2017

Applying Intelligence Amplification to the problem of Schema Matching

MASTER THESIS BUSINESS AND IT

JOHAN BUIS (S0166308)

Graduation committee:

University of Twente:
M.E. Iacob
M.J. van Sinderen
A. Dobrkovic

CAPE Groep:
L.O. Meertens

Acknowledgement

This thesis is the final product of my student career. It was an interesting thesis to write with an even more interesting problem at its centre. This meant learning a lot of new concepts which made it a challenging task. However, putting all my thoughts on paper proved to be the toughest part. It was at those times the Instant Gratification Monkey took control of the steering wheel. For those who never heard of this friendly monkey I recommend looking him up on YouTube. Seriously, watch that TED talk.

I would like to thank Maria, Marten and Andrej for their great support throughout the process. You've always provided helpful feedback which helped me to further write this thesis. Lucas, thanks for your daily support helping me to point me to the right directions.

I would also like to thank all my colleagues at eMagiz and CAPE Groep. And thanks Rob for giving me the opportunity to write my thesis at CAPE. I am looking forward starting my job at CAPE after my upcoming holiday.

Finally, I want to thank my family friends and everyone who helped me through this. It wasn't always easy but I'm glad I can finish my studies.

Johan Buis
Enschede, August 2017

Abstract

A task often occurring at CAPE Groep is the task of schema matching. Schema matching is the problem of finding pairs of attributes (or groups of attributes) from a source schema and attributes of a target schema such that pairs are likely to be related. At present, this time-consuming task is done manually. This thesis explores the possibilities for partially automating this process thus saving time and, eventually, money.

Fully automating the task of schema matching has proved to be difficult. We therefore apply the concept of Intelligence Amplification to the problem of schema matching. Intelligence Amplification is a field which focuses on a symbiotic relationship between human and machine. A clear definition is currently lacking in literature and after assessing extracting key features we created our own definition: *"Intelligence Amplification focusses on a close collaboration, with complementary contributions, between human and machine to empower humans in the decision-making process".*

For the problem of schema matching we found two major moments where interaction between humans and machine occur: during the stage of pre-processing and during the matching stage. Pre-processing happens at the begin of a matching scenario. Steps included in pre-processing include expanding abbreviations or translation of attribute names. In the matching stage, a machine calculates a set of candidate mappings. In our IA driven approach, the user can opt to invoke several software agents, either get better results or to have a different software agent for a subset of the matching scenario.

A reference architecture was developed to aid in development of such tools. Using this reference architecture, we developed our own prototype. This prototype contained a machine learning approach. We trained a neural network to predict candidate mappings. Evaluation of this method has showed there is still room for improvement as for some scenarios the neural network was not able to generate any candidate mappings.

Evaluation of the prototype was done using two metrics: effectiveness and efficiency. For effectiveness we look at precision and recall. Precision is a metric for the quality of results. It indicates the percentage of correct predictions that were made by the machine as part of the total amount of predictions made. Recall tells something about the completeness of results. It indicates how many correct predictions were made as part of the total amount of correct predictions which should be made.

The second evaluation criteria, efficiency, is looking at the time aspect. First a baseline is established. In our case this is the time it takes a user to manually complete a matching scenario. When using an automated approach, we again look at the total time it takes to complete a scenario and compare this against the baseline. Using this feature a performance improvement score is calculated.

It was found the prototype needs several improvements. We tried an approach using a trained neural network and one with a heuristic to create candidate mappings. We have not found a single approach which works best for every situation. For CAPE Groep we recommend the most important next step is to improve the user interface so it is better able to handle the input of an auto-mapping application. Sliders for the various metrics should be included. This allows a user to directly see the effect of any change they make and tweak the settings such that it fits the scenario they work on. This should then be extended by further pre-processing steps to research what the benefit is of certain pre-processing actions.

Contents

1 Introduction

This section starts with the context in section 1.1 followed by the problem statement in section 1.2. Next, the relevance of the project is discussed in section 1.3. This is followed by a research model which is discussed in section 1.4 and the research questions which are addressed in section 1.5. Section 1.6 provides the methodology and section 1.7 gives an outline of the rest of the thesis.

1.1 Context

The early days of automation started with the objective of looking which tasks could best be replaced by computers because the computer could do it better or against lower cost (Casini, Depree, Suri, Bradshaw, & Nieten, 2015). One of the research fields seeking to achieve this objective is the field of Artificial Intelligence (AI) (Fischer, 1995). This is now a well-established field and a lot of research effort has been put in it. However, the focus of AI is to replace human reasoning which is not always possible (Garcia, 2010). This stands in contrast to Intelligence Amplification (IA) which does not aim to replace a human but amplify the human intelligence (Breemen, Farkas, & Sarbo, 2011). This is a different view in which the point is not to assess which tasks are better suitable to be undertaken by a human or a computer, but to see how tasks can best be shared by both humans and computers (Casini et al., 2015). This is the concept of symbiosis and the roots of it can be traced back to Licklider (1960). In his paper, he discusses the symbiosis between human and machine. As example of a symbiosis Licklider (1960) cites the fig tree which is pollinated by a larva. This larva lives in the ovary of the tree on which it eats. Thus, the two form a productive and thriving partnership.

Even though the concept is more than 50 years old it hasn't been entirely realized today (Cerf, 2013). However, this could be contributed to recent advances in in computer technology and psychological theory which make the subject possible (Griffith & Greitzer, 2007). In this thesis, we take the concept of Intelligence Amplification and apply it to the problem of schema matching which is introduced in the next section.

1.2 Problem statement

Schema matching is a basic problem in many database applications, such as data integration, data warehousing and semantic query processing (Rahm & Bernstein, 2001). It aims at identifying semantic correspondences between metadata structure or models, such as database schemas or XML message formats (Rahm, 2011). Often this process is carried out manually costing a lot of time and user effort (Rahm & Bernstein, 2001). It is an inherently difficult task to automate because the exact semantics of the data are only completely understood by the designers of the schema, and not fully captured by the schema itself (Madhavan, Bernstein, Doan, & Halevy, 2005).

CAPE Groep offers eMagiz to integrate various software applications. eMagiz makes this process easy and intuitive by offering a graphical user interface which makes the product suitable to be used without extensive programming knowledge. Each implementation of eMagiz starts with the design of the internal data model which is referred to as a Canonical Data Model (CDM). Applications can be connected to the CDM thus creating a hub-and-spoke architecture (Weske, 2012).

The process of connecting a new application to the CDM involves creating a schema mapping between the incoming (or outgoing) system and the CDM. At present this is done manually and therefore is very time-consuming (Duchateau & Bellahsene, 2016). An example of such a mapping can be seen in figure 1.

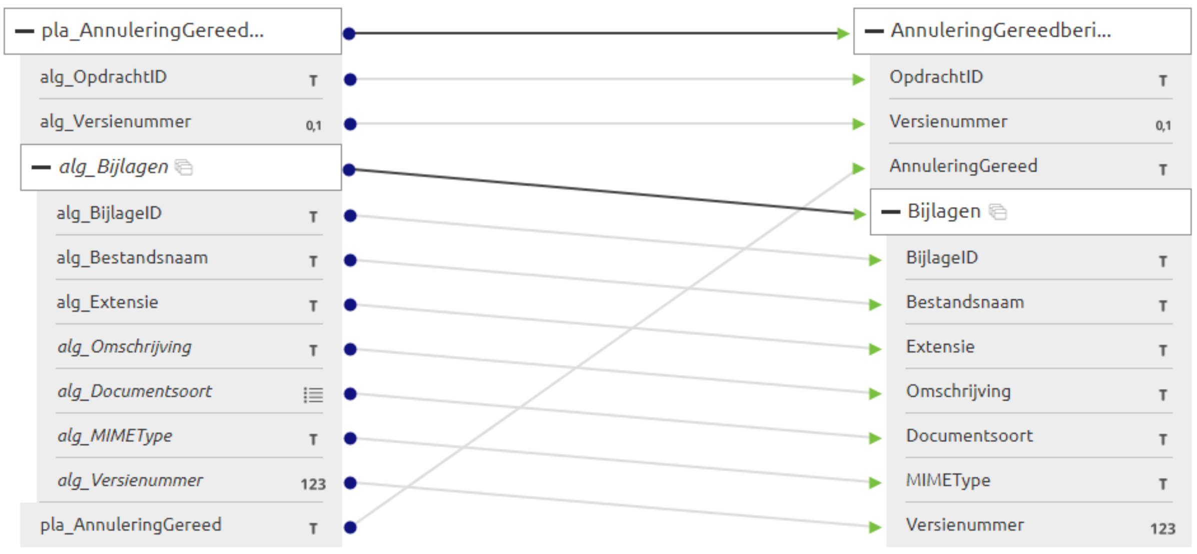


Figure 1: Example mapping in eMagiz

Each line represents a mapping between the two schemas. On the left we find the source schema which needs to be mapped to the destination schema on the right. To enhance this process a schema matcher would be useful. Research into schema matching mostly focusses on systems which are capable of proposing possible matching without human involvement (Rahm & Bernstein, 2001). Many approaches make wrong choices which could have a cascade effect and lead to further errors (Jimenez-Ruiz, Grau, Zhou, & Horrocks, 2012). When accuracy is important user intervention during the matching process becomes essential and earlier research indicate this intervention could significantly improve matching result (Jimenez-Ruiz et al., 2012) This symbiosis between tool and user has been gaining more prominence but research in this area is still in its infancy (Falconer & Noy, 2011; Rahm, 2011; Rodrigues, da Silva, Rodrigues, & dos Santos, 2015).

In this thesis, we aim to create a symbiotic method between tool and human for schema matching. This approach is uncommon in science and gives a CAPE Groep a possibility to reduce time consultants need to perform the task of schema matching. This main research goal is to develop a reference architecture for using the concepts of Intelligence Amplification and database schema matching which should aid in the task of matching.

1.3 Relevance

The project has both scientific and practical relevance. First, we will look in literature and assess the current state-of-art in Intelligence Amplification research. This will be combined with knowledge from schema matching to deliver a framework for an IA driven approach for schema matching.

Practical relevance is for CAPE Groep for which we develop a functional prototype which should partially automate the schema matching process thus leading to a time reduction needed for consultants performing the task.

1.4 Research model

Based on the problem statement we formulate a research model (Verschuren & Doorewaard, 2007) shown in figure 2.

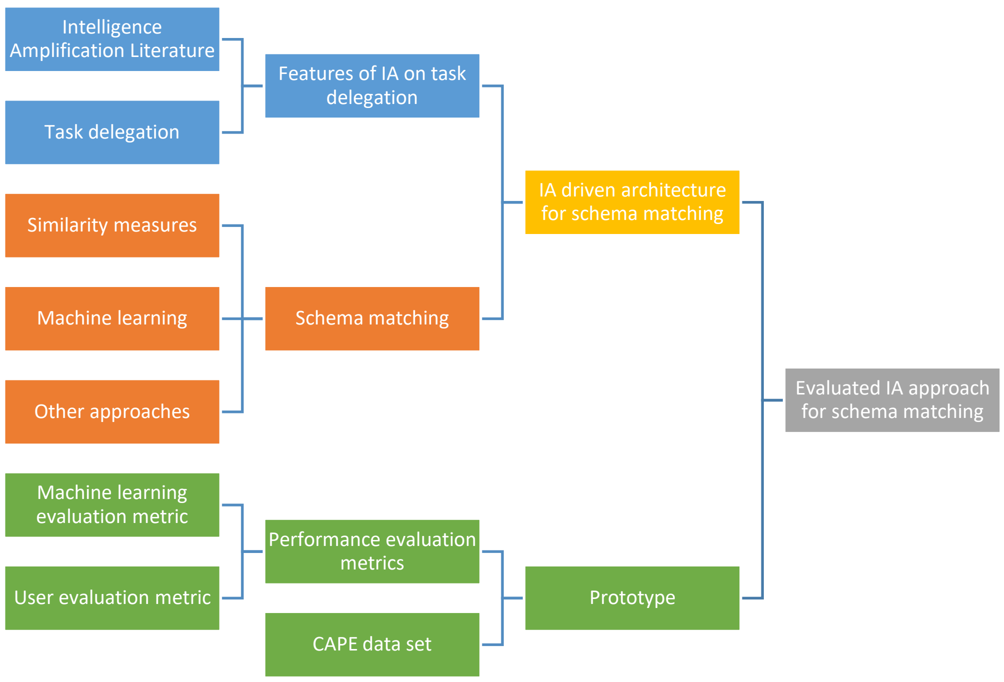

Intelligence Amplification Literature
Task delegation
Features of IA on task delegation
Similarity measures
Machine learning
Other approaches
Schema matching
IA driven architecture for schema matching
Evaluated IA approach for schema matching
Machine learning evaluation metric
User evaluation metric
Performance evaluation metrics
CAPE data set
Prototype

Figure 2: Research model

The main goal of the thesis is shown on the right. This is to have an evaluated IA approach for schema matching. To achieve this a IA driven approach to schema matching is created and a prototype.

The IA driven approach is split in two tasks, namely features of IA on task delegation. The latter is split in literature concerning Intelligence Amplification and task delegation. Next, a closer look is taken at schema matching. We are curiosity driven to look for a solution involving machine learning. Also, a look is taken at different approaches and similarity measures. These are measures indicating how similar two text string are which are further discusses in section 2.

To create a prototype, we used the IA driven approach defined above. We use the data set at CAPE for machine learning. To evaluate our approach, we look for performance metrics, these are split in metrics for machine learning and user evaluation.

Colour coding is used to link the tasks to the research questions defined in the next section.

1.5 Research Questions

Based on the problem statement and research model we formulate the following main research question:

How can we combine the concept of Intelligence Amplification with database schema matching to create a reference architecture for IA driven schema matching?

To answer this question several sub-questions are formulated. First, we focus on Intelligence Amplification. We start with a literature review to assess the current- state-of-art. At present, there is no universally accepted definition. Therefore, a literature study is conducted to find various definitions and highlight the difference.

1a. Which definitions exist in literature?
1b. What is the current state-of-art in Intelligence Amplification research?
1c. How does IA affect the delegation of tasks between human and machine?

Our next step is related to the task for which we will try to build a solution, namely database schema matching. We would like to use machine learning for this purpose so therefore we provide a general introduction. Before we can use the data for machine learning we look which possibilities have been identified in literature for performing schema matching using machine learning. These questions deal with the data currently present and how to prepare it for extraction by a machine learning algorithm.

2a. Which solutions for schema matching have been proposed in literature?
2b. Which pre-processing steps are needed to prepare data for schema matching?
2c. Which issues has literature identified in database schema matching?
2d. Which machine learnings algorithms are available?

Using the knowledge from the first two research question we describe a general architecture which describes how we can create an IA driven approach for schema matching.

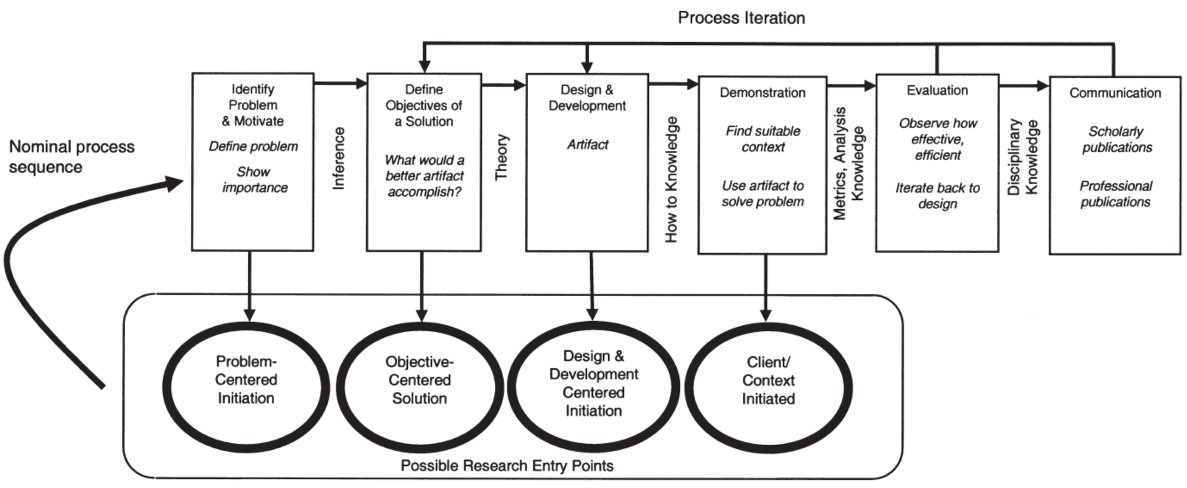
3a. How to design an IA driven approach and architecture for schema matching?
3b. In which stages do we include the user in the approach?

We then use our general approach to develop a prototype. The prototype is implemented and evaluated.

4a. Which parts of the architecture will we use to build a first concrete architecture?
4b. Which data is available at CAPE?
4c. Which machine learning algorithm produces the best result?
4d. Which metrics can we use to measure performance?
4e. Which improvements to the initial implementation can be made?

1.6 Methodology

To structure the report we use the Design Science Research Model which is displayed in figure 3 (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2008).



Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 3: DSRM process model (Peffers et al., 2008)

The model consists of the following steps:

- Problem identification and motivation: this step defines the research problem and is used to justify the value of a solution. This is covered in this chapter.

- Define the objectives and solution: from the problem identification and motivation the objectives for a solution are inferred. This looks at what is possible and feasible. This is covered in research question 1 and 2.
- Design and develop: in this stage, an artefact is developed. This start by determining its desired functionalities and architecture. Initially we derive a general approach discussed in question 3. Next out prototype is created in question 4a-c.
- Demonstration and evaluation: a demonstration shows the artefact in a single act to prove it works. A more formal method is an evaluation. This is covered in research question 4d-e.
- Communication: the last step consists of communicating the outcome to relevant stakeholders. In this case, this would be the final report and presentation which are covered with all research questions.

1.7 Document outline

The rest of the document is outlines as follows. First, a literature study is conducted which is presented in section 2. Based on the literature a reference architecture for an intelligence amplification driven method is derived in section 3. Based on this reference architecture we build a prototype which we discuss in section 4. The prototype is then evaluated which is discussed in section 5. We conclude this thesis with a conclusion and future work in section 6.

2 Literature

This chapter describes the literature search. First, we discuss literature around the topic of intelligence amplification in section 2.1. Next is an overall introduction to the concept of machine learning in section 2.2. This is followed by a literature section about the topic schema matching in section 2.3. The chapter is concluded with a summary in section 2.4.

2.1 Intelligence Amplification

The topic of the first part of the literature search is Intelligence Amplification.

2.1.1 Questions and selection process

We define the following questions:

- Which definitions of IA related exist in literature?
- Is there a suitable task delegation which can be applied in IA?
- Has literature identified requirements or frameworks for human-machine collaboration?

For the search process Scopus is used. Scopus provides many options for searches and has one of the biggest databases on scientific literature. For articles for which the full text was not available a search on Google Scholar was conducted. The search was conducted in February 2017 and the following five keywords were used: "intelligence amplification", "cognitive augment*", intelligence augment* OR "augment* intelligence ", "*man computer symbiosis" OR "*man machine symbiosis" and "*man computer collaboration" OR "*man machine collaboration". The search yielded 370 results which were first narrowed down on title, next on abstract, next on the availability of the full text article and finally the article was read. An overview of the result set can be found in figure 4.

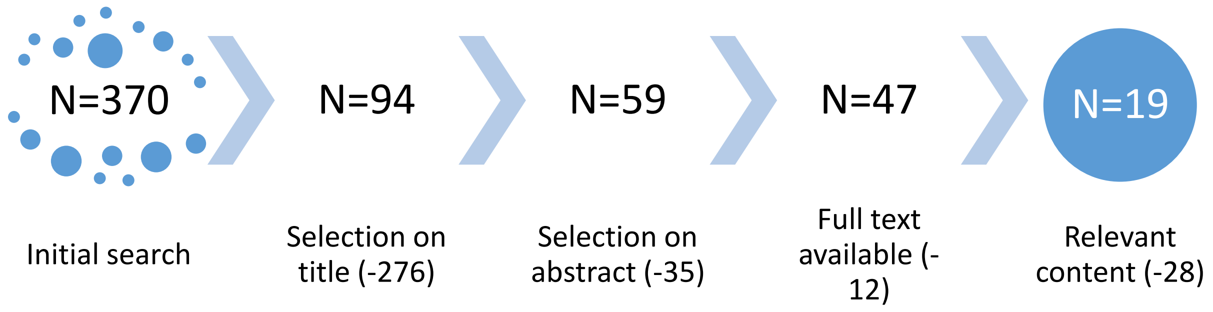

N=370
Initial search
N=94
Selection on title (-276)
N=59
Selection on abstract (-35)
N=47
Full text available (-12)
N=19
Relevant content (-28)

Figure 4: search results

Results are classified based on the questions.

