universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Text analytics for conceptual modelling“

verfasst von / submitted by

Julia Baginski BA

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft UG2002
Betreut von / Supervisor:	o. Univ.-Prof. Dr. Dimitris Karagiannis

Schriftliche Versicherung

Ich habe mich bemüht, sämtliche Inhaber der Bildrechte ausfindig zu machen und ihre Zustimmung zur Verwendung der Bilder in dieser Arbeit eingeholt. Sollte dennoch eine Urherberrechtsverletzung bekannt werden, ersuche ich um Meldung bei mir.

Ich versichere, dass die Arbeit von mir selbständig verfasst wurde und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Weiterhin wurde diese Arbeit keiner anderen Prüfungsbehörde übergeben.

Wien, 14. Juni 2018

Acknowledgement

At this point, I would like to thank everybody who has supported me while writing this thesis.

This particularly includes my mom and brother for their continuous moral support and their help in taking other tasks off my hands so that I could focus on my thesis; my colleagues at work whose chocolate-based conditioning has undoubtedly helped to keep my mind focused; Dipl.-Ing. Patrik Burzynski for his valuable input, guidance and reassurance that kept me on track; and Univ.-Prof. Dr. Dimitris Karagiannis for his patience and the opportunity to dive into this interesting topic. Finally, I want to extend my sincere gratitude to all other friends and family who have studied with me, motivated me and supported me in any other way.

Sincerely, thank you all.

Table of Contents

List of figures

List of tables

List of abbreviations

ADL	ADOxx Development Language
BPMN	Business Process Model and Notation
EPC	Event-driven Process Chain
ER	Entity Relationship
GUI	Graphical User Interface
KDD	Knowledge Discovery in Databases
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS	Part of speech
RDF	Resource Description Framework
SQL	Structured Query Language
UML	Unified Modeling Language
XML	Extensible Markup Language

1. Introduction

According to Han, Pei, and Kamber (2011), the general procedure of uncovering interesting patterns and information from data while using various techniques, such as machine learning and different algorithms, is commonly referred to as data mining. As a sub-category, text mining focuses on uncovering interesting patterns and information from textual data specifically. It is a field which has enjoyed vast popularity in recent years, due to the recent surge in available text data and the increasing interest in its utilization. Aggarwal and Zhai (2012) argue that this interest can be attributed to the recent rise of online platforms, advances in hardware and software technology, and the focus on user-created content. Because of which, the amounts of textual data that is generated and can be stored and analyzed everyday has increased substantially. Textual data is very easy to create. However, due to its unstructured nature, as well as ambiguity and implied inference in natural language itself, it is not as easy to analyze. This is why many researchers have spent considerable time and efforts on coming up with increasingly sophisticated methods and techniques to effectively process large amounts of text data, as discussed by Shwartz and Schank Roger (1987) Given the recent advances in natural language processing, text analysis is now performed on various data sources and finds applications in many different domains. However, when considering the conceptual modelling domain, text mining seems to have been underutilized until now.

Bearing in mind that any model starts as a general description of the modelling objective, usually expressed in natural language, it seems almost natural to try to apply natural language processing techniques to these descriptions. Supporting the model design process already at this stage can have various benefits. To provide an example, the analysis of initial text specifications within its individual conceptual considerations, could potentially enable automated model generation based on natural language input alone. Even technology-averse individuals would then be able to generate models of processes, system or objects easily and with no or minimal knowledge of the underlying modelling processes or languages. While some efforts have already been made to apply text mining techniques to specific types of models (see Omar, Hanna, and McKevitt (2004) or Friedrich, Mendling, and Puhlmann (2011) for examples) or for specific purposes (see Sagar and Abirami (2014) or Hornsby and Li (2009) for examples), a comprehensive approach on how to utilize text mining techniques for conceptual modelling in general still seems to be missing.

Essentially, this work thus aims to provide a more general overview of text mining technologies and their potential application to the conceptual modelling domain. Throughout this work, various opportunities and challenges of utilizing text analytics for conceptual modelling are discussed and a

generic concept for text mining in conceptual model generation is presented. The purpose of the generic concept is to enable further research into this area and to ease the development of such text mining applications in the future. Additionally, a comprehensive example of how this generic concept can be utilized for application development is presented. The example illustrates how text mining techniques can be employed to assist the modelling process and thus ultimately shows how the discussed theory can be put to practice. For this purpose, a tool has been developed which extracts entity-relationship modelling elements from natural language specifications. Essentially, the thesis aims at answering the central research question, *"How can text analytics be utilized for conceptual modelling?"*

The first part of the thesis, encompassing chapters one through two, is of explanatory nature with the central objective of introducing the reader to the domains of conceptual modelling and text mining in general. This part is based on an extensive literature review. Information has been drawn from various literature sources, such as books, journals, academic papers and relevant articles. Chapter one provides a brief introduction into the topic, motivation for the thesis and the paper structure. Chapter two provides the theoretical background for the rest of the work. Thus, after reading chapter two, the reader should have a sound understanding of both domains.

The second part of the paper is driven by an exploratory approach. Chapter three focuses on merging the two areas and giving suggestions on how conceptual modelling could benefit from current text mining advances. Hereby first, the rationale for merging the two domains and potential application areas are discussed. Then, one of the application areas, namely the utilization of text analytics throughout the model design process itself is presented in detail and a generic text analysis approach for conceptual model generation is introduced. Additionally, a selection of free tools which are available for both conceptual modelling as well as text analytics is presented at the end of this chapter.

Finally, chapter four provides an illustrative example of a text analysis and extraction tool for the purpose of demonstrating the usability of the generic approach presented in the previous chapter. The tool's functionality focuses on the identification of relevant modeling concepts from text specifications and aims at enabling automatic ER model generation out of natural language input alone. Given the entity-relationship context, the tools objective is to identify entities, relationships, attributes and cardinalities. The implementation of this tool is hereby based on the generic concept presented in chapter three and extensively documented to enable the reader to follow each step of the implementation easily. Finally, an evaluation and discussion of the developed tool is presented and suggestions for future improvements and research opportunities are made.

2. Theoretical background

To be able to apply text analytics to conceptual modelling, it is important to first familiarize oneself with the fundamentals of both domains. This chapter thus focuses on reviewing basic concepts and providing the theoretical foundation for both the conceptual modelling domain itself, as well as text analytics in general. The first subchapter concentrates on providing a clear understanding of what conceptual models essentially are, what they are used for, how they are presented and how they are evaluated. In the second subchapter, fundamental principles, frequently used techniques, current possibilities as well as limitations of text analytics are discussed.

2.1 Conceptual modeling

2.1.1 Definition and fundamental objectives

"The idea came to me as one switches on a light, one day when by chance there fell into my hands an old dusty diagram, the work of some unknown predecessor of mine. Since a chemist does not think, indeed does not live without models, I idly went about representing them for myself, drawing on paper the long chains of silicon, oxygen, iron and magnesium, with the nickel caught between their links, and I did not feel much different from the remote hunter of Altamira who painted an antelope on the rock wall so that the next day's hunt would be lucky." (Levi 1984)

Models, as purposeful abstractions of reality, have been used by humanity for centuries and found applications in many different domains. Given their usefulness in aiding decision making and enabling the understanding of highly complex systems, they can now be found in virtually every area of life. Scientists use mathematical models, economists use chart models, philosophers use mental models and software engineers use data models, to name just a few. See Figure 1 below for examples of the previously mentioned models.

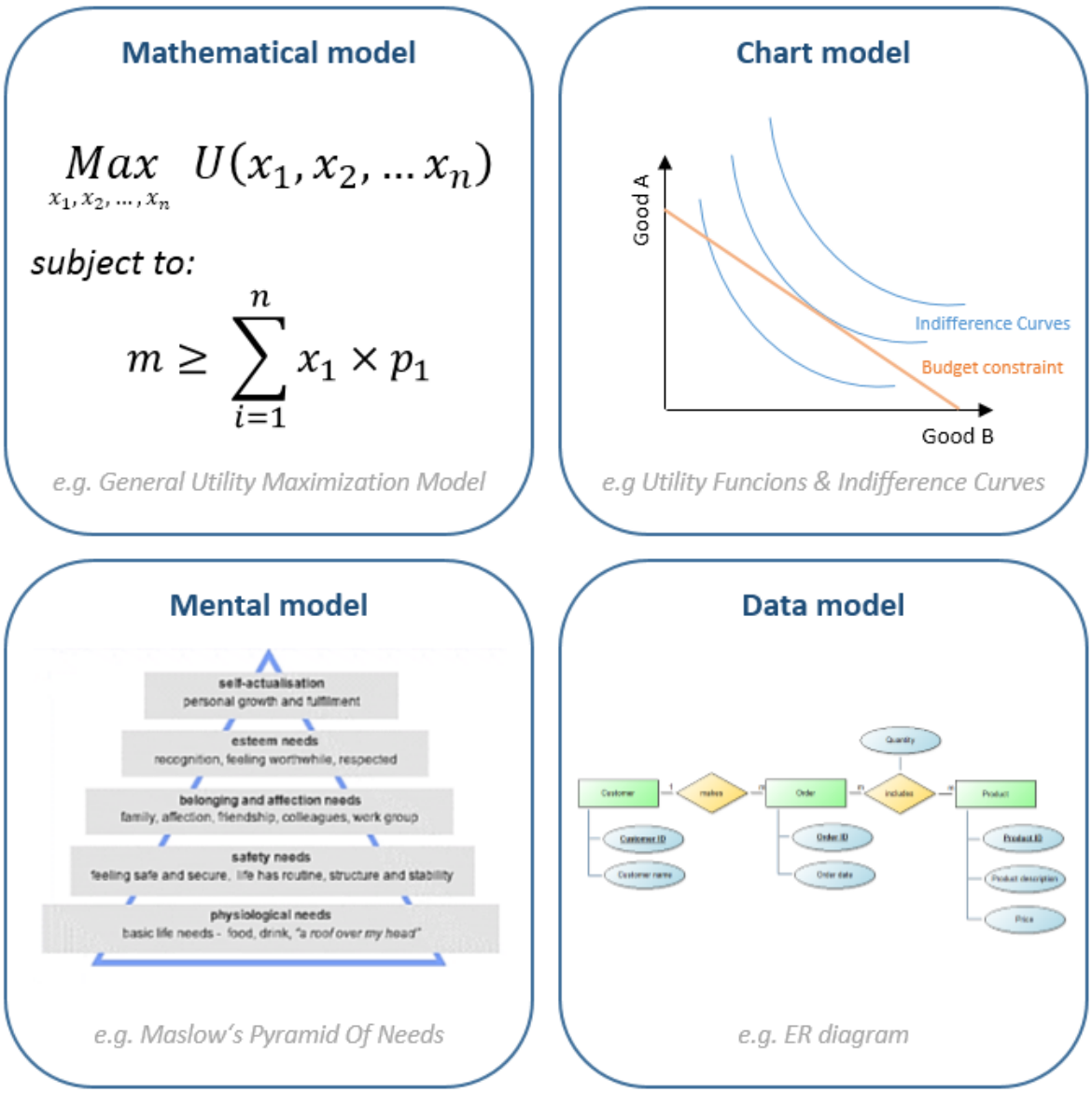

Figure 1 Examples of models [1]

Particularly in the business world, with its fast-changing environment, models continue to enjoy a high popularity. This can be attributed to the fact that their inherent nature allows the representation of complex systems in a clear and understandable manner, finding uses in training, system or product development as well as general decision making. This thesis will hereby focus on conceptual models. Additionally, to deepen the reader's understanding of general principles, the example of a data model from Figure 1 above will be used to illustrate basic modelling concepts in the following sections.

[1] Note that the depiction of "Maslow's Pyramid of Needs" was taken from Google sites: https://sites.google.com/site/psychologyofpersonalityperiod6/home/humanistic-theories/maslow, accessed on September 15th 2017

Conceptual modelling itself is defined by Mylopoulos (1992) as *"the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication."* This is in line with Wand et al. (1995)'s definition which focuses on the formal description of some part of a real-world system. Given this general definition, conceptual modelling in the business domain can have various forms and various applications. Its purpose can be to capture process flows, to explain how a database is or should be structured, or to describe a system, environment or any other subject matter to understand and communicate it. According to Mylopoulos (1992) the distinctive features of conceptual models are on one hand the human-centric approach and on the other, the importance of semantic representation. This is meant in the sense that ultimately humans are the users of conceptual models and thus these models should be built in a way that people can easily derive the meaning of any given representation. Thus, a focus is laid upon the simplicity and efficiency of conceptual models. *"The key concern is to structure the representation of knowledge about a subject consistently to the way humans structure the same knowledge and to make sure the procedures that use these representations draw the same, or at least a subset of the inferences people would draw when confronted with the same facts."* (Mylopoulos 1992)

Furthermore, this paper tries to stay consistent with Seidewitz (2003) formal definitions of general modelling terms discussed below. According to him a model is generally defined as *"a set of statements about some system under study."* Hereby, he distinguishes two types of models. According to him, models can be descriptive, in the sense of simply describing a specific system under study, or they can be prescriptive, thus providing specification for a system to be developed. This distinction is particularly relevant when evaluating a given model. In its descriptive sense, a model can be a "*correct*" representation of the system under study, if all the statements within the model are true for the specific system it aims to describe. If, however, a model serves as specification, then typically the system under study which has been built based on those specifications is examined for validity not the model itself. The system can then be considered "*valid*" if it does not contradict any of the model's statements. An example given by Seidewitz is the construction of a rocket, which has been built based on a model and its calculations. If the model for example described what trajectory the rocket should take, and the rocket diverges from it, then most likely the rocket's design could be considered invalid not the model itself.

Another useful distinction is provided by Kühne (2006). According to him, models can represent abstractions of either real or language-based systems. An example of a model based on a real system, or an iconic model, would be a small physical model of a building to be constructed. A language-based model, on the other hand, could be a process flow for the installation of doors in said building. The latter type, namely models based on language-systems, can typically be found in software engineering

or the business domain and thus represents the focus of this research paper. Whether physical or linguistic, Stachowiak (1973) argues that in order to be considered a "*model*", the abstraction must possess three distinct features: a mapping, a reduction and a pragmatic feature. Firstly, the model must be mapped to, thus represent, a specific system. Secondly, only certain parts of the system should be described by the model. Thus, it must provide a reduced view of the system and must only focuses on certain characteristics of the represented system. Which characteristics of the system are put in focus depends on the final purpose of the model. Finally, the pragmatic feature dictates that the model must serve some specified purpose and thus have been designed with this final purpose in mind. Figure 2 below provides a brief overview of the modelling concepts discussed above.

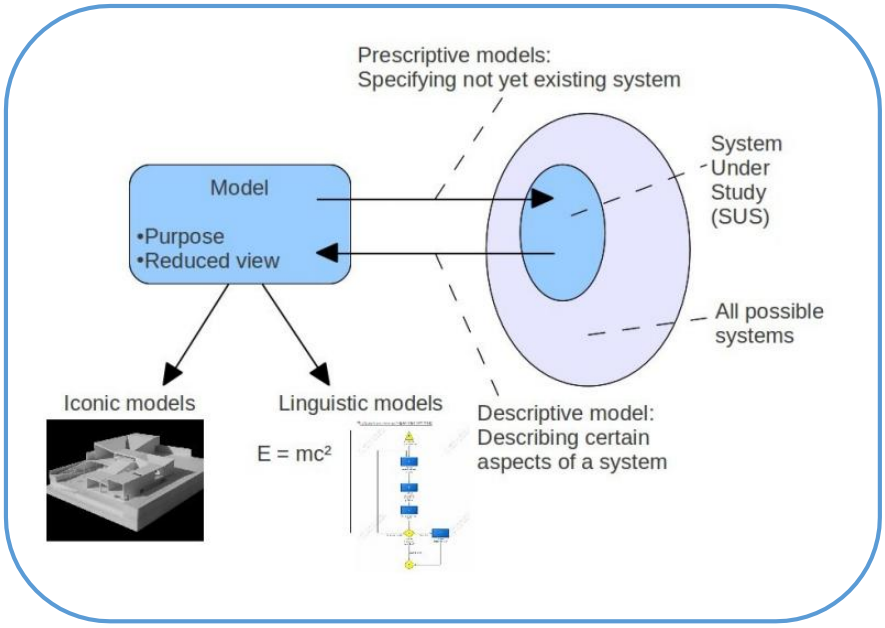

Figure 2 Models and SUS taken from Burzynski (2013)

2.1.2 Model design process

At the beginning of the model design process the modeler is typically presented with some sort of textual description of the system under study, often expressed in natural language. For example, to reference back to the data model from Figure 1, the owner of a new online shop may describe to the web developer in what way customers should be able to make orders and what information the system needs to be able to store about these orders. Depending on the purpose in mind, the modelling approach as well as the chosen language, different parts of the text or conversation might be important for constructing the model. However, irrespective of the previously mentioned aspects, each model design process typically goes through three distinct stages as described by Karagiannis, Burzynski, and Miron (2017): text annotation, mapping and model design.

