Language. This may cause the confidence interval to grow, and makes that we cannot say anything about these ranges. Also, higher polynomials are accepted to better incorporate these points, but this does not change much to the beginning of the line.

When entities have many equivalent matches, precision tends to decrease in a similar fashion as the schema lengths have. This does not hold for attributes, which may be an effect of how the structure similarity measure is set up. We elaborate on this in our discussion (Chapter **??**).

5.3.4 Relationship between recall and predictor variables

The performance of the recall due to the predictor variables on entity and attribute is assessed in the same way as we do for the precision. Section 5.3.3, about the DNN entities, gives a more detailed description of this method.

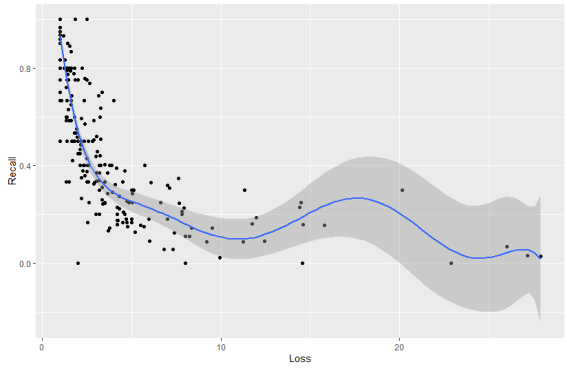
DNN entities

Multiple linear regression on entity data with recall as the response variable, indicates Ratio, Cardinality, Loss and Language as significant (Figure 5.19. Nevertheless, simple linear regression shows that only Loss and Language remain significant when all other predictor variables are not taken into account (Appendix F Figure F.3).

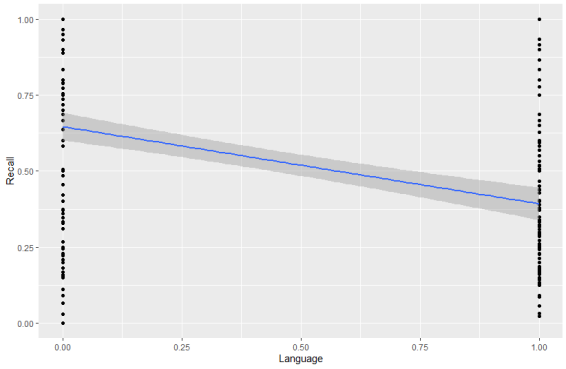
Polynomial regression shows that a polynomial of 7 is the best fit for Loss, while language can only remain linear since it is binary. We see a strong declining trend at the beginning of the x axis values, and a confidence interval that widens along the way. This is probably, just like previously discussed figures, due to the lack of data points. The rapid decline can be linked to the 1:1 cadinality constraint of our model. Our model can only select a single match, and therefore recall drops if multiple matches should have been selected. The Language plot on the use of multiple languages again shows to lead to lower performance, this time the recall.

```
Call:
lm(formula = Recall ~ Source + Destination + Ratio + Combinations +
    Cardinality + Loss + Abbreviations + Language, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7169 -0.1594 -0.0017  0.2019  0.9842

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.973e-01  2.229e-02  35.777  < 2e-16 ***
Source        -2.261e-04  5.366e-04  -0.421 0.673824
Destination   -1.011e-03  5.397e-04  -1.873 0.062204 .
Ratio          2.785e-03  1.288e-03   2.163 0.031441 *
Combinations   6.455e-07  4.729e-06   0.136 0.891537
Cardinality   -2.180e-02  6.464e-03  -3.372 0.000858 ***
Loss          -3.134e-02  5.097e-03  -6.148 2.90e-09 ***
Abbreviations -1.071e-01  5.445e-02  -1.966 0.050319 .
Language      -2.003e-01  2.976e-02  -6.730 1.06e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2257 on 263 degrees of freedom
Multiple R-squared:  0.5035,    Adjusted R-squared:  0.4884
F-statistic: 33.34 on 8 and 263 DF,  p-value: < 2.2e-16
```

Figure 5.19: DNN entities: Multiple linear regression with recall as the response variable



(a) 7th order polynomial regression entity data, loss on recall



(b) 1st order polynomial regression entity data, language on recall

Figure 5.20: DNN entity: Polynomial linear regression of all significant predictor variables on recall

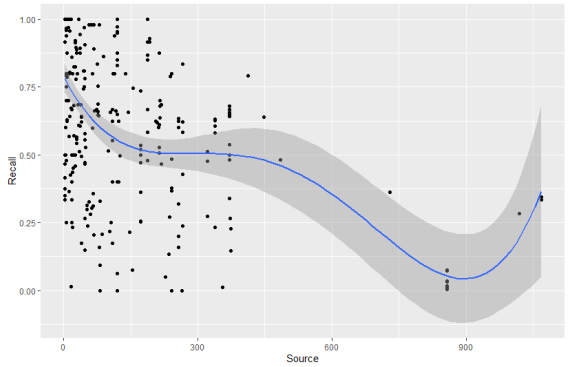
DNN attributes

A multiple linear regression on attribute data with recall as the response variable shows totally different significant predictor variables (Figure 5.21). The second step of simple linear regression indicates that only Source, Destination, Ratio, Cardinality and Loss remain as significant ones (Appendix F Figure F.4).

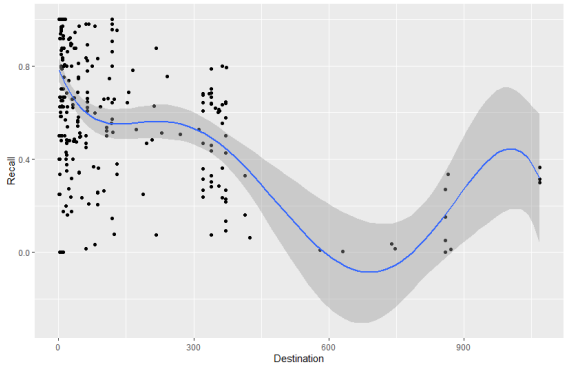
Just as in the previous sections, we increase the order of the polynomial regression until we get a best and significant fit. These fits are displayed in Figure 5.22 and show a behaviour that we have seen before at other polynomial regressions. The regression lines all start of with a steep descent, and meander off with an increasing confidence interval. Of all five graphs, the Cardinality and Loss have the least wide spread data points at the beginning of the x-axis. We observe some relatively high order polynomials, which resulted in a better fit but this is mainly due to the parts where there are few data points. A lower order would still be a good fit, and could look quite differently except for the first part of the regression line. The most important thing is that this is captured, and we can say something about the response variable behaviour in this stage.

```
Call:
lm(formula = Recall ~ Source + Destination + Ratio + Combinations +
    Cardinality + Loss + Abbreviations + Language, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.74101 -0.09346  0.02168  0.15415  0.45539

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.115e+00  3.362e-02  33.166  < 2e-16 ***
Source        -4.581e-04  1.050e-04  -4.362 1.83e-05 ***
Destination   -3.308e-04  8.662e-05  -3.819 0.000166 ***
Ratio         -1.705e-03  8.657e-04  -1.969 0.049975 *
Combinations   3.177e-08  2.306e-07   0.138 0.890513
Cardinality   -2.630e-01  2.538e-02 -10.360  < 2e-16 ***
Loss          -2.854e-03  1.204e-03  -2.370 0.018462 *
Abbreviations -9.473e-02  3.850e-02  -2.461 0.014484 *
Language      -2.528e-02  2.651e-02  -0.953 0.341240
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1893 on 275 degrees of freedom
Multiple R-squared:  0.5859,    Adjusted R-squared:  0.5739
F-statistic: 48.64 on 8 and 275 DF,  p-value: < 2.2e-16
```

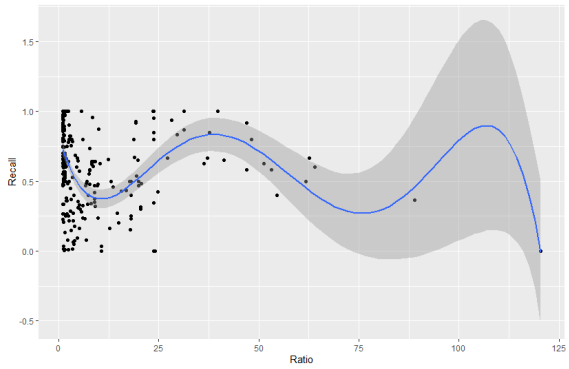
Figure 5.21: DNN attributes: Multiple linear regression with recall as the response variable



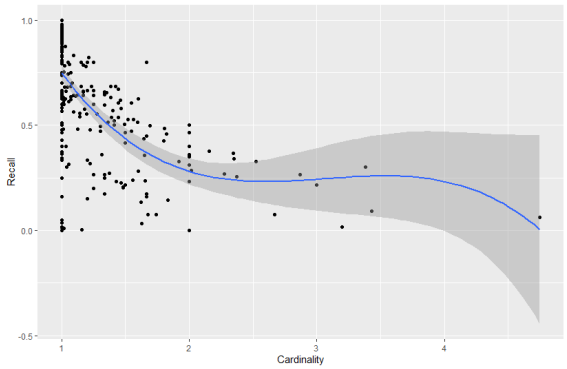
(a) 4th order polynomial regression attribute data, source on recall



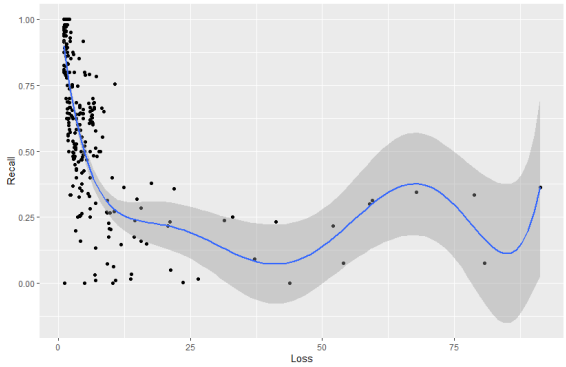
(b) 5th order polynomial regression attribute data, destination on recall



(c) 5th order polynomial regression attribute data, ratio on recall



(d) 3rd order polynomial regression attribute data, cardinality on recall



(e) 6th order polynomial regression attribute data, loss on recall

Figure 5.22: DNN attribute: Polynomial linear regression of all significant predictor variables on recall

Comparison between found precision relationships

The relationship comparison between entity and attribute data shows a combination of very different predictor variables. Ratio and Cardinality were not chosen in all other situations except for attribute data on recall. Nevertheless, it is not unusual that Cardinality and Loss appear together, since they are linked to the same category of characteristics. Especially due to the 1:1 cardinality constraint of our model, the Cardinality and Loss can quickly decrease which may explain their behaviour.

5.4 Net benefit of the IA solution

To complete research question 5 that we deal with in this chapter, we assess the approximate benefit of the IA situation we created. Since this depends on multiple variables, we cannot give a single representation. Therefore, we provide an estimation based on different scenarios.

We make some assumptions on the general schema matching process to make estimations less cumbersome and easier to understand.

1. The time to analyze both schemas for mappings is equal when the schema matcher is used or not. In case of no use, the human has to evaluate all possible combinations. When the schema matcher is used, the human still has to evaluate all possible combinations to intercept all possible false negatives and false positives.

2. Creating a mapping by hand (i.e. drawing a line in the eMagiz iPaaS) takes two seconds. This of course differs per turn and person, but is a safe estimate according to experienced users.

3. Deleting a mapping also takes two seconds. This is done with a right mouse click on the mapping and then selecting 'Delete...'.

With the assumptions in place, we can make a time estimation on the two different scenarios: Schema matching done by just a human and schema matching with our proposed IA solution. We take five integrations that differ in size and we have access to in the eMagiz iPaaS, to get an indication of the computation behaviour. Per integration, we replicate the schema matching situation three times to create an average of the total time it takes for our solution to follow the whole automated process of schema matching. The outcomes of these runs are given in 5.5, and are plotted in a graph in Figure 5.23. Although the number of combinations grow exponentially with an increase in source or destination schema length, the actual computation behaviour is linear. The line has an intercept of approximately 0.24 seconds, which is given by the smallest integration possible of 1 entity and 1 attribute at both sides. This is shown in 5.5 by the integration of 4 elements. Besides the minimal average waiting time of 0.24 seconds, the line of computation speed has a slope of 0.0004 seconds per combination. This is obtained by applying a linear trend line in Microsoft Excel. Since the number of combinations is a product of the number of source and destination schema elements, we also express it that way.