Author	IA definitions	Task delegation	Requirements
(Baker, 2016)			
(Barca & Li, 2006)		●	●
(Breemen et al., 2011)	●		
(Casini et al., 2015)		●	
(Crouser & Chang, 2012)		●	
(Cummings, 2014)		●	
(Dekker & Woods, 2002)		●	
(DiBona, Shilliday, & Barry, 2016)		●	●
(Dobrkovic, Liu, Iacob, & van Hillegersberg, 2016)	●		●

(Fischer, 1995)			●
(Garcia, 2010)	●		
(Greef, Dongen, Grootjen, & Lindenberg, 2007)	●		
(Griffith & Greitzer, 2007)	●		●
(Jacucci, Spagnolli, Freeman, & Gamberini, 2014)	●		
(Khabaza, 2014)	●		
(Kondo, Nishitani, & Nakamura, 2010)			●
(Lesh, Marks, Rich, & Sidner, 2004)			●
(Paraense, Gudwin, & Goncalves, 2007)	●		●
(Stumpf et al., 2009)			●

2.1.2 IA definitions

In the previous chapter we introduced IA as the symbiosis between humans and machines. At present no universal definition exists. We analyse the papers and highlight which definitions is used. These definitions are presented in table 1.

Table 1: Overview of definitions

Author	What	Definition
(Breemen et al., 2011)	IA	IA is a field of research aiming at 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.
(Garcia, 2010)	IA	Artificial intelligence (…) working in partnership with people to reach rationally superior solutions by helping them better explore the solution space.
(Dobrkovic et al., 2016)	IA	Enhance human decision-making abilities through a symbiotic relationship between a human and an intelligent agent.
(Greef et al., 2007)	Augmented Cognition	The symbolic integration of man and machines in a closed-loop system whereby the operator's cognitive state and the operational context are to be detected by the system. In this integration, there is a dynamic division of labour between human and machine which can be reallocated in real-time in order to optimize performance.
(Griffith & Greitzer, 2007)	Human information interaction	A new vision of symbiosis – one that embraces the concept of mutually supportive systems, but with the human in a leadership position, and that exploits the advances in computational technology and the field of human factors/cognitive engineering to yield a level of human-machine collaboration and communication that was envisioned by Licklider, yet not attained.
(Jacucci et al., 2014)	Symbiotic interaction	A new generation of resources to understand users and to make themselves understandable to users
(Khabaza, 2014)	IA	Intelligence Amplification refers to the idea that the products of Artificial Intelligence will be used initially, not to create fully intelligent machines, but to amplify or increase the power of human intelligence.
(Paraense et al., 2007)	IA System	Computational systems performing some sort of intelligent decision making based on the cooperation provided by an ongoing dialogue between a human user and a computer system.

Based on these results we can see the field is diverse and draws characteristics from various research fields. From the results, we derive five distinct features, which are artificial intelligence, decision making, problem solving, partnership/collaboration or symbiosis and human empowerment. We list these feature in table 2.

Table 2: Comparing features

	Artificial Intelligence	**Decision making**	**Problem solving**	**Partnership/ collaboration**	**Empowering human**
(Breemen et al., 2011)			●		●
(Garcia, 2010)	●		●	●	●
(Dobrkovic et al., 2016)		●		●	●
(Greef et al., 2007)	●				●
(Greef et al., 2007)				●	●
(Griffith & Greitzer, 2007)				●	
(Jacucci et al., 2014)				●	
(Khabaza, 2014)	●				●
(Paraense et al., 2007)		●		●	

We first take a closer look what symbiosis between humans and machines entails. Jacucci et al. (2014) looked at different paradigms related to symbiotic collaboration. For this purpose, three different frameworks were discussed, namely that of telepresence, affective computing and persuasive technologies. Telepresence is the research about the subjective experience of being in an environment that is mainly supported by digital resources. Affective computing refers to computing that relates to, arises from, or deliberately influences emotions. Lastly, persuasive technologies deals with the persuasive power a computer possess to persuade a human to undertake action. A symbiotic relationship draws upon all these frameworks. A comparison between system properties of each framework and how they relate to a symbiotic relationship is shown in figure 5. The 'greyer' a box the more a property applies to said feature.

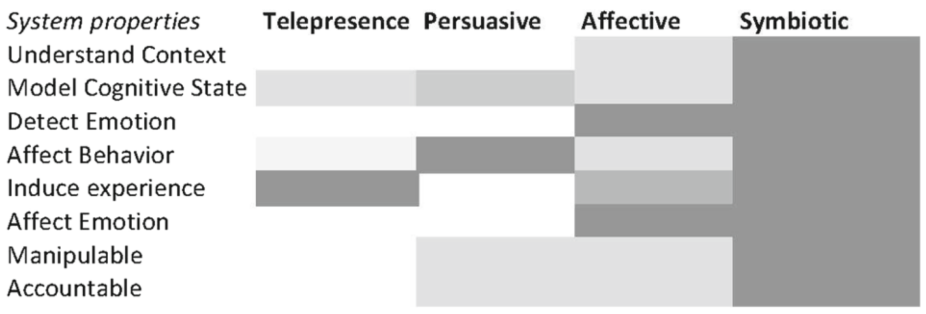


Figure 5: Comparing system features for several frameworks (Jacucci et al., 2014)

The features listed in figure 5 can also be found in the definition of Greef et al. (2007) and Griffith & Greitzer (2007). Almost all list a form of collaboration between human and machines. Two authors focus on AI in their definition. Where Cristina & Garcia (2010) indicates a partnership with an AI agent and a human Khabaza (2014) uses AI as a starting point to empower human intelligence. The same suggestion is made by Breemen et al. (2011) who indicates a human is empowered in its problem-solving ability. The same applies to empowerment of humans. Griffith & Greitzer (2007) keep their definition at a more abstract level by only focusing on a partnership; they do not indicate what this partnership can be used for.

Next, we group the five features by their function. We consider decision making and problem solving as a goal. Artificial intelligence and partnership are considered as means to achieve the goal. Finally, human empowerment is seen as *raison d'être* of Intelligence Amplification. Concluding the above we use the following definition for IA in this thesis:

> *Intelligence Amplification focusses on a close collaboration, with complementary contributions, between human and machine to empower humans in the decision-making process.*

2.1.3 Task delegation

A first attempt to distinguish tasks which are suitable for humans or machines was made in 1951 by Paul Fitts and are known as the Fitts' lists (Casini et al., 2015; Cummings, 2014). This list is considered out-of-date (Crouser & Chang, 2012) and behind it is a false idea that humans and computers each have strengths and weaknesses whereby human weaknesses are eliminated or compensated by machines (Dekker & Woods, 2002). The list might suggest humans and machines are antithetical, however they are better seen as complementary (Crouser & Chang, 2012).

Instead, automation creates new human strengths and weaknesses (Dekker & Woods, 2002). Failing to take this into account could lead to a situation where an engineer will envision the future in which *only* the predicted consequences will occur (Dekker & Woods, 2002). An update to the Fitts list has been proposed dubbed the "un-Fitts list" (Casini et al., 2015). This list is presented in table 3.

Table 3: The "Un-Fitts" list (Hoffman et al., 2002)

Machines	
Are constrained in that	**Need people to**
Sensitivity to context is low and is ontology-limited	Keep them aligned to the context
Sensitivity to change is low and recognition of anomaly is ontology-limited	Keep them stable given the variability and change inherent in the world
Adaptability to change is low and is ontology-limited	Repair their ontologies
They are not "aware" of the fact that the model of the world is itself in the world	Keep the model aligned with the world
Humans	
Are not limited in that	**Yet they create machines to**
Sensitivity to context is high and is knowledge- and attention-driven	Help them stay informed of ongoing events
Sensitivity to change is high and is driven by the recognition of anomaly	Help them align and repair their perceptions because they rely on mediated stimuli
Adaptability to change is high and is goal-driven	Affect positive change following situation change
They are aware of the fact that the model of the world is itself in the world	Computationally instantiate their models of the world

Strengths of humans are creativity (DiBona et al., 2016), self-reflection and the ability to perform a variety of tasks (Barca & Li, 2006). Machines can be designed to perform a specific task, can easily be replaced and can perform non-stop routines (Barca & Li, 2006).

Automation does not transform technology and the people who adapt; human practice get transformed as they adapt technology to fit their local demands and constraints (Dekker & Woods, 2002). Allocation of task should not be the focus, design for harmonious human-machine cooperation should be (Crouser & Chang, 2012).

A solution to task delegation is provided by Crouser & Chang (2012) by looking at *affordances*. An affordance is defined as action possibilities that are readily perceivable by a human operator. Affordances are relational and exists between human and machine; they do not exist separate form that relationship. They suggest a non-exhaustive list of human and machine affordances which we list in table 4.

Table 4: human and machine affordances (Crouser & Chang, 2012)

Human affordances	Machines affordances
Visual perception	Large-scale data manipulation
Visuospatial thinking	Collecting and storing large amounts of data
Audio linguistic ability	Efficient data movement
Sociocultural awareness	Bias-free analysis
Creativity	
Domain knowledge	

Visuospatial thinking is our ability to visualize and reason about the spatial relationships of objects in an image.

2.1.4 Human machine collaboration

The key to automation is to turn systems into team players (Dekker & Woods, 2002). Effective collaboration between humans and machines is essential to become team players. Or, as put by Griffith & Greitzer (2007), the goal is to create a neo-symbiotic interaction between the human and information. This raises the question what is required to achieve this goal.

Casini et al. (2015) list four requirements which are observability, directability, predictability and learning. Observability concerns our ability to understand and evaluate what is currently happening whereas directability is our ability to implement our goals for what we want to happen in the future. The latter two are closely related to each other and indicate (partial) results should be predictable and we should be able to learn from them. A lack of observability can lead to high complexity and undesirable machine behaviour (Greef et al., 2007). Next Casini et al. (2015) list three different intervention forms for collaboration between humans and machines. In the first situation, the system can ask the human operator for clarification. Secondly, a human can perform a random inspection and finally a human can perform a drill-down. In the last situation, a human is curious what led the machine to make a certain decision and he can inspect the processing chain which led to a specific assertion and conclusion.

Kondo et al. (2010) discusses a human machine collaboration in a kitchen for recognizing objects. Three assumptions are made:

1. the system should be able to uniquely a target object in good condition
2. the user can improve the conditions
3. the user can evaluate the result of the object recognition.

After testing a prototype, they indicate the key concept is to provide information feedback consisting of recognition status and suggestions for improvement.

Another list of requirements for collaboration is proposed by Dobrkovic et al. (2016):

- The human entity is given the master role, and it oversees the artificial intelligence,
- The artificial intelligence is given the assistant role,
- The human is responsible for the strategic decision making,
- AI is responsible for the tactical/operational tasks,
- The human is also responsible for the creative tasks that AI cannot handle,
- The AI is pre-processing the data, and brings awareness to the human component,
- The AI acts upon meta instruction given by the human,
- The AI analyses the human output in context using the available input, and learns to recognize and adapt to the human's behaviour,
- Depending on the level of autonomy of the AI, the machine will either automatically complete all computational tasks that conform with the strategic goals set by the human, or will suggest a solution for the human to verify, executing only the tasks that the human has approved,
- If the AI neither can understand the input, nor can process the task, it will ask for human assistance,
- The human can overrule the AI.

Using these requirements a hierarchical organization overview is created which we present in figure 6.

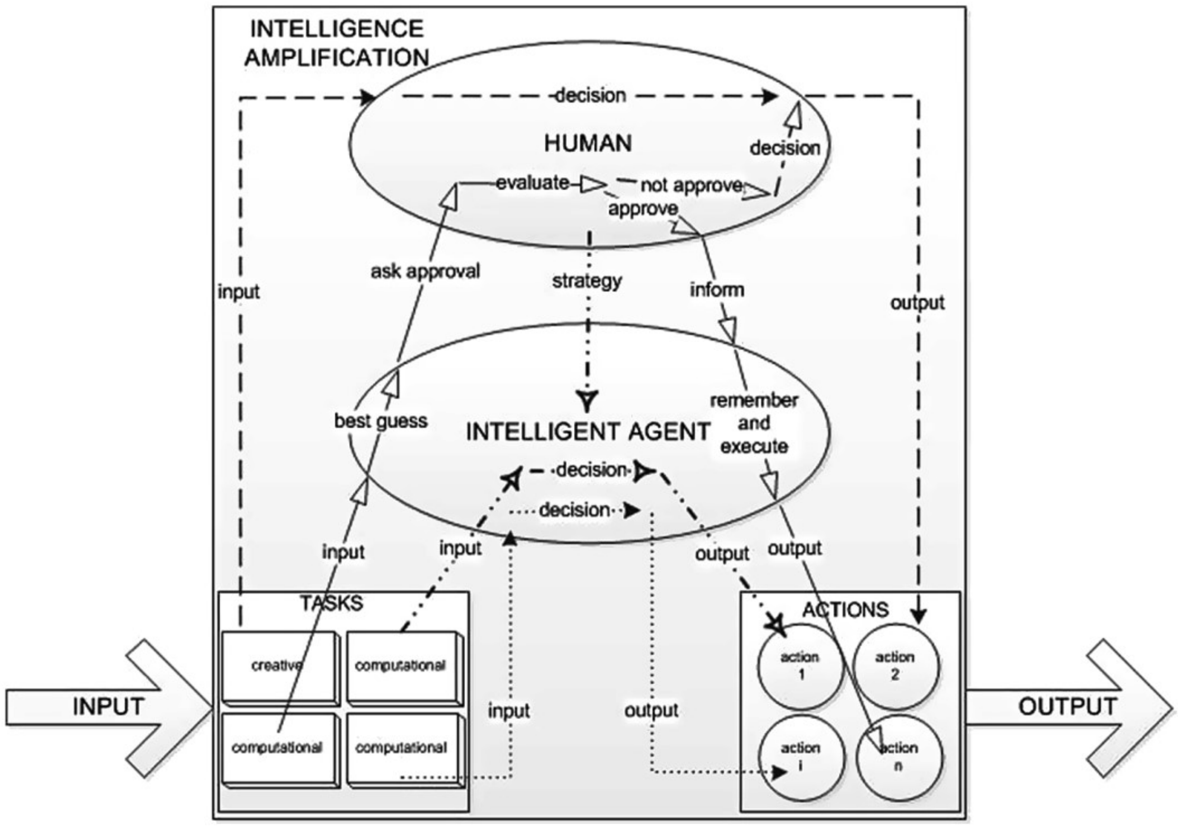


Figure 6: hierarchical overview of intelligence amplification (Dobrkovic et al., 2016)

When machine learning is used as intelligent agent the system should be capable of explaining the results it made (Stumpf et al., 2009). The user should be able to correct the machine if it is wrong. Paraense et al. (2007) state the final decision is not made by the human nor the machine but should be seen as an offspring of the collaboration between the two of them. Whether or not the AI should be able to overrule the human is open for debate (Barca & Li, 2006).

DiBona et al. (2016) investigated how information can effectively be shared between humans and machines. For this purpose, they propose the Proactive Autonomy Collaboration Toolkit (PACT) model (figure 7). The model is constructed using four primary elements: goals, work product, context, and information. A human (referred to as analyst) starts with *goals*. Initially these are limited, but they will become clearer further in the process. These goals lead to hypothesis which are fed to the *work product*. The latter is the collaborative environment where the human and computer (referred to as autonomy) work together to test hypothesis. Eventually the fully realized hypothesis becomes the goal of the research. Important to the collaboration with the machine is *context* which consists of two aspects: the actions and decisions of the analyst, and the *information* itself. Information fed to the work product by the human helps the machine understand what interest a human has and can help suggesting which information is relevant. This information serves as evidence for a hypothesis.

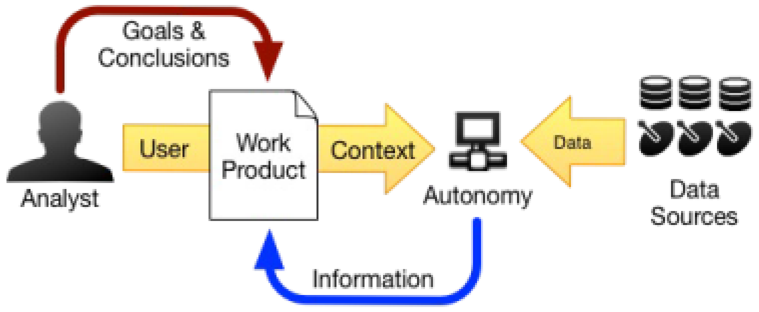


Figure 7: PACT model (DiBona et al., 2016)

In order to effectively allow collaboration between humans and machines a common language is need (Fischer, 1995). This language requires a data representation of hypotheses a human want to test which should be readable by both the human and machine.

2.2 Machine learning

We use machine learning to predict new mappings. Another option would be to predefine a set of rules. For example, when the name of two attributes are equal a new mapping should be created. This process of writing rules is what we refer to as heuristics and stands in contract to what we want to achieve by using machine learning. In other words, we don’t want to tell the machine what to do but we want the machine to learn these rules by itself so it can recognize when to make a new mapping.

The book of Kubat (2015) provides a good introduction to the various concepts. We use his example to give an introduction about machine learning. Figure 8 provides a list of pies Johnny likes and dislikes. What we want is to induce a *classifier* which is an algorithm capable of predicating whether Johnny likes a future pie. The examples given in figure 8 constitute the *training set*. This is what we use to train a classifier. For this case, there are two different class labels: positive and negative. A classifier capable for these problems is therefore referred to as a two-class classifier. Other options are multi-class classifiers (used when a minimum of three different class values are predicted) or one-class classifiers. The latter are also known as anomaly detection classifiers. They are used when there is plenty training data regarding one class, but little of another class. An example is a bank which is trying to predict fraudulent transactions. Since the overwhelming majority of transactions are not fraudulent it makes sense to supply a classifier the ‘good’ examples and derive a pattern from those. When a new example comes in it is checked whether it is an anomaly (i.e. possibly fraudulent) or not. We have used this type of classifier as well. These results are discussed in section 4.4.

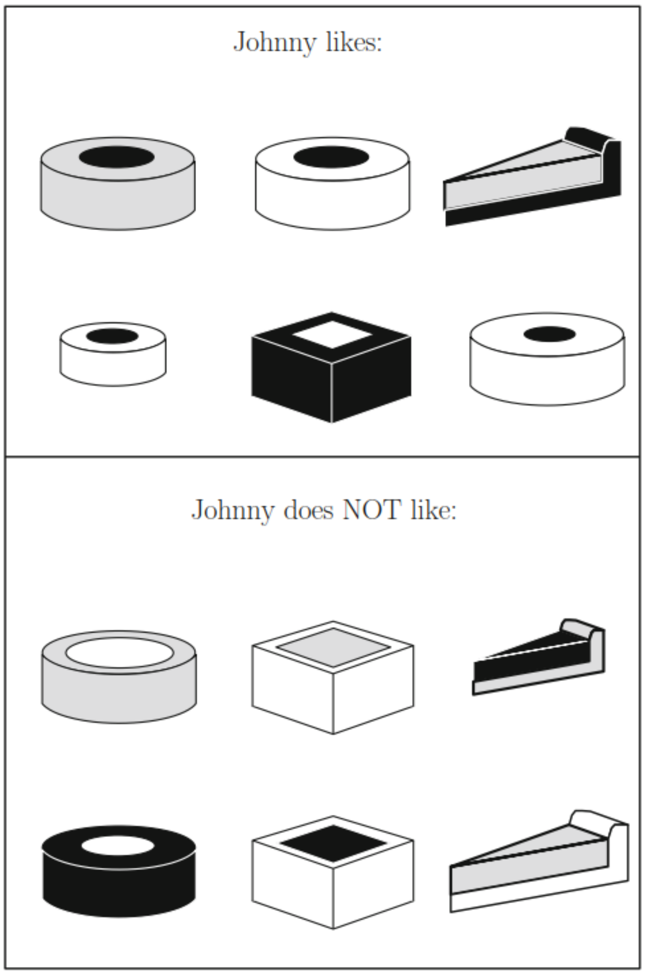


Figure 8: overview of pies Johnny likes and dislikes (Kubat, 2015)

For the machine to be capable of recognizing the various pies we describe features (Kubat refers to this as attributes, however, we prefer the term features because in the rest of the thesis we use the term attribute for a different purpose). Examples of features are the shape (round, rectangular or triangle) or crust-size (thin or thick). Using these features, we create a table which describes the training examples, see table 5.

Table 5: Twelve training examples expressed in a table (Kubat, 2015)

Example	**Shape**	**Curst - Size**	**Curst - Shade**	**Filling - Size**	**Filling - Shade**	**Class**
Ex1	Circle	Thick	Gray	Thick	Dark	Pos
Ex2	Circle	Thick	White	Thick	Dark	Pos
Ex3	Triangle	Thick	Dark	Thick	Gray	Pos
Ex4	Circle	Thin	White	Thin	Dark	Pos
Ex5	Square	Thick	Dark	Thin	White	Pos
Ex6	Circle	Thick	White	Thin	Dark	Pos
Ex7	Circle	Thick	Gray	Thick	White	Neg
Ex8	Square	Thick	White	Thick	Gray	Neg
Ex9	Triangle	Thin	Gray	Thin	Dark	Neg
Ex10	Circle	Thick	Dark	Thick	White	Neg
Ex11	Square	Thick	White	Thick	Dark	Neg
Ex12	Triangle	Thick	White	Thick	Gray	Neg

Many classifiers exist. We group them together and discuss the type of classifier below (Herrera et al., 2016; Witten, Frank, & Hall, 2011a):

Bayesian learning: methods that are based on Bayes Theorem. Most notable is NaiveBayes which assumes mutual independence between attributes. In practice this means all attributes make an equal contribution to the decision (Witten et al., 2011a). All though this assumption never holds in practice it turns out NaiveBayes can yield good performance (Herrera et al., 2016).