The first step, text annotation, focuses on text comprehension. Any description of the modelling objective first needs to be read and ultimately understood. Annotating the text while reading it can help filtering out useful information from abundant one and add helpful content if necessary. Hereby "*annotation*" should not only be understood as adding notes to the original text, other forms of text transformation such as highlighting the text or changing its format are also frequently utilized and can be very useful. It is however important to keep the ultimate modelling goal in mind during this phase. Only relevant information that supports the specific modelling purpose should be added, highlighted or elsewise annotated.

The next step concentrates on mapping the domain specific information derived from the previous phase to the specific concepts and elements of the chosen modelling language. Hereby a thorough understanding of the modelling elements, rules and particularities of the chosen modelling language is required. A conscious decision, on which part should be included in the model and in what way, needs to be taken. Hereby the focus is not only laid upon abiding by the rules and concepts of the modelling language, but also to extract and translate the most important information in an efficient manner. Keeping the final modelling objective in mind helps with the decision process at this stage as well.

Finally, after the text has been annotated and mapped to the respective modelling language elements, the actual model design can begin. Hereby it is important to point out that this is an iterative process. Models can be designed in various ways and different levels of detail. Typically, the first draft of a model is improved upon based on feedback from others or new considerations oneself might have. Often there is also more than one correct representation of any given system, however one representation might be more suitable than the other if one considers the final domain the model will be used in. (Karagiannis, Burzynski, and Miron 2017)

2.1.3 Modelling languages

As previously defined, models consist of statements about any given system under study. Modelling languages thereby provide a way to express these statements in a more efficient way than natural language does (Seidewitz 2003). The modelling language chosen hereby depends on the desired outcome of the modelling process and ultimately on the model's purpose. Different modelling languages should be chosen depending on whether a process needs to be captured, a system described or whether the goal is to build a data model. This section briefly describes the basic concepts underlying modelling languages and then gives a brief overview of five commonly used conceptual modelling languages.

Similarly, to natural languages, modelling languages consist of a certain set of rules that, as with any language, dictate the spelling, grammar and meaning of certain language elements. These rules are needed not only for textual languages, but also for graphical or diagrammatic languages. In the modelling domain spelling, grammar and meaning are typically covered by the notation, syntax and semantics of modelling elements. We will have a closer look at our data model example (see Figure 3 below), which has been drawn using the Entity Relationship (ER) modelling language, to discuss these principles.

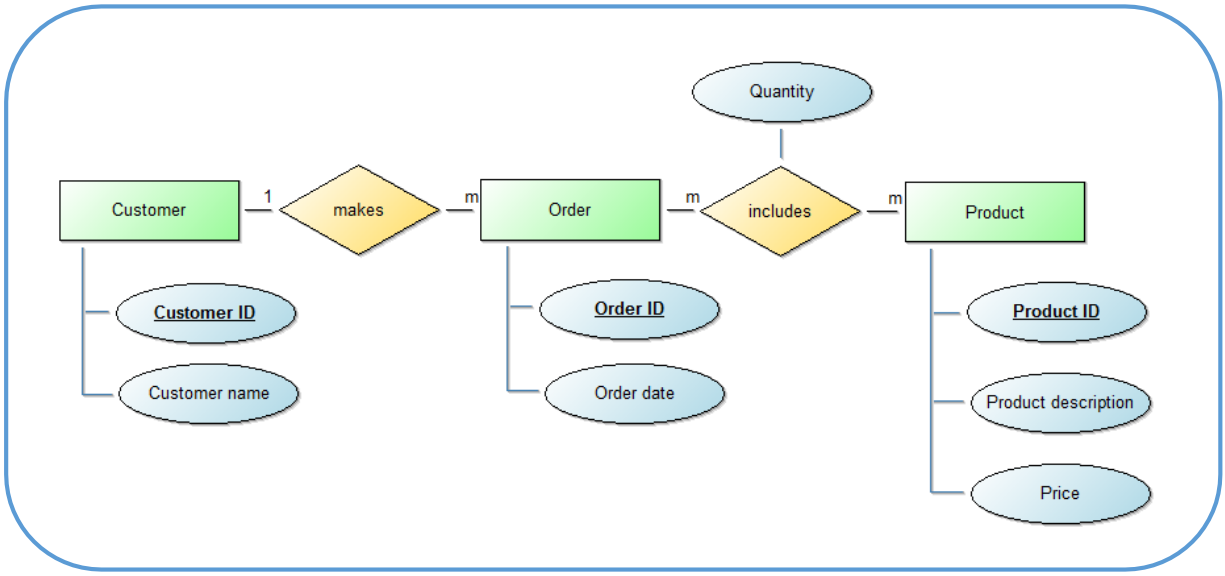

Figure 3 Example of a data model: An ER- diagram

As described by Karagiannis and Kühn (2002), the notation determines the visualization of a modelling language, thus in what way the modelling constructs should be presented or drawn. Karagiannis and Kühn discuss that in static modelling approaches, this mainly concerns the shape and appearance of the modelling elements. In more dynamic approaches, the model state can be considered as well when determining the correct visualization, e.g. by allowing the specification of rules which adapt the depiction based on the model state. Referring to the example above, the notation would hereby specify that entities need to be depicted as rectangles, relationships as diamond shapes and attributes as ovals. Additionally, the notation would also specify how these elements are connected, in this case through a solid line, and how cardinalities are shown, namely as annotations next to the connecting line.

While the notation covers the spelling part of the modelling language, the syntax, on the other hand, is representative for the grammar of the language. Hereby, the different modelling elements are described and the rules which need to be applied when constructing syntactically valid models are

discussed. Basically, the syntax describes in what way the various modeling elements can be put together to construct valid models. A syntactical rule which is applicable to our exemplary ER-model is for example that each entity must have a key attribute which uniquely identifies an instance of the entity, e.g. each "*Customer*" has a unique "*Customer ID*" or each "*Order*" can be identified by the "*Order ID*". Another syntactical rule could for example be that only connections of relationships can have cardinalities. For describing syntactical concepts of modelling languages, metamodels are frequently used.

However, before having a closer look at metamodels, another very important concept, namely the semantics of a given modelling language needs to be mentioned. The semantics hereby cover the meaning of given constructs. According to Karagiannis and Kühn (2002) meaning can be described in various ways, e.g. by using mathematical concepts, ontologies or simply natural language text. Furthermore, they stress that it is important that domain-specific semantics are considered when evaluating a model for semantic validity. This is supported by Harel and Rumpe (2004), who stress that, particularly in computing, the usability of languages and ultimately precision of communication, highly depends on the clear definition of rules not only for the allowable syntax, but also the meanings associated with syntactically correct expressions. However, they also point out how difficult it is to define semantics in the modelling domain and that because of this language definitions often lack this important component. Because of the difficulty, frequently the chosen approach remains to define semantics with natural language descriptions. Finally, it is worth to consider that while a model can be syntactically valid and be using the correct notational concepts, it might not make sense on a semantic level or vice versa. To provide an example, in Figure 3 the cardinality between "*Customer*" and "*Order*", which specifies the maximum number of relationships between these two entities, could have been entered as 1:1, meaning that every customer can only place a maximum of one order. This would still be a syntactically valid model, as 1:1 is a cardinality which can be used in ER models. However, when considering the application domain, namely an online shop, then it is evident that this is not correct, as a customer should be able to make more than one order. Thus, while being valid syntactically; this model would not make sense semantically. Therefore, when evaluating if a model has been correctly transcribed using a specific modelling language each of the three language aspects, namely notation, syntax and semantics, needs to be considered separately, before the model can be determined to be valid.

To come back to the subdomain of syntactic validity, while the ways of determining semantic validity is still limited, syntactical concepts are typically well defined and thus can be checked easily. A good way of checking whether a model is syntactically valid is by considering the metamodel of the used modelling language. In this regard Kühne (2006) discusses that as per the linguistic definition of "*meta*"[2], a metamodel can be understood as a model where the modelling process has been performed twice, thus "*a model of models*". A more narrow, yet commonly accepted definition is provided by Seidewitz (2003), who argues that metamodels specify *"what can be expressed in valid models of a certain modeling language"*. According to this definition, a metamodel can be seen as a model of a modelling language, expressed in yet another language, namely the meta-modelling language. Coming back to the construct of validity, this means that a given model in a specified modelling language is only "*valid*" if it does not falsify any of the statements in the metamodel. This follows from the definition of a specification model, considering that the system under study of a certain metamodel is the modelling language itself. Karagiannis and Kühn (2002) further discuss that the syntactic rules of metamodels can themselves be described by a meta-meta modelling language or a meta² modelling language. Generally speaking, there is no limit for how many layers a meta-modelling hierarchy can have, however typically four layers including the system under study itself, the model of said system, its metamodel and a meta² model, as visualized in Figure 4, are sufficient to conceptualize the needed concepts while maintaining a practical level of abstraction.

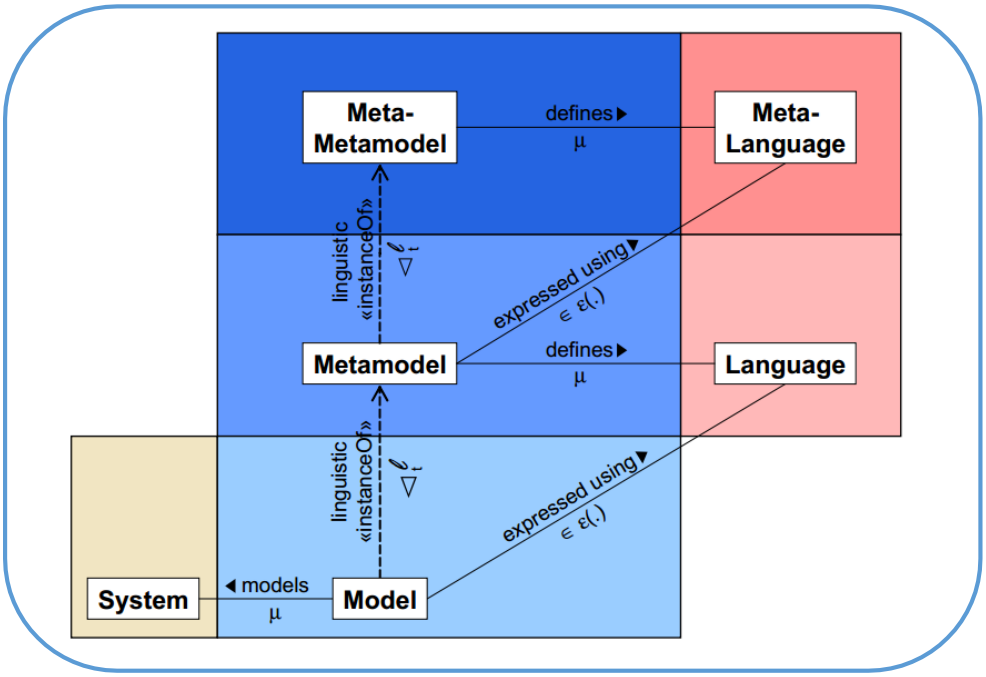

Figure 4 Language Definition Stack taken from Kühne (2006)

[2] As per Kühne (2006) the prefix „meta" is mainly used in cases where an operation has been applied twice.

The below table gives a brief overview of five fundamental conceptual modelling languages as defined by Karagiannis et al. (2016).

Table 1 Fundamental conceptual modelling languages

MODEL TYPES/ LANGUAGES	DESCRIPTION
ER – Entity Relationship diagrams	ER diagrams describe categorical concepts and how they relate to one another. They further allow the specification of characteristics of these concepts or of their relationships. **Core elements:** entities, relationships and attributes. **Main application area:** Data modelling / database design
EPC – Event-driven Process Chain diagrams	EPC diagrams describe processes as alternating sequences of functions and events, with an emphasis being put on the identification of states (events) which trigger or result from the various actions (functions). Additionally, EPC diagrams allow the addition of enterprise context to the various functions, e.g. the system used to perform a certain function. The focus hereby lies on understandability by making the process depiction very user friendly, e.g. through the use of colors and shapes. **Core elements:** functions, events and operations. **Main application area:** Business process management
BPMN – Business Process Model and Notation	BPMN is used for process descriptions with a heavy focus being laid upon formal notation and business- orientation. BPMN was designed to be easily understood by both, business users as well as developers. In contrast to EPC diagrams, the focus is hereby laid upon the tasks themselves, rather than the identification of states. **Core elements:** tasks, events and gateways. **Main application area:** Business process management
UML – Unified Modeling Language	UML is one of the most popular and widely-used general-purpose modelling languages for describing structural and behavioral aspects of software systems. **Core elements:** static/structural diagram types (objects, attributes and relationships) and dynamic/behavioral diagram types (objects and object changes). **Main application area:** Software engineering / object-oriented programming

Petri Nets	Petri Nets provide a more abstract graphical modelling method, which is based on set mathematical concepts. Again, the focus lies on states/ places and changes of states through transitions. Hereby tokens are used to depict the dynamic aspects of the process flow. Its high level of abstraction allows cross-domain applicability, however at the expense of user-understandability. **Core elements:** places, transition, arcs and tokens. **Main application area:** Process dynamics/ Academics

2.1.4 Modelling method

However, when constructing a model, choosing and applying the appropriate modelling language is not the only thing that needs to be carefully considered. The modelling language is typically applied with a modelling procedure, which together make up the modelling technique. Additionally various algorithms and mechanisms are applied throughout the modelling process. Karagiannis and Kühn (2002) have developed a generic modelling method framework which tries to encompass all these aspects. According to them, modelling methods consists of two main parts, namely a modelling technique, as well as mechanisms & algorithms used throughout the modelling process and on the final model itself. An overview of the generic modelling method framework proposed by Karagiannis and Kühn (2002) can be seen on Figure 5 below.

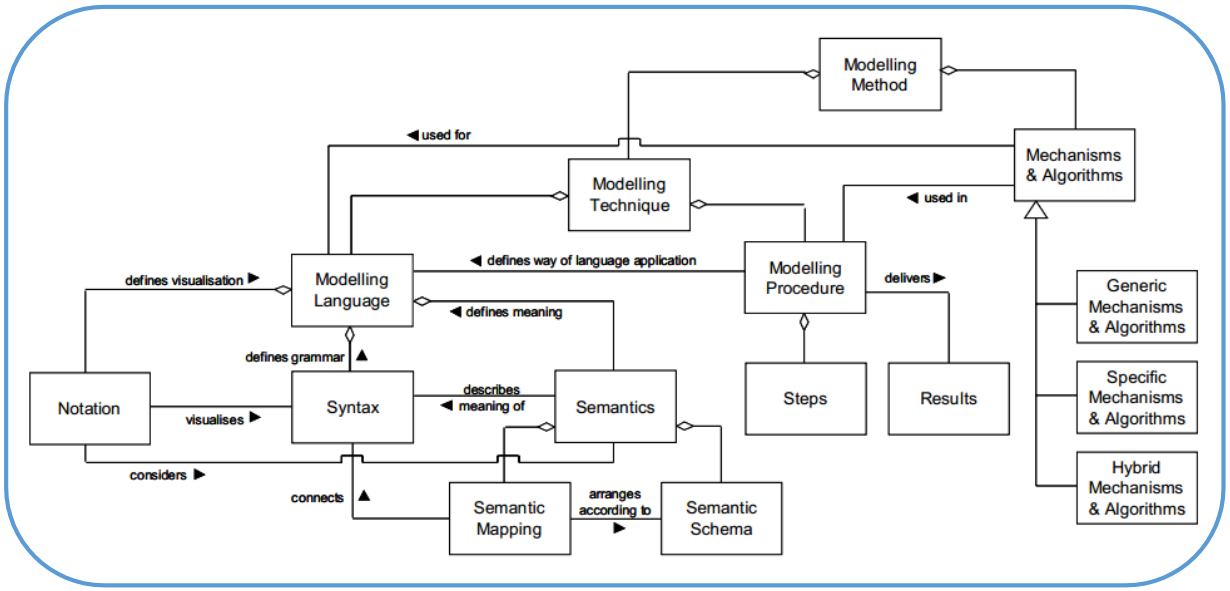

Figure 5 Components of modelling methods taken from Karagiannis and Kühn (2002)

The modelling technique hereby refers to the modelling language chosen and the modelling procedure itself. The modelling language, described by its notation, syntax and semantics, has already been examined in detail in the previous section. The modelling procedure on the other hand, mainly describes the various steps taken to generate the resulting model and thus the way the language is applied. An example for a high-level modelling procedure is the model design process discussed in section 2.1.2 Modelling procedures can however be far more detailed.