Table 5.5: Measurements of computation speed on 5 different combination sizes

Number of combinations	4	625	5146	12533	32100
Source × Destination elements	2 × 2	25 × 25	62 × 83	151 × 83	214 × 150
3 measurements in seconds	0.265949	0.347252	1.691011	4.744875	13.68379
	0.229536	0.341076	1.809653	5.050569	14.31551
	0.219936	0.348597	1.992374	4.834622	14.92644
Average in seconds	**0.238474**	**0.345642**	**1.831013**	**4.876689**	**14.30858**

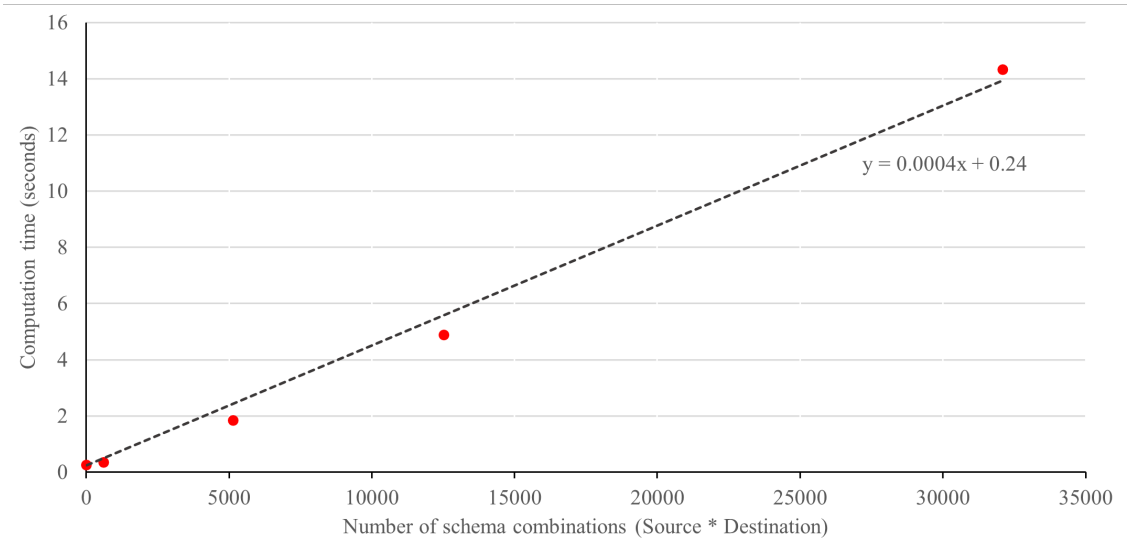


Figure 5.23: Computation speed in seconds per number of combinations

Previously we mentioned the two different scenarios of schema matching. We want to compare the human situation with the IA situation, leaving out the time needed for analysis. First, the time it takes for a human to draw all the mappings (T_H), based on our assumptions, is simply the number of mappings that have to be made (M) multiplied with the time it takes to draw a mapping:

$$T_H = 2 \cdot M$$

We define the Intelligence Amplification schema matching time (T_{IA}) to be a four part expression. There is a fixed average computation time of 0.24 seconds, a 0.0004 seconds computation time per schema element combination, a 4 seconds correction time for false positives and a 2 second mapping time for false negatives. The 4 seconds needed for false positives comes from the 2 seconds it takes to delete a mapping, and an extra 2 seconds to draw a new mapping. With this, we can write the following expression of (T_{IA}) :

$$T_{IA} = 0.24 + 0.0004 \cdot S \cdot D + 4 \cdot F_p + 2 \cdot F_n$$

Here, the source and destination schema lengths are expressed as S and D respectively. F_p and F_n are the number of false positives and false negatives respectively.

When trying to compare both situations, a high number of possible combinations between false positives and false negatives are possible. Therefore, we only explore the IA situations where these are minimal and maximal, but also how many false positives and false negatives are allowed to equal the situation without IA. We use the integrations from Table 5.5 and from now on, we refer to these integrations as Int 1 to Int 5. The number of mappings that should be made for each of these (M) are 2, 25, 83, 132 and 209 respectively. Table 5.6 shows the T_H, T_{IA} minimal and T_{IA} maximal per integration, which are computed according to the expressions above. The minimal situation uses 0 false positives and false negatives, the maximal situation uses the maximum number of false positives since these take the most time to correct. The table indicates that the IA solution can greatly outperform the human situation, given that performance is highest. The other way around shows that with a bad performance, the human situation is preferred.

Table 5.6: Benefit per integration

Situation	**Int 1**	**Int 2**	**Int 3**	**Int 4**	**Int 5**
T_H	4	50	166	264	418
T_{IA} minimum	0.2416	0.4900	2.2984	5.2532	13.0800
T_{IA} maximum	8.2416	100.4900	250.2984	337.2532	613.0800

The maximum number of false positives is something we can calculate due to our choice of mapping cardinality. A total number of mappings can be made less than or equal to the minimum of source or destination schema elements.

$$F_p \leq min(S, D)$$

Furthermore, the number of false positives can never exceed the number of false negatives since each false positive results in a lost match and thus in a false negative.

$$F_p \leq F_n$$

The maximum number of false negatives depends on how many mappings should be created and the number of true positives, which is the following:

$$M = T_P + F_n$$

With these constraints we can compute an error margin, an expression of how low the precision and recall can be given that $T_H = T_{IA}$. This is merely an indication, since there are too many combinations of precision and recall imaginable to consider. For this indication, we set the precision and recall to be equal and compute them with the goal seek function of Microsoft Excel. If precision and recall are both higher than the obtained values, it means that the IA situation is beneficial. If they are both lower than these values, it means that schema matching done by human took less time. Table 5.7 displays the computed values, which should be interpreted as one of the combinations of precision and recall being equal to the human situation. Note that we have to relax the integer property of the T_p, F_p and F_n to take on real numbers in order to let $T_H = T_{IA}$. In a real life situation, these values should be rounded to the nearest integer.

Table 5.7: Computation of error margin

Parameter	**Int 1**	**Int 2**	**Int 3**	**Int 4**	**Int 5**
S	2	25	62	151	214
D	2	25	83	83	150
M	2	25	83	132	209
Tp	1.3736	16.7483	55.7164	88.8755	141.5133
Fp	0.6264	8.2517	27.2836	43.1245	67.4867
Fn	0.6264	8.2517	27.2836	43.1245	67.4867
Precision	0.6868	0.6699	0.6713	0.6733	0.6771
Recall	0.6868	0.6699	0.6713	0.6733	0.6771

The table gives us an indication of what the performance of the IA schema matching solution should be for it to be beneficial on the given integrations. Nevertheless, in a practical situation the user does not know what the precision and recall are beforehand. Moreover, the 'beneficial boundary' as we describe it also depends on the number of schema elements S and D, and on the number of mappings to be created M. We can therefore conclude that such a comparison can only be done retrospectively. Describing the net benefit of the IA solution is thus only of use theoretically. The user should decide on whether to use the solution, given the estimation

he or she has made on the findings of the solution behaviour described in Section 5.3. Given the findings we presented, the IA solution can be really beneficial if performance is above a certain level. If it is not, it can have a negative impact on the schema matching problem if the user decides to manually correct every error.

5.5 Conclusions RQ 6

RQ 6 What is the performance of the proposed solution design and when is it beneficial?

The choice of cardinality for our model has its effects on the training possibilities and the interpretation of outcomes. The number of mapping samples for both entities and attributes shrinks in size, and the maximum average recall than can be achieved lowers to 0.644 and 0.826 respectively. Mappings that otherwise would have been classified as a true positive, are now classified as false negative if there was a more confident matching pair related to a corresponding source or destination schema element.

Training set sizes are chosen according to the rule of thumb proposed by Kavzoglu (2009) and consist of equal parts *match* / *no-match* samples due to undersampling. The validation set size is chosen to be the same for all model settings, and at least 20% of the total sample size in all cases. Training and validation is replicated 5 times, all with a different but consequent use of random seeds. The average of these 5 replications is used as a representative performance.

Confidence thresholds that are assessed in combination with our model are based on the typical confidence distributions the models produce. The batch size and number of epochs needed for training both models are obtained by pivoting with the settings until the training error converges for all training settings. Convergence implies that the limit of learning is reached and it is not necessary train even further.

The experiments of the model sizes resulted in a pattern for both models where the higher number of samples generally produced better results than their counterparts. Therefore, we assessed the performance on both models for an extra, higher, number of samples. This additional experiment resulted in slightly worse performances, indicating that both models have an underlying optimum curve. The average performances in the DNN entities show a couple of unusual scores, that could be explained by a replication in which no predictions were made due to confidences scores being too low. A display of average standard deviations across f-measures is made to get a better understanding of the fluctuations in the f-measure behaviour. The average standard deviations are close to each other and do not provide us extra information or value in making a decision on the optimal parameter settings. We found a setting for the DNN entities of 2 hidden layers, 10 nodes per hidden layer and 1800 training samples to be optimal. For the DNN attributes, 1 hidden layer, 18 nodes per hidden layer and 2500 traing samples gave the best average result.

Threshold settings experiments for both DNN's resulted in a somewhat similar performance, where the recall increasingly declines and the precision is relatively stable. Our hypothesis that predictions with a higher confidence hold more true positive classifications is not true. The optimal threshold settings for the DNN entities and DNN attributes are 0.8 and 0.85 respectively.

With the optimal settings, we tested the DNN's on the complete dataset. We replicated this five times, where each replication had a different training and test set that did not overlap. An average over four measurements is calculated per integration. With these averages we calculated a total average. For the entity DNN, an average precision and recall of 0.72651 and 0.53909 are achieved respectively. Performances are spread with no clear distribution visible, other than a large spike at 1.00 for all performance measures. The attribute DNN scores an average precision and recall of 0.65206 and 0.61465. They show a similar pattern as the entity DNN, where performances are scattered across the range. The entity DNN has a better average

performance that the one for attributes, and also shows the smallest gap with the maximal average recall reachable.

We applied linear regression to 8 different integration characteristics to describe the performance by possible relationships. The precision seemed to be significantly impacted by the number of schema elements in source and destination schemas and also by the use of one or multiple languages. A rapid decline can be seen when moving from the smallest schemas to some larger ones. Also, the confidence interval widens and we can say increasingly less about the performance. This is at least an effect of the fewer data points we have in that area. The use of a single language in both schemas is proven to yield better precision based on the whole data set.

Taking recall as the response variable, outcomes on relationships differ slightly. On the attribute data, we observe the same behaviour as with the precision response variable, with an added insight on the Ratio, Cardinality and Loss. The Loss also appears in the entity data and it is no coincidence that it is linked with recall. Discarding matches with a high confidence can have a side effect of discarding correct matches, resulting in more false negatives. Although there are fewer similarities to be found in predictor variables when comparing on recall, we observe a strong decline at each graph which is a useful insight on our solution.

In the last section of Chapter 5 we provided an estimation of the net benefit of our IA solution. This is done on three assumptions that generalize and simplify the quantification of schema matching with and without our solution. We conclude that the time needed for analyzing the schemas takes approximately equally long in both situations. Drawing or deleting a mapping in the eMagiz iPaaS takes about 2 seconds, which results in 2 seconds of human effort for each false negative and 4 seconds for each false positive.

Next to the assumptions, provided an approximation of the computation time based on 5 integrations that differ in size. A linear behaviour of 0.0004 seconds per element combination and a base computation time of 0.24 seconds describes the solution best computation time wise. If we compare the IA solution to the average human according to our expressions, mixed results can be observed. When performance is excellent, the IA solution greatly outperforms the human. On the other side, if performance is bad, the human schema matching is preferred.

We attempted to give a practical comparison overview, but there are too many factors that play a role for displaying it in a concise manner. An indication is given on the 5 integrations earlier used which showed that precision and recall should be both around 0.67 for the IA solution to level or outperform the human. Also here, too many combinations of precision and recall are possible to give a clear rule. Comparison can be best done in retrospective, something that does not suit well to the practical aspect. Therefore, the user should decide on whether to use the solution, given the estimation he or she has made on the found characteristic relationships with performance.

6 Conclusions

In this final chapter, we can conclude our research with answering the main research question in Section 6.1. A critical reflection on our choices and accompanying results is provided in Section 6.2. This is followed up by an overview of the encountered limitations in Section 6.3. Section 6.4 elaborates on the contributions to theory and practice. Lastly, Section 6.5 and 6.6 sum up recommendations for future research and eMagiz respectively.

6.1 Findings for the main research question

This research, which is a result of a practice-inspired problem on schema matching encountered by eMagiz, had multiple aims of gaining knowledge and creating a tangible product. The knowledge that we needed to obtain is used for shaping a suitable solution to the schema matching problem. With the knowledge we also wanted to create a blueprint and a realized artifact of the schema matcher. For reaching these research objectives, we answered multiple research questions during the chapters, with which we can answer our main question.