Instance Based Learning: sometimes known as lazy learners. These classifiers do no train or construct a model, but store all training data and compute distance measures between them. When a new attribute comes in the closest training example(s) is located and the outcomes are aggregated into a prediction.

Rule Induction: construct a set of rules to predict the outcome. Rules have the advantage they are very easy to understand by humans. The easiest to understand rule classifier is OneR. This uses one feature to construct rules. Even though this is a very rudimentary approach it comes up with quite good results for characterizing the structure in data (Witten et al., 2011a).

Decision trees: this group also provides models which are easy to interpret by humans. Decision trees are constructed by making multiple divisions at nodes and ending up at a leaf node. The leaf node provides the prediction (Herrera et al., 2016). Decision trees compare well to rules but there are differences. In a multi-class case a decision tree split takes all classes into account in trying to maximize the purity of the split whereas a rule-generating method concentrates on one class at a time, disregarding what happens to other classes (Witten et al., 2011a).

Logistic regression: uses numeric attributes to construct classifiers. However, nominal attributes can be used as well. Logistic regression builds a linear model based on a transformed target variable.

Support vector machines: a more recent category of classifiers. They construct a hyperplane such that a maximal separation is achieved between the classes. This has the advantage they are not prone to overfitting.

Neural network: the working of our brain inspires these methods, or, more specifically the working of the neurons. An artificial neuron receives many weighted inputs and provide one aggregated outcome. A neural network has at least an input and output layer and between are hidden layers. This makes it possible to model complex data.

To train a classifier a training set is used. After the classifier is trained we want to get an idea how it will perform. There are two different approaches for doing so. First, we split the data in a training set and a test set. As the names imply we use the train set to train the classifier, subsequently the classifier then scores the instances in the test set. By comparing the outcome of the classifier in the test by the outcome of the ground truth contained in the test set we get an idea of its performance. The other option is to use k-fold cross validation. This method is better suitable when limited training data is available. This is not the case in our thesis, therefore we omit a further explanation and we refer the keen reader to Witten et al. (2011b)

Now we know how to test the performance we are interested in the metrics used for this purpose. The easiest and best to understand method is accuracy. Accuracy is defined as the amount of correctly classified instances w.r.t. the total amount of instances (Kubat, 2015). For the problem of schema matching literature suggests three metrics are often used. These are precision, recall and F-measure (Duchateau & Bellahsene, 2016). Precision calculates the proportion of relevant matches among those which have been discovered. Recall measures the relevant correspondences between all relevant ones. A graphical overview of is displayed in figure 9. Finally, the measure is the harmonic mean of precision and recall and can be calculated as follows:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

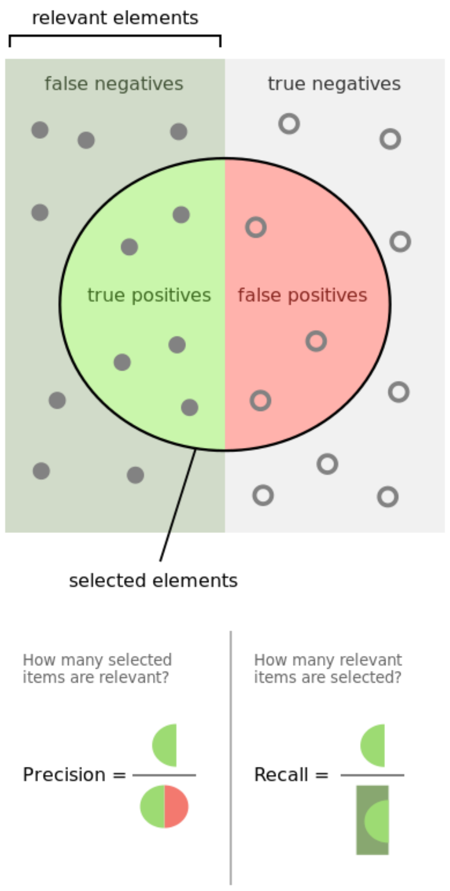


Figure 9: graphical overview of precision and recall (Wikipedia, 2017)

A popular tool for performing machine learning and one that is often used in scientific publications is Weka (Duchateau & Bellahsene, 2016). Weka provides a number of different classifiers which can easily be trained without programming experience (Witten, Frank, & Hall, 2011c).

2.3 Database schema matching

The second part of our literature study focusses on the concept of database schema matching.

2.3.1 Questions and selection process

During the literature search we want to answer the following questions:

- What techniques for schema matching have been proposed?
- Which similarity measures are available?
- Which approaches for schema matching have been proposed?

To answer the questions, we started with a literature search on Scopus as we did in section 2.1. Articles not available on Scopus were looked up using Google Scholar. The search was conducted mid-March 2017. Many approaches have been proposed (Assoudi & Lounis, 2015) and we are interested in those that use some form of machine learning. Therefore, the following keywords were used ("schema matching" AND "machine learning"). After we read the documents we retrieved additional documents by using a forward and backward search on the relevant articles. This yielded a set of 8 relevant documents. The search process is displayed in figure 10.

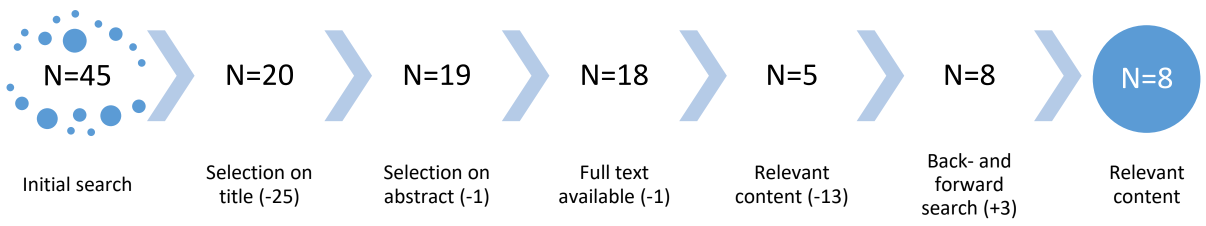


Figure 10: Overview of literature search

2.3.2 Techniques

Schema matching is the problem of finding pairs of attributes (or groups of attributes) from a source schema and attributes of a target schema such that pairs are likely to be related (Assoudi & Lounis, 2015). It is a basic problem which can be found in many application domains (Rahm & Bernstein, 2001). A schema is a set of elements connected by some structure. For schema matching many approaches are possible and a taxonomy is provided by Rahm & Bernstein (2001) which we present in figure 11.

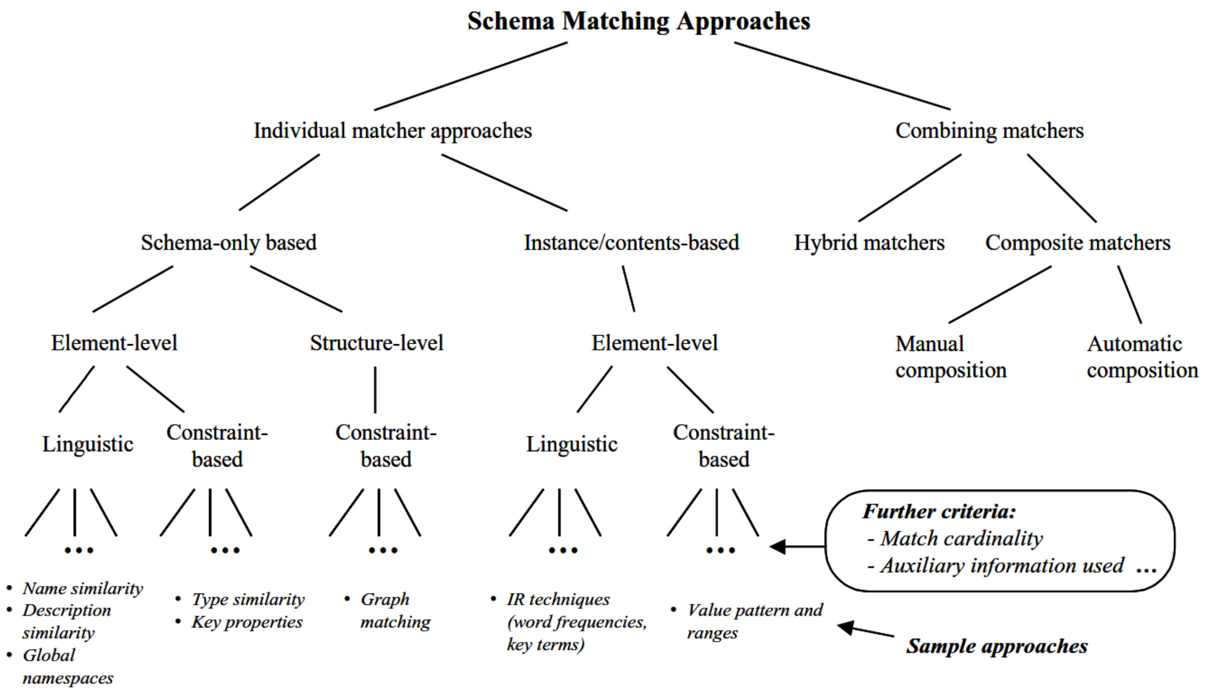


Figure 11: Taxonomy for schema matching approaches (Rahm & Bernstein, 2001)

At first a distinction is made between individual and combining matchers. The latter uses multiple individual matchers to get to its final result. An individual matcher computes a mapping based on a single matching criterion. A hybrid matcher uses multiple criteria to create a match whereas a composite matcher combines multiple results to create its final match. The taxonomy of the individual matchers makes the following splits (Rahm & Bernstein, 2001):

- Schema vs. instance based: the first case only takes the schema information is into account whereas the latter also uses instance data.
- Element vs. structure matching: element matching uses individual schema elements such as attributes; structure matching is performed for combinations of elements.
- Language vs constraint: a matcher can use a linguistic- based approach (e.g., based on names and textual descriptions of schema elements) or a constraint-based approach (e.g., based on keys and relationships).
- Matching cardinality: the overall match result may relate one or more elements of one schema to one or more elements of the other, yielding four cases: 1:1, 1:n, n:1 and n:m. In addition, each mapping element may interrelate one or more elements of the two schemas. Furthermore, there may be different match cardinalities at the instance level.
- Auxiliary information: a matcher could use auxiliary information such as dictionaries, global schemas, previous matching decisions and user input.

2.3.3 Similarity measures

In section 2.2 we discussed the need for features when performing machine learning. For schema matching these features are similarity measures. A similarity measure takes strings requires as input and the output is an integer describing the similarity. Similarity measures are bundled in the Second String project (Cohen, Ravikumar, & Fienberg, 2003). Several methods exist which we discuss briefly. The first category of methods are edit distance functions. A distance function maps a string *s* to a string *t* by calculating a real number *r*. A low value of *r* indicates high similarity. This stand in contrast to a similarity function for which a high value of *r* indicates a high similarity. We make a further distinction between edit distance functions and token based functions.

Edit based distance functions

The most well-known edit distance is the Levenshtein distance. Levenshtein counts the amount of edit operations needed to convert string *s* into string *t*. An edit operation is a character insertion, deletion or substitution. In the basic form each operation has a cost of 1 (Christen, 2006). A more advanced edit distance function is Monge-Elkan which normalizes the score between [0,1]. It is an affine variant which means a sequence of insertions or deletions are given lower cost.

Jaro is another popular similarity function which is not based on edit distance. It is based on the number of, and order of, common characters between two strings. Winkler proposed a variant, called JaroWinkler, which emphasizes the similarity at the beginning of the strings. It does so by using the length of the common prefix. Both Jaro and JaroWinkler are intended for short strings.

Token based distance functions

The methods we discussed so far look at characters in a string. However, often strings consist of multiple words (or tokens). Token based functions compute a similarity by looking at the words rather than the characters. An example is the Jaccard similarity which computes how many words occur in both string *s* and *t* and divide this number by the total amount of words in both strings. There are more token based similarity functions, however, since we are not using them for the purpose of this thesis we refer to the work of Cohen et al. (2003).

Hybrid distance functions

A hybrid function uses tokens as input and computes edit based distance functions of all possible combinations of words. First strings *s* and *t* are broken down into substrings $s = a_1,\dots,a_K$ and $t = b_1,\dots,b_L$. Similarity is then computed using the following formula:

$$sim(s,t) = \frac{1}{K}\sum_{i=1}^{K} max_{j=1}^{L}(sim'(A_i, B_j))$$

Sim' is a secondary distance function. In the Second String project the Monge-Elkan, Jaro and Jaro-Winkler are used as secondary functions. These functions are referred to as level two distance functions. To illustrate how such a function works we compute the level 2 similarity score for the following two strings: *'carriersuser firstname'* and *'carriercontact firstname'*. As secondary distance function we use JaroWinkler. Each string contains two tokens (words) thus the total amount of possibilities is 2 * 2 = 4. For each combination we calculate the JaroWinkler score. After computing the scores the maximum is taken for each token in the first string. This is shown in table 6.

Table 6: calculating a Level 2 similarity score

Token – string 1	Token – string 2	JaroWinkler
carrieruser	carriercontact	0,82
carrieruser	firstname	0,43
Maximum for first token in string 1		**max(0,82; 0,43) = 0,82**
firstname	carriercontact	0,5
firstname	firstname	1
Maximum for second token in string 1		**max(0,5; 1) = 1**

Finally, the average is calculated from the obtained maximum:

$$\frac{1}{2} * (0{,}82 + 1) = 0{,}91$$

The level 2 similarity score for these two strings using JaroWinkler as secondary distance function is 0,91.

2.3.4 Schema matching approaches

Next, we look at several examples found in literature. First, we describe a general approach to schema matching in figure 12.

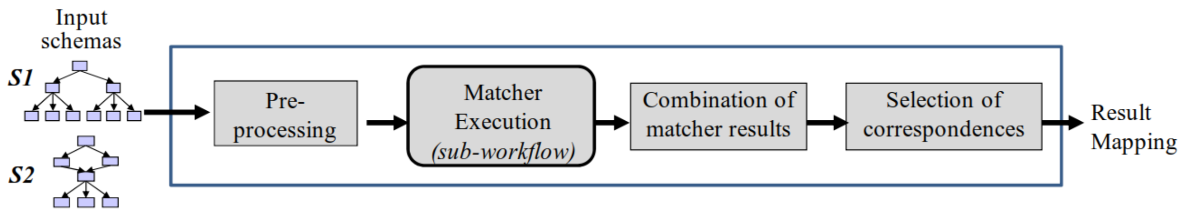

Input schemas
S1
S2
Pre-processing
Matcher Execution (sub-workflow)
Combination of matcher results
Selection of correspondences
Result Mapping

Figure 12: general workflow of a schema matcher (Rahm, 2011)

The input consists of two schema which are processed into an internal processing format (Rahm, 2011). Possible different pre-processing steps can be applied such as tokenization or a dictionary lookup. Next a matcher determines correspondences. When multiple matchers are used the results are combined and based on these results a selection of correspondences constitute the result mapping. Many different approaches have been developed prior to 2001 and a summary of these methods can be found in Rahm & Bernstein (2001). Below we discuss several more recent and successful approaches.

A well-known example of a schema matcher often referred to in literature is COMA (Rodrigues et al., 2015). COMA uses heuristics to combine the result of different matching algorithms to determine matching instances. Internally input schemas are converted to trees for structural matching (Duchateau & Bellahsene, 2016). An overview how COMA works is given in figure 13.

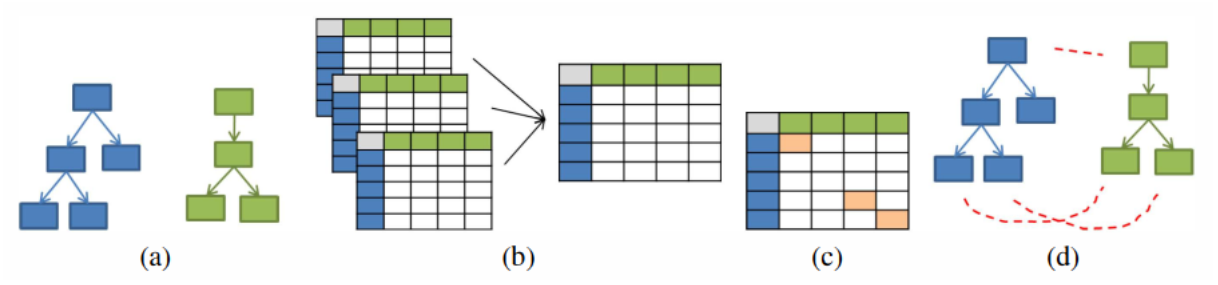


Figure 13: COMA matching operation: (a): two input schemas, (b) matrices aggregation, (c) candidate selection and (d) output (Rodrigues et al., 2015)

COMA receives two input schemas (a) for which it makes a matrix of all possible combinations. Each matrix is labelled using different similarity measures. All matrices are aggregated in a single matrix according to a chosen criterion (e.g. maximum, average, minimum). From the aggregated matrix candidates are selected for which the value exceeds a certain threshold (c) which are then presented to the user (d). Since COMA relies solely on heuristic the approach was not used in our approach since we want to incorporate a machine learning approach.

A recent advancement using machine learning is YAM, short for **Y**et **A**nother **M**atcher (Duchateau & Bellahsene, 2016). YAM is a schema matcher generator designed to generate a tailor-made matcher when making a new mapping. Optionally user input can be specified to integrate user preferences or requirements. YAM uses more than 20 different classifiers using Weka and over 30 similarity measures using the Second String Project. An architecture overview of YAM is presented in figure 14.

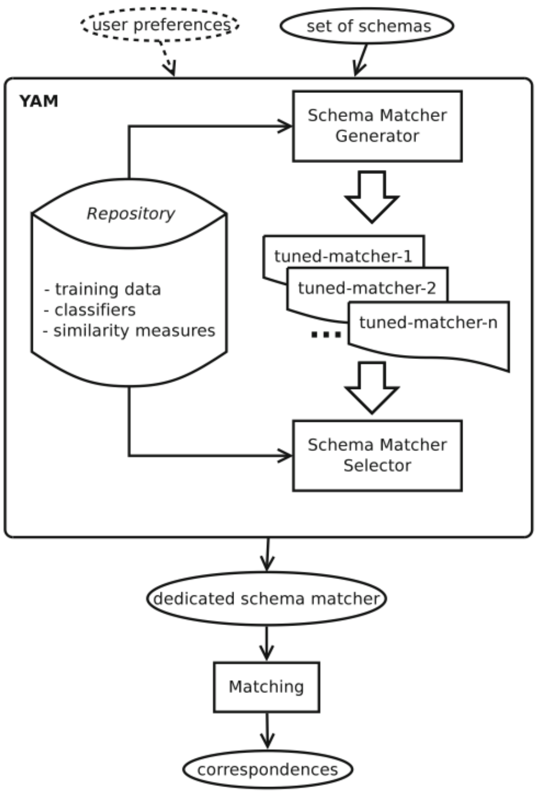


Figure 14: Architecture of YAM (Duchateau & Bellahsene, 2016)

Internally YAM stores a repository of schemas (training data), classifiers and similarity measures. When a new schema is presented these are used to generate matchers. Additional user preferences can be included by the user. For example, a user could indicate whether he favours precision over recall or include additional training data (expert correspondences). The output of YAM is a dedicated schema matcher. In this thesis, we aim to create a dedicated schema matcher by using a plethora of expert correspondences and thus we haven't used YAM.

Similarity flooding is an approach based on structural matching which is given by Melnik, Garcia-Molina & Rahm (2002). The algorithm makes use of the hierarchal relationships found in XML schemas to derive mappings between schema elements. This allows for an algorithm which works with schemas from different domains. Other methods based on heuristic are often fine-tuned which costs significant time and resources. Structure matching is therefore promising, but needs further research (Zhao & Ma, 2017).

2.4 Summary

In this section, we first looked at intelligence amplification. A definition was extracted based on a selection of articles. Intelligence Amplification is not a well-defined term and as such it draws upon literature from various domains.

Important for Intelligence Amplification is the close collaboration between humans and machines. This led to the idea of investigating the task delegation between the two. Some argue the idea of exploiting strengths and weaknesses of humans and machines should not be looked at we tend to disagree. Naturally each has their own strengths and weaknesses and exploiting them results in an effective collaboration. To model this collaboration the PACT framework discussed explains how a machine and computer could collaborate on the same product. The framework provided by Dobrkovic et al. (2016) provides further guidelines.

A general introduction was given into machine learning. We explained what a classifier is, what is needed to train a classifier and how performance of the classifiers can be evaluated on a test set by looking at precision and recall.

Finally, a literature search was conducted for database schema matching. Many approaches have been proposed over the recent years. However, they all focus on the matching task, not on the involvement of users in this task (Falconer & Noy, 2011).