Furthermore, the framework considers various mechanisms and algorithms which are used at different stages of the modelling process, enabling the evaluation and further use of the designed models. These mechanisms and algorithms can be generic in nature, thus used regardless of modelling language; they can be specific, thus used for a certain modelling language or meta-model; or they can be hybrid, thus built upon similar concepts but adapted to the specifics of a certain meta-model or language. Examples of mechanisms and algorithms include the simulation of processes or documentation as described by Karagiannis and Kühn (2002). In their modelling example of an insurance provider, they describe how linking process models with administrative data and an IT infrastructure model allows the running of process simulation algorithms. This allows the determination of estimates for resource allocation, cycle and response times. Another mechanism mentioned in their example is model export and documentation for the purpose of constructing an operations manual.

2.2 Text analytics

2.2.1 Origin and overview

According to Hotho, Nürnberger, and Paaß (2005), whose survey on text mining is the main source of information for this part of the thesis, text mining can be understood as machine –supported knowledge discovery from textual data. As they mention, the terms "*text mining*" and "*knowledge discovery in text*" (KDT) are, in fact, often used interchangeably and can thus be considered synonyms. This section focuses on providing an explanation of what "*knowledge discovery in text*" means, where it came from, as well as why and how it is performed. Essentially, the main objective of text mining is the automated discovery of interesting and useful insights or patterns through the analysis of textual or linguistic data. Text mining is rooted in both, traditional data analytics, which focus on knowledge discovery from various sorts of data, as well as natural language processing, which has the ultimate goal of enabling computers to understand natural language.

For humans, natural language, as their primary means of communication, is relatively easy to understand. Though Shwartz and Schank Roger (1987) argue that it is certainly not an elementary capability. Linguistic content is often filled with ambiguities, irregularities and implications, however

in most cases, humans have no trouble with discerning basic linguistic concepts, grasping contextual meaning and comprehending the main content of what is communicated as natural language was designed precisely for that purpose, namely the communication between humans. Computers, on the other hand, have difficulties handling the fuzziness of natural language, particularly as natural language understanding is not solely reliant on linguistic rules, but also heavily on world knowledge and the ability to use this knowledge to infer and discern the meaning of linguistic input. However, as Gupta and Lehal (2009) have correctly identified, computers have one major advantage to us humans, namely the ability to process large volumes of data at a very high speed. However, since traditional data analytics only provided means to analyze structured content, and textual data is often unstructured in nature, the utilization of these resources was rather limited in the early years. Nevertheless, the notion that valuable insights could be gained by finding ways to automatically analyze and discern the abundance of this readily available data, has spurred lots of academic and commercial research interest in the area.

Consequently, over the years, text mining has emerged as a highly interdisciplinary field, utilizing principles, algorithms and techniques from many different subject areas as discussed by Gupta and Lehal (2009) and at present finding its main applications in the medical field, in business, education and the Government sector. As described by Fan et al. (2006), text mining was already used successfully in 1987 by Dan Swanson who uncovered a linkage between migraines and magnesium deficiency by applying text mining techniques to shift through numerous scientific papers on migraines looking for significant keywords, and then repeating the process by shifting through scientific papers which focused on those keywords. See Swanson (1987)'s publication for more details. Subsequent studies proved the validity of his hypothesis. As the availability of data has grown, the application areas have grown as well. To provide a more recent example, also from Fan et al. (2006), governments now use various text mining techniques for national security by shifting through various web pages, comments, messages and usage logs to uncover terrorist threats ahead of time. Another example they mentioned are question and answering system which allow natural language input and return natural language answers, used for various commercial and non-commercial purposes. Amongst others, text mining hereby draws on developments and the continuous advancement of knowledge management, database technology, statistics, machine learning, and data mining, all for the ultimate purpose of knowledge discovery from unstructured, textual data. However, before diving into the specifics of the various techniques and the knowledge discovery process itself, it may be useful to take a step back and provide a clear distinction between the terms "*data*", "*information*" and "*knowledge*" in order to ensure a clear understanding of what *"inferring knowledge from textual data"* essentially means.

Bellinger, Castro, and Mills (2004), who have sought to establish clear definitions of the above-mentioned terms, define data as raw constructs, statements or facts, which simply stand alone and do not have any meaning by themselves, e.g. "*red*" or "*#FF0000*". They are constructs which exist without any further significance and were typically generated through some sort of measurement or data capturing process. No conclusion can be made based on seeing a simple datum. When data is however processed and put in relation to or in some way connected to other data, meaning is added to the whole construct and its usefulness increases. According to Bellinger, Castro, and Mills (2004) this understanding of some sort of relationship is the distinctive factor between mere data and information. An example of information is given by, "*The pedestrian light on 1st street has just turned red.*" This can be considered information, as it provides a useful relational understanding of what the original datum "*red*" meant. One can then further add context to it, and connect the information to other information, such as the information that "*I am walking towards 1st street."* This would make the whole construct more relevant and actionable, demonstrating the way how information can become knowledge. For example, knowledge would thus be the further connection of the previously mentioned information, namely that "*The pedestrian light that I am walking toward has just turned red.*" Subsequently, if one were to further increase the connectedness, apply the knowledge to other knowledge one might have and understand the fundamental principles that the knowledge embodies, to generate some sort of idea or decision, in that case, such a construct could be considered wisdom. To finalize the example above, wisdom in that case would be to "*Stop walking.*"

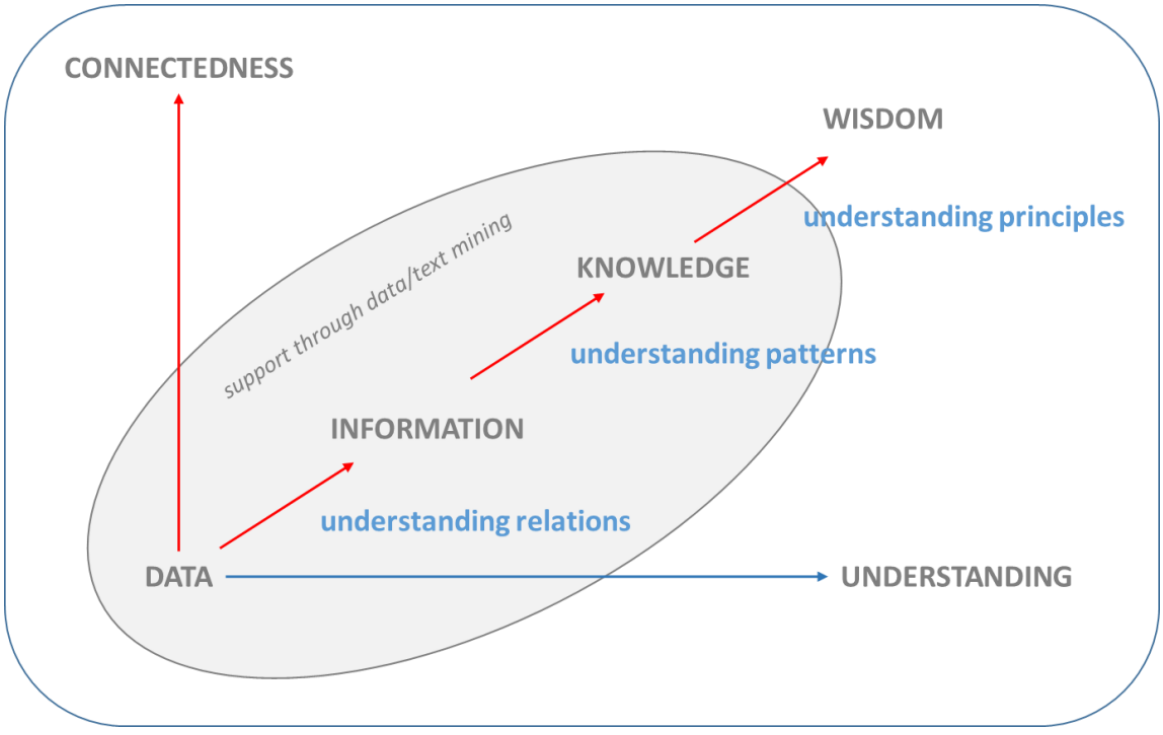

Figure 6 Transition from data to wisdom, adapted from Bellinger, Castro, and Mills (2004)

Figure 6 above, which has been adapted from Bellinger, Castro, and Mills (2004), provides a basic overview of the data to wisdom transition process. Text mining hereby focuses on gaining knowledge from textual data specifically, thus on the identification of useful patterns within textual data. According to Fayyad, Piatetsky-Shapiro, and Smyth (1996) the concept of "patterns" can hereby be understood as the extraction or generation of a description of some subset of the original data set, either in the form of fitted models or some other, for example linguistic description. They further stress that the identified patterns do not only have to be correct, but they also should be useful and understandable in order to be considered knowledge. To continue down this path, once knowledge has in fact been inferred, it is typically used for decision making, as a basis for further action or the subsequent development of ideas and judgments, often based on underlying values and beliefs. Many efforts are currently made to further extend the scope of data analytics to even infer wisdom from data alone, see various smart applications as described by Cook and Das (2004) for example. An illustrative example is hereby given by a smart fridge which monitors your fridge content, its usage and expiration dates and alerts you what you need to buy when you walk past a grocery store or suggest recipes for meals you can cook with the things which are expiring soon. Nevertheless, some researchers remain skeptical on whether extending data analytics to the realm of automated wisdom inference can and should truly be done. Some researchers still agree with Ackoff (1989), who is often considered to have coined the distinctions between data, information, knowledge and wisdom, when arguing that *"wisdom—which is essential for the pursuit of ideals or ultimately valued ends—is the characteristic that differentiates man from machines."*

It should thus be clear by now that data by itself is not very useful on its own, but needs to be analyzed and refined to gain something useful from it. This is where text mining comes in to play. Text mining emerged as an extension of data mining with the shared objective of machine supported knowledge discovery, in this case, from textual data. The main distinctive factor which helps differentiate text mining from data mining is given in the type of data processed, and the added steps and unique approaches needed to discern this textual content. While data mining focuses on knowledge discovery from all sorts of data, text mining is specifically applied to textual data. Frequently, this type of data is unstructured in nature and thus requires various preprocessing, information extraction or natural language processing methods to first extract relevant data sets from the original text, transform them into a more practical structure, before the text can then be further analyzed with various data mining methods.

The reason why this field has now gained popularity, can partially be attributed to the increasing amounts of textual data that has become available as a result of the world's continued digitalization. Daily an almost unimaginable amount of textual data is being created by various devices, applications and individuals. The business world has been affected by this dramatic development as well. Virtually all communication is now done via email and online collaboration platforms, customer profiles can be found online and their opinions are voiced in the form of written reviews or blog posts, news stories about the business sector and competitors are readily accessible and internal operating procedures, meeting minutes and many other corporate documents are all stored in textual form on company computers or in the cloud. As Grimes (2008) has mentioned, in the text mining domain, it is generally believed that approximately 80% of data is unstructured in nature, mainly in textual form. The growing availability of textual data has thus left a large number of researchers and individuals trying to find new and more efficient ways to capture, analyze and utilize this data, due to the believe that a lot of valuable insights can be gained from these ever-growing data mountains. Furthermore, the advancement in data storage and data mining techniques, which enabled the automated knowledge discovery from very large volumes of data at a speed and efficiency that was previously not imaginable, further contributed to the exploration of unstructured text data as a basis for knowledge discovery. It is without any doubt that research on text mining techniques will continue to grow as our ability to capture, store and analyze large datasets continues to grow as well.

2.2.2 Opportunities and challenges

That insights can be gained from analyzing unstructured, textual content has been mentioned a few times already. However, the question remains, what kind of insights can be attained, with how much invested effort and where are the limits of this knowledge exploration? Additionally, one might wonder why, when it is that valuable, not every company has started with textual analytics yet? The following section thus provides a very brief overview of potential opportunities, application areas and challenges.

Opportunities

Everyday more and more news and content on companies, industries and consumers can be found and readily accessed on the web. Utilizing this data influx has become an increasingly important focus of companies when it comes to developing their competitive intelligence. Next to competitive intelligence, other application areas are records management, sentiment analysis, targeted advertising, search & retrieval and knowledge management. However, text mining is not only used by businesses, but is equally as important for governments and the academic field. In the government sector, national security is the main application area of text mining. When it comes to academics,

plagiarism checking, publishing and the opportunity for new research methods are among the main opportunities. See Kasemsap (2017) for further examples and more detailed descriptions of current opportunities and applications.

Challenges

Whilst there are many opportunities in analyzing textual data, the unique properties of natural language and current technological limitations, still limit the current possibilities of text analytics. When it comes to computerized natural language understanding the main problems according to Shwartz and Schank Roger (1987) are still given with the ambiguity of words and phrases as well as the inherent presupposition and inference of our natural language. Ambiguity is hereby explained as the uncertainty which meaning should be assigned to a specific word, as a word or a phrase by itself may have various meanings. Dr. Stephen Clark, who was cited in an online article (Research 2013), even stated *"Ambiguity is the greatest bottleneck to computational knowledge acquisition, the killer problem of all natural language processing"*. The example of the word "*run*" was provided, which is claimed to have 606 different meaning, including *"moving at a speed faster than walking"*, *"a journey accomplished"*, *"to flow rapidly"* or *"a quick, casual trip"*. Which meaning is essentially the right meaning, depends on the context. Word-sense disambiguation is something that comes naturally to us, as we apply our world knowledge to the problem at hand, rely on our past experience and consider all potential contextual clues. If a person were to say, "*I will run in the next race",* then we immediately know that the meaning of run is "*moving at a speed faster than walking"*. Computers on the other hand have large difficulties discerning the right meaning of words. Particularly, because words may not just be semantically ambiguous, but also syntactically as Shwartz and Schank Roger (1987) argue. A word such as "*run"* may be a noun, a verb, or an adjective, depending on its presented context and usage. The other problem mentioned is the inherent presupposition and inference of our natural language. From the previously provided example, "*I will run in the next race*", one might infer that the person will participate in the next race, even though it is not explicitly mentioned. Another example, which demonstrates the concept of inference rather clearly is an individual saying, "*I like chocolate"*. Another person may easily infer that the person likes *"eating"* chocolate, even though it was not mentioned in the original sentence. Humans hereby rely on their world knowledge, namely knowledge about the world that they have accumulated over their lifetime, to infer the correct meaning. While humans can make errors as well, typically they do grasp the correct meaning. The challenge is how to enable computers to understand the sentence in such cases.

Now to consider the technical side of things, according to a recent study by Akilan (2015), another open problem in text mining is the complexity of transforming unstructured data into a more usable, intermediate form. Particularly, the scalability of semantic analysis during data preprocessing poses a

considerable challenge. Despite recent advances in big data technologies, semantic analysis of textual data still requires a large computational effort. Nevertheless, one of the main aspirations in text mining is the fast analysis of very large document libraries. Therefore, this is a challenge that needs to be overcome in order to ensure the usefulness of text mining applications which rely on semantic or in depth linguistic analysis. Another challenge mentioned by Akilan (2015) is the language component which is uniquely prominent in text mining. Text mining applications are typically written for one specific language, particularly because the syntax and semantics of languages differ from one another. Multilingual applications are rare, however present a potential largely untapped opportunity. Finally, one more open problem is the current lack of domain knowledge integration into text mining applications. The integration of which could however vastly improve text mining results and the quality of the derived models.

2.2.3 Text mining process

The knowledge discovery process provides a holistic view of the transformation of low-level data to high-level knowledge, and while not specific to text mining, it can be applied in any case where knowledge is inferred from data, thus regardless of what type of data is processed. While there are many different variations, this thesis will focus on the KDD (knowledge discovery in databases) process as described by Fayyad, Piatetsky-Shapiro, and Smyth (1996). See Figure 7 below for a slightly adapted version of the process depiction that can be found in their article. While some researchers use different terms, skip or add additional steps, the general process mainly remains the same. In some literature, the simplified version of (1) preprocessing, (2) data mining and (3) post-processing, or slight variants of it are sometimes preferred. See Sukanya and Biruntha (2012) for example.