MQ *How can the process of schema matching be augmented with machine learning techniques to provide highly accurate and time saving results, when integrating systems in an iPaaS?*

This research has shown that the process of schema matching in an iPaaS can be augmented with our machine learning solution. The solution can be called a hybrid schema matcher since it makes use of multiple similarity measures, taking into account different perspectives of the matching problem. A linguistic, constraint-based and structural approach are taken in comparing all schema elements, divided over a total of 6 different similarity measures. These are used since there is information hidden in different perspectives and which can help make a distinction between similar looking cases. Two distinct Deep Neural Networks, one for both entities and attributes, are suitable options for predicting matches. They utilize the similarity measures as input and produce a list of possible matches with their confidence. Out of all the matching options, only the most confident one is selected to satisfy a 1:1 mapping cardinality.

Besides the impact of the structure of the solution, the estimation of the user also plays a big role in whether the augmented schema matching process is beneficial. Integrations within an iPaaS can vary widely between its characteristics, that a human could combine to assess 'the difficulty' of the schema matching problem. We have studied the behaviour of the solution performance given integration characteristics, because performance is scattered and can hardly be explained by a single statement. It is up to the user to take into account the outcomes of the performance behaviour and to assess whether the integration would benefit from the IA solution.

The performance and how beneficial the IA solution is both depend on a lot of factors. Unfortunately, that it makes it only possible to assess these in hindsight. Depending on the integration, precision and recall roughly have to be around 0.65 to 0.68 in order for our solution to match the old process time wise. If all matches are predicted correctly and thus precision and recall are 1.0, the schema matching process takes only a couple of seconds. It is then significantly better than schema matching by hand, and can even save minutes for larger integrations. On the other hand, if all matches are incorrectly predicted, it takes the user roughly double the amount of time it would have taken without the IA solution. This is due to the corrections that the user has to do manually. These indicated performance boundaries show us that outcomes can vary from wanted to unwanted, which emphasizes the role of the user in this situation on assessing a fit for using the IA solution.

6.2 Implications and discussion of results

In this section we discuss the interpretation of our results and what the implications are that come with our choices.

Intelligence Amplification

The use of Intelligence Amplification as the area we place our solution design in, might not be widely familiar in science. The terminology is not consistently used in literature, and most of them refer to the same concept but with a different name. There are two downsides to this, the first being a harder understandability of the concepts of this research. It takes the reader an extra step to relate the theory to already familiar constructs. The second downside is that there is not much written about Intelligence Amplification, thus relevant literature is harder to find. It is therefore harder to assess whether the found literature exactly fits the research context. We could have chosen to not use Intelligence Amplification as our framework, which could work out from what we found in relevant literature. However, the concept really fits the research context and helped a lot in getting clear what was needed in our design.

Our developed generic IA schema matching framework is based on the process flow of Rahm and Bernstein (2001), which is a simplistic view on schema matching. Although the greater part of situations follows these steps, there are situations where it is no continuous process or parts of it are more in depth than they appear in this simplistic view. Therefore, our model could create the feeling that a schema matching process should exactly follow these steps, leaving no room for the more creative ideas or exceptional cases. Although a more complex process flow can still be fitted into our framework, it would not be very representative. Thus, our framework stimulates the creation of simple solutions and does not feel quickly relatable to the more complex designs and situations.

The choice of our used methodology, the Action Design Research, has led to multiple user feedback sessions on the design of the model. This helped in getting it into the right direction, but it is not fully executed as intended by the methodology. Due to the time limits of this research, a full implementation in the platform is not feasible. Therefore, the artifact that came from guided emergence is still only tested in a local environment and has no implemented aspects of proper design or user experience. Other methodologies therefore could have been useful up to the same phase. Nevertheless, the ADR is still the most fitting to this research context.

Choices in solution design

With our systematic literature review we aimed to find a machine learning technique that could handle great complexity, is scalable and easy to handle. Together with the findings of previous research a variation of Neural Networks, Deep Neural Networks, checked the boxes and seemed to have no major downsides. However, during the setup experimentation in the training phase, we discovered the limits on the amount of data that could be taken into consideration. The optimum discovery of 1800 and 2500 training samples for entities and attributes respectively, meant that only a really smart part of the total data is represented. Cases that are not represented in the training phase are more likely to cause bad performance, which causes the concern of what to include in the training set. With the current solution structure it is impossible to represent integration data of even more companies. We have found optima, although with a small chance of being local, that indicate the limits of this prediction technique. Feeding it more samples from different cases will not solve the problem of under-representation, but possibly a different prediction technique such as Naive Bayes will.

The similarity measures in our solution design represent the logical and interpretable perspectives on element comparison. As we found in literature, there are many more that could have been included that are all designed for variations or completely new views. Although we indicated that the black box behaviour of the Deep Neural Network was a proper choice for eMagiz, it made it very hard to assess the value of individual similarity measures on the outcome.

Therefore, we do not know if this combination of similarity measures covers all information that should be covered, if some are redundant or that new ones should be added. A more statistical approach would be useful for assessing the added value of the similarity measures. For now, we cannot say what the relation between the solution design and the outcomes is and where there is improvement possible in choosing the similarity measures.

The elaboration on the choice of the mapping cardinality was clear: to reduce complexity. A characteristic of complexity in non-1:1 mapping would be to decide how many matches to apply and how many to leave out, considered per element. This could result situations that are far worse than a 1:1 mapping case with each mapping predicted wrong. On the other hand, it would also mean that the maximum reachable recall is higher. With 1:1 mapping we leave out matches that are not the most confident one, but could still be perfect matches. We did not assess the quality of these left out matches, only the most confident ones.

Data

There is a trade-off in the choice on how much data must be used to make statements about the model performance. As explained earlier, more data is not likely to be represented by the model since it can only handle a relatively small number of samples. More data would in that case be counterproductive. On the other hand, the description of the relationship between integration characteristics and the performance would benefit from it. We saw that confidence intervals grew when data points got fewer, leading to a polynomial regression line that is losing its worth along the x-axis. More data of the less presented cases would lead to better predictions, giving the user a more secure grip on whether to engage the IA solution.

From empirical experiences and statements in literature about schema matching, the belief was that the complexity of the dataset used in this research was high. We adjusted the choice of prediction technique and other smaller decisions to this perceived level. Obtaining results and working with our solution design gave us insights on the actual complexity. A better description of this dataset would be 'high in variability'. We developed and tested our solution multiple times on a select and uniform set of integrations, on which the DNN had no trouble in quickly learning and predicting the right outcomes. For instance, integrations were only included if they had a length of less than 100, to reduce computation time and make debugging easier. With this setting, results were much higher and averaged around 0.90 for both precision and recall. Nevertheless, when all integrations were included we obtained the results as they are displayed in this report. The result outcomes are highly variable, because the schema matching problems are highly variable too. This creates the thought that a single DNN is simply not enough to cover all the aspects that come with schema matching at eMagiz. Therefore two options are possible: decrease the scope of the schema matcher to make it specific for only a selection of cases, or develop a composite schema matcher. In hindsight, it would have been better for the development of this solution to knew this earlier, if it was possible at all. A real score or feel on the complexity and variability of the data was not present and is hard to obtain. Therefore, in pursuit of a better schema matcher, this solution design is a product of a cycle that comes with useful feedback on the data.

Results

Assessing the properties of our data, we concluded that a maximum recall of 0.644 and 0.826 could be achieved for entities and attributes respectively. In pursue for these numbers, we trained and validated both DNN's of which the following settings were found optimal: 1800 training samples, 2 hidden layers and 10 nodes for the entity DNN and 2500 training samples, 1 hidden layer and 18 nodes for the attribute DNN. However, results were close and a different setting would not have made a big difference. Stating that these are the best settings is not significant, definitely not when not a large area of settings is explored. The settings that are explored give a good view on the found optimum, but it does not immediately rule out that

this is a local optimum.

In the section about the relationship between integration characteristics and the performance, we did not include interaction effects. This is the result of the many combinations that can be made between all predictor variables and the time limits in this research. It could be that there is a hidden relationship that clarifies much about performance, and it could also be that there is another important integration characteristic that we did not think of. With the most obvious ones analyzed, the descending patterns of all graphs can give the user the needed insight. To say more about the parts in the graphs that have a large confidence interval, more data was needed. Here a decision must be made, if new data is still classified as in the same sector and representative, or that another approach must be taken. The choice of the order of the polynomial in the graphs is a questionable point. A smaller order is easier for interpretation and larger orders tend to represent an overfit on the data. There is a chance that more data solves this issue.

The expression of the net benefit of the IA solution is highly indicative since a lot of approximations and assumptions needed to be made. To start off, we do not know how much time is needed to analyse schemas before matching and to correct false positives. In that way, we also do not know what the share is of the approximated net benefit in the total schema matching process. The two seconds committed to draw or remove a line is based on user experience and also on the user not making a fault. Drawing a line can fail, which may take more seconds than our indication. This can have its negative influence on the schema mapping performance of the human. The computation speed line is roughly drawn on 5 random different integrations, but this does not matter much to the outcome. The intercept and gradient are expressed in such a small number of seconds, that a deviation from this still makes no significance. Therefore, this line and many variations of it can be correctly used. The computation of the error margin of the IA solution depends on a couple of variables, which are related in some way. We thought about setting up a single formulation to capture the behaviour, but the complexity seemed too high and practicality too low.

6.3 Limitations

A limit in the guided emergence of our artifact is that development of features in the platform takes a long time. Full implementation in the iPaaS is not possible at this moment, leaving the artifact in its bare minimum state. It made it hard to let others use it, play with it and give feedback. Therefore, these moments had to be rearranged for it to provide useful feedback. Also, a time-wise comparison of the tool was hard to arrange, for which we choose to work out the theoretical alternative in Section 5.4.

The second limit that came with the long process of developing the platform, is that the feedback on appending the thesauri is not established. Since the information must come from validated integrations, the connection within the platform is outside the scope of this research and is never addressed. The moment our solution is fully integrated within the platform, the possibility for the feedback loop regarding updating the thesauri is enabled, that should then be created by the developers of the platform.

Much effort has been put in minimizing the computation time of the model's Python script. Still, testing a lot of integrations could take hours. Therefore limits had to be set on what model settings to analyze. With our used method, we have obtained a decent overview on a found optimum. It would have been nice to explore areas around this overview, to check for possible other optima or to discover patterns. The use of more replications also would improve the data even more, although our 5 replications are sufficient to see patterns.

In displaying the relationship between integration characteristics and performance, we ran into the limit of having little data. Especially for the higher values on the x-axes, we were constantly lacking data for the outlying cases. This resulted in a wide confidence interval,

leaving few relevant words to say about the behaviour. To tighten the confidence interval, more data on these cases is needed, that does not deviate too much from the used set.

6.4 Contributions to theory and practice

With this research we made several contributions to both the scientific community and the practical environment where the artifact is suited for.

First of all, we created a generic framework for intelligence amplification in schema matching, with an accent on learning. The framework is based upon the basic principles of schema matching, but differentiates from it by emphasizing the responsibilities of both human and computer. With the incorporation of feedback loops and the knowledge base, it is no longer a static model and learning or adapting to the environment is stimulated. It can be a good starting point for new research on this topic.

Second, a unique hybrid approach for schema matching is constructed and validated. The model differentiates itself from other models by the combination of similarity measures, but more importantly by the ability to overcome language barriers and abbreviations without the use of external applications. This last feature could be applicable to any schema matching problem with a solid database of mapping records, that encounters the use of multiple languages, abbreviations or simply synonyms and technical sub-languages. The provided algorithms along the descriptions make the application of these similarity measures more accessible.

Third, the method for discovering relations between integration characteristics and performance is unique for the schema matching field, based on the absence of it in related literature. Papers mostly focus on the explainability of their input variables or the average performances. Sometimes, characteristics are addressed as subgroups on which performance is measured, but never is this done with the techniques of regression. An advantage of the use are the deeper insights it provides in the behaviour of the model and where the improvement areas are.

Fourth, a large practical contribution which is tailor made to the eMagiz environment is a working solution design. This solution design encompasses every aspect that is described in this thesis. The design is constructed and validated locally, and there has been tight cooperation for a knowledge and product transfer so it can be implemented to work in the platform. A total of two Python scripts, two thesauri, two deep neural network model files, a sql query overview and a Mendix project are presented together with instructions.

Fifth and last, a practical overview on the behaviour and benefits of our schema matcher is presented. With this, the user can create a quick understanding of the relations between integration characteristics and possible performance. Although it totally depends on the user whether to use the artifact or not, it indicates whether the human-computer cooperation might be beneficial. This encourages to establish the concept of Intelligence Amplification.