3 Reference architecture

In this chapter, a reference architecture is defined for performing schema matching using Intelligence Amplification. Section 3.1 introduces the concept of a reference architecture. Next, the approach to the reference architecture is discussed in section 3.2. This is divided in two stages, first is pre-processing discussed in section 3.3 and matching in section 3.4. Section 3.5 gives a summary.

3.1 Introduction

A reference architecture is a generic architecture for a class of systems used as a foundation for the design of concrete architectures from this class (Angelov, Grefen, & Greefhorst, 2012). A concrete architecture is an architecture specifically designed for a software application. The purpose of a reference architecture is to provide guidance for future development (Cloutier et al., 2009), provide standardization of concrete architectures and facilitation of the design of concrete architectures (Angelov et al., 2012). A reference architecture is defined at an abstract level (Angelov et al., 2012). This level of abstraction is the cause of one of the main challenges of a reference architecture, namely to make them concrete and understandable (Cloutier et al., 2009). For an extensive list of benefits and drawbacks we refer to the paper of Martínez-Fernández, Ayala, Franch, & Marques (2017).

Our reference architecture is developed using the ArchiMate language. The core of ArchiMate consists of three layers, namely the business, application and infrastructure layer (Iacob, Jonkers, Quartel, Franken, & Berg, 2012). The *business layer* offers products and services to external customers that are realized in the organization by business processes. It shows how the organization is internally organized. Next is the *application layer* which delivers the services to realize it's business added value modelled in the business layer. Lastly the *infrastructure layer* realizes infrastructure services on which applications can be build. Enterprise architecture shows the relation between these layers. These concepts are from the ArchiMate core (version 1); the language has been further expanded with a motivation, implementation & migration layer (version 2) and more recently a strategy and motivation layer (version 3). The reference architecture is based on the business and application layer.

3.2 Approach

The architecture is based on the general approach to schema matching proposed by Rahm (2011) which was discussed in section 2.3.4. Two main actions are distinguished: a pre-processing stage and a matching phase. When discussing the interaction between human and machine we refer to the PACT model discussed in section 2.1.4. Both actions contain a work product which both machine and human work on. This is indicated by creating a green coloured data object.

3.3 Pre-processing

Pre-processing is the task of cleaning the input labels. Auxiliary information could be used here such as thesauri or dictionaries (Rahm & Bernstein, 2001). The drawback of thesauri or dictionaries is they often do not contain domain specific words, the ability to expand abbreviations or the ability to expand compound nouns (a compound noun is a word composed of more than one word) (Sorrentino, Bergamaschi, Gawinecki, & Po, 2010). Employing pre-processing steps can greatly improve results (Sorrentino et al., 2010).

The process of pre-processing first starts with the user who indicates which pre-processing steps are needed. For example, if the source schema is in Dutch whereas the destination schema is in English a translation for the source schema could be performed. The process steps are graphically presented in figure 15.

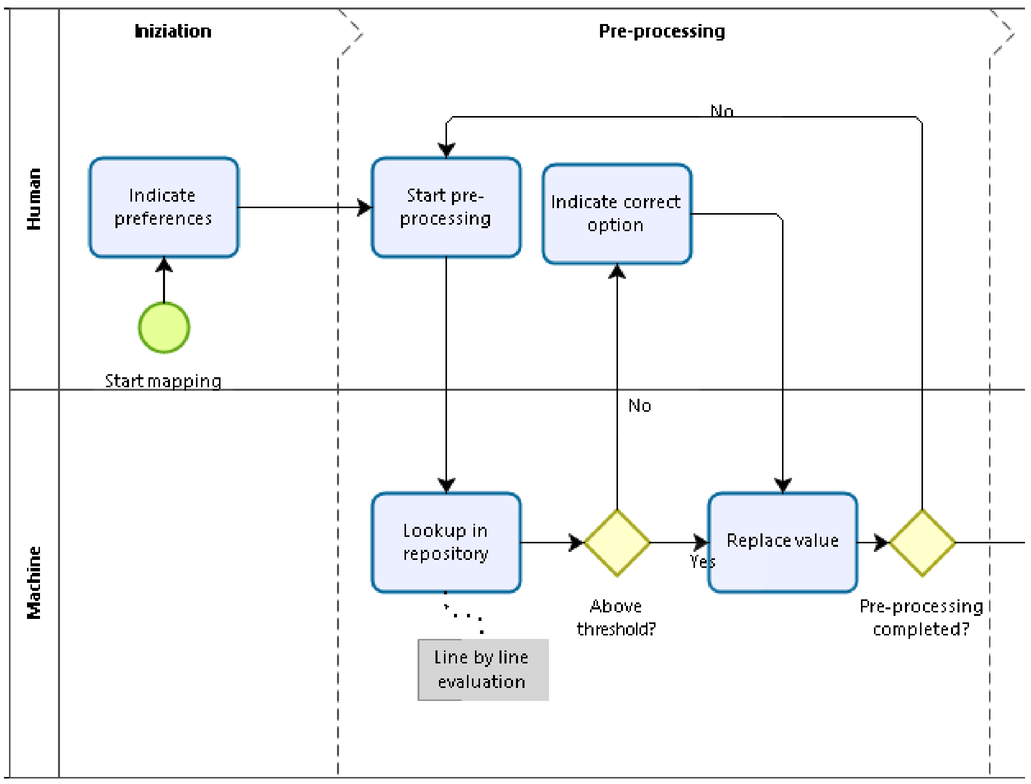


Figure 15: Steps for pre-processing

For each pre-processing step, the machine performs a lookup in a repository. When a lookup value is found, or when a highly similar value is found, it is automatically replaced. However, when the machine is not certain about a replacement value the user is involved and is asked to indicate the correct option.

To effectively leverage the power of the human in the process the work product (DiBona et al., 2016) is the list of schemas which need to be cleaned. The computer first tries to clean the text by performing a lookup in a repository (1.2). For each lookup, a score of certainty is generated. When the score is above a predefined threshold (1.4) the option should automatically be selected (1.6). When in doubt the user is invoked who decides which option is best, or, when no option is presented, indicate the result (1.5).

We note this stage of pre-processing can occur multiple times (1.7). For example, first compound words could be cleaned, next abbreviations expanded and finally a translation is made. As said this is indicated by the user in the preference (0.1). Which pre-processing task yield best results depends on each situation and is open for further research because different pre-processing steps lead to different mappings (Zhao & Ma, 2017). For this reason, only the use of pre-processing steps is indicated in the reference architecture. We consider this to be the essence and therefore include it in the reference architecture (Cloutier et al., 2009). Which pre-processing options are implemented should as such be part of a concrete architecture.

The architecture for pre-processing is displayed in figure 16. The functions in the architecture refer to the tasks in figure 15.

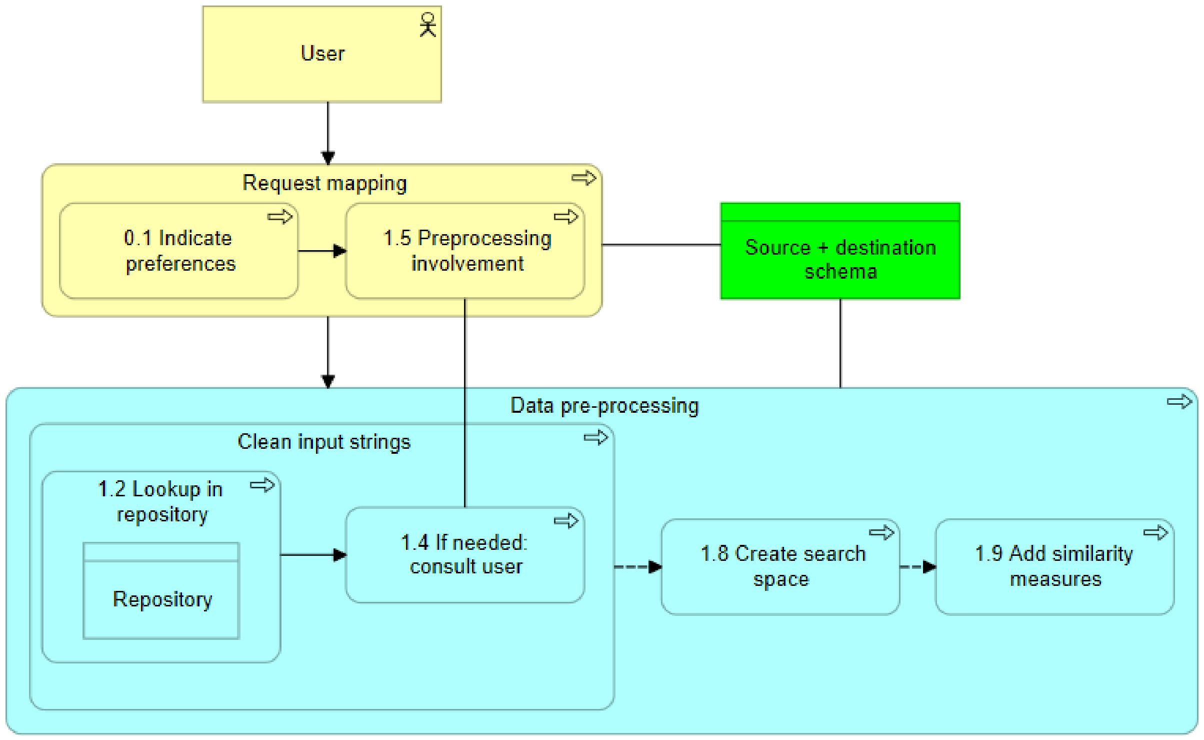


Figure 16: Architecture for pre-processing

As mentioned, the user triggers the mapping process which starts by indicating the preferences (0.1). At the pre-processing function, the first function (clean input string) is coloured green. This indicates there is a work product which the human and computer jointly work together. In this case these are the source and destination schema. After the data has been cleaned a search space is created and similarity measures are added. The boxes are coloured blue which we use to indicate this is a computer only task.

3.4 Matching

After the data has been pre-processed a software agent is invoked. When using machine learning each pair of schema elements is considered a machine learning object where its attributes are the similarity values computed by a set of similarity measures of these elements (Duchateau & Bellahsene, 2016). An active learning approach could also be used as intelligent agent. Compared to traditional machine learning, where user intervention is required afterwards, active learning is requested while the method is running (Rodrigues et al., 2015). This goes beyond the goal of this thesis and for now we discuss agents which operate independently. It is important to note the actions for generating candidate mappings can be repeated and are therefore iterative (Falconer & Noy, 2011). The process diagram is shown in figure 17.

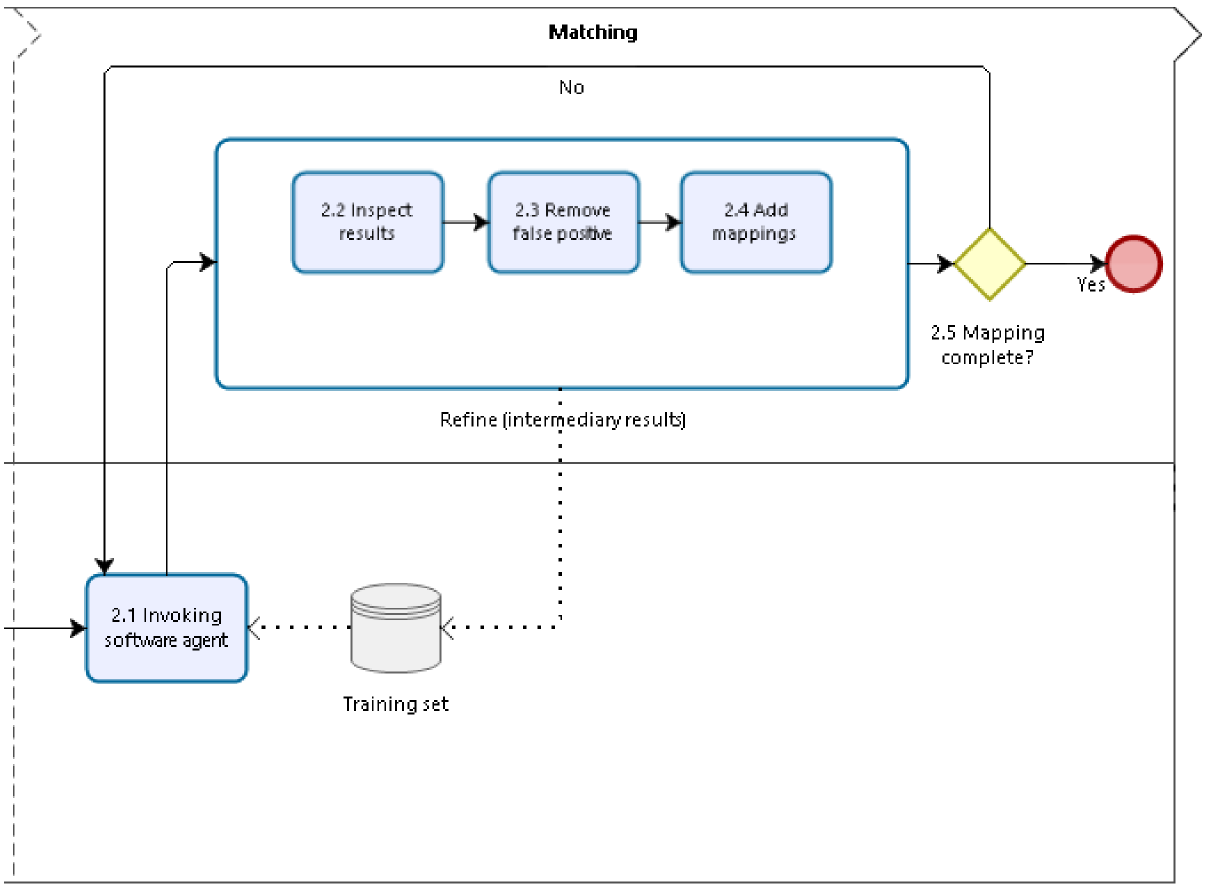
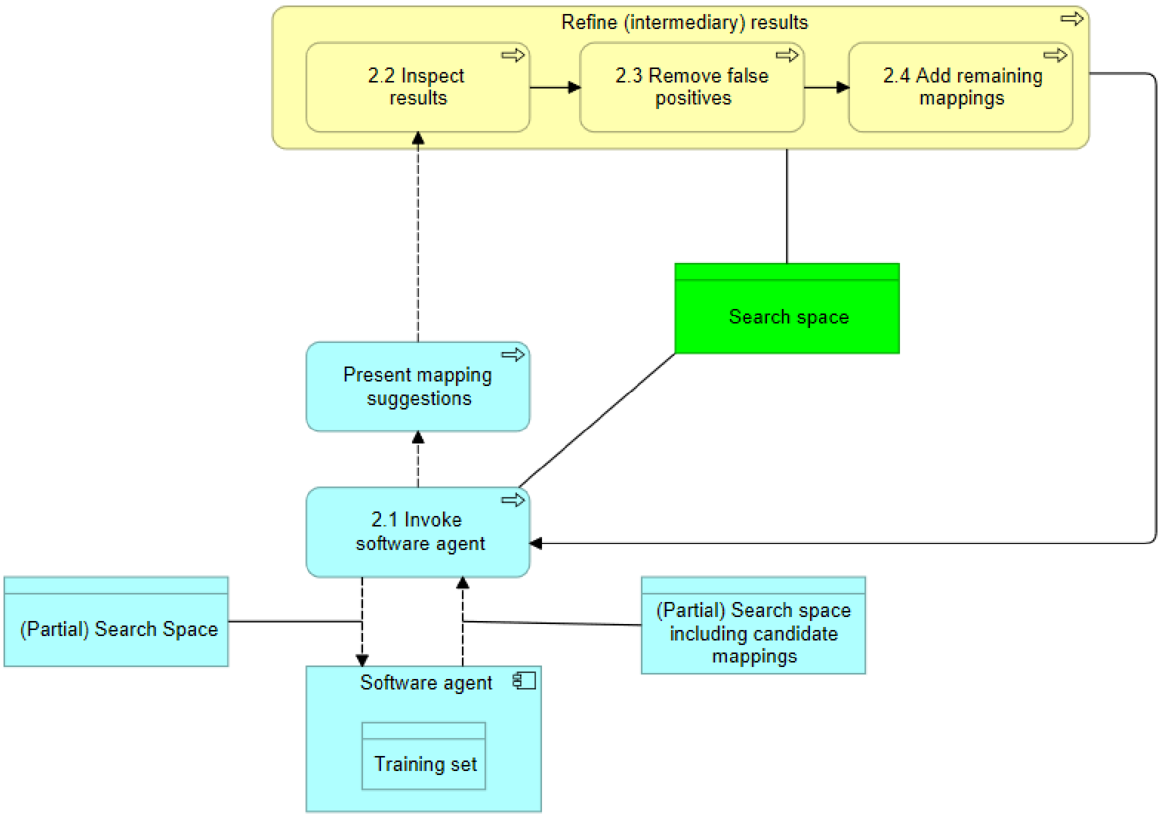

Matching
No
2.2 Inspect results
2.3 Remove false positive
2.4 Add mappings
2.5 Mapping complete?
Yes
Refine (intermediary results)
2.1 Invoking software agent
Training set

Figure 17: Process overview of matching stage

Initially, the software agent generates a list of candidate mappings (2.1) which are presented to the user, who inspects them (2.2), removes false positives (2.3) and remaining mappings (2.4). However, this doesn't necessarily have to complete the mapping scenario. A user could opt to invoke a different software agent or make a selection for which he needs refinement (2.5). In this case the actions repeat itself. This loop, to re-invoke the software agent, is what distinguishes the Intelligence Amplification approach from other existing approaches (Falconer & Noy, 2011). The architecture for realizing such a process is displayed in figure 18.



Refine (intermediary) results
2.2 Inspect results
2.3 Remove false positives
2.4 Add remaining mappings
Search space
Present mapping suggestions
2.1 Invoke software agent
(Partial) Search Space
(Partial) Search space including candidate mappings
Software agent
Training set

Figure 18: Architecture for matching

When invoking the software agent (2.1) the search space is send and as a response the search space including suggested mappings is returned. The software agent could use a training set when making selections (Rahm & Bernstein, 2001). This is used when the software agent is a machine learning classifier, but is not needed when a heuristic is used. After the agent created a list of suggested mappings these are presented to the user. Visualisation is very important in this stage. Presenting all schema matching correspondences to a user at once could be too overwhelming and in fact annoys the user as they become frustrated sifting through all the false positives (Falconer & Noy, 2011). Completing the mapping is a task which is both time consuming and cognitively demanding (Falconer & Noy, 2011). An explanation of the reason why the software agent suggested a mapping is considered an important feature to help the user but this still is a feature where many approaches fall short (Falconer & Noy, 2011; Ivanova, Lambrix, & Åberg, 2015).

The green coloured search space is the work product machine and human provide input to. The machine starts with the initial input which the user refines and the machine re-adds knowledge. When the matching is complete the search space is kept by the machine so it can learn upon it in future iterations.

3.5 Summary

The overall process diagram if shown on the next page in figure 19.

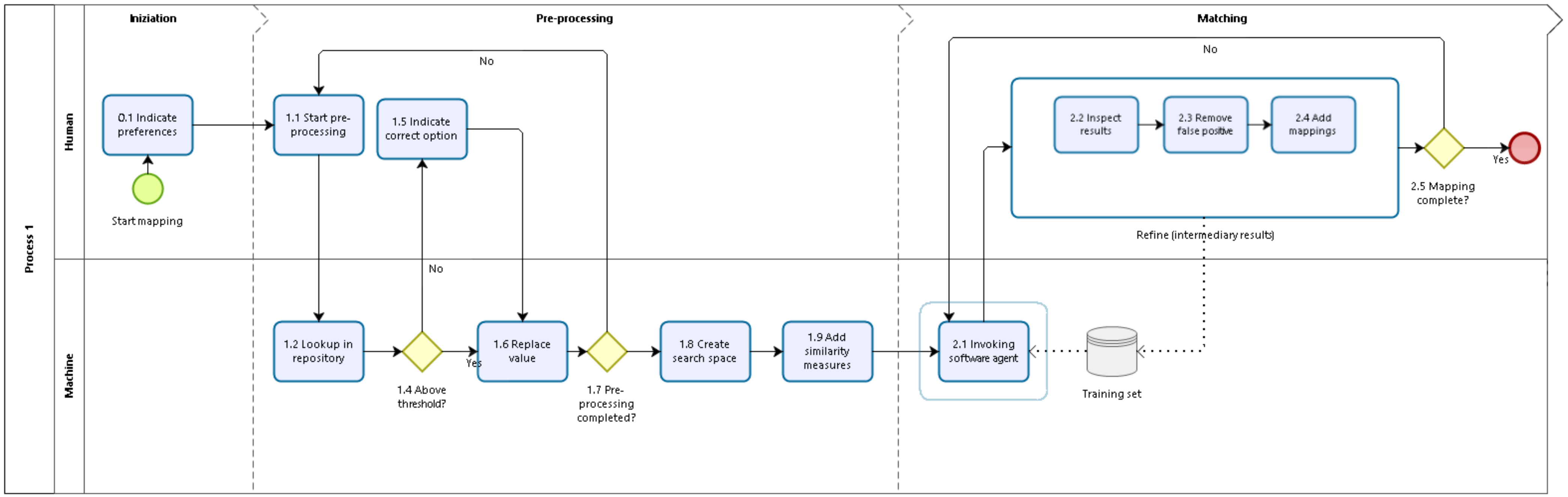


Figure 19: Total process diagram

Finally the complete reference architecture is displayed on the below in figure 20.