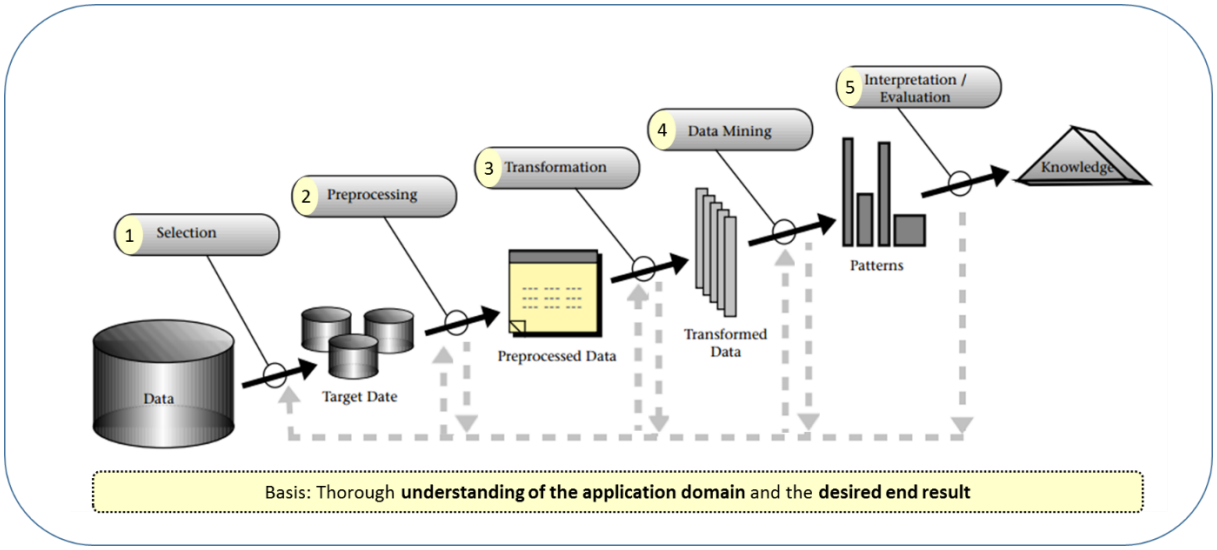

Figure 7 KDD process, adapted from Fayyad, Piatetsky-Shapiro, and Smyth (1996)

Fayyad, Piatetsky-Shapiro, and Smyth (1996)'s description of the KDD process emphasizes its highly iterative and interactive nature. At the beginning, a thorough understanding of the application domain and the desired end result is needed. This understanding provides the basis on which target data is then selected, which is the first step in the KDD chain. Before the data can however be analyzed, preprocessing and data cleaning have to take place. Hotho, Nürnberger, and Paaß (2005) even state that this is where the most time while analyzing textual data is spent, namely on preprocessing and transforming it into an analyzable, and typically more structured dataset. Considering that textual data is typically processed by computers as a simple succession of various character strings, it does not come as a surprise that various preprocessing steps and data cleaning algorithms may be needed to enable the computer to understand the data and retrieve the desired knowledge. After preprocessing, the next step, as described by Fayyad, Piatetsky-Shapiro, and Smyth (1996), focuses on dimensionality reduction and data transformation with the goal of finding a data representation which ideally represents only the needed data concepts. This is then considered transformed data. Then, depending on the desired end result, data or text mining algorithms and methods need to be selected and applied to the transformed data set to discover the desired patterns. Thus, data mining is essentially only a step in the entire chain of knowledge discovery, though some researchers like to consider the entire KDD process when defining the term data mining. At this stage, there is an endless repertoire of algorithms and techniques which can be used, including for example data clustering, classification, summarization as well as various other supervised and unsupervised learning methods. Which techniques and algorithms should be applied highly depends on the desired end result and the approach of the analyst. This stage is often considered the modelling stage and frequently characterized by explorative analysis. Once the patterns are extracted, they then need to be further analyzed and any of the previous processing steps might need to be revisited to refine or adjust the model at any time. Visualization can be of great help at this stage as well as it provides a more intuitive access or representation of the mined patterns. Finally, the extracted knowledge needs to be put to use, archived, put in relation to other knowledge or used in some other form for further action. In the end, the result must have some kind of purpose and should be useful or beneficial to the end user, else the knowledge discovery process cannot be considered completed.

2.2.4 Text mining techniques and operations

There are various text mining activities and operations which can be performed for knowledge discovery purposes. This section focuses on providing a brief, high-level overview of the most common operations and particularly tries to clarify the three most time-extensive steps of the KDD process, namely preprocessing, transformation and data mining. It should however be noted that this is by no means supposed to be a comprehensive list, and with the continued research interest in this field, new

methods are being developed frequently. This section mainly serves the purpose of introducing the reader into the topic and explaining underlying concepts of various text mining applications. Furthermore, it should be pointed out, that many of the methods used in the text mining domain originate from different fields of research. In this regard, text mining methods heavily rely on statistics, artificial intelligence, machine learning, and data mining for example.

Preprocessing operations

As already mentioned before, preprocessing plays a prominent role in text mining and is one of the most time-consuming steps in the entire process of discovering knowledge from text. The underlying objective of applying preprocessing steps is to make the data more machine-readable and preparing the data for further analysis. After all, one should keep in mind that natural language evolved as the ideal way for humans to communicate and understand each other, and as Shwartz and Schank Roger (1987) argues natural language understanding is anything but a primitive cognitive function. It requires not only a large vocabulary and an understanding of the associated concepts behind words, but also extensive world knowledge to be able to discern hidden meaning, deal with ambiguity and correctly infer what may only be implied. A sentence which uses the same words may have two completely different meanings depending on the context it is presented in. Amongst others the ambiguities and implied contextual meanings are the very reason why machines have a hard time understanding natural language. Thus, the available data first needs to be preprocessed, structured and annotated in a way that machines can make sense of it. To put this back into the context of the KDD process in its entirety, after the selection of target data has been done based on a thorough understanding of the application domain, preprocessing is the second step in the KDD process.

Generally, there are different types of preprocessing methods and operations that can be applied to textual data. Which methods are relevant ultimately depends on the text mining task at hand, however typically a mixture of these techniques needs to be applied. Below, a selection of predominantly used preprocessing methods will be discussed.

As described by Hotho, Nürnberger, and Paaß (2005), general preprocessing steps can include, but are not limited to the following:

- **Tokenization:** This is typically the first step when analyzing a document or text data. Hereby all unnecessary characters, such as periods, commas, brackets and such get deleted from the text, so that only single words separated by single spaces remain. These words then make up the dictionary of the document.

- **Filtering**: Removes words from the dictionary or document representation, based on a given set of rules. For example, prepositions and articles can typically be removed as they do not contain much information.
- **Lemmatization**: Hereby words are transformed into a uniform form, i.e. verbs are transformed into their infinite tense and nouns into their singular form. Example would hereby be that *"am, are, is"* would all be transformed to *"be"* or "*girls, girls', girl's*" would all be transformed to *"girl".* This however requires large computational effort and the identification of the part-of-speech the word belongs to. Because of this, stemming is more frequently applied.
- **Stemming**: Hereby words are reduced to their stem or root typically by cutting of parts which contain little information. This is done by applying a set of rules which remove prefixes and suffixes or the plural *"s"* from words for example. The stem of *"calling"* and *"calls"* would in both cases be *"call"*.

Additionally, various linguistic preprocessing steps can be applied. The used methods hereby come from the Natural Language Processing (NLP) domain, which is defined by Bird, Klein, and Loper (2009) as *"any kind of computer manipulation of natural language"*. Typically, these methods use dictionaries and rule-based approaches to get their desired result. Some of the most frequently used linguistic preprocessing methods, as described by Hotho, Nürnberger, and Paaß (2005), include:

- **Part-Of-Speech (POS) Tagging:** Hereby every word is tagged based on the part of speech it corresponds to, e.g. noun, verb, adjective, preposition. Thus, every sentence is examined from a grammatical perspective.
- **Parsing:** Parsing goes one step further than POS tagging, as the sentence is transformed into a full parse tree. The parse tree shows each word's function within the sentence and how words are related to one another. An example of a simple parse tree is given in Figure 9 on page 33.
- **Word Sense Disambiguation:** Hereby an effort is made to determine the meaning of words and dissolve potential ambiguities. The same word may have various different meanings, e.g. "bank" can refer to a financial establishment or the piece of land that runs alongside a river. If the word is said in proximity to words such as "money" or "investment", these contextual clues help us understand what kind of "bank" is meant. Word sense disambiguation thus tries to identify the correct definition or interpretation based on contextual information.

Text encoding or transformation of textual data

Various approaches exist to make text more machine-readable and to ultimately allow computers to "understand" the meaning behind it. However fundamentally, text is stored and read by computers as subsequent strings of characters. Single words can be enriched with meaning by providing dictionary

definitions and interpretations. However how does a computer deal with sentences, passages or even lengthy text documents? How can text documents be represented so that machines can easily analyze them? As already mentioned in the previous section, to ultimately enable computerized textual understanding, the strings of characters first have to be preprocessed, structured and encoded in a way that allows further analysis. This part of the section thus focuses on text encoding or transformation of textual data into a more machine-readable format. This covers the third step of the KDD process. Hereby it is important to keep in mind that preprocessing and transformation of data are highly iterative steps, thus one should not consider the order of the KDD process as strictly sequential, but rather see the conjunction of the two steps as the preparation stage of text processing.

Now in regard to transformation, Hotho, Nürnberger, and Paaß (2005), who are the primary source for the description and information that follows, argue that in most text mining applications, text mining is now done built upon the idea that *"a text document is described based on the set of words contained in it".* This is also called *"bag-of-words representation*". They argue that frequently, vector space models are used for this purpose, particularly when it comes to information retrieval. Hereby the core idea is that a document can be represented as a vector whose elements or dimensions are the words contained within it. The simplest form would be binary vectors. Hereby the vector dimensions are all words included in the dictionary or document library. For any given document each vector element is then set to 1 if the word from the dictionary is mentioned in the document and it is set to 0 if the word is not mentioned in the document. To provide a simple example, if the dictionary contains the following words "**bee, keeper, honey, summer**", then the sentence "The **bees** are busy producing **honey**." would be represented by "1, 0, 1, 0", while the sentence "I see a lot of **bees** in the **summer**." would be represented by "1, 0, 0, 1". Encoding the text in such a way enables the use of vector operations for the comparison of different documents and many other applications, e.g. the similarity of documents can then be computed by calculating the distance between the vectors. This is particularly useful in information retrieval. Hereby the search query then has to be encoded in a similar way as the queried documents. However, this is just the simplest form of a vector space model. Additionally, for most text mining operations it is important to determine which words are more meaningful than others. For this purpose, certain words are often excluded from the dictionary or, in other words, a "stop word" list is used to eliminate or assign very low meaning to frequently used words such as prepositions, articles or generally common words. The importance of a word can also be represented within the vector space model, by assigning numerical weights to the words used. A word may seem more important or distinguishable for a document if it appears frequently throughout the document, measured by its term frequency, or if it is rather rare in the whole document collection and only occurs in a few documents, measured by the inverse document frequency. Furthermore, the document length should

be considered as well, which is why length normalization is often built into the weighing scheme. However, while this is one of the most frequently used models, it has its disadvantage, such as the fact that term independence is assumed and the connections between words are typically lost. Finally, it should be noted that there are many other ways to encode text documents for further processing and depending on what the final objective is the appropriate way should be chosen.

Main text mining operations

Once the text document has been prepared for the main analysis, one can move on to the fourth step of the KDD chain, namely data and text mining. Various different operations can be performed at this stage in order to extract the desired knowledge from the transformed data set. Most applications are hereby focused on one or multiple of the following operations as described by Gupta and Lehal (2009).

- **Information extraction:** Information or feature extraction focuses on the identification of facts and relations in textual documents. As the name suggests, hereby the main features of the text are extracted through the use of various algorithms and techniques. Usually, this is done by using dictionaries in combination with linguistic rules. The identified features hereby include named entities such as objects, people and places; relationships and associations; events and any other meaningful information or facts. While the main objective of a text mining application can be to extract specific features from text, typically some sort of feature or information extraction can be found in most text mining application.
- **Search and retrieval:** Search and retrieval methods enable users to search through large document libraries and easily retrieve the desired information through using various search functions. To enable an efficient search the documents first have to be indexed. As Hotho, Nürnberger, and Paaß (2005) mentioned, which words from a document should be indexed is typically determined by various index term selection algorithms. To provide an example, key terms may be selected based on the relative entropy of each word, namely a measure which quantifies the ability of a word to separate the document from other documents in searches. Additionally, Gupta and Lehal (2009) mention that search and retrieval may make use of various other methods, ranging from basic search functions such as the use of Boolean operators or wildcard symbols to more advances search functions which require linguistic processing or fuzzy searches for example.

- **Categorization (Supervised Classification):** Hereby documents are classified into distinctive, predefined categories to identify the main themes of a document. The categories are hereby either predefined by the application developer or based on user input. However, what is distinguishing between categorization and the later mentioned clustering is that the categories are fixed in number and predefined. Thus, the documents are sorted into these categories based on the best fit. Each document can hereby belong to one or multiple categories. As Gupta and Lehal (2009) point out, categorization techniques typically do not need the text to be understood, but rather work on the basis of word counts and the consideration of related terms and synonyms. Categorization further utilizes machine learning techniques. Often supervised learning algorithms are applied, which first require a training set of labeled documents from which the program derives its classification rules or the categorization model. Based on this model, unlabeled documents are then categorized automatically. To determine the quality of the model, typically a testing set of labelled documents is set aside and not used for training. The results of the categorization algorithm are then evaluated based on this testing set. Some frequently used classification concepts, as described by Hotho, Nürnberger, and Paaß (2005) are decision trees, Naïve Bayesian Classifiers, Nearest Neighbor Classification and Support Vector Machines. These concepts apply to both supervised classification (categorization) and unsupervised classification (clustering). Additionally, it is important to note that classification methods, regardless whether supervised or unsupervised can be applied to separate text passages or sentences as well. In this case the sentence would be considered the "document" to be classified.
- **Clustering (Unsupervised Classification):** Clustering also has the classification of documents as its objective. As Gupta and Lehal (2009) point out, hereby the categories are not predefined, but instead the documents are classified based on similarity measures. Since there are no predefined categories, there is also no labelled training set that can be used as basis for classification. The examples provided are unlabeled and the program has to identify suitable categories hidden in the text. This is called an unsupervised classification. The more similar documents are to each other, the more likely they share the same cluster. Thus, if a document is not similar to any of the already identified categories, then a new category is made. This would not be possible with a given set of predefined categories and has the result that no document remains uncategorized. Additionally, it should be mentioned that documents can be in more than one cluster, as clusters can overlap. Clustering algorithms hereby include e.g. hierarchical clustering, fuzzy clustering and self-organizing maps. An application example would be given by user profiling on the web, where information on users is often easily

accessible. Based on how much data about a user can be collected, he or she can get assigned to a cluster of similar users, which allows the prediction of future behavior to some extent.

- **Summarization:** Summarization focuses on providing a short and concise description of the textual data, which includes the most important aspects of the text. In most summarization users can determine how much the text should be shortened, e.g. by specifying the approximate % of words to be used as compared to the original text or by specifying the number of sentences. Gupta and Lehal (2009) argue that there are two types of summarization approaches: shallow approaches which simply try to extract the most important sentences and deeper approaches which analyze the text on a semantic level and perform more complex operations to generate a summary. An example of a shallow approach could be a basic summarization application which simply looks for key phrases such as *"in conclusion"* or *"to sum up"* and then extract the paragraph that follows these phrases. More sophisticated approaches may identify the most important words in the heading or abstract and then find the most significant sentences or concepts based on that, utilizing linguistic concepts and focusing on the semantics of the text. Figure 8 below provides an example of what steps can be included in a text summarization process.