6.5 Recommendations for future research

We mentioned earlier in Section 6.2 that the data of eMagiz can be described as highly variable. The current solution is not capable of addressing all the scenarios with a somewhat similar performance. Therefore, a good option is to investigate in developing a composite matcher, which is basically a number of different hybrid matchers tailor made to the scenario at hand. Right now the user has to assess whether the scenario is right for our IA solution, with a composite matcher the computer takes over this task. It could create a more consistent performance and as a result increase the relevance of the schema matcher in the iPaaS of eMagiz.

The use of integrity constraints defined in Doan et al. (2001) has a promising potential on shaping the solution to the environment it should be implemented in. Based on feedback from eMagiz iPaaS users, there are some unwritten schema matching rules that apply to almost every integration. These come with the working standards of the company, and could be obtained by collecting enough feedback from these users. These integrity constraints are rules that must be

applied after the prediction of the matches as a sort of final check. The value it would add is a possibly slightly improved precision and or recall.

With the problem of correctly representing data in training the DNN, it could be interesting to assess performance of various other machine learning prediction techniques that incorporate more data than the current solution. We mentioned earlier that Naïve Bayes would be an option and Support Vector Machines could be inserted here as well. It is no guarantee that this will overcome the complications with the variability of the data, but it possibly might deliver a slightly improved performance due to better representation of all cases in the dataset.

In the further development of this IA solution, it would be useful to explore a larger area of parameter settings. Especially the parameters that are related to the DNN's could possibly uncover other optima, or further confirm our optimum.

The effects of the integration characteristics on performance can be better analyzed in the regions where data is scarce. If more data is collected to evenly distribute the different integration cases, the confidence interval could be narrowed. This leads to drawing better conclusions on the displayed relationship and eases the use of the solution in these earlier underrepresented cases.

6.6 Recommendations for eMagiz

A recommendation for eMagiz on direct use of our developed solution, is to use the indicated settings that represent the building blocks for an optimal performance. Of these settings, the behaviour is analyzed and displayed, which may not be exactly the same for other settings. Should the developers of eMagiz decide to use a different data set to train the model on, then it would be wise to also experiment with model settings and adapt them according to the optimal settings.

In the case of the IA solution being implemented in the platform of eMagiz, it is wise to directly create the feedback loop that enables appending of both thesauri. This secures that the model is not static but instead learns from the validated cases. It is up to the developers of the platform to find a way that suits best to establish this feedback loop, since it multiple ways are possible that have their own effects on performance and usability of the application.

It would be good to know for eMagiz how the IA solution performs on other data than the four logistical companies we used. There is a good chance that performance does not differ too much, which means that its applicability can be widened to all platform users. This can be done by simply testing on the exact same parameters and changing the input data to the integrations and companies of interest.

Practical use of the schema matcher should be done in combination with an analysis of the integration at hand. Given the conclusions on integration characteristics, the user should match his or her expectations to the difficulty of the integration. The task for eMagiz is to make this as clear as possible to the user. This can be done for example by providing a short manual, or make the user assess the complexity of the matching problem. Probably a bit of experience should be needed in order to master the use of the tool. Nevertheless, users should be aware that there is always a chance it might not perform as they wanted it to perform.

Bibliography

Anam, S., Kim, Y., Kang, B., & Liu, Q. (2015b). Designing a knowledge-based schema matching system for schema mapping. In *Conferences in research and practice in information technology series* (Vol. 168, pp. 69–77).

Anam, S., Kim, Y., Kang, B., & Liu, Q. (2015c). Schema mapping using hybrid ripple-down rules. In *Conferences in research and practice in information technology series* (Vol. 159, pp. 17–26).

Anam, S., Kim, Y., Liu, Q., Jung, T. M., Lee, Y. S., & Cho, S. B. (2010). *Knowledge Management and Acquisition for Smart Systems and Services* (Vol. 6232). Retrieved from `http://www.scopus.com/inward/record.url?eid=2-s2.0-78049313196partnerID=tZOtx3y1` doi: 10.1007/978-3-319-13332-4_7

Anam, S., Kim, Y. S., Kang, B. H., & Liu, Q. (2015a). Designing a knowledge-based schema matching system for schema mapping. *Conferences in Research and Practice in Information Technology Series*, *168*(AusDM), 69–77.

Ashby, W. R. (1956). *An introduction to cybernetics.* New York,J. Wiley,. Retrieved from `https://www.biodiversitylibrary.org/item/26977`

Berkovsky, S., Eytani, Y., & Gal, A. (2005). Measuring the relative performance of schema matchers. *Proceedings - 2005 IEEE/WIC/ACM InternationalConference on Web Intelligence, WI 2005*, *2005*(June 2014), 366–371. doi: 10.1109/WI.2005.94

Berlin, J., & Motro, A. (2002). *Database schema matching using machine learning with feature selection* (Vol. 2348).

Bishop, C. (2006). *Pattern Recognition and Machine Learning.* doi: 10.1007/978-3-030-57077-4_11

Bonaccorso, G. (2017). *Machine learning algorithms.* Packt Publishing Ltd.

Bonifati, A., & Eds, E. R. (2011). *Schema Matching and Mapping.* doi: 10.1007/978-3-642-16518-4

Brownlee, J. (2018). Better Deep Learning. Train Faster, Reduce Overfitting, and Make Better Predictions. *Machine Learning Mastery With Python*, *1*(2), 539.

Buis, J. T. P. (2017). *Applying intelligence amplification to the problem of schema matching* (Unpublished doctoral dissertation). University of Twente.

Candidat, F. D. (2009). Towards a Generic Approach for Schema Matcher Selection : Leveraging User Pre- and Post-match Effort for Improving Quality and Time Performance.

Cheng, R., Gong, J., & Cheung, D. (2010). Managing uncertainty of XML schema matching. In *Proceedings - international conference on data engineering* (pp. 297–308). doi: 10.1109/ICDE.2010.5447868

Do, H.-H. (2006). Schema Matching and Mapping-based Data Integration. *Department of Computer Science*(August), 222. Retrieved from `lips.informatik.uni-leipzig.de/files/2006-4.pdf`

Do, H.-H., Melnik, S., & Rahm, E. (2003). Comparison of Schema Matching Evaluations. In A. B. Chaudhri, M. Jeckle, E. Rahm, & R. Unland (Eds.), *Web, web-services, and database systems* (pp. 221–237). Berlin, Heidelberg: Springer Berlin Heidelberg.

Do, H.-H., & Rahm, E. (2002). COMA — A system for flexible combination of schema matching approaches. *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*(January), 610–621. doi: 10.1016/b978-155860869-6/50060-3

Do, H. H., & Rahm, E. (2007). Matching large schemas: Approaches and evaluation. *Information Systems*, *32*(6), 857–885. doi: 10.1016/j.is.2006.09.002

Doan, A., Domingos, P., & Halevy, A. Y. (2001). Reconciling schemas of disparate data sources. , 509–520. doi: 10.1145/375663.375731

Dobrkovic, A., Döppner, D. A., Iacob, M.-E., & van Hillegersberg, J. (2018). Collaborative Literature Search System: An Intelligence Amplification Method for Systematic Literature

Search. In S. Chatterjee, K. Dutta, & R. P. Sundarraj (Eds.), *Designing for a digital and globalized world* (pp. 169–183). Cham: Springer International Publishing.

Dobrkovic, A., Liu, L., Iacob, M.-E., & van Hillegersberg, J. (2016). Intelligence Amplification Framework for Enhancing Scheduling Processes. In M. y Gómez, H. J. Escalante, A. Segura, & J. d. D. Murillo (Eds.), *Advances in artificial intelligence - iberamia 2016* (pp. 89–100). Cham: Springer International Publishing.

Duchateau, F., Coletta, R., Bellahsene, Z., & Miller, R. (2009). (Not) yet another matcher. In *International conference on information and knowledge management, proceedings* (pp. 1537–1540). doi: 10.1145/1645953.1646165

Ebert, N., Weber, K., & Koruna, S. (2017). Integration Platform as a Service. *Business and Information Systems Engineering*, *59*(5), 375–379. doi: 10.1007/s12599-017-0486-0

Engelbart, D. C. (1962). Augmenting human intellect: A conceptual framework. *Menlo Park, CA*.

Englebienne, G. (2019). *Machine Learning I Course 1: Introduction.*

Euzenat, J., & Shvaiko, P. (2007). *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-540-49612-0 doi: 10.1007/978-3-540-49612-0

Gal, A. (2011). *Uncertain Schema Matching* (Vol. 3) (No. 1). doi: 10.2200/s00337ed1v01y201102dtm013

Gal, A., & Sagi, T. (2010). Tuning the ensemble selection process of schema matchers. *Information Systems*, *35*(8), 845–859. Retrieved from http://dx.doi.org/10.1016/j.is.2010.04.003 doi: 10.1016/j.is.2010.04.003

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Retrieved from http://files.sig2d.org/sig2d14.pdfpage=5

Groothuis-Oudshoorn, K. (2020). *Introduction to Statistical Learning.* Course: Applied Statistical Learning.

Haibo, H., & Garcia, E. (2009). Learning from Imbalanced Data. *Proceedings - IEEE Transactions On Knowledge And Data Engineering*, *21*(9), 1263–1284. doi: 10.1109/ICTAI.2019.00131

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Two Paradigms on Research Essay Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–79.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. SPRINGER. doi: 10.1016/j.peva.2007.06.006

Jeong, B., Lee, D., Cho, H., & Lee, J. (2008). A novel method for measuring semantic similarity for XML schema matching. *Expert Systems with Applications*, *34*(3), 1651–1658. doi: 10.1016/j.eswa.2007.01.025

Joy, A. (n.d.). *Pros And Cons Of Reinforcement Learning.* Retrieved from https://pythonistaplanet.com/pros-and-cons-of-reinforcement-learning/

Jung, T. M., Lee, Y. S., & Cho, S. B. (2010). *Knowledge Management and Acquisition for Smart Systems and Services* (Vol. 6232). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-78049313196partnerID=tZOtx3y1

Kavzoglu, T. (2001). An Investigation of the Design and Use of Feed-Forward Artificial Neural Networks in. *Information Systems*(May), 1–307.

Kavzoglu, T. (2009). Increasing the accuracy of neural network classification using refined training data. *Environmental Modelling & Software*, *24*, 850–858.

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering* (Tech. Rep.). Keele: School of Computer Science and Mathematics, Keele University. doi: 10.1145/1134285.1134500

Lee, M. L., Yang, L. H., Hsu, W., & Yang, X. (2002). XClust: Clustering XML schemas for effective integration. *International Conference on Information and Knowledge Management,*

Proceedings, *117543*(065), 292–299.

Lewis, M. (1998). Designing for Human-Agent Interaction. *AI Magazine*, *19*(2), 67–78.

Licklider, J. C. (1960). Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, *HFE-1*(1), 4–11. doi: 10.1109/THFE2.1960.4503259

Marie, A., Gal, A., & Kapur, D. (2008). *Lecture Notes in Artificial Intelligence: Preface* (Vol. 5081 LNAI). doi: 10.1007/978-3-540-75410-7_5

Marie, A., Gal, A., & Smirnov, M. (2005). *Boosting schema matchers* (Vol. 3457) (No. PART 1). doi: 10.1007/978-3-540-88871-0_20

Mocanu, E. (2020a). *Machine Learning I Course 5: Support Vector Machines.*

Mocanu, E. (2020b). *Machine Learning I Course 8: Dimensionality Reduction.*

Mukherjee, D., Bandyopadhyay, A., Chowdhury, R., & Bhattacharya, I. (2020). Learning Knowledge Graph for Target-driven Schema Matching. In *Acm international conference proceeding series* (pp. 65–73). doi: 10.1145/3430984.3431013

Oldham, N., Thomas, C., Sheth, A., Verma, K., Benatallah, B., & Motahari Nezhad, H. R. (2005). *Interoperability in semantic web services* (Vol. 3387). doi: 10.1007/978-3-540-30581-1_3

Pan, Y. (2016). Heading toward Artificial Intelligence 2.0. *Engineering*, *2*(4), 409–413. Retrieved from `http://dx.doi.org/10.1016/J.ENG.2016.04.018` doi: 10.1016/J.ENG.2016.04.018

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, *30*(3), 286–297. doi: 10.1109/3468.844354

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. doi: 10.2753/MIS0742-1222240302

Piest, J. P. S., Meertens, L. O., Buis, J., Iacob, M. E., & van Sinderen, M. (2020). Smarter interoperability based on automatic schema matching and intelligence amplification. *CEUR Workshop Proceedings*, *2900*(July).