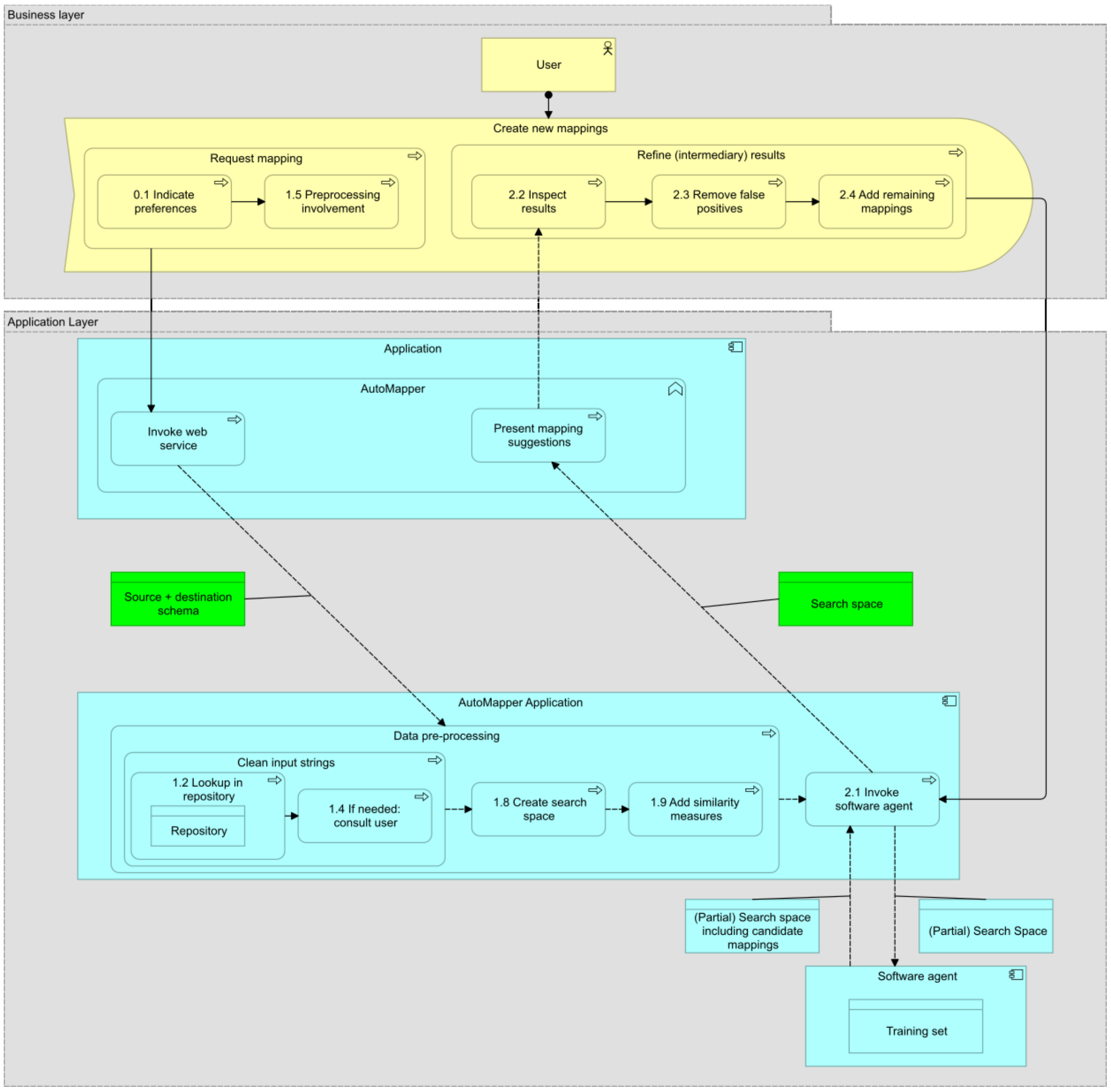


Figure 20: IA driven architecture for schema matching

The reference architecture consists of the two stages discussed above. The green coloured data objects coloured green is a work product user and computer jointly work together on (DiBona et al., 2016).

4 Prototype

This section first describes the architecture used to build the prototype in section 4.1. Next, the machine learning platform used is discussed in section 4.2 after which section 4.3 describes the training set. Section 4.4 discusses results of a one-class classifier (anomaly detection) and section 4.5 the results of the two-class classifiers. Finally, section 4.6 gives an overview of the prototype itself.

4.1 Architecture

The prototype is developed to be part of eMagiz which is the software product developed by CAPE Groep. eMagiz functions by using an Integrated Life Cycle management consisting of five different phases:

- **Capture:** the initial phase where requirements are captured. This gives a high-level overview of the integration
- **Design:** this is where mappings are designed. The prototype we have developed is used in this phase.
- **Create:** after the mappings have been designed they are refined and finalized. This also includes the routing process of messages.
- **Deploy:** the created mappings are deployed to a production environment.
- **Manage:** when deployed transformations are managed in this phase.

The system architecture provided in this section is based on the reference architecture defined in the previous chapter. A system architecture is focussed on a limited class of systems from the reference architecture and is used to design and engineer a system (Cloutier et al., 2009). Figure 21 shows the architecture.

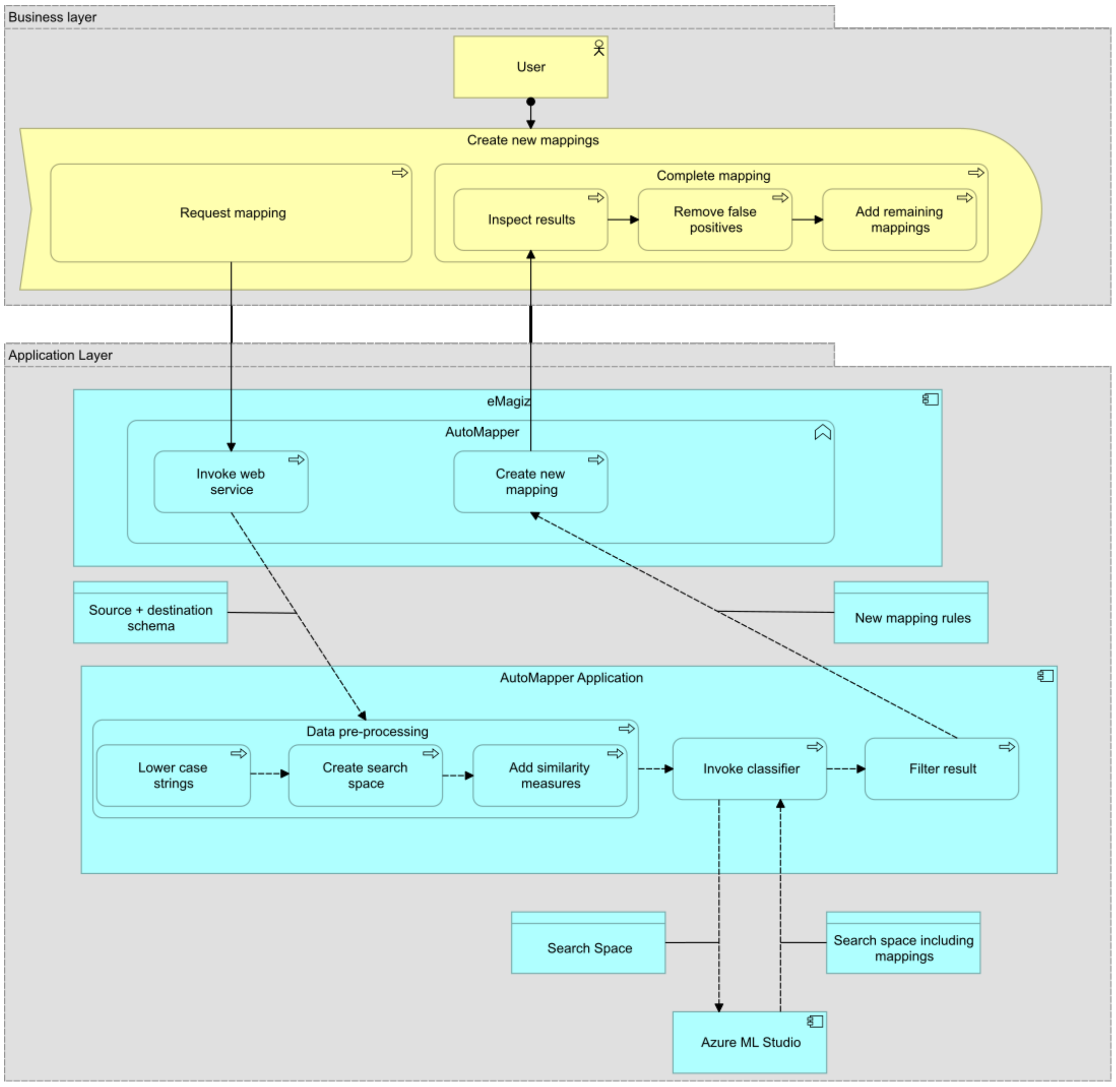


Figure 21: architecture of the prototype

As with the general architecture the user invokes a process to create new mappings. This is triggered by a button in eMagiz which calls the AutoMapper webservice. However, in our case the user is not yet involved in any pre-processing nor is it able to indicate any preferences.

During pre-processing the only step currently taken is to lower case the string. Next it iterates over all possible combinations by comparing every element of the source schema with every element of the destination schema (i.e. evaluation of the cross join) . This approach has at least a quadratic complexity and could lead to problems when using large schemas (Rahm, 2011). In these cases, reduction of the search space could be needed. In our prototype this isn't done.

Next similarity measures are added. Initially we have chosen for Levenshtein and JaroWinkler. The search space is wrapped in a web service and is send to the classifier. For this purpose, we use a trained classifier using the Azure Machine Learning platform from Microsoft. This platform offers plenty of possibilities for training a classifier and provides an easy to use interface. Section 4.2 provides detailed information about the platform and section 4.3 dives deeper in the training set. Figure 22 gives an overview of this process.

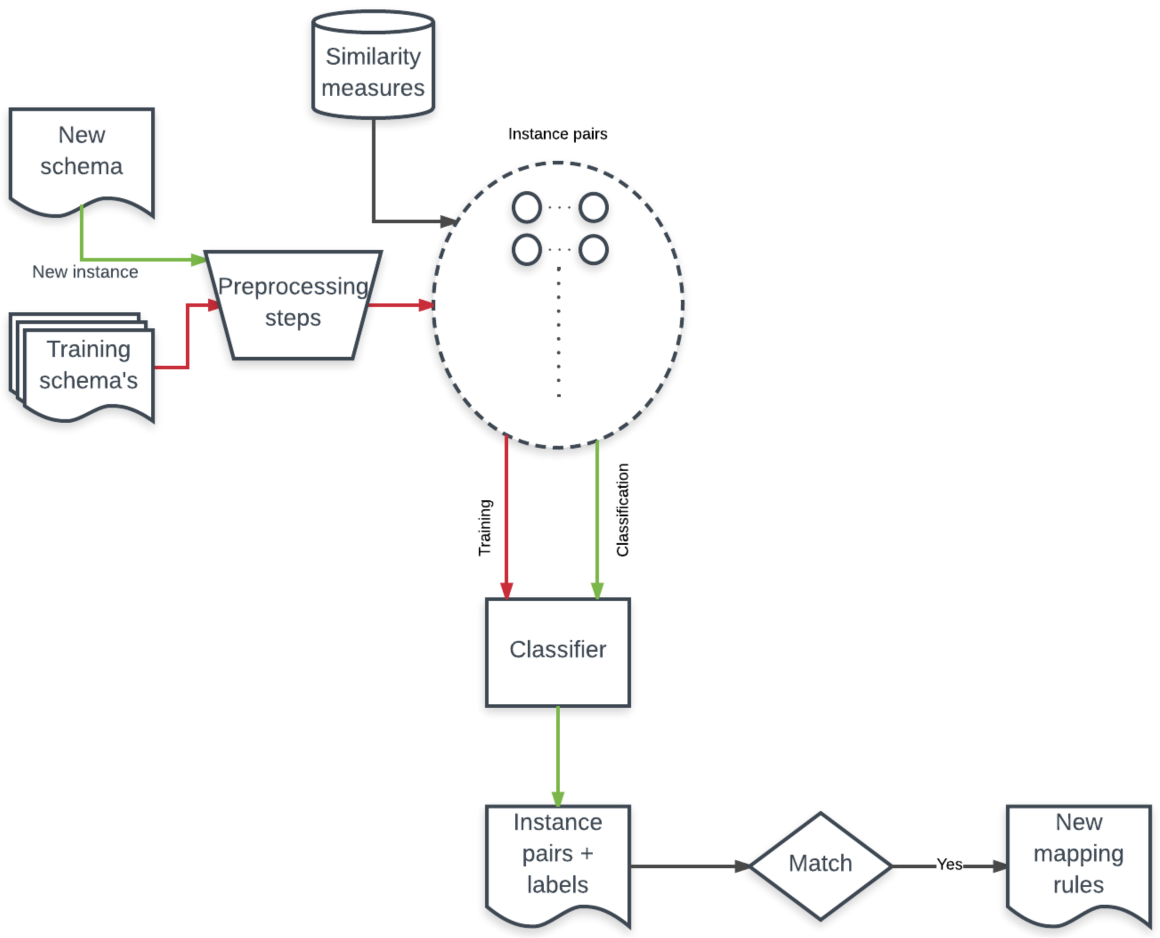


Figure 22: overview how data is prepared for machine learning solution

The first step is to extract several schemas to be used for training purposes (red coloured lines; described in section 4.3). For each schema pre-processing steps are applied. At this stage pre-processing only lower cases the string name. A list of all possible instance pairs is then created. If the amount of element in schema A is $|A|$ and for schema B is $|B|$ then the total amount of instance pairs is $|A| * |B|$. For each instance similarity measures are appended. Since we extract schemas which have already been mapped we label each instance pair whether it's mapped (1) or not (0). This comprises the training set which is used to train a classifier.

When a user invokes the auto mapper the same steps are followed. The green coloured lines indicate these steps. However, since the class label is now unknown we use the classifier to predict the label. The user is then presented with new mappings which were predicted by the classifier.

4.2 Machine learning platform

As mentioned previously we opted to use Azure Machine Learning studio. The ease of use complemented with the graphical 'drag and drop' user interface allows to build machine learning experiment which fit the needs at CAPE. Initially Weka was taken into consideration but the ease of implementing Azure ML into Mendix compared to Weka favoured our decision to use Azure ML. Several classifiers are available in Azure ML which can be grouped in the following categories (Microsoft, 2017b):

- Regression: used when trying to predict numerical values (e.g. stock price).
- Anomaly detection: used for finding unusual data patterns (e.g. detecting credit card fraud).

- Clustering: this is an unsupervised learning method which means no class label is used. Data with similar patterns are clustered together to discover structures in data. Can be used to detect which combination of groceries are often purchased at a supermarket.
- Two-class classification (or binary classification): used when trying to predict two categories. This applies to our case since we are trying to predict whether an instance pair is mapped or not.
- Multi-class classification: an extension of two-class classification, however, in this case there are at least three categories which the classifiers is trying to predict.

Our interest goes to two-class classification. However, we are curious to find out whether an anomaly detection model would give good results. At the time of writing this thesis several two-class classifiers are available which we list in table 7. To ease the process of selecting a classifier Microsoft indicated what unique features each classifier has. A brief comparison was made to Amazon Web Services (AWS) which only provides one classifier, namely logistic regression (Amazon Web Services, n.d.). For this reason, we did not dive deeper in using AWS.

Table 7: overview of classifiers available in Azure (Microsoft, 2017b)

Classifier	Supports >100 features	Linear model	Training time	Accuracy
Support vector machine	●	●		
Locally deep support vector machine	●			
Averaged perceptron		●	Fast	
Logistic regression		●	Fast	
Bayes point machine		●	Fast	
Decision forest			Fast	●
Boosted decision tree			Fast	●
Decision jungle				●
Neural network			Long	●

Each classifier has several characteristics. Both models of the support vector machines (SVM) scale very well to handle more than one hundred features. We only have three features, so this is not a relevant criterion (for an explanation of what features are we refer to section 2.2). A linear model assumes the problem space can be separated by a straight line (better known as a linear hyperplane). An illustration of a linear hyperplane is presented in figure 23.

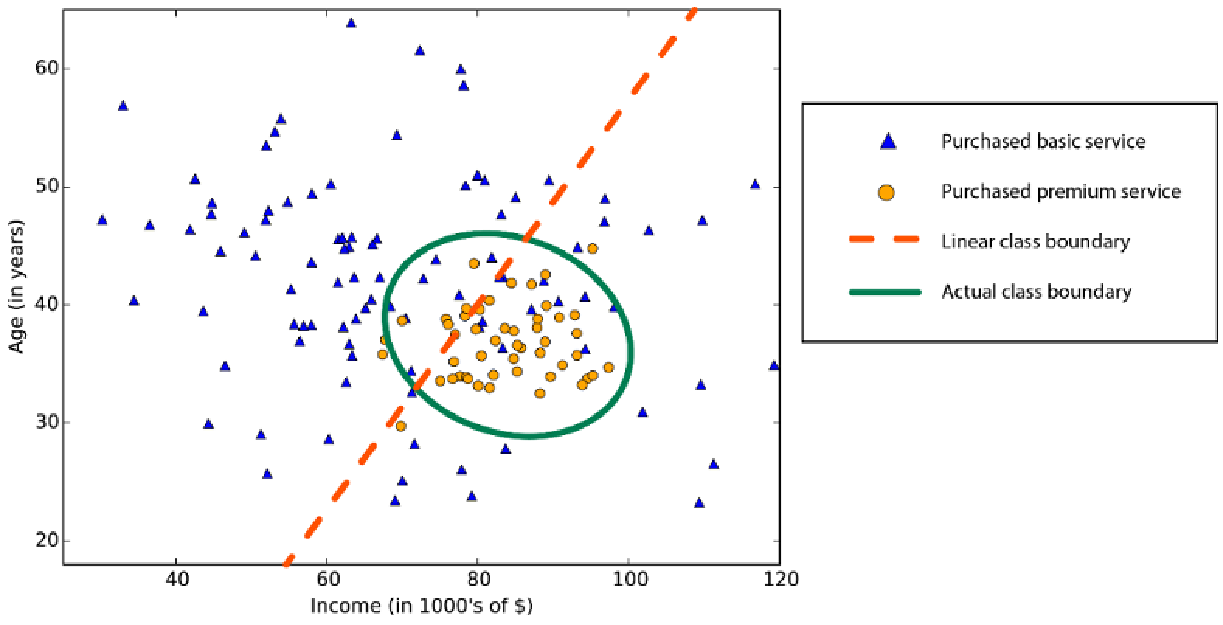


Figure 23: comparing a linear vs. a non-linear class boundary (Microsoft, 2017a)

The figure describes a problem by using two features, namely age and income. Looking at the figure we see a classifier which constructs a linear class boundary misclassifies a relatively high number of data points. A classifier which constructs a non-linear class boundary, in the figure denoted as the actual class boundary, would give better results because it captures almost all the data point belonging to the 'purchased premium service' class. Using a classifier which produces a linear boundary could be outperformed by a non-linear model.

The last two criteria found in table 7 are training time and accuracy. Most classifiers can be trained quickly, however, only for the neural network it is indicated training time could be long. However, as we can see this is met with a generally high accuracy. In section 4.5 we discuss the outcome of the classifiers.

4.3 Training set

Before a classifier can be trained, a training set must be created. This was done by extracted existing mapping results which were performed at BAM, Neele-Vat and PostNL. Since all these mappings have been put in production we consider them to be a reliable, expert verified, set. These results have undergone the pre-processing steps outlined in section 4.2. This yielded a training set with 2.354.498 records of which 19.486 were mapped. An example of this data is shown in table 8.

Table 8: Excerpt from the training set

AttributeIn	**AttributeOut**	**Type-Match**	**Leven-shtein**	**Jaro-Winkler**	**Class**
costcentrereceived	costcentercode	1	6	0,9	1
naam1	shipper	1	7	0	1
factuurspecificatieaankooppijsvaluta	lengte	0	33	0,42	0
documentid	buildingnumberdesignation	1	18	0,52	0
bevestigingid	results_of_audit	1	13	0,5	0
adrescontactfaxnummer	zeevervuilingindicator	0	20	0,46	0
servicelevel	adrestype	0	10	0,51	0

cityname	starttimedropoff	0	14	0,58	0
brutogewicht	douanedocumentnummer	0	17	0,49	0
conf_oms	conf_oms	1	0	1	1
height	gevaarlijkegoederenrailindicator	0	28	0,42	0
innerverpakking	douanedocumentcode	1	16	0,47	0

As can be seen the training set contains seven columns. The first is the AttributeIn which is the name of the attribute of the source schema. The AttributeOut columns contains an attribute name on the destination schema. Type match indicates whether the attributes are of the same type (i.e. string to string). Levenshtein gives the Levenshtein edit distance and JaroWinkler is the similarity score. Finally, the class indicates whether the combination of attributes is mapped (1) or not (0).