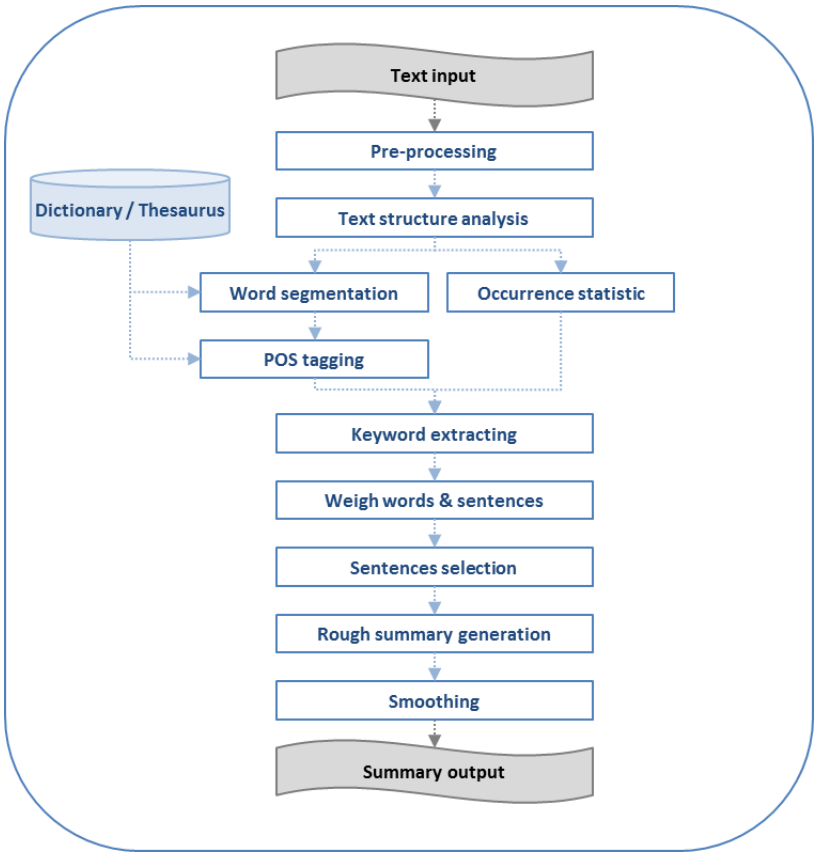

Figure 8 Example of text summarization process as adapted from Gupta and Lehal (2009)

- **Visualization:** Particularly in recent years a large focus has been put on the visualization of text mining results. This is largely due to humans being able to process graphical data more quickly than data which is presented in textual or tabular format. A focus is thus laid upon the understandability of text mining results. The uncovered patterns or knowledge are presented in some sort of visual form, e.g. a map or a hierarchical structure. As Gupta and Lehal (2009) mention, 2D or 3D representations with interactive features such as zooming or filtering are among the most common visualization types. The main objectives are to enable quick decision making and the exploration of large document sets. Hereby it is important that the visualization is designed in a way that the interpretation is intuitive for the end user.

3. Text analytics for conceptual modelling

This chapter now tries to merge the two previously introduced areas and apply text mining to conceptual modelling. First the reasons why this may be beneficial are identified and potential application areas are outlined. Afterwards, one area is examined more closely, namely the application of text analytics throughout the model design process itself. Hereby, a closer look is given to information extraction approaches, the nature of text as it appears in conceptual modelling and how linguistic concepts relate and could potentially be mapped to modelling constructs. Finally, a generic text analysis approach for conceptual modelling from linguistic input is presented. To serve as a starting point for further research into this area and to support the development of text mining applications for conceptual modelling in the future, this chapter is concluded by a list of free tools available for both, conceptual modelling and text analytics.

3.1 Rationale and application areas

To come back to the general definition of conceptual models, Wand et al. (1995) define conceptual models as formal representations of some part of a real-world system ultimately serving the purpose of communication and understanding. Given that conceptual models serve as means to communicate some knowledge about a particular domain or "system under study", it seems natural to look at the way humans communicate in general as a first step for trying to find a more efficient, correct and understandable representation of the knowledge to be conveyed. This is where natural language comes in to play. It is highly complex, filled with ambiguity and yet structured and organized in a way that understanding comes easy to us. To ultimately develop a conceptual model representation that seems natural to interpret and is easily understandable, Wand et al. (1995) proposed to first look at how humans organize knowledge, what general concepts are used to describe phenomena and how interactions, dependencies and motions are typically communicated. They propose that models of human knowledge should be considered as the foundation for conceptual modelling and particularly suggest ontology, concept theory and linguistics to provide the basis for conceptual understanding. Given that language and human understanding seems to play such a large role in conceptual modelling, it seems reasonable to assume that natural language processing could be of use in this domain. Nevertheless, it may not be immediately clear where text analytics can be applied. This subchapter thus focuses on outlining potential applications in the conceptual modelling domain. Two areas are hereby examined more closely, namely the model design process itself and the utilization of the conceptual model. While both areas are outlined within this subchapter, the rest of the thesis focuses on the area which provides the highest potential benefit, namely the model design process itself.

As discussed in section 2.1.2, the model design process consists of three distinct stages: annotation, mapping and model design. The first and most apparent potential application area is the automatic analysis of natural language descriptions or functional specifications at the beginning of the modelling process. Thus, mainly supporting the annotation and mapping stages of conceptual modelling. This can then even be taken one step further to enable the automated model creation from natural language descriptions alone. This is also where a lot of previous research efforts can be found, see Friedrich, Mendling, and Puhlmann (2011), Yue, Briand, and Labiche (2011) or Sukanya and Biruntha (2012) for examples. Particularly in the software development domain, one can observe many efforts being made to automate the entire modelling process from the beginning to the end. To provide an example, Sagar and Abirami (2014) consider the requirements gathering phase at the beginning of the software development process to be one of the most critical phases. This phase is characterized by lengthy documents or conversations describing the expected requirements in natural language. They emphasize the importance of clear communication and understanding between business users and developers at this stage and argue that conceptual models are often utilized to demonstrate the understanding of requirements, and can help to clear up any potential misunderstandings and as an early stage of product design. Considerable time is however spent gathering the needed information and creating the model. To provide another example of how time-consuming the modelling process can be, Friedrich, Mendling, and Puhlmann (2011) argue that up to 60% of time spent during a typical process management project is spent on the creation of as-is models. Undoubtedly, time savings in this region would be largely beneficial. Sagar and Abirami (2014) discuss that by automating the modelling process on the basis of natural language text alone, one could focus one's attention on improving, analyzing and refining the model rather than the actual modelling process itself. In addition to time savings, text analytics- enabled automated model generation also allows less tech-savvy users to create conceptual models by simply describing the process or system under study in textual form. Hereby it is further important to stress that everything that should be included in a model, should ideally be explicitly stated in the textual description as inference is still one of the major setbacks of computerized textual understanding. Nevertheless, the ultimate goal is that computers can deal with unrestricted natural language text.

However, enabling automated conceptual model generation by applying natural language processing techniques during the text annotation and mapping stages is not the only application area for text analytics in conceptual modelling. Text analytics can also be of use on already finished or generated conceptual models as most models still incorporate textual information in either the construct descriptions or additional information which is attached to the constructs. Text analytics can be particularly useful, because they enable semantic applications. Awad, Polyvyanyy, and Weske (2008),

for example, propose a concept for semantic querying of BPMN models, utilizing various text analysis techniques for this purpose. In their paper they represent the textual descriptions within the model with an enhanced topic-based vector space model and compute similarity measures to effectively perform search and retrieval functions on conceptual models. Search and retrieval is however not the only application area where text analytics can be useful after the model has already been designed. Bögl et al. (2014), for example, designed and patented a method and an apparatus for the automation of semantic annotation of process models. They demonstrate their methodology on a detailed example of EPC models. Further application areas for example include the identification of common model vocabulary, the computation of similarities between models, or the identification of model metrics.

3.2 Text analysis throughout the model design process

As the remainder of the thesis is focuses on text analytics throughout the model design process itself and the main difficulty hereby lies in identifying and extracting the needed information from natural language input, it may be useful to have a look at what methods can be used in this regard, how rules for modelling construct extraction can be derived and how to approach model generation from natural language in general.

3.2.1 Information extraction for conceptual modelling

As Hogenboom, Frasincar, and Kaymak (2010) pointed out, in general, there are three kinds of approaches for extracting information from natural language text: pattern-based approaches, statistical approaches and hybrid approaches. Pattern-based approaches are knowledge-driven. Hereby, information is extracted based on predefined patterns or rules, derived from human knowledge about the domain as well as linguistic and lexical concepts. Statistical approaches on the other hand are data-driven, with information being extracted based on quantitative methods and statistical evidence. Statistical approaches identify their own patterns or rules by utilizing machine-learning principles based on a large annotated training library. Hereby the size and nature of the training library is the decisive factor when it comes to the quality of the identified patterns. Unfortunately, obtaining a large, domain-relevant and already annotated or mapped text is the main challenge in this regard. Additionally, the lack of semantic consideration is one of the main reasons why pattern-based or hybrid approaches seem to be preferred. Either of these approaches can however be utilized for the purpose of conceptual modelling and should be considered when building such a system.

However, as pattern-based approaches are more prevalent, it is important to understand how these approaches work in detail. In general, when using pattern-based approaches for conceptual modelling, a set of heuristics or rules should be identified for each modelling construct. This set of heuristics is then used to determine whether a certain word or phrase belongs to that particular construct. Ideally, both syntactic and semantic information should be considered, however employing only syntactic-based heuristics has shown to yield satisfactory, although not ideal, results as well. See Omar, Hanna, and McKevitt (2004) for example. There are various ways to derive useful heuristics or rules for pattern-based approaches. Heuristics can, for example, be built upon linguistic concepts, they can be based on previous research in the area or be a result of the verbalization of the annotation thought process. To provide an example of a heuristic, the following rule as described by Chen (1983), can be used amongst others to identify entity types in Entity-Relationship models: *"A common noun (such as "person," "chair") in English corresponds to an entity type in an ER diagram."* However, one heuristic is not enough to confidently determine whether a word is relevant for any given model and which category a word belongs to. Rather a set of heuristic needs to be built for each construct. Furthermore, as some heuristics may be better indicators for a specific construct than others, this type of concept is often built as a weighted approach. Hereby weights are assigned to each heuristic, depending on how reliable it is as an indicator. The words are then tagged based on the cumulative weight. The main problem with pattern-based approaches is however that rules can only effectively be applied if the information needed to test the heuristic is complete and accurate. In the case of heuristics which are based on linguistic concepts, it is assumed that each text can be fully and correctly parsed and all the tags needed to determine whether a heuristic is met or not are correctly identified. It thus assumes that computers already understand human language to such an extent that parsing a text and identifying the correct part-of-speech a word belongs to, for example, is not an obstacle. Unfortunately, while many great parsing tools exist, see section 3.3.2 for examples, the inherent ambiguity and inference in natural language is still impairing the performance of such tools. Thus, a strong syntactic and semantic text analysis is the needed foundation of such an approach and it may make sense to consider the fact that certain words might be parsed wrongly when evaluating certain heuristics.

3.2.2 Linking linguistic concepts to modelling constructs

Thus, for the purpose of discovering underlying patterns, it can be useful to understand how linguistic concepts relate to modelling constructs in general. To gain an understanding of what linguistic concepts are particularly relevant, an overview of basic lexical categories is provided in the table below and a refresher on sentence structures and grammatical decomposition is provided afterwards. The

lexical category hereby refers to the parts-of-speech (POS) a word can belong to, which determine the word's function or meaning within a sentence.

Table 2 Main POS categories

POS category	Definition [3]	Examples
Adjective	"An adjective modifies or describes a noun or pronoun."	nice, healthy
Adverb	"An adverb modifies or describes a verb, an adjective, or another adverb."	quickly, well, thoroughly
Conjunction	"A conjunction joins words, phrases, or clauses."	and, or, but
Determiner	"A determiner is a word that introduces a noun."[4]	A, an, the, many
Interjection	"An interjection is a word used to express emotion."	Oh, My, Uh
Noun	"A noun is the name of a person, place, thing, or idea."	beekeeper, hives
Preposition	"A preposition is a word placed before a noun or pronoun to form a phrase modifying another word in the sentence."	by, until, from
Pronoun	"A pronoun is a word used in place of a noun."	he, she, it, they
Verb	"A verb expresses action or being."	made, flies, eat

In general, sentences are made up of words which belong to a certain POS category depending on the context. These words however cannot always be analyzed independently of each other, as they often appear in the form of phrases. Phrases consist of multiple words which form a unit within a sentence and typically follow a certain structure. Thereby one word is the root of the phrase which is accompanied by further words belonging to the root. For each POS category, there is a corresponding phrasal category with a certain root. This means that there are noun phrases, verb phrases and adjectival phrases amongst others. For example, a typical noun phrase would be "the hard-working beekeeper". Hereby the noun "beekeeper" is the root and is accompanied by a determiner "the" and an adjective "hard-working". Multiple phrases can further be joined to construct clauses. A clause hereby must at least contain one noun phrase, which constitutes the subject of the sentence, and one verb phrase, which constitutes the predicate. Each grammatically correct sentence must at least have one clause.

[3] The definitions for all categories other than the one for the determiner, were taken from Butte College (2018)'s website http://www.butte.edu/departments/cas/tipsheets/grammar/parts_of_speech.html, accessed on February 24th 2018

[4] The definition of a determiner was taken from https://en.oxforddictionaries.com/grammar/determiners, accessed on June 10th 2018

An example is hereby given in Figure 9, where a sentence (S) is split into a noun phrase (NP) and a verb phrase (VP), whereby the verb phrase is further split into a verb (V) and a noun phrase, with the latter consisting of a determiner (Det) and a noun (N).

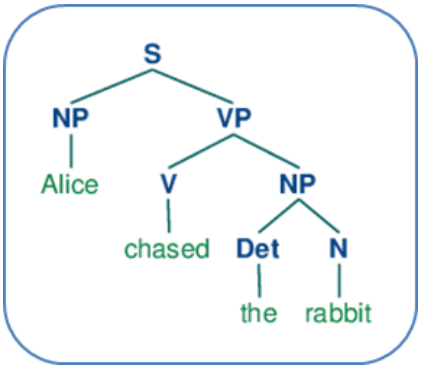

Figure 9 Example of a parse tree taken from Bird, Klein, and Loper (2009)

There are various open-source and proprietary tools for text analytics which already provide a wide range of functionalities and methods that help identify linguistic concepts in free-flowing text, e.g. part of speech (POS) taggers, various parsers and named entity recognition tools. While no tool is perfect, as each tool still struggles with ambiguity and inference, amongst other problems, they provide a great starting point for computerized text understanding, can be improved upon, enriched with semantic analysis and adapted for one's own specialized needs. Having a strong basis to test patterns against is very important for the successful extraction of relevant concepts, thus when designing a system that automatically extracts modelling concepts, paying attention to the preprocessing stage and the quality of the parser is of great importance.

As a next step, modelling constructs can be examined more closely, to identify whether it is possible to derive general transformation rules which could be used to map identified linguistic concepts to the respective modeling constructs. As previously mentioned, Wand et al. (1995) point out that the constructs and semantics of a modelling language determine which aspects of the system under study are represented and in what way. Thus, when presented with a textual description of a system under study, one has to consider the modelling constructs of the modelling language one will use to determine which parts of a text are relevant and in what form. Being able to derive certain rules either statistically or manually, which ideally map modelling constructs onto free-flowing text is one of the main challenges in text-to-model generation. Hereby, it is important to keep in mind that different languages have different modelling constructs, and while some concepts may be similar, they have small differences which need to be considered when mapping linguistic concepts to modelling constructs. Furthermore, modelling elements typically have to adhere to specific rules and naming conventions defined within the language's syntax. In this regard, Bögl et al. (2014) argue that these naming conventions are important for standardization and lower the influence or subjectivity of the people constructing the models. Standardizing the output and bringing the identified words into the correct form may become important in the mapping stage of the model design process.

Table 3 below, provides a list of some typical modelling constructs and some loose linkages of the modelling constructs to the previously presented POS categories. The linkages were hereby partially

derived on the basis of previous work from Chen (1983) for ER models and Bögl et al. (2008) for EPC diagrams, from the construct definitions or from the verbalization of exemplary annotation processes. One thing that becomes evident when looking at the table, is that some constructs within the same modelling language can be derived from the same POS categories, thus the subtle differences in definition are important for constructing sound transformation rules or heuristics which would enable the correct mapping to the corresponding modelling construct. The table below can however be used as a starting point for constructing such rules.