Rahm, E. (2011). Towards large-scale schema and ontology matching. In *Schema matching and mapping* (pp. 3–27). Springer.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, *10*(4), 334–350. doi: 10.1007/s007780100057

Rahm, E., Do, H. H., & Maßmann, S. (2004). Matching large XML schemas. *SIGMOD Record*, *33*(4), 26–31. doi: 10.1145/1041410.1041415

Rodrigues, D., Da Silva, A., Rodrigues, R., & Dos Santos, E. (2015). Using active learning techniques for improving database schema matching methods. In *Proceedings of the international joint conference on neural networks* (Vol. 2015-Septe). doi: 10.1109/IJCNN.2015.7280630

Sahay, T., Mehta, A., & Jadon, S. (2020). Schema matching using machine learning. In *2020 7th international conference on signal processing and integrated networks, spin 2020* (pp. 359–366). doi: 10.1109/SPIN48934.2020.9071272

Schmidts, O., Kraft, B., Siebigteroth, I., & Zündorf, A. (2019). Schema matching with frequent changes on semi-structured input files: A machine learning approach on biological product data. In *Iceis 2019 - proceedings of the 21st international conference on enterprise information systems* (Vol. 1, pp. 196–203). doi: 10.5220/0007723602080215

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R., & Sein, M. (2011). Action Design Research Action Design Research1. *Source: MIS Quarterly*, *35*(1), 37–56.

van Breemen, A. J., Farkas, J. I., & Sarbo, J. J. (2011). Knowledge representation as a tool for intelligence augmentation. *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*(January 2010), 321–341. doi: 10.4018/978-1-60960-551-3.ch013

Van Rijsbergen, C. J. (1979). *Information Retrieval, second ed.* London: Butterworth.

Verschuren, P., & Doorewaard, H. (2007). Het ontwerpen van een onderzoek. Vierde druk. *Den Haag: Lemma uitgevers*.

Wieringa, R. J. (2014). *Design science methodology: For information systems and software engineering*. doi: 10.1007/978-3-662-43839-8

Xia, C., & Maes, P. (2013). The design of artifacts for augmenting intellect. *ACM International Conference Proceeding Series*, 154–161. doi: 10.1145/2459236.2459263

Zheng, N.-n., Liu, Z.-y., Ren, P.-j., Ma, Y.-q., & Chen, S.-t. (2017). Hybrid-augmented intelligence: collaboration and cognition. *Frontiers of Information Technology & Electronic Engineering*, *18*(2), 153–179.

A ADR Methodology

(1) Identify and conceptualize the research opportunity
(2) Formulate initial research questions
(3) Cast the problem as an instance of a class of problems
(4) Identify contributing theoretical bases and prior technology advances
(5) Secure long-term organizational commitment
(6) Set up roles and responsibilities

Figure A.1: Tasks in the Problem Formulation Stage

(1) Discover initial knowledge-creation target
(2) Select or customize BIE form
(3) Execute BIE cycle(s)
(4) Assess need for additional cycles, repeat

Figure A.2: Tasks in the Building, Intervention, and Evaluation Stage

(1) Reflect on the design and redesign during the project
(2) Evaluate adherence to principles
(3) Analyze intervention results according to stated goals

Figure A.3: Tasks in the Reflection and Learning Stage

(1) Abstract the learning into concepts for a class of field problems
(2) Share outcomes and assessment with practitioners
(3) Articulate outcomes as design principles
(4) Articulate learning in light of theories selected
(5) Formalize results for dissemination

Figure A.4: Tasks in the Formalization of Learning Stage

B Levels of automation framework

Table B.1: Generic Levels of Automation

	LoA	Automation description (Interpretation/Evalution/Selection/Creation)
High	10	The computer (...) everything, acts autonomously, ignoring the human
	9	Informs the human only if it, the computer, decides to
	8	Informs the human only if asked, or
	7	(...) automatically, then necessarily informs the human, and
	6	Allows the human a restricted time to veto before automatic execution, or
	5	Executes that suggestion if the human approves, or
	4	Suggests on (...)
	3	Narrows the selection down to a few, or
	2	The computer offers a complete set of (...) alternatives, or
Low	1	The computer offers no assistance: human must make all (...) for the computer