4.4 Anomaly detection model performance

Azure ML provides two anomaly detection classifiers, namely the one-class SVM and a PCA-Based anomaly detection classifier. We are interested in the initial performance before we dive deeper in the exact working and differences between the two classifiers. The training set outlined in section 4.3 was used. Unfortunately, these results were appalling with a respective F-measure of 0,4% and 1,6%. We therefore decided not to take a closer look at anomaly detection possibilities and move on to the two-class classifiers in the next section.

4.5 Two-class classifier performance

Most of the classifiers which we listed in section 4.2 were constructed to compare performance. We ddin't use the Bayes Point Machine because it threw an internal error message and as such couldn't be used. For each classifier we wrote down precision, recall and F-measure. Testing was done by performing a stratified split for which 70% of data was used for training and the remaining 30% was used to test the trained classifier. The results are presented in figure 24.

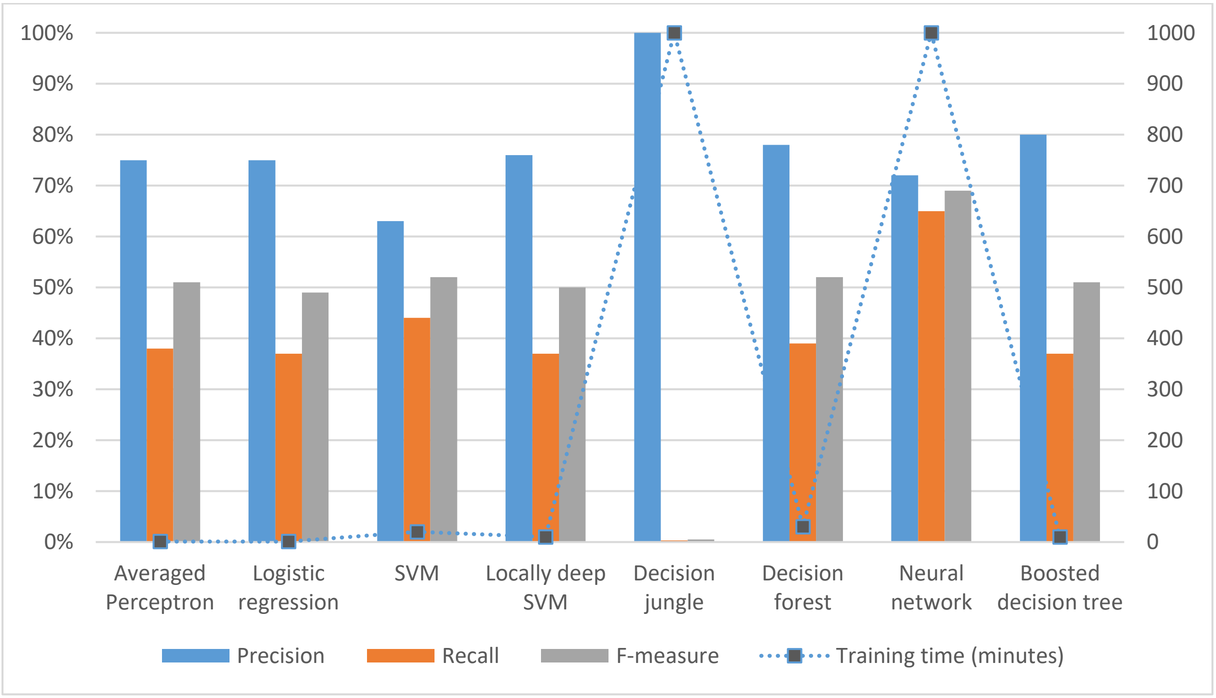


Figure 24: comparing classifier performance

Analysing the results, we see the best results (in terms of F-measure) came from the neural network. It outperformed all other classifiers with an F-measure of 69%. Other classifiers scored around 50%, except the decision jungle which scored appalling. However, as can be seen, there is a catch which is the long training time. The boosted decision tree gave good results in terms of precision (80%) but had a lower recall (37%). For each trained classifier, a scored column is appended. This is an indication how 'certain' the classifier is of its results. Altering the threshold for which the classifier labels the class as either mapped or not mapped has an impact on the precision and recall. By default, the threshold is set to 0,5. In other words, if the scored probability is greater than or equal to 0,5 the assigned label is 'mapped'. Else the label would be 'not mapped'. Azure ML generates a graph which compares precision and recall of two classifiers by altering the scored threshold. This graph is presented in figure 25 for the neural network and boosted decision tree.

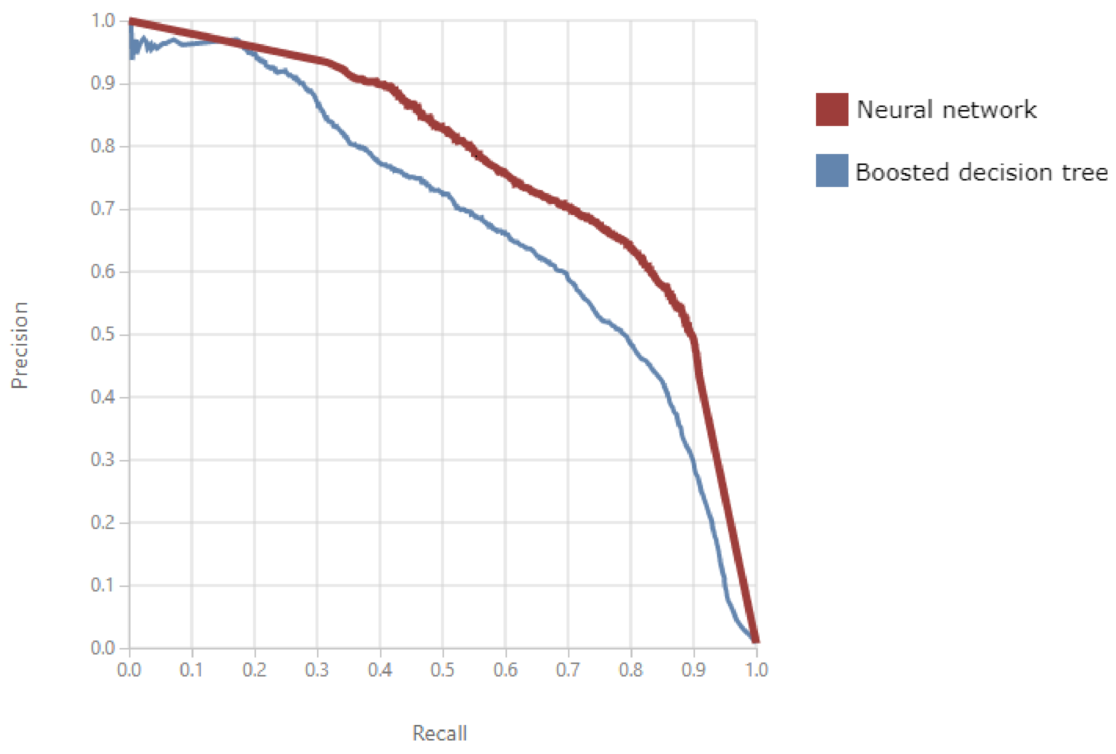
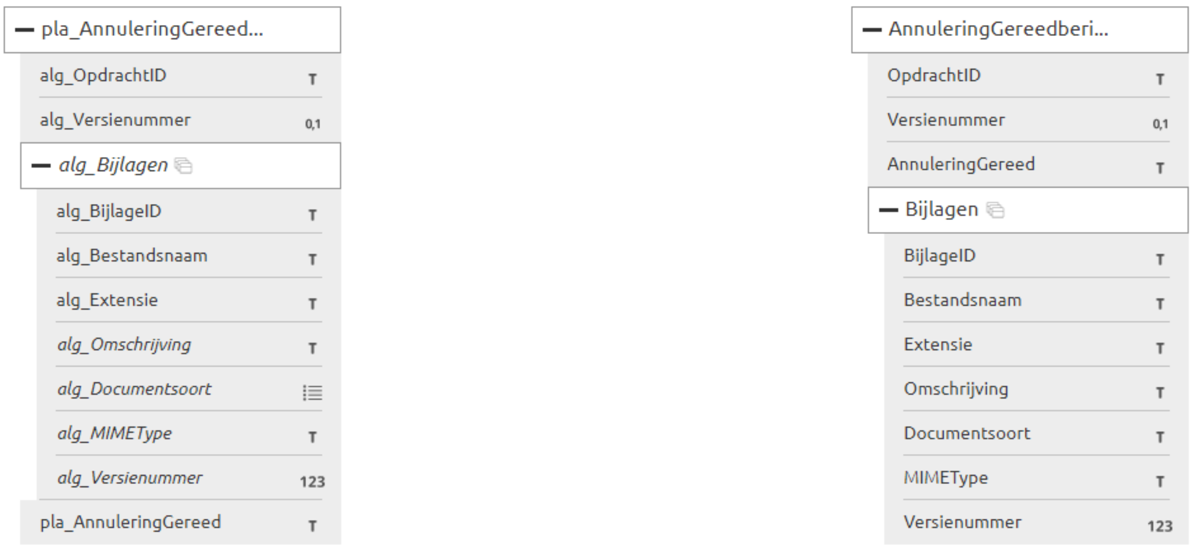


Figure 25: Comparing precision and recall of the neural network to the boosted decision tree by varying the scored threshold

Based on this result we see the neural network outperforms the boosted decision tree. As mentioned earlier the neural network took more time to train. Since cost are incurred when training a classifier this means the neural network is more expensive to train. At the time of writing the cost of training a classifier equate to €0,85 per hour. Therefore, training the classifier would cost about €15 which is still acceptable. Retraining the classifier is not needed on a daily basis, but we estimate it could be needed once every quarter. However, this is ball-park estimate and further testing is needed to determine the frequency of retraining. Based on the results it is decided to use the neural network.

4.6 Prototype

This section shows how we created the prototype. It is build using Mendix which is the same platform in which eMagiz is developed. The working of the AutoMapper is illustrated using the example in figure 26.



- pla_AnnuleringGereed...
  - alg_OpdrachtID T
  - alg_Versienummer 0,1
  - *alg_Bijlagen*
    - alg_BijlageID T
    - alg_Bestandsnaam T
    - alg_Extensie T
    - *alg_Omschrijving* T
    - *alg_Documentsoort*
    - *alg_MIMEType* T
    - *alg_Versienummer* 123
  - pla_AnnuleringGereed T

- AnnuleringGereedberi...
  - OpdrachtID T
  - Versienummer 0,1
  - AnnuleringGereed T
  - Bijlagen
    - BijlageID T
    - Bestandsnaam T
    - Extensie T
    - Omschrijving T
    - Documentsoort T
    - MIMEType T
    - Versienummer 123

Figure 26: example of two schema's which have to be mapped

The first step when creating a Mendix app is to create the domain model. For the AutoMapper app this is shown in figure 27.

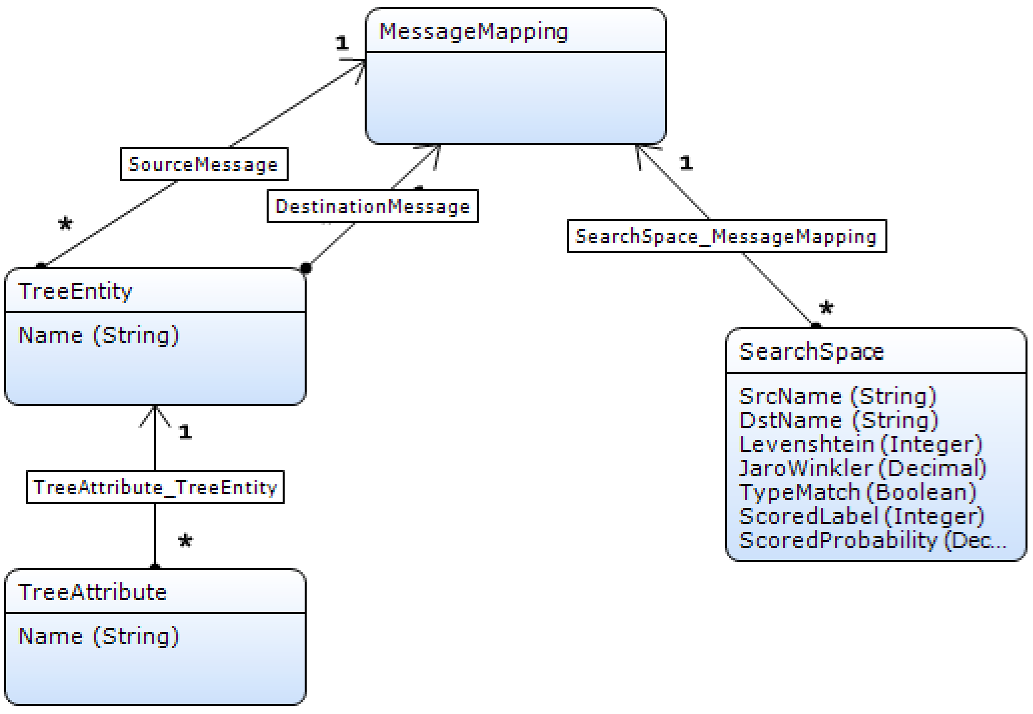


Figure 27: domain model in the AutoMapper app

At the top of the domain model is the message mapping. A message mapping has a source and destination message. This consists of an entity, and an entity contains attributes. When the search space is created this is linked to the message mapping. This data is stored in the SearchSpace entity. This includes the Levenshtein edit distance, the JaroWinkler similarity measure and whether the types of the source and destination match. The last two columns are the result of the machine learning classifier. First is the scored label (i.e. match or no match) and the probability of the score.

The AutoMapper application contains a webservice which is called by eMagiz when the user requests a mapping. This action sends the two schemas the user is working on to the AutoMapper application. In this first iteration no pre-processing steps occur, other than lowering upper cases. Next, the search space is created and the Levenshtein and JaroWInkler similarity measures are added. The complete search space is send to Azure ML. It is processed and send back including the class label and scored probability.

After the AutoMapper app receives the results a choice can be made which candidate mappings to return to eMagiz. During the evaluation two methods are used, namely the result of the neural network (those with a class label of mapped) and a heuristic approach. In the latter case, the set of candidate mappings consist of those with a JaroWinkler similarity score of at least 0,8.

The effect of the AutoMapper on the schema provided in figure 26 is shown in figure 28. For this scenario, the heuristic approach was chosen since the neural network did not provide a result. The reason for this is discussed in the next section.

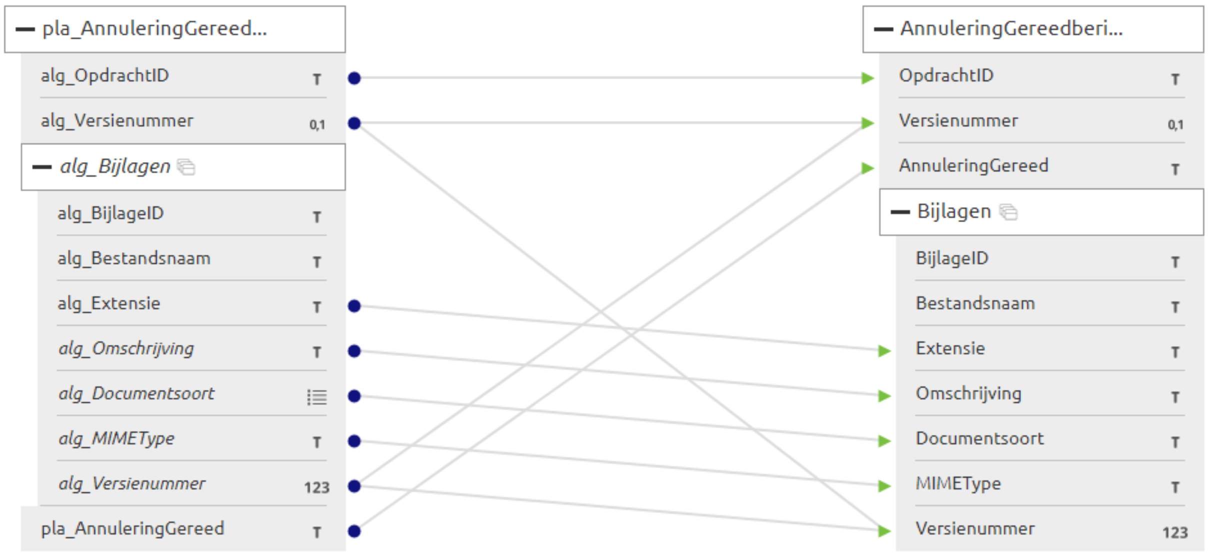


Figure 28: AutoMapper suggestion based on a JaroWinkler score

As can be seen ten candidate mappings are suggested. Next, the user completes the mapping. For this case, the user had to remove two candidate mappings and add two mappings. Please note this means adding two attribute mappings. The first iteration of our prototype is only capable of handling attributes; entities are ignored. The result is displayed in figure 29.

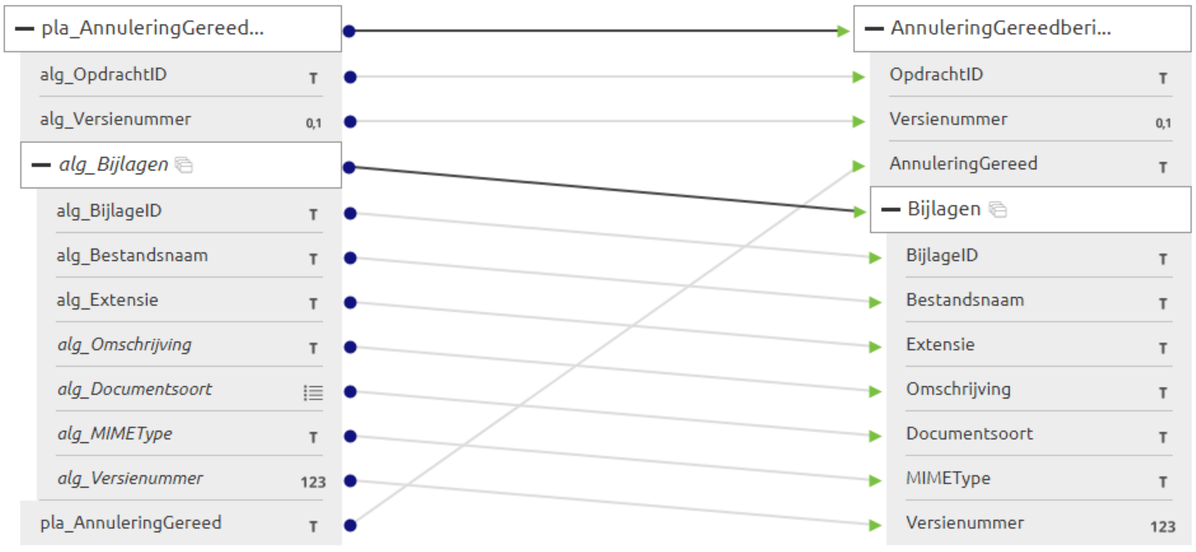


Figure 29: the final result

5 Prototype evaluation

In this section, the prototype is evaluated. This starts by defining the evaluation criteria in section 5.1. Section 5.2 describes the results and section 5.3 discusses improvement suggestions.

5.1 Evaluation criteria

Software development requires a measurement mechanism for feedback and evaluation (Basili, Caldiera, & Rombach, 1994). For schema matching two major challenges exist: achieving both good effectiveness and efficiency (Rahm, 2011). *Effectiveness* (match quality) requires the correct and complete identification of mappings. *Efficiency* is the second challenge which is about the time it takes to perform a mapping scenario. The IA driven approach proposed in this thesis however stresses the benefit of including the user in the process and therefore also focusses on efficiency. At present, evaluation of such tools is lacking (Falconer & Noy, 2011).

Both are required since measuring productivity cannot be viewed in isolation without any accompanying assessment of product quality (Fenton & Pfleeger, 1998). To construct the criteria used to evaluate the tool the goal, question, metric (QGM) method is used (Basili et al., 1994). In our case, we define two goals, namely to evaluate effectiveness and efficiency. This leads to the following two main questions:

- What is the effectiveness of the prototype?
- What is the efficiency of the prototype?

5.1.1 Measuring effectiveness

Measuring effectiveness tells us something about the quality of the result. For schema matching this is measures in terms of precision and recall (Falconer & Noy, 2011). From these two figures the harmonic mean, or the F-measure, is calculated. Most tools are evaluated based on these metrics and, as a result, are only judged on effectiveness (Duchateau & Bellahsene, 2016).

5.1.2 Measuring efficiency

Evaluating schema matchers based on efficiency is new and hardly done (Falconer & Noy, 2011). We base efficiency on time and define the following question:

- What is the time needed to perform a matching scenario?