Table 3 Potential POS linkages for modelling constructs

Model	Construct	Description and Potential POS Linkage
Entity- Relationship Model (ER)	Entities	Entities are important constructs, objects or things about which information needs to be stored. Examples include “course”, “student” or “lecturer”. **Potential linkages:** common nouns (entity types) or proper nouns (entities)
	Relations	Relationships in ER-diagrams typically describe how entities are linked to one another or, in other words, their associations, e.g. “takes” or “teaches”. **Potential linkages:** transitive verb or verb + preposition
	Cardinalities	Cardinalities describe the degree of a relationship or how many such relationships can exist. Examples include “one to one” or “many to one”. **Potential linkages:** noun form (singular/plural) or certain adjectives
	Attributes	Attributes are concepts that capture important characteristics of an entity or relationship. As each instance of an entity must be uniquely identifiable, typically entities have one attribute or a combination of them that serves as the primary key, uniquely identifying an instance of an entity. Examples include “course number”, “student name”. **Potential linkages:** common nouns, adjectives (entities) or adverbs (relations)
	Links	As Attributes belong to entities or relationships, there are links connecting them to these constructs. **Potential linkages:** certain verbs such as “have” or position in noun phrase

Event-driven Process Chains (EPC)	Functions	Functions describe some sort of action or transformative process that needs to happen in order to change the state of something, e.g. “prepare invoice”. **Potential linkages:** verb phrase
	Events	Events describe a certain state or circumstances given that are either the trigger or the result of a function. An example would be “invoice sent”. **Potential linkages:** noun/noun phrase + verb (past tense)
	Operations	Operations in event-driven process chains, describe logical relationships, such as the “and”, “or” or “xor” operators which control the flow of the process or incite decisions. Potential linkages: conjunction
	Links	Opposed to ER diagrams, links in EPC diagrams mainly focus on the order or the flow of either functions and event (control flow) or information and data (information flow), e.g. “X happens after Y” and “ Y happens after Z”. Occasionally, links do also appear to show connections between certain constructs, e.g. “X uses A” or “B belongs to Y”. **Potential linkages:** preposition, adverb and conjunction
Business Process Model Notation (BPMN)	Activities	In the Business Process Model Notation, activities are tasks or steps which represent a unit of work performed as part of a process. Similar to the previously introduced functions in EPC, they typically are of a transformative nature. An example would be “send email” or “check invoice”. **Potential linkages:** verb phrase
	Events	Events represent something that happens within a process that can be either a trigger or a result of some process steps or that can be triggered by external circumstances. In BPMN there are start, intermediate and end events, with many subtypes depicted by certain symbols. **Potential linkages:** noun/noun phrase + verb (past tense)
	Gateways	Gateways are used in BPMN to control the flow of a business process by allowing process branching. They specify whether at a certain point a task splits into various simultaneous paths, or takes on a certain course depending on the circumstances for example. There are parallel (“and”), inclusive (“or”), exclusive (“xor”) or event-driven gateways. Potential linkages: conjunction
	Flow	The flow in process diagrams mainly focus on the order of activities, events or information, with the occasional linking of objects to those constructs. **Potential linkages:** preposition, adverb and conjunction

Petri Nets (PN) – Process oriented view	Places	Places determine the various states that a process can be in, e.g. “order is received”. They can hold tokens, whose position within the petri net determines the current state of the process, e.g. if an order is received the above place holds one token. **Potential linkages:** noun/noun phrase + verb
	Transition	Transitions are events or things that may happen if certain circumstances are met. The circumstances hereby depend on the number of available tokens in the input places. Once triggered a transition generates a certain number of tokens in its output places. To provide a simple example, one token in the “order is received” place could be enough to trigger the transition “start processing” which would then generate an output token in the place “order is being processed”. **Potential linkages:** verb + noun/ noun phrase
	Arcs	A petri net is connected through directed arcs that specify in which direction a process moves and what conditions need to be met in order to transition to the next state. In this sense, arcs connect places to transitions and transitions to places. **Potential linkages:** preposition, adverb and conjunction
	Arc weights	Each arc has a specific weight assigned to it, which determines how many tokens need to be available for a transition to be activated and how many tokens are the result of a transition. The default arc weight is one. Thus if one token is available for a default arc, the transition will be enabled and the corresponding output tokens will be generated. **Potential linkages:** numerical attribute or pronominal, e.g. “there are three”

3.2.3 Generic text analysis approach for conceptual model generation

In this section a generic approach for the utilization of text analytics for conceptual model generation is presented. The introduced approach provides a very high-level depiction of text-mining enabled model design as is not only focused on the extraction of the relevant constructs, but on the full model design process. It is not a step-by-step solution, but should rather be seen as a starting point which highlights the main components needed and their influencing factors for potential future applications or research. As with other generic concepts, its generalized nature enables system architects, developers or researchers to easily adapt the given concept based on their individual needs, implementation environment and application specifics. Chapter 4 will later provide an illustrative example of how this generic concept can be used as the basis for building a specialized application, namely a text mining-enabled tool for semi-automated ER model generation.

First of all, already available literature was examined and specific examples of text to model generation were identified. Many approaches are hereby applying pattern-based extraction methods. To provide a detailed example, Sagar and Abirami (2014) demonstrate how conceptual models can be derived from natural language requirement specifications. In the first step of their approach, general preprocessing methods, such as tokenization, are applied on the requirement specification text. Then linguistic preprocessing or syntactic feature extraction is applied and a natural language parser is utilized to provide a grammatical context and perform part-of-speech tagging. In the next step a set of rules is applied to extract the relevant design elements such as objects and relationships. An example of a design rule is given by "*Any Noun that appears as a subject is always a class*". Afterwards relation type classification is performed. They further suggest some post-processing steps to refine the results and remove elements which do not necessarily add anything of relevance to the model. Finally, the results are visualized and the conceptual model is generated. As can be seen from the example above, the automation of the model design process would not be possible without the application of a variety of text analysis techniques. However, to deal with the known problems in text analytics, their model still requires the text specifications to be adapted beforehand. The specifications hereby must consist of grammatically correct sentences, each sentence should name all needed references to avoid the need for reference resolution, words such as "*him, her or it*" should therefore be avoided and no negative statements should be entailed. With these given constraints for the input text, their model yields satisfactory results, nevertheless it is missing the incorporation of semantic factors and constricts the use of true natural language.

Montes et al. (2008), who also deal with the objective of automated conceptual modeling from requirement specifications, proposes two steps to be applied on the natural language input text, to lessen the needed constraints on the input and produce an ambiguity-free and labeled text which can further be used as the basis for constructing the model. As can be seen from Figure 10 below, they suggests that both a syntactical and semantic analysis should be performed on input text. Syntactical analysis would thereby decompose the input text into simple structures and produce a labelled text, whereas semantic analysis would deal with ambiguities and inferences.

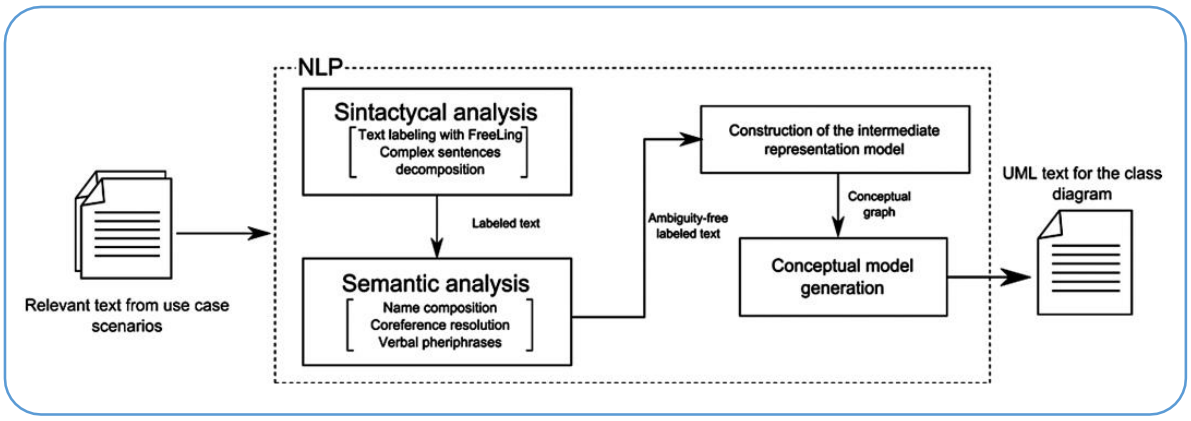

Figure 10 Transformation method of requirements models into conceptual models from Montes et al. (2008)

In the literature, various other approaches for the automation of conceptual modelling from text specifications can be observed and various implementations for specific modelling languages are presented. See Yue, Briand, and Labiche (2010) for an example of automated UML modelling from text, Friedrich, Mendling, and Puhlmann (2011) or Sintoris and Vergidis (2017) for BPMN modelling and Omar, Hanna, and McKevitt (2004) for ER modelling.

After examining previous research, the generic concept presented here was derived with two things in mind: firstly, a high level of abstraction was desired in order to keep the approach as generic as possible, and secondly, a focus was put on the model design process itself to stress the human-centric nature of conceptual models and to identify how text analytics could support each stage of the process. To meet the first objective, thus to remain generic, the presented concept has been designed in a way to be independent of technical details such as modelling language or implementation platform. A higher-level and more business-oriented perspective was targeted from the onset. The approach in this section is thus presented as platform independent concept, providing a high degree of domain abstraction and focusing on generic processes and activities. This is also the aspect that differentiates this work from others, as previous research has mainly focused on applying text analytics to specific modelling languages or specific types of models, see the examples mentioned at the end of the previous paragraph. Secondly, other approaches presented to date, such as Btoush and Hammad (2015) or the previously explained approaches from Sagar and Abirami (2014) and Montes et al. (2008), mostly focus on the text mining process itself with the application to the modelling domain being an environmental factor. While considering the text mining process is undeniably important for the construction of solid text mining applications, only focusing on this perspective has two downsides: on one hand, it shifts the focus from modelling to text mining, potentially undermining the importance of conceptual modelling considerations; on the other hand, while it may give a good procedural overview of how an application could ultimately function, it does not cover all the necessary factors that come into play when building such an application. This is the point which the generic approach tries to

address. Nevertheless, it might be helpful to understand how the text mining process could look like in respect to text-to-model generation. To provide a high-level overview of such a process, Fayyad, Piatetsky-Shapiro, and Smyth (1996)'s KDD process has been adapted for conceptual modelling in Figure 11 below.

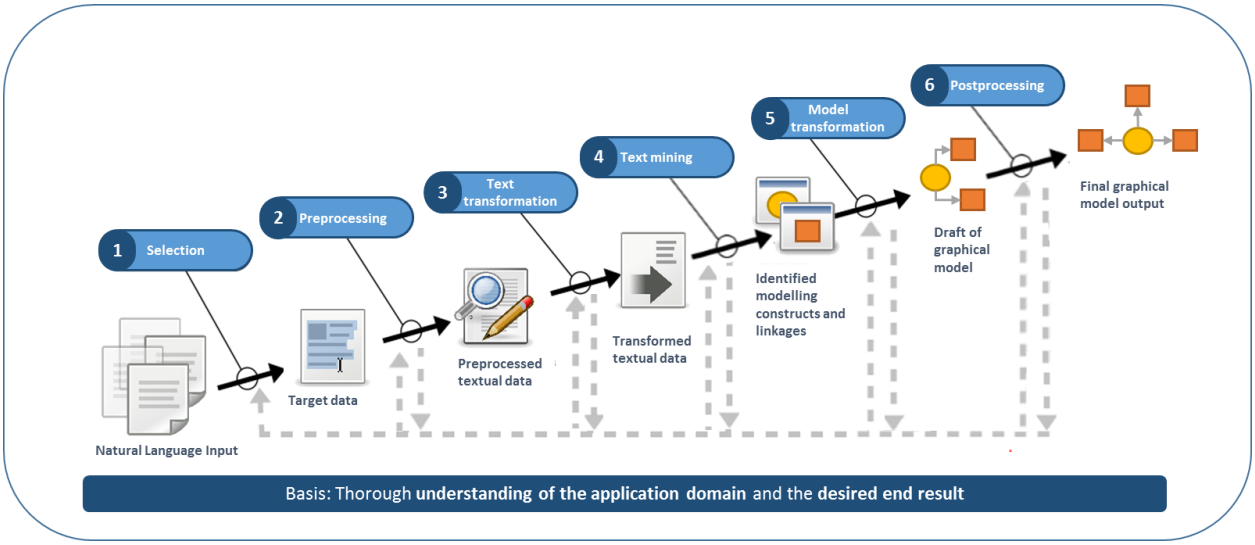

Figure 11 Fayyad, Piatetsky-Shapiro, and Smyth (1996)'s KDD process adapted for Conceptual Modelling

As in the KDD process, the basis for text mining enabled conceptual modelling is built upon a thorough understanding of the application domain and the desired end result. Textual data expressed in natural language constitutes the input for the entire process. Based on the objective, target data is then selected as a first step. This mainly relates to the selection of text paragraphs relevant for the conceptual model. For simpler applications, this step could potentially be covered by implementing input constraints and requiring the rewriting of complicated or ambiguous texts into a simpler text form by the users themselves. For more automated solutions, this could be done on the basis of set patterns and algorithms. In the end, target data should ideally consist of only the text specifications relevant for the model, omitting any unnecessary paragraphs. The second step, is focused on preprocessing the data and putting it into an analyzable format. This includes the various general and linguistic preprocessing operations discussed in section 2.2.4, namely tokenization, stemming, word sense disambiguation, POS tagging and parsing amongst others. The next step, text transformation deals with getting the text into a more machine-readable format as typically algorithms and text mining operations require a more structured data format. As a fourth step, the main text mining operations come into play. Of particular importance is hereby information extraction, which has been discussed in detail in section 3.2.1. Information extraction can be pattern-based, statistical or hybrid. As Allahyari et al. (2017) mentioned, in statistical and hybrid information extraction, classification- based approaches are commonly used given that their underlying concepts can also be applied to text

passages within documents and not just to a collection of different documents. Classification methods hereby include supervised classification (categorization) and unsupervised classification (clustering) and can be used to determine whether a sentence is particularly relevant for a given category and how to classify each word within the sentence for example. The result of this step would be to arrive at a set of identified modelling constructs and their linkages. Again, the highly iterative nature of the entire process should be stressed, as different text mining operations may require a different data format for example making the looping over previous steps necessary before arriving at a satisfactory result. Once, the constructs and linkages have been identified, the fifth step, model transformation, can begin. This step is mainly focuses on applying the language syntax and notation to translate the results of the text mining operations into a draft graphical output. Hereby consideration should also be given to the initial positioning of the elements, as overlaps should be avoided for example, and constructs which are connected to one another should be kept in each other's proximity. Finally, the last step, post-processing focuses on model refinement. As Seidewitz (2003) has mentioned, ultimately *"an interpretation of a model gives a model meaning."* Thus making sure that a model really represents what one is trying to depict and communicates its features effectively, is a very important step of the process. Taking the time to refine the model and adjust it for one's needs is essential. This can include the addition of missing constructs, the exclusion of redundant ones, the re-positioning of elements and other adjustments based on the evaluation of the given model. Often at this stage, user involvement is beneficial.