C Solution structure

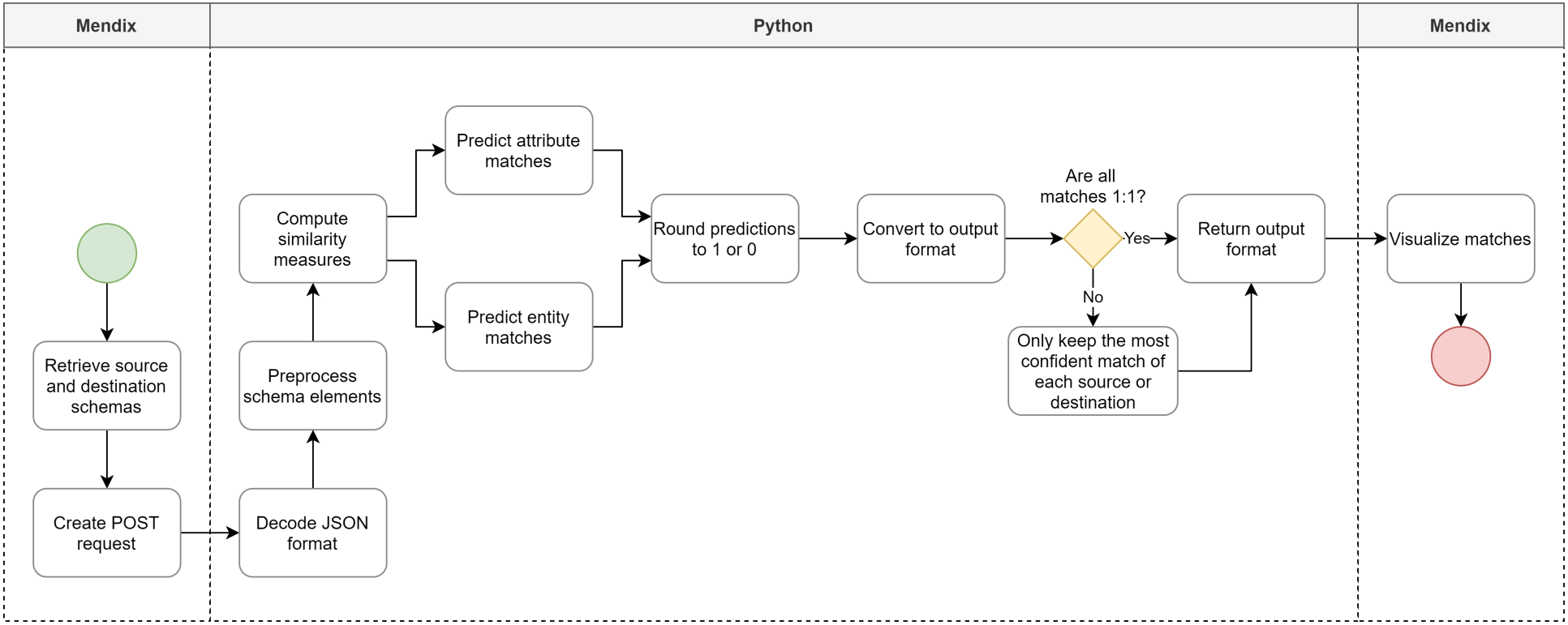


Figure C.1: Process flow of prototype in production

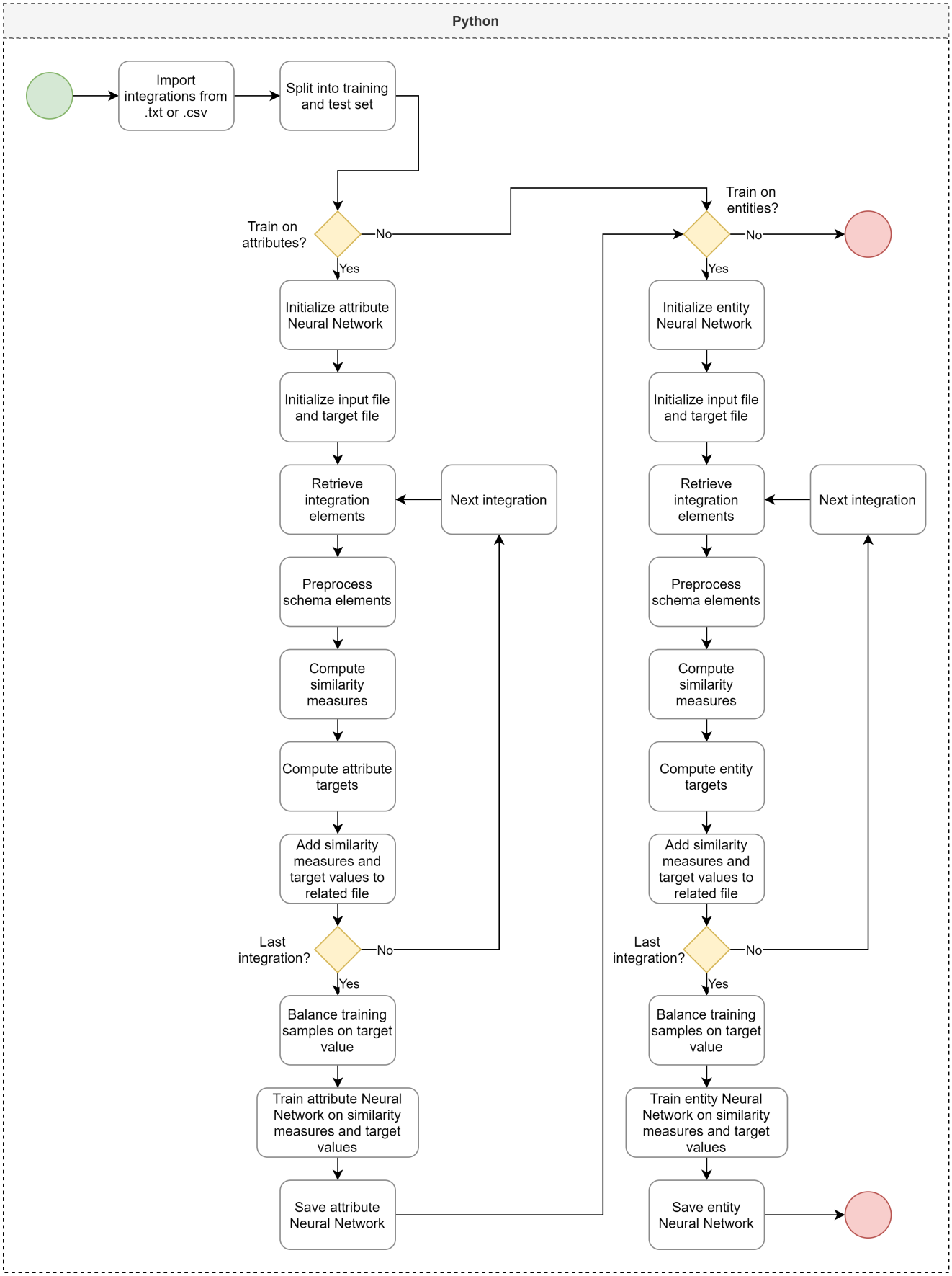


Figure C.2: Process flow of training prototype

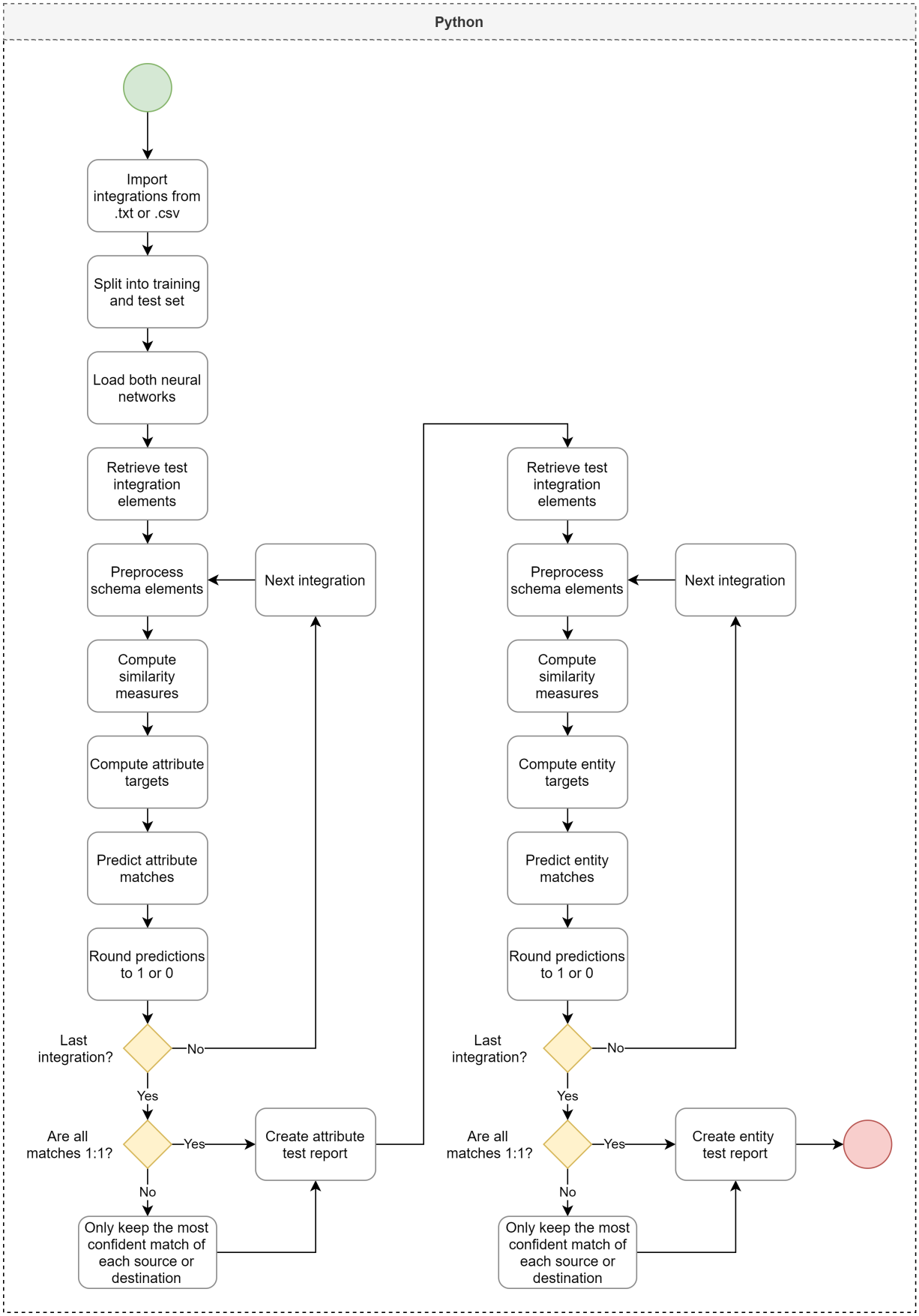


Figure C.3: Process flow of testing prototype

D Database retrieval

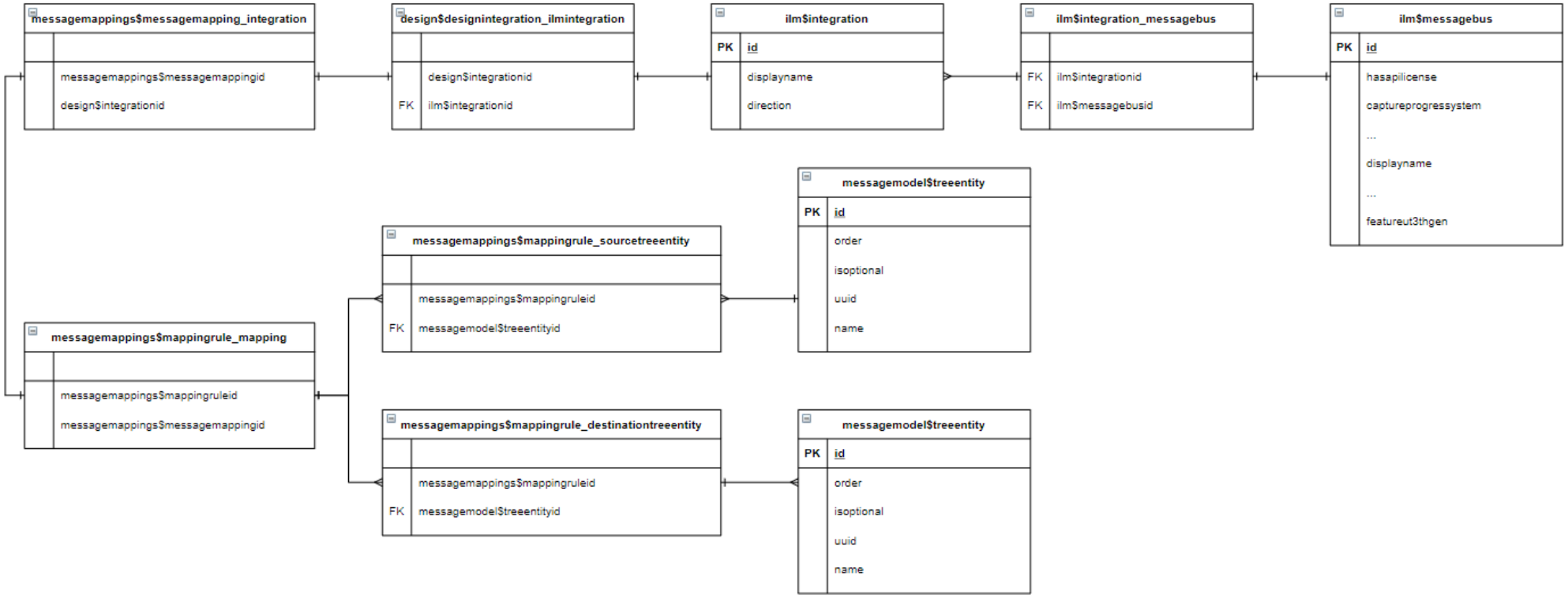


Figure D.1: Retrieval of entity training and test data from database

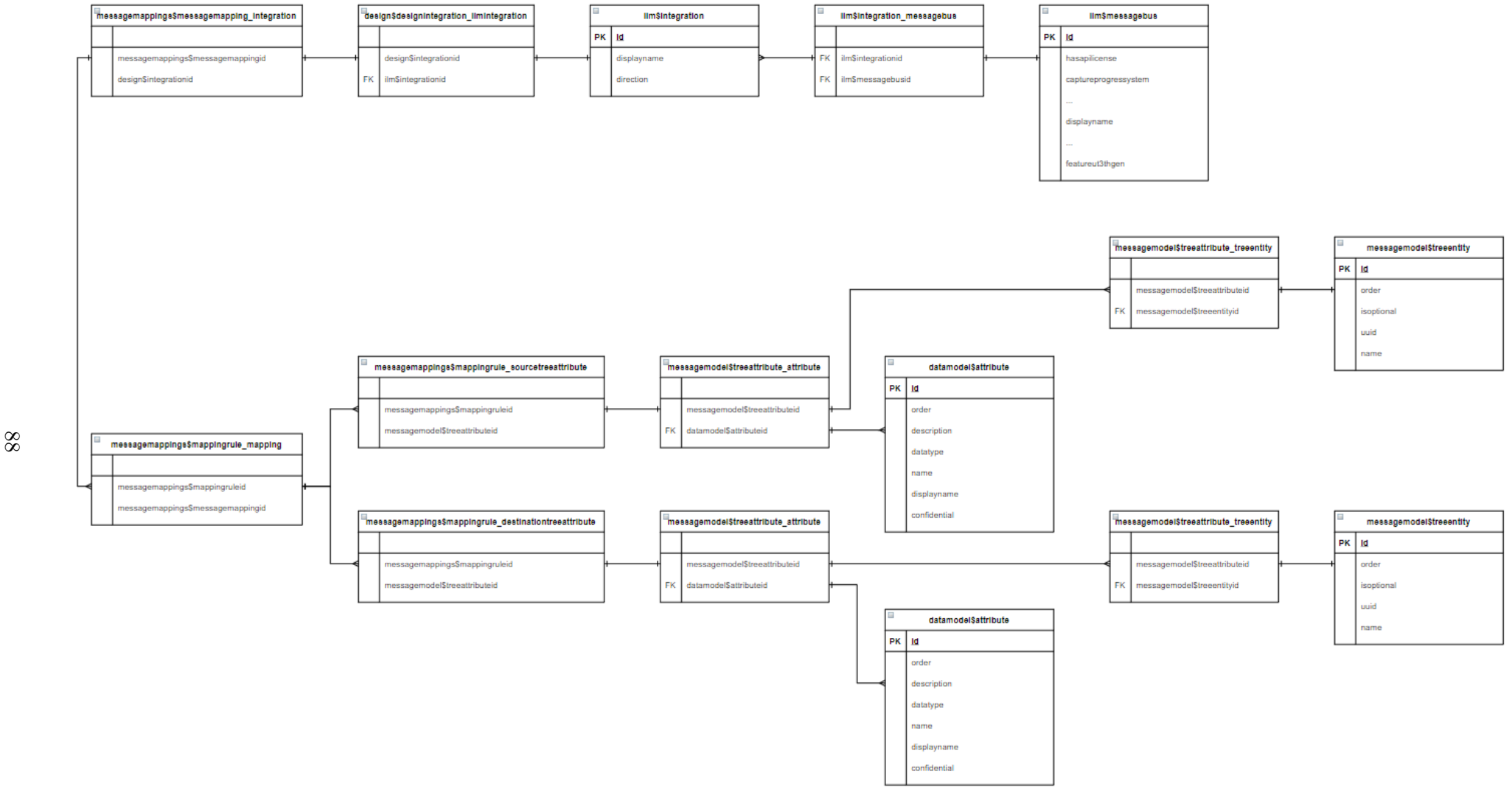


Figure D.2: Retrieval of attribute training and test data from database

E Results Parameter Configuration

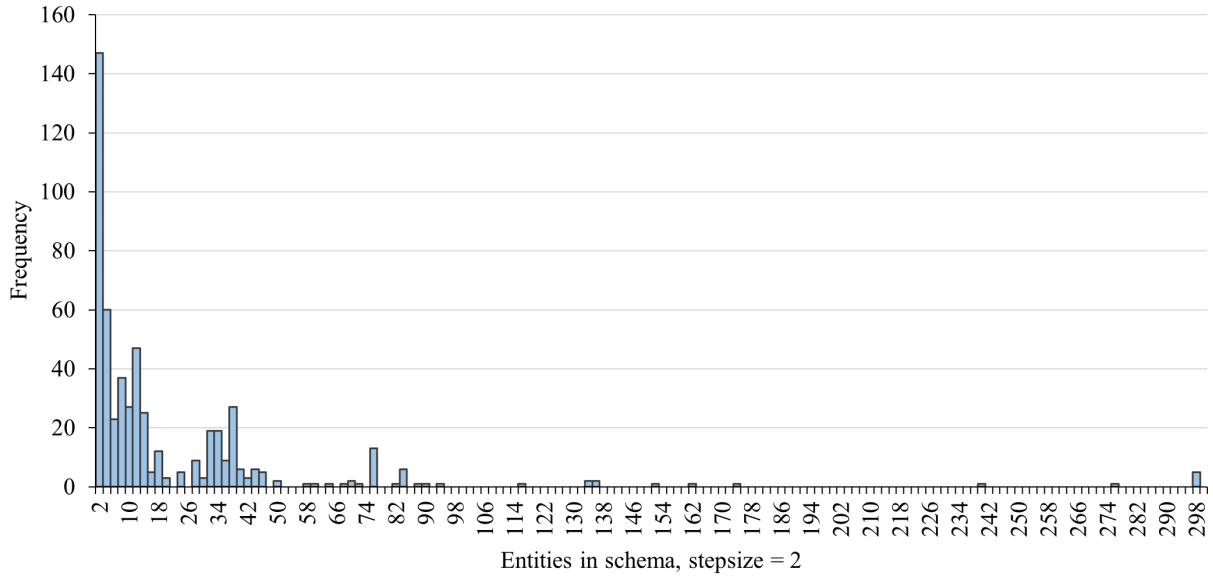