The total time it takes to complete a scenario is ideally split in four metrics:

1. Pre-processing: time it takes a to conduct the pre-processing task.
2. Time for the software agents to create candidate mappings
3. Time for a user to remove the false positives
4. Time for a user to add the missing mappings (i.e. false negatives)

However, in practice we noticed the third and fourth intertwine when a user completes a mapping scenario and therefore are difficult to measure independently. As such, we decided to merge these two metrics to the time it takes a user to complete the mapping.

Using only total time makes it impossible to compare different scenario's. To compare the efficiency of different scenario's the time improvement is calculated. Let t_1 be the total time of the base line and t_2 the total of a different method. The time improvement (in percent) is then calculated using the following formula:

$$Improvement\ w.r.t.baseline\ (\%) = \frac{t_2}{t_1} * 100\%$$

If performance were to degrade the outcome is greater than 100%.

The evaluation criteria are summarized in figure 30.

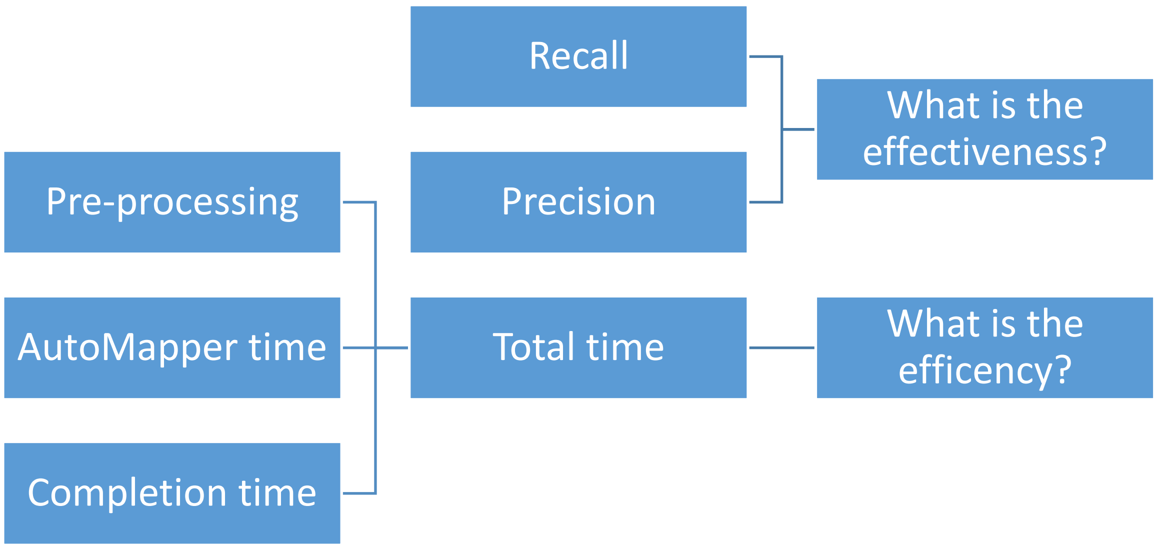


Figure 30: evaluation criteria

5.2 Evaluation

The evaluation was conducted using four different schemas. The size of the schemas is presented in table 9.

Table 9: Schema sizes

Scenario	Schema	Attributes	Entities
1	Source	7	3
	Destination	7	3
2	Source	10	2
	Destination	10	2
3	Source	40	6
	Destination	42	9
4	Source	136	44
	Destination	138	44

The first scenario is relatively small. The schemas have different languages, the source schema language is Dutch whereas the destination schema language is English. The second scenario is entirely in Dutch where names are mostly similar. In the source schema name elements have the tag "*alg_*" as a prefix. For example, a mapping should be created between *alg_bijlageid* and *bijlageid*. One can see the prefix is what makes them different, otherwise the strings are mostly similar. The third scenario is somewhat larger and is a schema which contains order information. The final scenario is a mapping for a large work order schema. Names of corresponding matches are highly similar.

For each mapping thee user groups were selected:

- Group 1: performing each scenario manually, i.e. without using a software agent

- Group 2: performing each scenario using the trained neural network
- Group 3: performing each scenario using JaroWinkler heuristic

The first group provides the baseline measurement. The second group indicates the result of the neural network. The third group only uses the JaroWinkler heuristic metric. This scenario is included for situations where the source and destination schemas are highly similar. The experiment for the second and third user group was performed on a local machine.

Each user group contains one user and a different user was selected for each scenario to avoid a time benefit because of a learning effect.

5.2.1 Scenario 1

The first scenario is relatively small schema in which many names of the source and destination schema were similar. The first user took 36 second to complete the scenario. Involving the neural network took 9 seconds but did not yield any result. The total was therefore 45 seconds. The third user had to wait two seconds for the AutoMapper to complete and it took him 28 seconds to complete the mapping thus the total time was 30 seconds.

5.2.2 Scenario 2

The second scenario is a relatively small schema as well and is the schema used to demonstrate the working of the AutoMapper in section 4. It took the first user 28 seconds to complete the mapping. As with the first scenario the neural network did not yield any results.

The third user achieved good results. It took the AutoMapper two seconds to complete and this yielded 10 candidate mappings. Two were false positives and two had to be added thus precision and recall were both 80%. I took the user an additional 10 seconds to complete the scenario bringing the total time to 12 seconds.

5.2.3 Scenario 3

The third scenario contains a schema of moderate size. The first used needed 208 seconds or 3 minutes and 28 seconds to complete the scenario. The second user first had to wait 17 seconds for the AutoMapper to create the candidate mappings (34 in total). It then took the user an additional 176 seconds to complete the mapping. In the process 13 false positives were removed and 19 mappings were added (out of a total of 40). This yielded a precision of 62% and a recall of 53%.

For the third user, it took the AutoMapper five seconds to generate results. This yielded 45 candidate mappings of which 24 were correct and 16 had to be added. Precision is 53% and recall 60%. Completing the mapping took more than five minutes or 311 seconds. This is much higher than the baseline. When observing the user this is probably due to the slow response of the system. Removing a false positive took considerable time thus the user often had to wait for the system to complete. In total 21 candidate mappings had to be removed so waiting took much time.

5.2.4 Scenario 4

The fourth scenario contains a large schema. The first user needed 28 minutes and 34 seconds to complete the scenario. When the second user wanted to run the AutoMapper an error message was shown so no candidate mappings were generated. We suspect this is due to the large memory foot print (Rahm, 2011). The source schema contains 136 attributes and destination schema 138 attributes creating a search space containing 136 * 138 = 18.768 possible mappings.

The third user had to wait 3 minutes and 34 seconds for the AutoMapper to generate results which yielded 993 candidate mappings. This time, the AutoMapper was capable of delivering a result because the search space was reduced at runtime (Rahm, 2011) by not adding a candidate mapping if the JaroWinkler score was less than 0,8.

After completion, the user started removing false positives. Unfortunately, the system became unresponsive as soon as the user wanted to do so. Each false positive which had to be removed incurred a waiting time of 20 seconds. As such the experiment was aborted because a quick scan by the user indicated many candidate mappings had to be removed. It was estimated this would take considerable time. The user also indicated he would discard the results were this to happen in his daily job routine.

5.2.5 Discussion of results

This section discusses the results obtained above. First, we present the results per scenario. For the first and second scenario, the neural network (second user group) did not yield any results. Therefore, these results have been omitted from the overview. For these experiments, the precision as well as recall would be 0%. The heuristic (third user group) did provide results so these are presented. For the third scenario both the neural network and heuristic provided results. The fourth scenario did not obtain any result and is therefore omitted as well. The obtained results are plotted in figure 31.

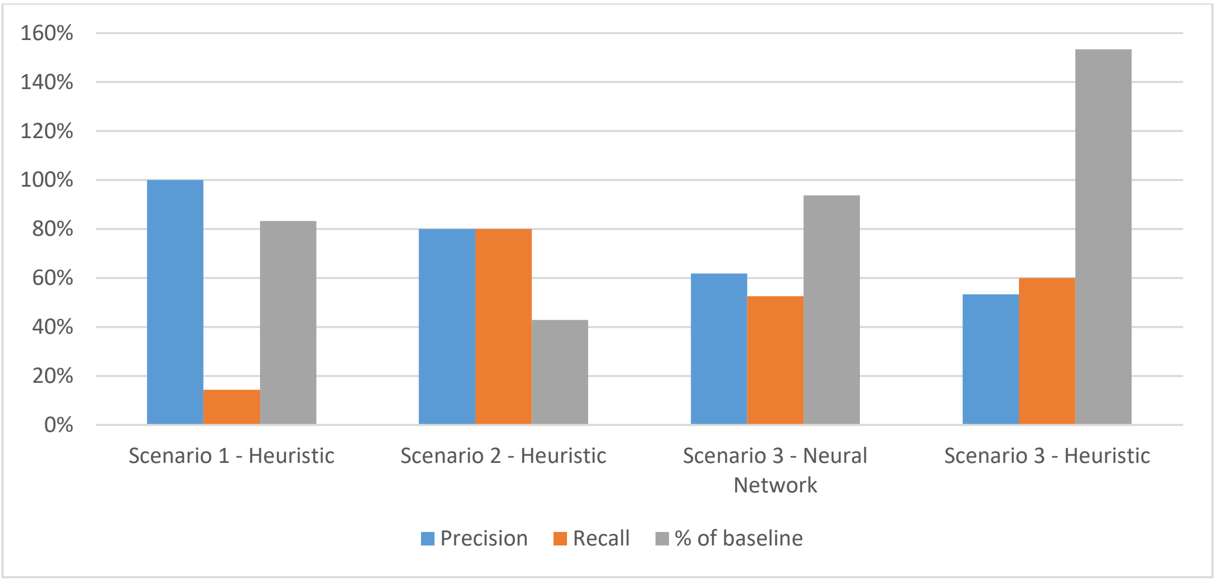


Figure 31: effectiveness and efficiency of scenarios

When analysing the graph, a correlation between precision, recall and the time improvement could be suspected. When both precision and recall are high, a high time reduction is seen. This makes sense, because a high value for effectiveness indicates little effort is needed from the user to refine the mapping scenario. This then leads to a reduction in time. However, this raises the question which method would work well on a given situation and wouldn't on another.

First, the neural network approach is discussed. We suspect one of the reasons this approach was not able to provide results for the first and second scenario can be contributed to the language. Since both scenarios both contained at least one schema in Dutch no candidate mappings were resolved. We analysed the training data and found the vast majority of trainings examples are English. This contributes to our assumption the language is a cause for the poor performance. The neural network managed to provide somewhat satisfactory results in the third scenario. However, only a slight time improvement was achieved. When precision and recall would be higher we expect a further time improvement can be achieved. A possible explanation for the lower scores on this scenario could be contributed to the huge training set used. Perhaps, if we training examples were chosen from schemas similar to the ones used in the scenario a better score would be achieved.

The second approach, using the JaroWinkler similarity as heuristic, yielded mixed results. In the first scenario only one candidate mapping was found. Due to the fact there was a difference in language (Dutch to English) the similarity between two words was generally low. We note similarity is computed by looking at the letters of a word; the meaning of a word is not taken into account.

The second scenario showed good results. The reason two false positives were discovered lies in the fact the entity name is not taken into account. We reprint the AutoMapper result for the second scenario in figure 32 below. When looking at the source schema (left side) the attribute name *'alg_Versienummer'* is included twice. Once under the entity *'pla_Annuleringgereed'* and once under *´alg_Bijlagen'*. The same applies for the destination schema where the attribute 'Versienummer' is seen twice. Since the entity name is not taken into account the AutoMapper cannot distinguish the two.

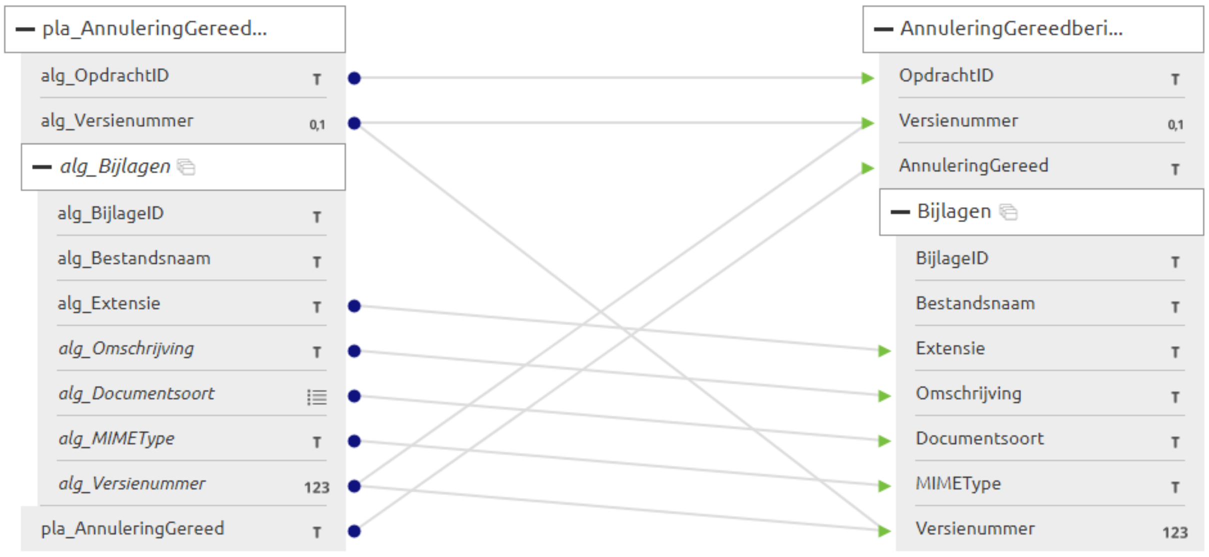


Figure 32: AutoMapper result for the second scenario

The fact the entity name is not taken into account is also one of the reasons the third scenario contained a number of false positive candidate mappings. What's interesting to see is the time it took the user to complete the mapping. When the user removed a false positive result, he had to wait a couple seconds before the candidate mapping was removed. Since 22 candidate mappings had to be removed the user incurred a long waiting time.

Next, we take a closer look at the efficiency of our tested method. This time we also include the efficiency score for the neural network for the first and second scenario. This value is computed by adding the time it took invoking the AutoMapper plus the baseline time. The latter is chosen since the AutoMapper did not provide and results and the mapping had to be completed as if the AutoMapper never ran (i.e. the baseline situation). Figure 33 compares the efficiency of the scenarios.

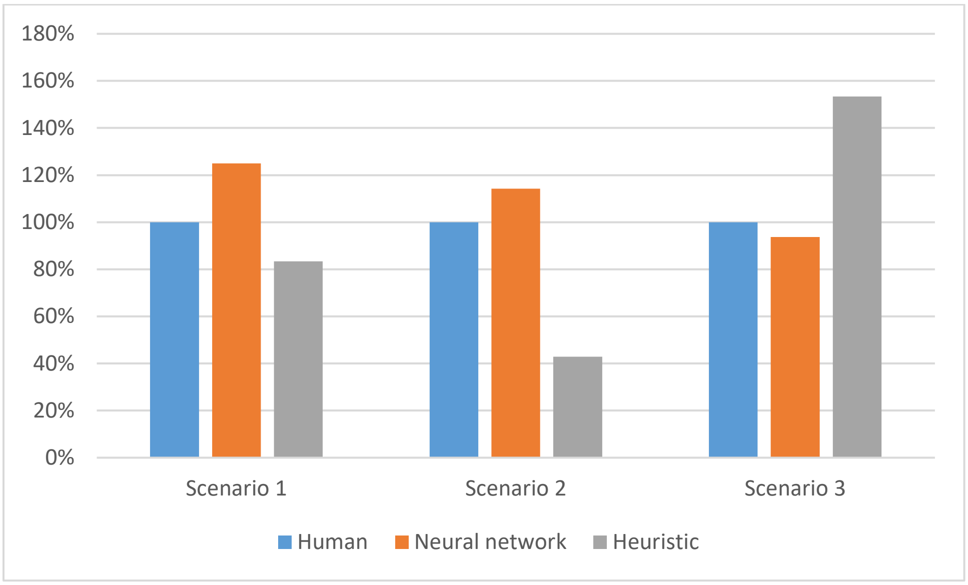


Figure 33: comparing efficiency for the three scenarios

The performance of the human is the baseline and is therefore set to 100%. In all scenarios, an improvement in performance was achieved. However, which method works best depends on the scenario. From the results above we see a scenario in which attribute names are highly similar and overlap a heuristic approach works best. With a scenario containing English attribute names, which do not overlap very well, the neural network approach works better.

Another important topic is scalability. Our fourth scenario was too big to be properly processed by the neural network AutoMapper. The heuristic approach did yield result but contained too many false positives for which the user indicated the result was unusable. In section 4.1 we discussed the size of the search space was the cartesian product of the number of attributes in the source schema times the number of attributes in the destination schema. Therefore, when the number of attributes in both schemas is doubled, the size of the search space is quadrupled. Rahm (2011) suggest two approaches of search space reduction: early pruning of dissimilar pairs and partition-based matching. In the first option a possible mapping is discarded from the search space if the similarity is low. In the second approach, a divide-and-conquer approach is used. The schema is divided section. For example, the search space only contains attribute pairs if the parent entities have been mapped. These options are not included in this version of the prototype, but are worth investigating in future work.

5.3 Improvements

Based on the results above we conclude further optimizations are needed. Some have been suggested already, but for completeness we repeat them below.

5.3.1 Entity mapping

At the moment, entities are not taken into account. Mapping entities not only aids the completeness of results, but may also improve results. In the previous section we used the example of the attribute ‘postal code’. When a destination schema has the attribute ‘postal code’ in both the entity ‘delivery address’ and ‘pickup address’ the entity name provides additional information. In this section, we discussed the impact of this issue using the second scenario.

5.3.2 Further pre-processing steps

We already noted the only pre-processing step taken is to lower case strings. However, we suspect further pre-processing is beneficial for results. This has also been suggested by literature (Sorrentino et al., 2010). First, we would start using a translation service and tokenize strings. Next, we would look at possibilities to expand compound nouns and perform domain specific dictionary lookups. Which of these steps improve, and by how much, performance should be part of future research. In this research we did not manage to address these issues due to time constraints.

5.3.3 Hybrid matchers

Combining multiple approaches in one matching solution, which we earlier referred to as hybrid matchers, could further improve results which this evaluation shows and has also been found in literature (Sutanta, Wardoyo, Mustofa, & Winarko, 2016).

5.3.4 Further graphical support

At present, candidate mappings are presented to the user as if they were drawn by the user. Also, we do not use the scored probability value generated. In a graphical user interface (GUI) candidate mappings can be presented to the user by giving them an option to adjust the threshold for various metrics. The use of a GUI in schema matching is growing (Sutanta et al., 2016). For example, if Levenshtein, JaroWinkler and a neural network are used the user should be presented with three different sliders. This allows the user to adjust the various thresholds and directly see the impact of these changes. The user is then able to select the best criteria for a given scenario. When a user is satisfied with a set of candidate mappings he accepts them and goes to the stage of refinement.

6 Conclusion

This section concludes the research and we summarize each research question.

6.1 Intelligence Amplification

First, we answer the research questions related to IA.

1A. What is the current state-of-art in Intelligence Amplification research?

In our literature review we found Intelligence Amplification is a rather new and undefined field in scientific literatur. We therefore focussed on a definition and the effect of close collaboration.

1B. Which definitions exist in literature?

We found several definitions in literature. Some definitions where directly described as a definition of intelligence amplification, others discussed the same topic but under a different name. The structure of the definitions was critically assessed and five unique features were extracted, namely: artificial intelligence, decision making, problem solving, partnership/collaboration and empowering human. Using these features, we created our own definition which we used in this thesis: *"Intelligence Amplification focusses on a close collaboration, with complementary contributions, between human and machine to empower humans in the decision-making process.".*

1C. How does IA affect the delegation of tasks between human and machine?