While being aware of the above process is helpful when diving into the specifics of developing an application, the generic approach presented below provides a more high-level perspective, and thus might be a better starting point for future research. The generic approach hereby specifies how text analytics can be of help in each step of the model design process: from annotating the model, over mapping the modelling constructs to the actual model design. The reason why the concept is so closely linked to the model design process as opposed to the text mining process, can be traced back to the ultimate purpose of conceptual models, namely to create a representation of a system under study that is used by humans for communication and understanding. Text mining is hereby seen as an enabler for automation, not as the main method for model creation. Furthermore, as Mylopoulos (1992) has pointed out, conceptual models should be structured similarly to the way we humans structure knowledge about any given subject. Thus, to end up at a well-structured model, it may make sense to follow the same steps a human would take to extract the most important concepts, structure the information given and ultimately to create a given model. For this purpose, this thesis suggests the following generic text analysis approach for conceptual model generation:

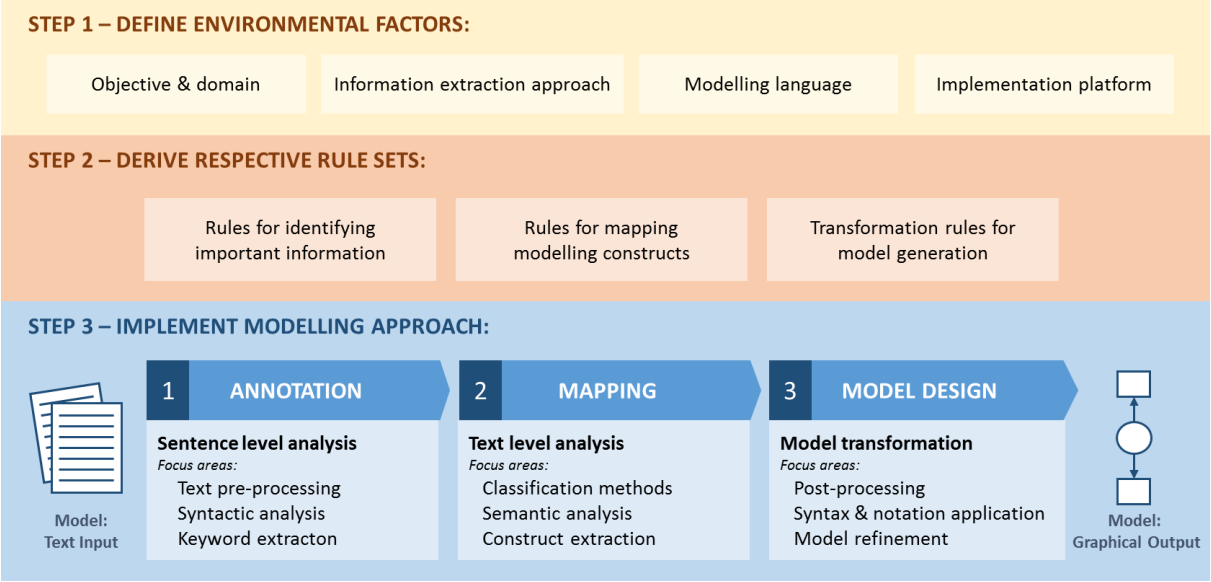

Figure 12 Generic text analysis approach for conceptual model generation

Overall, the generic approach is structured into three distinctive steps or stages. The first step is focused on preparation by defining the environmental factors. Particularly the following four categories should be discussed and decided upon in the beginning as they can have a large influence on the application implementation: (1) the objective & domain, (2) the information extraction approach, (3) the modelling language, and the (4) implementation platform. A sound understanding of the objective and domain builds the basis for any knowledge discovery process, and thus also the basis for conceptual model generation. One should have a clear understanding of what one wants to achieve, including the context it will be used in. Any further design needs to be done with this in mind and it is recommended to revisit this factor periodically through each step of the process, ensuring that the entire application is still in line with the overall objective. Secondly, while defining the environmental factors, one should decide on an information extraction approach, as the requirements and prerequisites for pattern based approaches differ largely from their statistical counterparts. As discussed in section 3.2.1, there are three types of information extraction approaches: pattern-based approaches, statistical approaches and hybrid approaches. A decision about the information extraction approach to be taken should be taken early in the process. Thirdly, the modelling language which will be used needs to be determined. Hereby, either only one language could be relevant or one could desire to create a hybrid tool, allowing the use of various languages. In each case knowing which language or languages the tool will have to be able to deal with is needed, particularly when it comes to the graphical model generation, but also in the early stages of the text mining process, should a pattern based approach have been chosen. Finally, the implementation platform also influences the final application design.

The second stage of the presented approach is the derivation of the respective rule sets. Three rule sets are hereby of importance: (1) Rules for identifying important information, (2) Rules for mapping modelling constructs, and (3) Transformation rules for model generation. Figure 13 below, depicts the influencing relationships that exist between the first two stages of the generic approach. The rule sets for identifying important information and for mapping modelling constructs largely depend on the objective and domain, the information extraction approach and modelling language, defined in the first step of the approach. The transformation rules for model generation on the other hand largely depend on the modelling language and implementation platform.

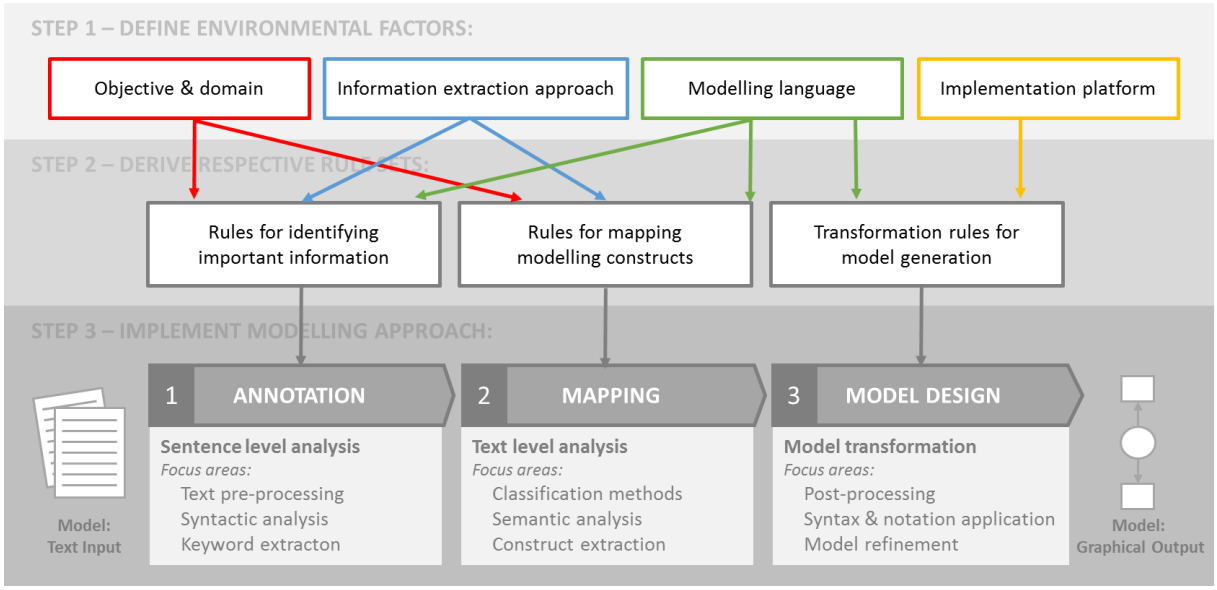

Figure 13 Influencing relationships of the generic text analysis approach for conceptual model generation

To provide an example, the objective & domain may influence what information is important. If an application is built for a highly specialized or complex domain, then the inclusion of lexicons and knowledge bases into the rules might be necessary to discern relevant information from irrelevant one and to help with ambiguity resolution. Domain-specific words may, for example, be assigned a higher weight during rule evaluation, indicative of their importance, whereas for simple domains this part may be omitted. Another example of an influencing factor, is the information extraction approach which determines how the rules for identifying important information and for mapping constructs are derived. If they are derived statistically, then a large training set of annotated data has to be provided. Should it be a highly specialized domain, then getting such examples may however prove to be very difficult. On the other hand, for pattern based approaches, rules can be derived from either literature review, the verbalization of the modelling process or the linguistic linkages presented in section 3.2.2. The rule patterns on the next page provide examples of commonly found rule structures, and while only few are presented, this could be a future starting point for constructing and testing various rules.

PATTERN A: **<Part of speech / grammatical structure>** may indicate **<modelling construct>.**

Examples: *ER: A common noun may indicate an entity.*
BPMN: A noun phrase followed by a verb may indicate an event.
EPC: A conjunction may indicate an operation.

PATTERN B: Certain words, namely **<list of words>**, may indicate a **<modelling construct>.**

Examples: *ER: Certain words, namely "number, no, code, date, type, volume, birth, id, identifier, name" may indicate attributes.*
BPMN: Certain wsords, namely "and, or, either" may indicate gateways.

PATTERN C: **<Modelling constructs>** are likely to be mentioned in the same sentence as the **<modelling constructs>** they are linked to, or in their proximity.

Examples: *ER: Attributes are likely to be mentioned in the same sentence as the entity they are linked to, or in their proximity.*
Petri Nets: Arc weights are likely to be mentioned in the same sentence as the arcs they are linked to, or in their proximity.

PATTERN D: A sentence structured as **<sentence structure>**, may indicate a **<modelling constructs>.**

Examples: *ER: A sentence structured as "subject-verb-object", may indicate that the subject is an entity and the objects its attributes.*
BPMN: A sentence structured as "subject-verb-object", may indicate that the verb and object describes an activity.

PATTERN E: If a **<part of speech / grammatical structure/certain words>** is mentioned at **<a certain position in a sentence>,** this may indicate the presence of a **<modelling construct>.**

Examples: *ER: If a determiner is mentioned before an entity, this may indicate a cardinality.*
EPC: If an adverb is mentioned at the beginning of a clause, this may indicate a link.

Some of these rules may be more important for the identification of important information, e.g. the rules that fall into pattern A. Others, may be more important for mapping the modelling constructs, such as the rules that fall into pattern C. In any case, the development of a rule-based text mining application will require considerable effort to be invested in identifying and testing various rules.

The third and final stage of the generic approach focuses on the implementation of the modelling approach, while highlighting the focus areas of each step. In the annotation stage of the model design process, sentence level analysis needs to be performed. When thinking back to the text mining process presented in Figure 11 (page 39), the application would hereby already have run through the first four steps of the process. Namely, selection, preprocessing, text transformation and text mining. A focus needs to hereby be laid upon preprocessing the text and fundamental syntactical analysis. The output is presented as extracted keywords which may be of importance for certain modelling constructs. The mapping stage of the modelling approach would then typically require another iteration through the text mining process, this time focusing on the text as a whole, rather than single sentences. The key success factors would hereby entail the incorporation of semantic analysis and the correct classification of constructs and their linkages through the use of a soundly developed rule set for mapping modelling constructs from textual descriptions. After this stage, both the modelling constructs, as well as their linkages should be available. It is important to note, that this approach does not suggest that these steps have to be strictly separated, both could be performed in parallel for example, however separating them and allowing for manual adjustments in between these stages may improve the end results. Finally, the model design stage would then conclude the modelling approach, with model transformation being the main consideration. The previously identified transformation rules for model generation along with the already identified modelling constructs and linkages would hereby provide the needed input for the generation of a draft graphical output. An iteration over various post-processing steps, particularly model refinement, would then finally yield the desired graphical model output.

3.3 Selection of available tools

As this thesis is supposed to provide a basis for future work in this domain, tools which can be used for either conceptual modelling or text mining in general, which are available on a cost-free basis are presented below. The presented list is not meant to be comprehensive, but rather informative and helpful if one wants to gain a basic overview of what tools are currently available for free and what functionalities these tools provide. Listing all tools would be not only excessive, but also outside of the scope of this thesis. Which is why instead, a handful of useful tools are presented, making it easier to pick the most suitable tools for one's research purpose. Furthermore, in the context of this thesis, the illustrative example in the next section is also built upon two of the below mentioned tools.

3.3.1 Tools for conceptual modelling

Considering the large academic and commercial interest in conceptual modelling and the breath of modelling languages and uses, there are numerous proprietary as well as open-source conceptual modelling tools available. Given that this thesis further focuses on models typically used in the business domain, the below tools have been chosen because of their relevance for business modelling and their wide range of applicability and functionality. An effort has been made to not only identify tools available at no cost, but tools for which the source code was readily accessible too. The main reason why mainly open-source tools have been chosen, as opposed to proprietary software, is that for further research, having access to the source code is undoubtedly beneficial. It should however be pointed out that there are numerous other tools which have been designed for different domains or specific purposes; which may be more useful depending on the given modelling objective.

Table 4 Tools for Conceptual Modelling

TOOLS FOR CONCEPTUAL MODELLING
BEE-UP
Developers: OMILab/University of Vienna, Patrik Burzynski and Dimitris Karagiannis **License**: Parts under ADOxx, Apache 2.0 and JDOM license **Website:** http://austria.omilab.org/psm/content/bee-up/info **Description:** BeeUp is a free modelling tool available from the Open Models Laboratory which is based on ADOxx and available for Windows. It is a standalone application that provides a hybrid modelling environment, which allows the user to create and evaluate models according to five common modelling languages: BPMN, EPC, ER, UML and Petri Nets. Further functionalities provided are process simulation (including path and capacity analysis), SQL code generation from ER models, model querying abilities and the import and export of models in various formats (such as RDF, ADL, XML or JPG). Thus it is a good choice for academic or business users looking to create and analyze detailed models in the five above mentioned languages. (Karagiannis, Burzynski, and Miron 2017)
DIA
Developers: Dia developers (originally developed by Alexander Larsson) License: GPL Implemented in: C **Website:** https://wiki.gnome.org/Apps/Dia **Description:** Dia is an open-source and free modelling or diagramming software available for Windows, Mac OSX, Linux and Unix. Taking its original inspiration from Microsoft Visio, the

developers have created a simple tool which allows the drawing of various diagrams. The user can choose shapes from various packages already available, which provide shapes supporting common conventions of ER or UML models for example, or users can create their own shapes in XML. The software itself is implemented as a drawing tool, not restricting users to use certain shapes together or certain connectors or symbols with certain shapes. This further emphasizes that this tool is meant for a more casual use. Thus if users are looking for diagrammatic freedom, this software allows users to express their modelling creativity readily and can be a great tool for simple modelling objectives, whereas if a user is looking for conformity with certain modelling languages and analytical abilities then this software may be lacking some functionalities. It does however support written Python extensions. (George 2002)
DRAW.IO
Developers: Gaudenz Alder and David Benson **License**: Apache 2.0 **Website:** https://www.draw.io/ **Description:** Draw.io is a free online diagramming tool, which allows users to quickly create a variety of diagrams and save them locally, on Google Drive, OneDrive, Dropbox or GitHub. It provides an extensive range of shapes, with support for common modelling languages such as BPM, UML, ER or network diagrams as well as common shapes and symbols used for graphical illustrations. Additionally, users can create new shapes by drawing or importing images of shapes into the application. Again, the functionality is mainly focused on creating or drawing diagrams. (Alder and Benson 2018)
MODELIO
Developers: Modeliosoft **License**: GPL, Apache 2.0 **Website:** https://www.modelio.org/ **Description:** Modelio is an open source tool, mainly designed for UML modelling and geared towards the business user. The tool provides support for BPMN and UML model generation. Community-built add-on modules provide further functionalities, such as integration of enterprise architecture, Java code generation from UML or reverse engineering. While the core packages are for free and open-source, Modeliosoft provide further packages for purchase, which are tailored for specific business needs with added functionalities. (Bridgwater 2011)

TERRAER
Developers: Richardo Terra and Henrique Rocha License: GPL Implemented in: Java **Website:** http://www.terraer.com.br/ **Description:** TerraER is a free and open-source Entity-Relationship Modeling Tool which has been developed as part of a study course as an aid for students to understand ER concepts better and design ER models based on the concepts learned at university. It is thus mainly geared towards academic use. Regarding availability, it is available for Windows, Linux, and Mac OS and in English as well as Portuguese. The generated models follow Chen's notation and are saved in XML format, allowing cross-plattform use. (Rocha and Terra 2013)
UMBRELLO
Developers: The Umbrello Team **License**: GPLv2+ Implemented in: C++ **Website:** https://umbrello.kde.org/ **Description:** Umbrello UML Modeller is, as the name suggests, a Unified Modelling Language diagramming tool, which mainly focuses on providing support for the analysis and model design stages. It is available for Linux (part of the KDE SDK module), Windows and Mac OS X and a free open-source project. The Umbrello UML Modeller supports various model types, including class diagrams, sequence diagrams, use case diagrams, activity diagrams, component diagrams and ER diagrams among others. (Hensgen 2013)

3.3.2 Tools for text analytics

As with tools for conceptual modelling, tool support for text analytics is vast as well. Kaur and Chopra (2016)'s "Comparison of text mining tools" and Ingersoll (2015)'s article on open-source tools for natural language processing provided the basis for the identification of suitable tools in this domain. The tools listed below were chosen based on their relevancy for conceptual modelling tasks, the availability of their source code and their scope of functionality provided. Again, the focus was laid upon the identification of open-source tools. While many other useful text mining tools exist, they may be of greater use in other domains or may provide solutions for highly specialized tasks.