Figure E.1: Frequency of schema lengths based on entities

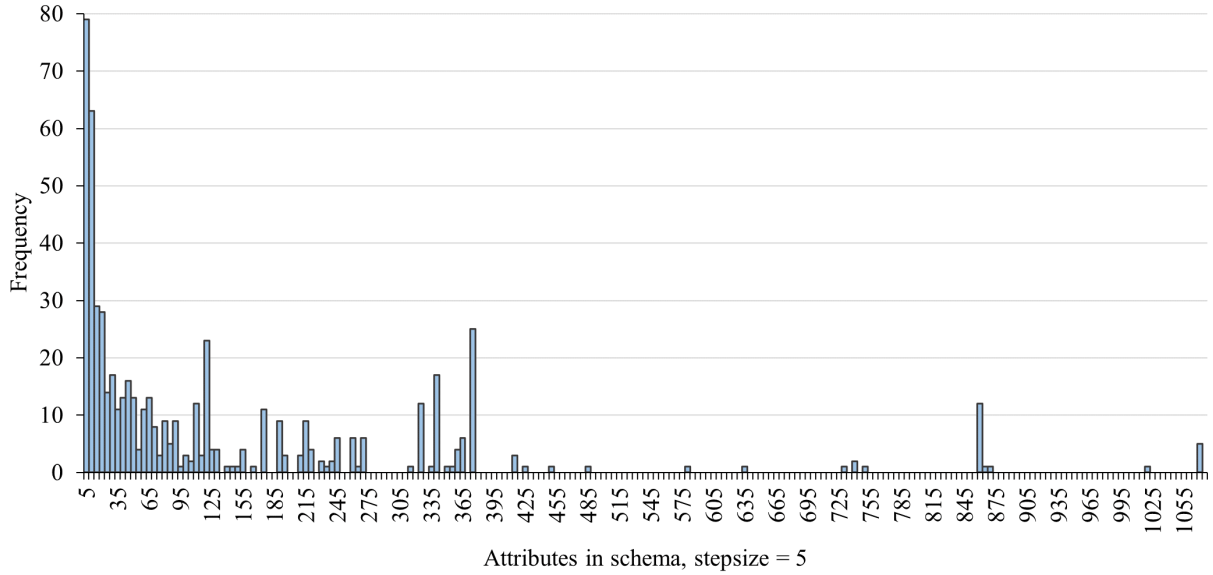


Figure E.2: Frequency of schema lengths based on attributes

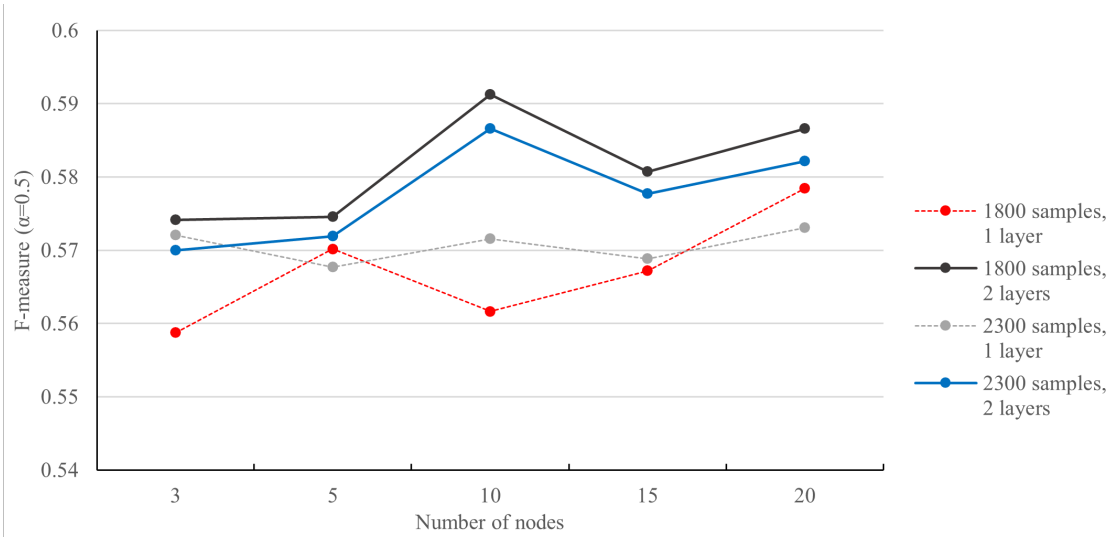


Figure E.3: Comparison of average f-measure of 1800 and 2300 samples. Confidence threshold = 0.8

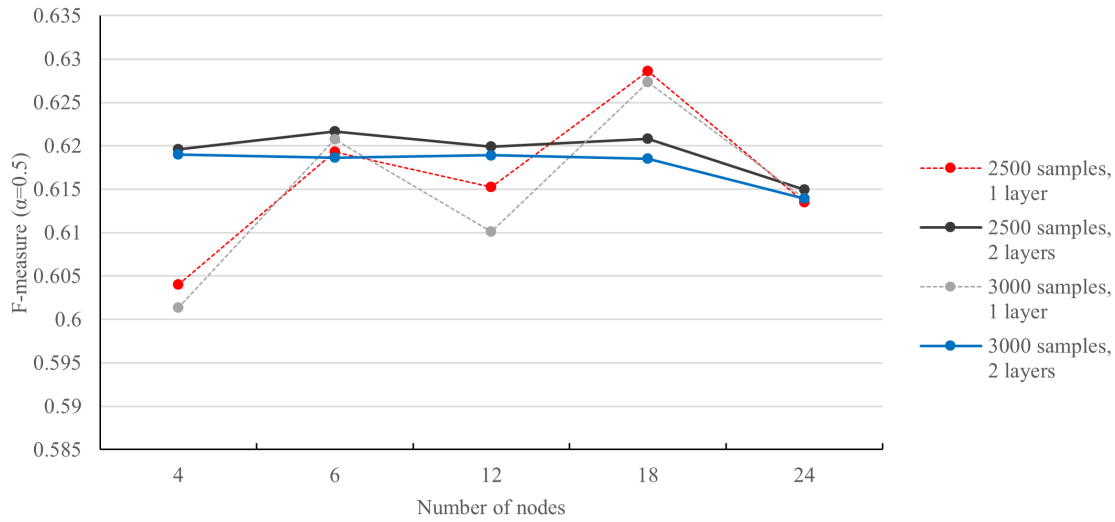


Figure E.4: Comparison of average f-measure of 2500 and 3000 samples. Confidence threshold = 0.8

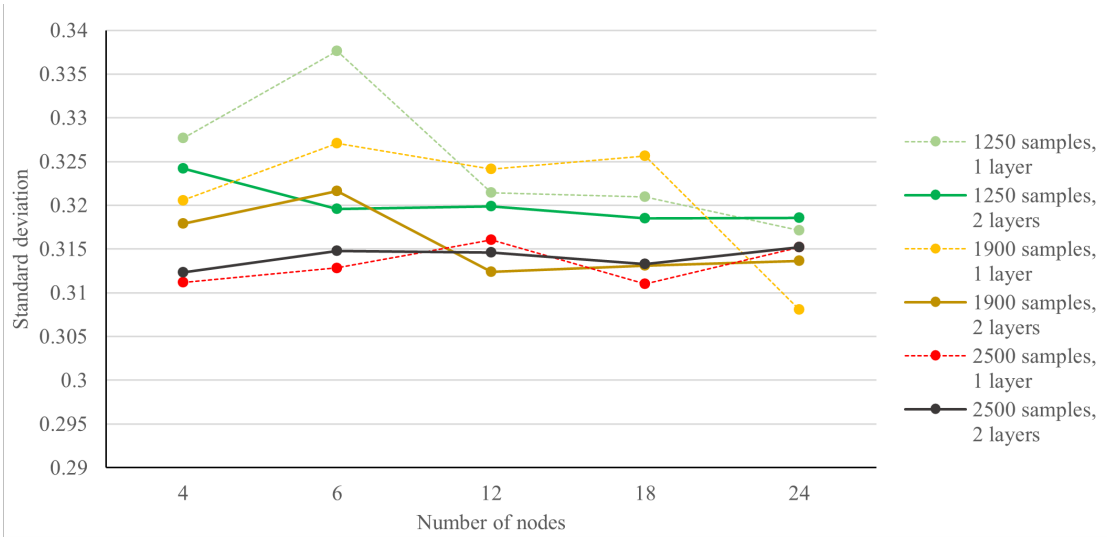


Figure E.5: Comparison of average standard deviation all parameter setting combinations. Confidence threshold = 0.8