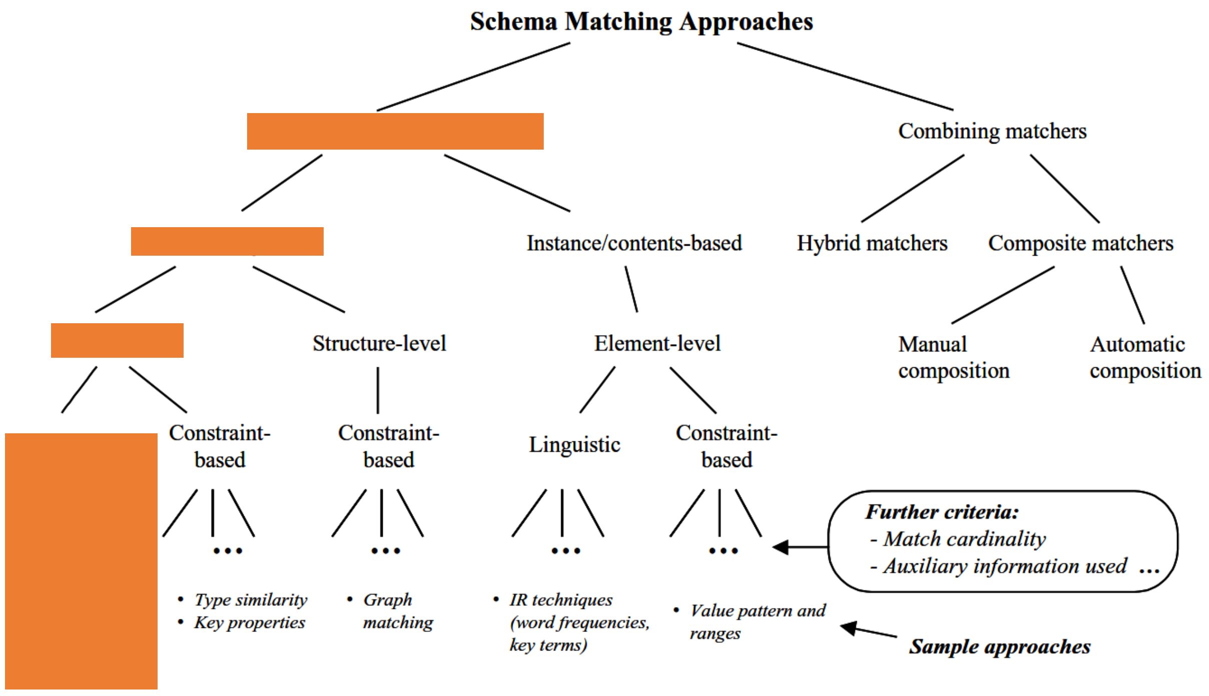
In our definition, we argue IA focusses on a close collaboration between human and machine. This raises the question how this collaboration should be shaped. Before we started looking at this question we took a closer look at task delegation. This took us back to the 50s with what's known as the HABA-MABA (humans are better at – computers are better at) list introduced by Paul Fitts. There has been critique to this list and updates have been proposed. We liked the idea of affordances proposed by Crouser & Chang (2012). An affordance is defined as action possibility that is readily perceivable by a human operator. They exist between human and machine and cannot be seen separate from that relationship. A model how to share information between computer and machines is given by Dibona et al. (2016) which they call the Proactive Autonomy Collaboration Toolkit (PACT).

6.2 Schema matching

Next, we answer the questions related to schema matching.

2A. Which solutions for schema matching have been proposed in literature?

At first, we included a taxonomy which describes schema matching approaches. We reprint this taxonomy below and indicate where our prototype fits in.



Over the years many approaches have been defined. More recent and successful approaches we discussed are COMA and YAM. The first is based on heuristics. The latter is a schema matching factory which produces a machine learning classifier each time two schemas are matched. Though this yields good results it takes more time to perform the schema matching task (Rodrigues et al., 2015).

2B. WHICH PRE-PROCESSING STEPS ARE NEEDED TO PREPARE DATA FOR SCHEMA MATCHING?

The framework by Rahm & Bernstein (2001) includes several suggestions for data pre-processing. Name similarity is the first we discuss, which is a value to express how similar two strings are. These are also known as edit distances and we have noted a number. In our prototype, we used Levenshtein and JaroWinkler. Other pre-processing steps could include the use of a dictionary or thesaurus.

2C. WHICH ISSUES HAS LITERATURE IDENTIFIED IN DATABASE SCHEMA MATCHING?

The main issue with schema matching is that it cannot be fully automated. Including the user in the process therefore is a necessity. Recent advancement indicates the limit has been reached what can be done in terms of reaching precision and recall. At this stage research is needed in ways the user can effectively incorporated in the process.

2D. WHICH MACHINE LEARNINGS ALGORITHMS ARE AVAILABLE?

Many classifiers exist for the task of machine learning. We grouped them together and discussed some older, easier to understand classifiers, as well as more recent and more complex classifiers.

6.3 General architecture

We now combine the knowledge of the sections above by answering the research questions related to our IA driven approach for schema matching.

3A. HOW TO DESIGN AN IA DRIVEN APPROACH AND ARCHITECTURE FOR SCHEMA MATCHING?

Using our definition and the PACT framework we defined an architecture which focusses on a close collaboration between human and machine for schema matching. The user is involved in pre-processing steps, as well as at the end for refinement of the results. The user can opt to reinvoke the software agent many times in the matching process. Important as well is the feedback loop to the machine which it should use to further improve upon its mistakes.

3B. IN WHICH STAGES DO WE INCLUDE THE USER IN THE APPROACH?

As we noted above we now included the user at the start in the pre-processing stage as well as the matching phase.

6.4 Prototype and evaluation

We use our initial approach and create a prototype which we then evaluate. We answer the research questions related to this task below.

4A. WHICH PARTS OF THE ARCHITECTURE DO NEED TO BUILD A MINIMUM VIABLE PRODUCT?

When building our prototype, we first focussed on a solution which is able of automatically suggesting mappings. The involvement of the user only occurred at the end of the steps when the need was to refine the results. However, we did create a highly modular prototype which allows for further development.

4B. WHICH DATA IS AVAILABLE AT CAPE?

CAPE has a dataset containing many expert verified mappings. We extracted those from which have been implemented at BAM Infra, Neele-Vat and PostNL.

4C. WHICH MACHINE LEARNING ALGORITHM PRODUCES THE BEST RESULT?

For our prototype, we decided to use the Machine Learning platform offered by Microsoft for its ease of use and implementation. We tested the classifiers which are available and found we got the best result from the neural network.

4D. WHICH METRICS CAN WE USE TO MEASURE PERFORMANCE?

Schema matching performance is measured in effectiveness and efficiency. Effectiveness is the performance of a classifier and can be measured using precision and recall. Precision measures the quality of the results whereas recall tells us something about the completeness of the results.

In our evaluation, we added the time component. This is split in the time for pre-processing, time needed for the AutoMapper to generate a list of candidate mappings and time needed to complete the mapping scenario by remove false positives and adding false negatives. We used four different scenarios for testing, of which three we got a result. We found that one scenario, which yielded a high score on both precision and recall, yielded a clear performance improvement.

4E. WHICH IMPROVEMENTS TO THE INITIAL IMPLEMENTATION CAN BE MADE?

At this moment, many improvements can be made. In this research we mainly focussed on an initial prototype for an IA driven method for schema matching. However, many improvements are yet to be made. First off, we haven’t included the matching of entities in the document. Also, the use of a structural matching approach is not yet included.

Further improvements to the pre-processing steps need to be researched. Possibilities are to automatically tokenize words, translation to one language and extend abbreviations. Other improvements are the use of multiple machine learning stages. In the current prototype we used one ‘do it all’ classifier. However, perhaps the concept of stacking, which uses a base learner and multiple different classifiers, could be used to further improve results.

The last improvement focusses around presentation of the results and the interaction with the user.

6.5 Main research question

Finally, we answer our main research question which was as follows:

How can we combine the concept of Intelligence Amplification with database schema matching to create a reference architecture for IA driven schema matching?

Using our definition of Intelligence Amplification, we designed a reference architecture to perform schema matching. In other words, we want a solution which focusses on a close collaboration between human and machine to empower the human in the process of matching two schemas. We used the idea of a work product proposed by Dibona et al. (2016) which is a product both human and machine can work on. Next, the process of schema matching was broken down in two stages: pre-processing and matching.

In the pre-processing stage strings are cleaned. Often, string names in schemas contain abbreviations, compound nouns or two schemas with different languages are used. Which steps are taken will depend on each matching scenario and in the reference architecture we make an abstraction by indicating various steps can be implemented in a concrete architecture. This is done by a machine which starts the pre-processing task. In case the machine is uncertain about a result it consults the user. This process could be repeated for various pre-processing steps. Which steps are needed are indicated by the user.

After pre-processing the search space is created and similarity measures are added. These are then send to a software agent which makes a decision which candidate pairs are elected to be candidate mapping. The candidate mappings identified by the software agents are presented to the user. The user then has to remove false positives and add the remaining mappings. The user could also choose to invoke a different agent or have a selection of the matching scenario run by a different software agent. At the end, the result is fed back to the machine to update its policies.

For the steps above we created a reference architecture which can be used to build a system capable of schema matching with a close collaboration between human and machine. Since user involvement in the task of schema matching is rather new this reference architecture is a starting point for designing systems with user involvement.

6.6 Limitations

As it goes with scientific research there are always limitations and our research is no different. The reference architecture is based on a vision based coming from a business need (Cloutier et al., 2009). We tried to mine the essence of existing architectures, but these are scarce. The reference architecture is kept a high abstraction since the exact steps which need to be taken varies for each schema matching scenario.

The developed prototype which was subsequently developed has several limitations. Unfortunately, we weren’t able to include the user in the pre-processing stages due to time constraints. The results gained from the machine learning platform also require further optimization, though we do believe more attention in the pre-processing stage will be beneficial to the results. At the matching stage, we haven’t yet dived deeper in visualizing the results which is something identified as in need for further research.

6.7 Future work

Based on the limitations we make several suggestions for future work. In the pre-processing stage, further research is needed to investigate which pre-processing steps work well and how to efficiently employ the user in the process. In our prototype, we only lower the upper cases, but the use of for example thesauri could aid performance. This could be part of a future version.

In our approach, the software agent used to create a list of candidate mappings is either heuristic based or a machine learning classifier. However, using an active classifier, which is capable of asking the user more information at runtime, is interesting (Rodrigues et al., 2015). As noted, presenting the results and guiding the user with visualizations needs future work (Ivanova et al., 2015). At the moment candidate mappings are presented as if they were drawn by a human. A study which options work best would be beneficial.

Our reference architecture provides a starting set of guidelines for developing applications based on schema matching with user involvement. When it is used for new applications this should result in feedback to update the document.

6.8 Recommendations for CAPE

The prototype created should be seen as a minimum viable product. It takes the approach where we use one method for all different scenarios. However, we already found some steps work better for a given scenario than others. Therefore, we first recommend updating the prototype with options for the user to create candidate mappings based on a heuristic or a machine learning classifier. Also, the use of a classifier trained on a domain specific dataset could further improve results. Second, visualization options should be incorporated to let the user know which mappings where automatically generated. A user interface which is better able to show various results of different software agents needs development.

7 References

Amazon Web Services. (n.d.). Learning Algorithm. Retrieved June 20, 2017, from http://docs.aws.amazon.com/machine-learning/latest/dg/learning-algorithm.html

Angelov, S., Grefen, P., & Greefhorst, D. (2012). A framework for analysis and design of software reference architectures. *Information and Software Technology*, *54*(4), 417–431. https://doi.org/10.1016/j.infsof.2011.11.009

Assoudi, H., & Lounis, H. (2015). Coping with Uncertainty in Schema Matching: Bayesian Networks and Agent-Based Modeling Approach. In M. Benyoucef, M. Weiss, & H. Mili (Eds.) (Vol. 209, pp. 53–67). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-17957-5_4

Baker, R. S. (2016). Stupid Tutoring Systems, Intelligent Humans. *International Journal of Artificial Intelligence in Education*, *26*(2), 600–614. https://doi.org/10.1007/s40593-016-0105-0

Barca, J., & Li, R. (2006). Augmenting the Human Entity Through Man/Machine Collaboration. In *2006 IEEE International Conference on Computational Cybernetics* (pp. 1–6). IEEE. https://doi.org/10.1109/ICCCYB.2006.305689

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, *2*, 528–532. https://doi.org/10.1.1.104.8626

Breemen, A. J. J. van, Farkas, J. I., & Sarbo, J. J. (2011). Knowledge Representation as a Tool for Intelligence Augmentation. In *Computational Modeling and Simulation of Intellect* (pp. 321–341). IGI Global. https://doi.org/10.4018/978-1-60960-551-3.ch013

Casini, E., Depree, J., Suri, N., Bradshaw, J. M., & Nieten, T. (2015). Enhancing decision-making by leveraging human intervention in large-scale sensor networks. In *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision* (pp. 200–205). IEEE. https://doi.org/10.1109/COGSIMA.2015.7108198

Cerf, V. G. (2013). Augmented Intelligence. *IEEE Internet Computing*, *17*(5), 96–96. https://doi.org/10.1109/MIC.2013.90

Christen, P. (2006). A Comparison of Personal Name Matching: Techniques and Practical Issues. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)* (pp. 290–294). IEEE. https://doi.org/10.1109/ICDMW.2006.2

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., & Bone, M. (2009). The Concept of Reference Architectures. *Systems Engineering*, *14*(3), n/a-n/a. https://doi.org/10.1002/sys.20129

Cohen, W. W., Ravikumar, P., & Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. Retrieved from http://secondstring.sourceforge.net/doc/iiweb03.pdf

Crouser, R. J., & Chang, R. (2012). An Affordance-Based Framework for Human Computation and Human-Computer Collaboration. *IEEE Transactions on Visualization and Computer Graphics*, *18*(12), 2859–2868. https://doi.org/10.1109/TVCG.2012.195

Cummings, M. M. (2014). Man versus Machine or Man + Machine? *IEEE Intelligent Systems*, *29*(5), 62–69. https://doi.org/10.1109/MIS.2014.87

Dekker, S. W. a., & Woods, D. D. (2002). MABA-MABA or Abracadabra? Progress on Human-Automation Co-ordination. *Cognition, Technology & Work*, *4*(4), 240–244. https://doi.org/10.1007/s101110200022

DiBona, P., Shilliday, A., & Barry, K. (2016). Proactive human-computer collaboration for information discovery. In B. D. Broome, T. P. Hanratty, D. L. Hall, & J. Llinas (Eds.), *SPIE 9851, Next-Generation*

Analyst IV (Vol. 9851, p. 985102). https://doi.org/10.1117/12.2222805

Dobrkovic, A., Liu, L., Iacob, M.-E., & van Hillegersberg, J. (2016). Intelligence Amplification Framework for Enhancing Scheduling Processes (pp. 89–100). https://doi.org/10.1007/978-3-319-47955-2_8

Duchateau, F., & Bellahsene, Z. (2016). YAM: A Step Forward for Generating a Dedicated Schema Matcher. In A. Hameurlain, J. Küng, & R. Wagner (Eds.) (Vol. 10120, pp. 150–185). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49534-6_5

Falconer, S. M., & Noy, N. F. (2011). Interactive Techniques to Support Ontology Matching. In Z. Bellahsene, A. Bonifati, & E. Rahm (Eds.), *Schema Matching and Mapping* (pp. 29–51). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-16518-4_2

Fenton, N. E., & Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach* (2nd ed.). Boston, MA, USA: PWS Pub. Co.

Fischer, G. (1995). Rethinking and reinventing Artificial Intelligence from the perspective of human-centered computational artifacts. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 991, pp. 1–11). https://doi.org/10.1007/BFb0034793

Garcia, A. C. B. (2010). AGUIA: Agents Guidance for Intelligence Amplification in Goal Oriented Tasks. In *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 338–344). IEEE. https://doi.org/10.1109/3PGCIC.2010.56

Greef, T. de, Dongen, K. van, Grootjen, M., & Lindenberg, J. (2007). Augmenting Cognition: Reviewing the Symbiotic Relation Between Man and Machine. In *Augmented Cognition* (pp. 439–448). https://doi.org/10.1007/978-3-540-73216-7_51

Griffith, D., & Greitzer, F. L. (2007). Neo-Symbiosis: The Next Stage in the Evolution of Human Information Interaction. *International Journal of Cognitive Informatics and Natural Intelligence*, *1*(1), 39–52. https://doi.org/10.4018/jcini.2007010103

Herrera, F., Ventura, S., Bello, R., Cornelis, C., Zafra, A., Sánchez-Tarragó, D., & Vluymans, S. (2016). *Multiple Instance Learning*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-47759-6

Hoffman, R. R., Feltovich, P. J., Ford, K. M., Woods, D. D., Klein, G., & Feltovich, A. (2002). A rose by any other name...would probably be given an acronym. *IEEE Intelligent Systems and Their Applications*, *17*(4), 72–80. https://doi.org/10.1109/MIS.2002.1024755

Iacob, M.-E., Jonkers, H., Quartel, H., Franken, H., & Berg, H. van de. (2012). Delivering enterprise architecture with TOGAF and ArchiMate.

Ivanova, V., Lambrix, P., & Åberg, J. (2015). Requirements for and Evaluation of User Support for Large-Scale Ontology Alignment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 9088, pp. 3–20). https://doi.org/10.1007/978-3-319-18818-8_1

Jacucci, G., Spagnolli, A., Freeman, J., & Gamberini, L. (2014). Symbiotic Interaction: A Critical Definition and Comparison to other Human-Computer Paradigms. In G. Jacucci, L. Gamberini, J. Freeman, & A. Spagnolli (Eds.), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8820, pp. 3–20). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-13500-7_1

Jimenez-Ruiz, E., Grau, B. C., Zhou, Y., & Horrocks, I. (2012). Large-scale Interactive Ontology Matching: Algorithms and Implementation. *Frontiers in Artificial Intelligence and Applications*, *242*, 444–

449.

Khabaza, T. (2014). From Cognitive Science to Data Mining: The First Intelligence Amplifier. In *Cognitive Systems Monographs* (Vol. 22, pp. 191–204). https://doi.org/10.1007/978-3-319-06614-1_13

Kondo, K., Nishitani, H., & Nakamura, Y. (2010). Human-Computer Collaborative Object Recognition for Intelligent Support. In *Media* (pp. 471–482). https://doi.org/10.1007/978-3-642-15696-0_44

Kubat, M. (2015). *An Introduction to Machine Learning*. *Machine Learning*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-20010-1

Lesh, N., Marks, J., Rich, C., & Sidner, C. L. (2004). "Man-Computer Symbiosis" Revisited: Achieving Natural Communication and Collaboration with Computers. *IEICE TRANSACTIONS on Information and Systems*, *87*(6), 1290–1298.

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

Madhavan, J., Bernstein, P. A., Doan, A., & Halevy, A. (2005). Corpus-Based Schema Matching. In *21st International Conference on Data Engineering (ICDE'05)* (pp. 57–68). IEEE. https://doi.org/10.1109/ICDE.2005.39

Martínez-Fernández, S., Ayala, C. P., Franch, X., & Marques, H. M. (2017). Benefits and drawbacks of software reference architectures: A case study. *Information and Software Technology*, *88*, 37–52. https://doi.org/10.1016/j.infsof.2017.03.011

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th International Conference on Data Engineering* (pp. 117–128). IEEE Comput. Soc. https://doi.org/10.1109/ICDE.2002.994702

Microsoft. (2017a). How to choose machine learning algorithms. Retrieved June 30, 2017, from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice

Microsoft. (2017b). Machine learning algorithm cheat sheet. Retrieved June 15, 2017, from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-cheat-sheet

Paraense, A. L. O., Gudwin, R. R., & Goncalves, R. de A. (2007). Designing Intelligence Augmentation System with a Semiotic-Oriented Software Development Process. In *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)* (pp. 84–89). IEEE. https://doi.org/10.1109/ISDA.2007.16

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, *24*(January), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Rahm, E. (2011). Towards Large-Scale Schema and Ontology Matching. In *Schema Matching and Mapping* (pp. 3–27). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-16518-4_1

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

Rodrigues, D., da Silva, A., Rodrigues, R., & dos Santos, E. (2015). Using active learning techniques for improving database schema matching methods. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN.2015.7280630

Sorrentino, S., Bergamaschi, S., Gawinecki, M., & Po, L. (2010). Schema label normalization for improving schema matching. *Data & Knowledge Engineering*, *69*(12), 1254–1273. https://doi.org/10.1016/j.datak.2010.10.004

Stumpf, S., Rajaram, V., Li, L., Wong, W.-K., Burnett, M., Dietterich, T., … Herlocker, J. (2009). Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, *67*(8), 639–662. https://doi.org/10.1016/j.ijhcs.2009.03.004

Sutanta, E., Wardoyo, R., Mustofa, K., & Winarko, E. (2016). Survey: Models and Prototypes of Schema Matching. *International Journal of Electrical and Computer Engineering (IJECE)*, *6*(3), 1011. https://doi.org/10.11591/ijece.v6i3.9789

Verschuren, P., & Doorewaard, H. (2007). *Het ontwerpen van een onderzoek* (4th ed.). Boom Lemma uitgevers.

Weske, M. (2012). Evolution of Enterprise Systems Architectures. In *Business Process Management* (pp. 25–69). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28616-2_2

Wikipedia. (2017). Precision and Recall. Retrieved June 16, 2017, from https://en.wikipedia.org/wiki/Precision_and_recall

Witten, I. H., Frank, E., & Hall, M. A. (2011a). Algorithms. In *Data Mining: Practical Machine Learning Tools and Techniques* (pp. 85–145). Elsevier. https://doi.org/10.1016/B978-0-12-374856-0.00004-3

Witten, I. H., Frank, E., & Hall, M. A. (2011b). Credibility: evaluating what’s been learned. In *Data Mining: Practical Machine Learning Tools and Techniques* (pp. 147–187). Retrieved from http://linkinghub.elsevier.com/retrieve/pii/B9780123748560000055

Witten, I. H., Frank, E., & Hall, M. A. (2011c). Introduction to Weka. In *Data Mining: Practical Machine Learning Tools and Techniques* (pp. 403–406). Elsevier. https://doi.org/10.1016/B978-0-12-374856-0.00010-9

Zhao, Z., & Ma, Z. (2017). A methodology for measuring structure similarity of fuzzy XML documents. *Computing*, *99*(5), 493–506. https://doi.org/10.1007/s00607-017-0553-x

8 Appendix

The appendix is left out due to confidentiality.