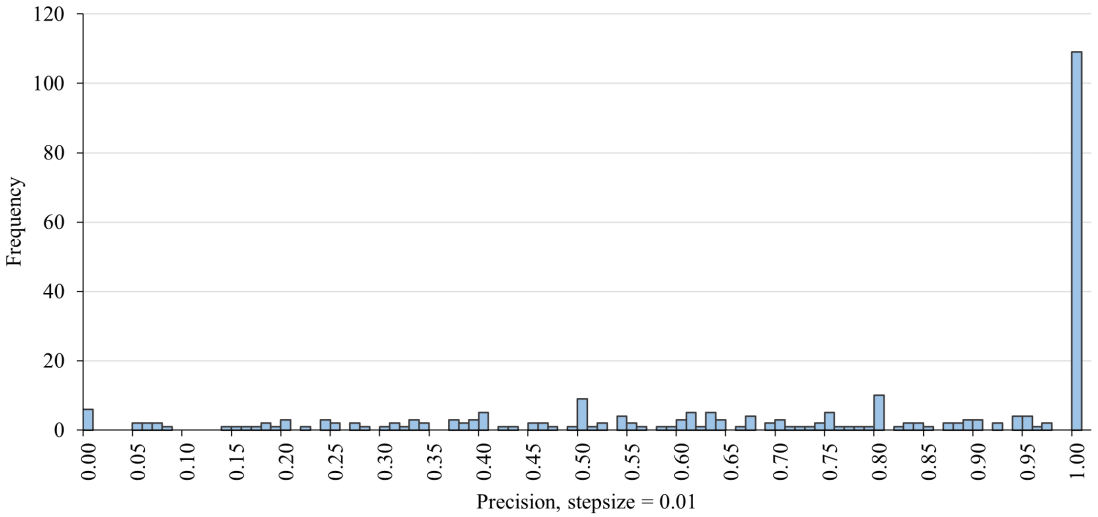


Figure E.6: DNN entities: Frequency of average precision on all integrations

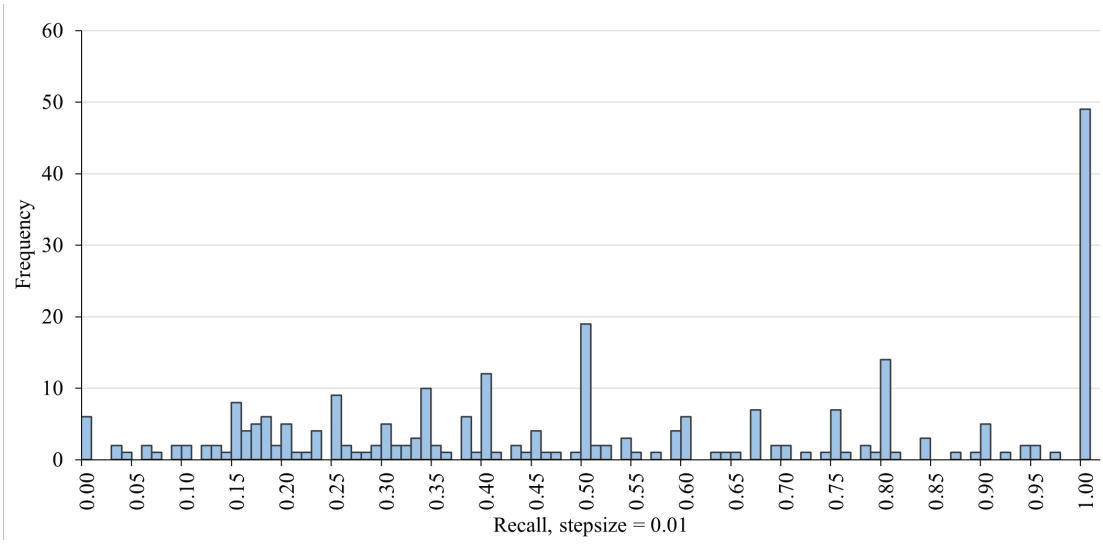


Figure E.7: DNN entities: Frequency of average recall on all integrations

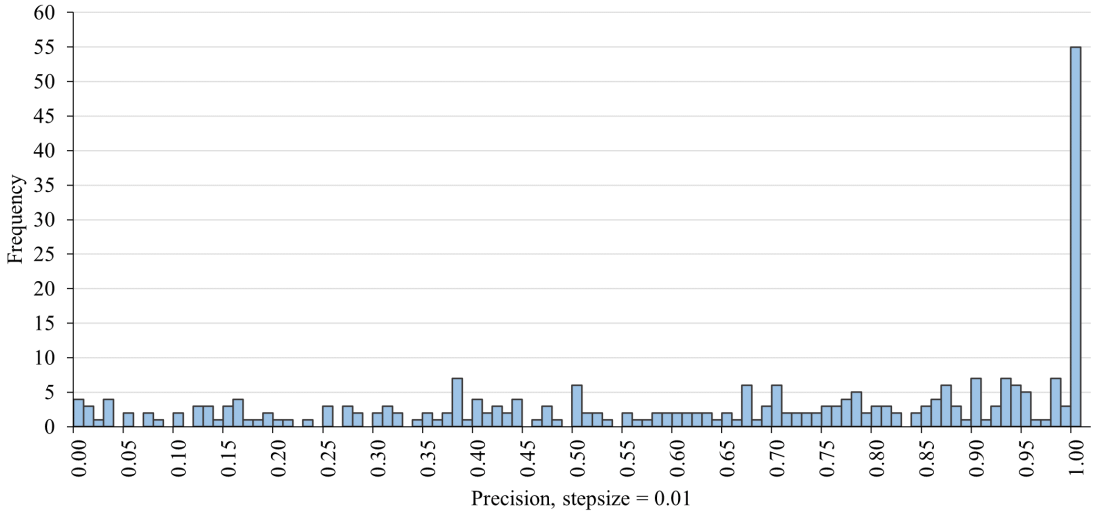


Figure E.8: DNN attributes: Frequency of average precision on all integrations

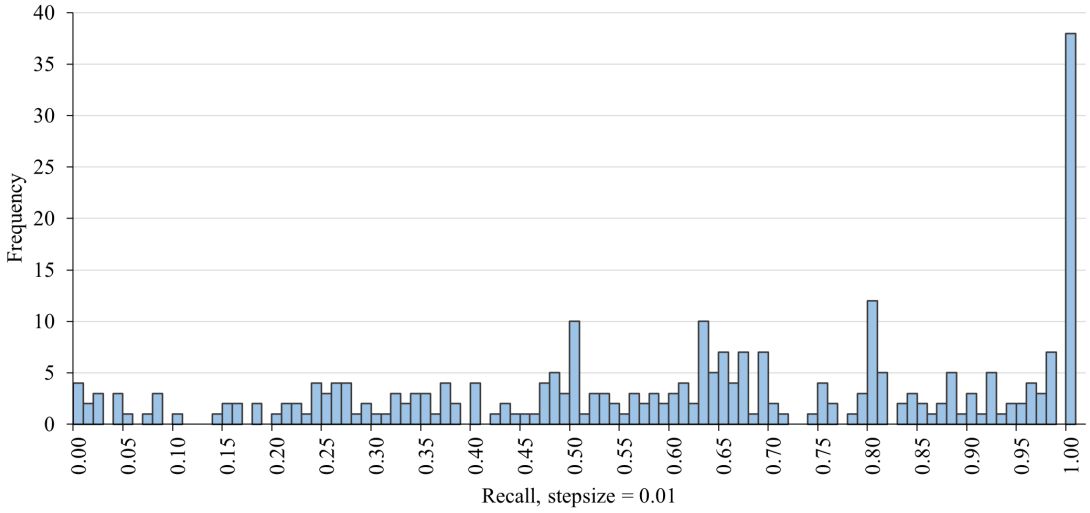


Figure E.9: DNN attributes: Frequency of average recall on all integrations

Table E.1: Frequency data of confidence scores, 1 layer, number of samples = 2500

Value	**Frequency**
<0.51	819150
0.52	0
0.53	5
0.54	5
0.55	10
0.56	15
0.57	0
0.58	10
0.59	0
0.6	5
0.61	10
0.62	10
0.63	5
0.64	10
0.65	10
0.66	15
0.67	0
0.68	5
0.69	0
0.7	10
0.71	0
0.72	10
0.73	0
0.74	5
0.75	0
0.76	0
0.77	0
0.78	0
0.79	60
0.8	60
0.81	60
0.82	85
0.83	75
0.84	40
0.85	90
0.86	70
0.87	75
0.88	215
0.89	110
0.9	270
0.91	230
0.92	285
0.93	370
0.94	265
0.95	295
0.96	825
0.97	1895
0.98	2055
0.99	5720
1	13759
Sum	**846199**

F Results linear regression

```
Call:
lm(formula = Precision ~ Source, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7923 -0.1869  0.1016  0.2109  0.6395

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.7953564  0.0196583   40.46  < 2e-16 ***
Source      -0.0031096  0.0004411   -7.05  1.5e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2814 on 270 degrees of freedom
Multiple R-squared:  0.1555,    Adjusted R-squared:  0.1523
F-statistic:  49.7 on 1 and 270 DF,  p-value: 1.497e-11
```

(a) Simple linear regression entity data, source on precision

```
Call:
lm(formula = Precision ~ Destination, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7822 -0.2091  0.1170  0.2206  0.8509

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.784977   0.019560  40.132  < 2e-16 ***
Destination -0.002805   0.000438  -6.404  6.7e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2853 on 270 degrees of freedom
Multiple R-squared:  0.1318,    Adjusted R-squared:  0.1286
F-statistic: 41.01 on 1 and 270 DF,  p-value: 6.702e-10
```

(b) Simple linear regression entity data, destination on precision

```
Call:
lm(formula = Precision ~ Ratio, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.76112 -0.22478  0.09803  0.27733  0.27843

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.720474   0.020234  35.607   <2e-16 ***
Ratio       0.001098   0.001473   0.746    0.456
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3059 on 270 degrees of freedom
Multiple R-squared:  0.002057,  Adjusted R-squared:  -0.00164
F-statistic: 0.5564 on 1 and 270 DF,  p-value: 0.4564
```

(c) Simple linear regression entity data, ratio on precision

```
Call:
lm(formula = Precision ~ Cardinality, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.73459 -0.22016  0.09126  0.26541  0.34475

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.742145   0.023484  31.602   <2e-16 ***
Cardinality -0.007556   0.006975  -1.083     0.28
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3055 on 270 degrees of freedom
Multiple R-squared:  0.004327,  Adjusted R-squared:  0.0006397
F-statistic: 1.173 on 1 and 270 DF,  p-value: 0.2797
```

(d) Simple linear regression entity data, cardinality on precision

```
Call:
lm(formula = Precision ~ Loss, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.78514 -0.17771  0.09773  0.17600  0.72131

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.862856   0.021107  40.881   <2e-16 ***
Loss        -0.038859   0.003948  -9.844   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2627 on 270 degrees of freedom
Multiple R-squared:  0.2641,    Adjusted R-squared:  0.2614
F-statistic:  96.9 on 1 and 270 DF,  p-value: < 2.2e-16
```

(e) Simple linear regression entity data, loss on precision

```
Call:
lm(formula = Precision ~ Language, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8152 -0.1867  0.1278  0.1848  0.3946

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.81522    0.02298  35.471  < 2e-16 ***
Language    -0.20981    0.03535  -5.936 8.94e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.288 on 270 degrees of freedom
Multiple R-squared:  0.1154,    Adjusted R-squared:  0.1122
F-statistic: 35.24 on 1 and 270 DF,  p-value: 8.943e-09
```

(f) Simple linear regression entity data, language on precision

Figure F.1: DNN entity: Simple linear regression of precision on all significant predictor variables

```
Call:
lm(formula = Precision ~ Source, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.71576 -0.20667  0.06056  0.23145  0.80121

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.720e-01  1.969e-02   39.20   <2e-16 ***
Source      -8.559e-04  8.201e-05  -10.44   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2696 on 282 degrees of freedom
Multiple R-squared:  0.2786,    Adjusted R-squared:  0.276
F-statistic: 108.9 on 1 and 282 DF,  p-value: < 2.2e-16
```

(a) Simple linear regression attribute data, source on precision

```
Call:
lm(formula = Precision ~ Destination, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.75626 -0.21947  0.05953  0.24529  0.56582

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.7577993  0.0195251  38.811   <2e-16 ***
Destination -0.0007717  0.0000788  -9.793   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2742 on 282 degrees of freedom
Multiple R-squared:  0.2538,    Adjusted R-squared:  0.2511
F-statistic:  95.9 on 1 and 282 DF,  p-value: < 2.2e-16
```

(b) Simple linear regression attribute data, destination on precision

```
Call:
lm(formula = Precision ~ Language, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.78438 -0.18159  0.09005  0.21562  0.46447

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.78438    0.02531  30.991  < 2e-16 ***
Language    -0.24885    0.03471  -7.169 6.63e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2919 on 282 degrees of freedom
Multiple R-squared:  0.1542,    Adjusted R-squared:  0.1512
F-statistic:  51.4 on 1 and 282 DF,  p-value: 6.626e-12
```

(c) Simple linear regression attribute data, language on precision

Figure F.2: DNN attribute: Simple linear regression of precision on all significant predictor variables

```
Call:
lm(formula = Precision ~ Ratio, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.76112 -0.22478  0.09803  0.27733  0.27843

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.720474   0.020234  35.607   <2e-16 ***
Ratio       0.001098   0.001473   0.746    0.456
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3059 on 270 degrees of freedom
Multiple R-squared:  0.002057,  Adjusted R-squared:  -0.00164
F-statistic: 0.5564 on 1 and 270 DF,  p-value: 0.4564
```

(a) Simple linear regression entity data, ratio on recall

```
Call:
lm(formula = Precision ~ Cardinality, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.73459 -0.22016  0.09126  0.26541  0.34475

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.742145   0.023484  31.602   <2e-16 ***
Cardinality -0.007556   0.006975  -1.083     0.28
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3055 on 270 degrees of freedom
Multiple R-squared:  0.004327,  Adjusted R-squared:  0.0006397
F-statistic: 1.173 on 1 and 270 DF,  p-value: 0.2797
```

(b) Simple linear regression entity data, cardinality on recall

```
Call:
lm(formula = Precision ~ Loss, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.78514 -0.17771  0.09773  0.17600  0.72131

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.862856   0.021107  40.881   <2e-16 ***
Loss        -0.038859   0.003948  -9.844   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2627 on 270 degrees of freedom
Multiple R-squared:  0.2641,    Adjusted R-squared:  0.2614
F-statistic:  96.9 on 1 and 270 DF,  p-value: < 2.2e-16
```

(c) Simple linear regression entity data, loss on recall

```
Call:
lm(formula = Precision ~ Language, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8152 -0.1867  0.1278  0.1848  0.3946

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.81522    0.02298  35.471  < 2e-16 ***
Language    -0.20981    0.03535  -5.936 8.94e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.288 on 270 degrees of freedom
Multiple R-squared:  0.1154,    Adjusted R-squared:  0.1122
F-statistic: 35.24 on 1 and 270 DF,  p-value: 8.943e-09
```

(d) Simple linear regression entity data, language on recall

Figure F.3: DNN entity: Simple linear regression of recall on all significant predictor variables

```
Call:
lm(formula = Recall ~ Source, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.68364 -0.19394  0.03318  0.22717  0.41827

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.110e-01  1.881e-02  37.804   <2e-16 ***
Source      -6.874e-04  7.831e-05  -8.777   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2574 on 282 degrees of freedom
Multiple R-squared:  0.2146,    Adjusted R-squared:  0.2118
F-statistic: 77.04 on 1 and 282 DF,  p-value: < 2.2e-16
```

(a) Simple linear regression attribute data, source on recall

```
Call:
lm(formula = Recall ~ Destination, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.70701 -0.18394  0.02452  0.21808  0.38580

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.084e-01  1.805e-02  39.235   <2e-16 ***
Destination -6.840e-04  7.286e-05  -9.388   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2535 on 282 degrees of freedom
Multiple R-squared:  0.2381,    Adjusted R-squared:  0.2354
F-statistic: 88.13 on 1 and 282 DF,  p-value: < 2.2e-16
```

(b) Simple linear regression attribute data, destination on recall

```
Call:
lm(formula = Recall ~ Ratio, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.62822 -0.19188  0.02325  0.25880  0.46396

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.635558   0.019746  32.187   <2e-16 ***
Ratio       -0.002509   0.001185  -2.117   0.0351 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2882 on 282 degrees of freedom
Multiple R-squared:  0.01565,   Adjusted R-squared:  0.01216
F-statistic: 4.483 on 1 and 282 DF,  p-value: 0.03512
```

(c) Simple linear regression attribute data, ratio on recall

```
Call:
lm(formula = Recall ~ Cardinality, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.70457 -0.10552  0.02675  0.18065  0.57846

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.02987    0.04075   25.28   <2e-16 ***
Cardinality -0.32530    0.02984  -10.90   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2436 on 282 degrees of freedom
Multiple R-squared:  0.2964,    Adjusted R-squared:  0.2939
F-statistic: 118.8 on 1 and 282 DF,  p-value: < 2.2e-16
```

(d) Simple linear regression attribute data, cardinality on recall

```
Call:
lm(formula = Recall ~ Loss, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.67832 -0.17447  0.00304  0.21611  0.71343

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.691633   0.017211  40.186   <2e-16 ***
Loss        -0.011414   0.001229  -9.285   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2542 on 282 degrees of freedom
Multiple R-squared:  0.2341,    Adjusted R-squared:  0.2314
F-statistic: 86.22 on 1 and 282 DF,  p-value: < 2.2e-16
```

(e) Simple linear regression attribute data, loss on recall

```
Call:
lm(formula = Recall ~ Abbreviations, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.62382 -0.20424  0.01684  0.25118  0.46914

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.62382    0.01807  34.524   <2e-16 ***
Abbreviations -0.09296    0.05755  -1.615    0.107
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2891 on 282 degrees of freedom
Multiple R-squared:  0.009169,  Adjusted R-squared:  0.005655
F-statistic:  2.61 on 1 and 282 DF,  p-value: 0.1073
```

(f) Simple linear regression attribute data, abbreviations on recall

Figure F.4: DNN attribute: Simple linear regression of recall on all significant predictor variables