Table 5 Tools for Text Analytics

TOOLS FOR TEXT ANALYTICS
AIKA
Developers: Aika Project **License**: Apache 2.0 Implemented in: Java **Website:** http://www.aika-software.org/ **Description:** Aika is a Java library which is mainly used for semantic text analysis. Its natural language processing algorithm can be used for syntax parsing, text categorization, named entity recognition, word sense disambiguation and semantic text enrichment, amongst others. Aika represents the textual information as an artificial neural network. Hereby the nodes represent linguistic concepts such as word meanings or categories, and the synaptic weights allow the definition of different functions within the network. (Molzberger 2017)
APACHE OPENNLP
Developers: Apache Software Foundation **License:** Apache 2.0 Implemented in: Java **Website**: https://opennlp.apache.org/ **Description:** Apache OpenNLP is a Java library which provides various functionalities for text processing, including preprocessing operations such as tokenization, part-of-speech tagging and chunking; as well as main text mining operations such as named entity recognition and coreference resolution. It is based on machine learning principles and thus provides functionalities for users to train as well as evaluate their constructed models. The main focus has hereby been laid upon providing a comprehensive toolkit for a variety of languages. (Community 2017)
GATE
Developers: GATE research team, Dept. Computer Science, University of Sheffield License: LGPL Implemented in: Java **Website**: https://gate.ac.uk/ **Description:** GATE stands for “general architecture for text engineering”. It is an open-source software which entails a development environment, web application, Java library, architecture and text mining process. As it has been heavily funded by private enterprises, many of its application cases focus on commercial interests, such as market analysis or text mining for recruitment. Detailed

documentation, process description and use cases are provided, enabling businesses and individuals to easily build their own applications. However originally being a University project, scientists and researchers have been using and further developing the software and its functionalities heavily as well. The main functionalities hereby revolve around web mining, information retrieval and information extraction, as well as semantic analysis. (Cunningham et al. 2013)

NLTK (NATURAL LANGUAGE PROCESSING TOOLKIT)

Developers: Team NLTK

License: Apache 2.0

Implemented in: Python

Website: http://www.nltk.org/

Description: The Natural Language Processing Toolkit is an open-source toolkit which consists of various program modules and libraries, providing statistical as well as symbolic natural language processing capabilities. Amongst others it entails the most common text preprocessing functionalities such as tokenization, stemming, POS tagging and parsing; as well as solutions for main text mining tasks such as categorization, classification or semantic text analysis. The NLTK was designed as part of a University course with its goal being the development of a comprehensive learning platform for computational linguistics. This is why it uses easy-to-understand language and is supplemented with numerous learning examples and extensive documentation, making it very suitable for beginners. (Bird, Klein, and Loper 2009)

STANFORD'S CORE NLP SUITE

Developers: The Stanford Natural Language Processing Group

License: GPL

Implemented in: Java

Website: https://stanfordnlp.github.io/CoreNLP/

Description: Stanford's Core NLP Suite provides a set of natural language processing tools which mainly focus on text understanding and text annotation. It incorporates a strong grammatical basis, supporting Arabic, Chinese, English, French, German and Spanish. It has been heavily used in academic research with its main functionalities including text preprocessing, part-of-speech tagging, named entity recognition, information extraction and sentiment analysis. Furthermore, while written in Java, Stanford's Core NLP Suite can be run from various other programming languages and is utilized in various other packages. (Manning et al. 2014)

4. Illustrative example: Utilization of text analytics for ER model generation

The objective of this chapter is to provide a proof of concept, for the previously introduced Generic text analysis approach for conceptual model generation. The proof of concept is presented in the form of the "ER Text Converter" tool, which has been implemented in Python and provides basic, text-analysis enabled Entity- Relationship (ER) model generation functionalities. In this chapter, first, an overview of the tool and its functionalities is provided. Then, the reader is guided through each of the three steps of the generic approach presented in section 3.2.3, namely the definition of environmental factors, the derivation of applicable rules and the implementation of the modelling approach. Finally, a brief result discussion is presented and areas for improvement are outlined.

4.1 Tool overview and functionalities

The "ER Text Converter" is a text converter tool, which enables semi-automated ER model generation from requirement specifications expressed in natural language. It has been designed to demonstrate how text mining techniques can be employed to assist the model design process. As a reminder, ER models were originally introduced by Peter Chen (1988) as "unified data model" and have been heavily used since as a basis for database design. When designing a database, the first step is a basic requirement analysis. In this step one should think about what kind of information the system should store and what categories this information falls into. This is usually provided in the form of a natural language description, which is however not practicable to be used as basis for the system implementation and is rather prone to lead to misunderstandings later down the line. Thus usually an abstraction is first sought out in the form of a conceptual model which is then used as a discussion vehicle before the derivation of a logical schemata and then finally, the implementation. ER models hereby provide an abstraction of the required information that is easy to understand for both business users and developers. The "ER Text Converter" tool has been developed to provide support for ER model generation and, in itself, is an example of how a (semi-) automated approach for this usually highly time-consuming task could be developed.

The tool was hereby developed by following the generic text analysis approach for conceptual model generation described in section 3.2.3 and its implementation is outlined in the following subchapters. The Graphical User Interface hereby takes the user through each stage of the model design process, while providing the following functionalities:

Text input: The user can enter simple text specifications and submit them for annotation.

Annotation: At this stage, the text is preprocessed and syntactic features are analyzed and tested against rules for identifying ER modelling constructs. As a result potentially important information for ER modelling is extracted and highlighted with different colors.

Mapping: Hereby, modelling elements are mapped based on a set of rules for mapping ER modelling constructs and an abstraction of the model is presented in a simple and editable format. The comparison to the annotated text and possibility to amend the suggested mapping is given.

Model design: In the final stage, standard notation and syntax of the ER modelling language are applied and an XML file is generated, which can be imported into the Bee-Up conceptual modelling tool[5], generating a visual ER model representation.

4.2 Definition of environmental factors

The first stage of the generic approach is the preparation stage, which deals with gaining a thorough understanding of environmental factors. This includes investigating and making a decision on the objective and application domain, the information extraction approach, the modelling language, as well as the implementation platform.

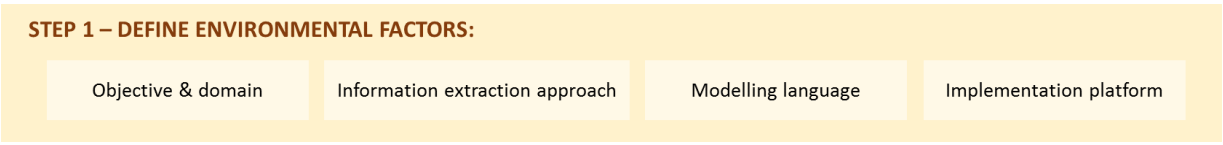

Figure 14 Step 1 - Define environmental factors

4.2.1 Objective & domain

The overall objective for this tool was to provide an example of how the generic text analysis approach for conceptual modelling can be used as the basis for a simple tool implementation, and by doing that, to illustrate the concept's usefulness. No specific application domain was envisioned, considering the rather generic objective and thus, it was determined that no special lexicon or knowledge base was needed for the implementation of the tool. Furthermore, the restriction to one modelling language and simple rule implementations would be sufficient to achieve this goal. On the basis of this understanding, a tool design that highlights each step of the process, potentially gives an interim result after each modelling step and visualizes the concept in detail was determined to be desirable.

[5] The Bee-Up conceptual modelling tool can be downloaded under the following link: http://austria.omilab.org/psm/content/bee-up/download?view=download (accessed on April 1st 2018)

4.2.2 Information extraction approach

In regard to the information extraction approach a pattern-based approach was chosen as not enough training and testing data was available. Furthermore, previous research suggested that pattern-based approaches yielded satisfactory results with simple data base problems, see Uduwela and Wijayarathna (2014) for examples. The rules or patterns for identifying and extracting the needed information are hereby mainly syntactic in nature and listed in subchapter 4.3.

4.2.3 Modelling language

The "ER Text Converter", as the name suggests, is a tool for generating data models in the ER modelling language. Thus, before constructing such a tool, one should study the ER modelling language to become familiar with the language's constructs and their purpose. In short, ER models use **entities** to describe categorical concepts, such as *"student*" or "*course*" and **relationships** between those entities to describe how they relate to one another, e.g. "*students takes courses*". Additionally, **cardinalities** can be used to specify how many of such relationships exist. The example before would be a *"many-to-many"* relationship as a *"student can take one or more courses"* and *"a course can be taken by one or more student"*. Furthermore, **attributes** can be used to describe relevant characteristics of the main concepts or of their relationships. To provide an example, for each "*student*" a university could choose to store information such as "*student ID", "student name"* or "*date of birth"*.

Thus, the core modeling constructs of the ER modelling language include entities, relations, attributes and cardinalities. However, before progressing to the derivation of heuristics to identify each type of construct, it is important to familiarize oneself thoroughly with each of the previously mentioned modelling constructs and the specific rules that apply to them on a syntactic, semantic and notational level. Figure 15 below shows the notation that is typical for these constructs.

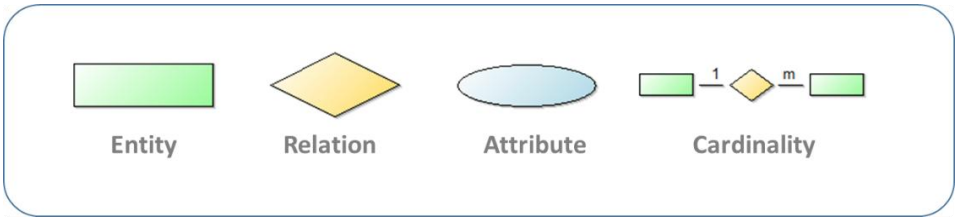

Figure 15 ER notation

Let us have a closer look at each of the mentioned constructs. According to Barker (1990), an **entity** *"is a thing or object of significance, whether real or imagined, about which information needs to be known or held.*" As can be seen from Figure 15, entities are typically displayed in a rectangle. This is in line with the Chen notation of ER modelling. The name of the entity type or class is written in singular form within the rectangle. Hereby it is important to point out that the name one sees when looking at an ER

diagram is that of the entire entity class, not of an instance. This is because the ER diagram is a model of the database structure, not of the data stored within it. To provide an example, an entity class which would be displayed in the model could be "*course*", whereas "*IT introduction*", "*Business fundamentals*" or "*Psychology 101*" would all be instances that belong to the entity class "*course*". Furthermore, entities are mutually exclusive. Thus, each thing or object can only be captured under one entity type. Additionally, each entity has to be uniquely identifiable, in the sense that each instance of an entity type has to be distinctly different from all others.

Relationships specify how entities relate to one another. As Barker (1990) mentions, in the context of database schemata, relationships are binary in nature, either relating one entity to another or to itself. While relationships can exist between more than two entities, e.g. "*a student takes a course during a semester",* if an ER model is to be used for database design, relationships need to be reduced to refer to either one or two entities. Notation-wise, typically a diamond shape is used to depict relationships with lines connecting the relation to its respective entities. An example of a relation would be the "*take*" in "*students – take – courses*". Relations hereby have a name and a cardinality. The **cardinality** of a relationship specifies how many of such relations can exist. According to the Chen Notation, there are three basic types of cardinalities: "*1:1", "1:n"* or *"m:n"*. The previously mentioned example would be a "*m:n"* cardinality. This is because each student can take multiple courses and each course can be taken by multiple students. Additionally, relationships can vary in terms of optionality, thus being either optional or mandatory in nature. Again, as with the other ER modelling constructs, the relation hereby represents a type of relation not any particular instance of the relation.

Information or any important descriptive detail about an entity or a relation is captured by specifying **attributes.** According to Barker (1990), an attribute is "*any detail that serves to qualify, identify, classify, quantify or express the state of an entity".* Attributes are depicted in an oval shape with the name of the attribute written within it in singular form. They are attached to either the entity or the relation that they describe by a simple connecting link. Examples would be the attributes "*StudentID", "name" and "date of birth"* for the entity "*student"* or the attributes "*semester", "attempt"* and *"grade"* for the relation "*take".* Hereby it is important to note that there can only be one attribute value for an instance of an entity or relation. Should there be a need for multiple attribute values for one entity instance, then it is usually advisable to create a new entity out of this attribute and to link it to the original entity via a relation. Furthermore, it is important to point out that attributes are used to uniquely identify an entity from the other instances. This is done through either one or a combination of multiple attributes. These attributes are called "unique identifiers" or "primary keys". An example of a primary key which helps identify each student is the "*StudentID*".

4.2.4 Implementation platform

Python

For the implementation of the example, the object-oriented programming language Python (version 2.7[6]) was chosen based on its simple syntax, scalability, cross-domain functionality and comprehensiveness. As described by Hofmann and Chisholm (2016), Python is a simple, open-source, general-purpose programming language that emphasizes readability and enjoys a wide-spread usage in academia as well as the industry. Numerous packages, libraries and tools are being released and updated on an ongoing basis, adding to Python's functionalities, some of them being part of the standard package and some of them having to be downloaded and installed separately.

To provide an example, the NLTK (Natural Language Processing toolkit), as described by Bird, Klein, and Loper (2009), is a python package that includes various libraries and modules that provide the user with a series of natural language processing functionalities such as tokenization, tagging, parsing and semantic analysis, which normally require the combination of several independent tools. Its modules can be chained together according to one's individual needs. In regard to this illustrative example, the NLTK for Python has been chosen to provide the text analysis functionalities needed. As it is not part of the standard package it has to be downloaded separately[7].

In addition to the NLTK module, the following packages were used: "Tkinter" for GUI functionalities, "re" to be able to use regular expressions, "sys" & "os" which provide functionalities for file naming, path addresses and directories. These packages are however part of the standard Python library, thus do not need to be downloaded separately. [8]

Bee-Up Modelling Toolkit

In respect of the modelling capabilities the Bee-Up conceptual modelling toolkit[9] which was built on ADOxx and is described in Karagiannis, Burzynski, and Miron (2017), has been chosen mainly due to its vast repertoire of model creation and utilization capabilities, its cost-free availability and support for multiple conceptual modelling languages. The toolkit supports ER, EPC, BPMN and UML models, as

---

[6] Python 2.7 can be downloaded under the following link: https://www.python.org/downloads/release/python-2714/ (accessed on 2 April 2018)

[7] NLTK can be downloaded under the following link: https://www.nltk.org/install.html (accessed on 2 April 2018). Once downloaded run the following command to download all libraries:

```
import nltk
nltk.download('all')
```

[8] The source code for the "ER Text Converter" tool is submitted on the enclosed CD, and requires the installation of Python 2.7 and the NLTK to be run.

[9] Bee-Up can be downloaded under the following link: http://austria.omilab.org/psm/content/bee-up/download?view=download (accessed on 4 April 2018)

well as Petri Nets. Not only is this tool thus suitable for ER models, but it opens up the possibility of extending the "ER Text Converter" to be able to handle various conceptual modelling languages in the future. Furthermore, in addition to providing a platform to create visual models, it comes with analytical functionalities and added features, such as SQL code generation for ER models. Finally, the Bee-Up tool includes XML import and export functionalities, which due to XML's wide- spread use as data exchange format and rather simple structure is easy to integrate into the "ER Text Converter" and gives flexibility to use and process the models with other applications as well.

4.3 Derivation of respective rule sets

The second stage of the generic approach is the derivation of identification and transformation rules. Hereby previous literature on automated and heuristic-based ER model generation was studied and commonly used rules for the identification of useful information and modelling linkages were identified. Additionally, rules were also formed based on the examination of modelling construct definitions and their linkages to linguistic elements, as well as textbook examples and lecture notes that describe how ER models are created.

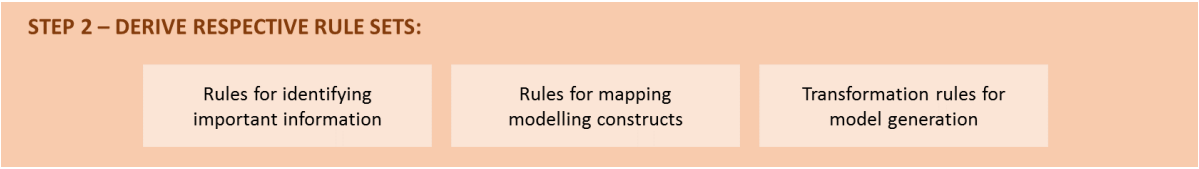

Figure 16 Step 2 - Derive respective rule sets

4.3.1 Rules for identifying potential ER modelling elements

The following rules for identifying and classifying important information have been implemented:

RULE 1: **Nouns** may indicate **entities or attributes**.

RULE 2: If a **noun** is mentioned more than once it may indicate an **entity**.

RULE 3: **Nouns** succeeding certain **verbs** ('have', 'include', 'involve', 'incorporate', 'contain', 'comprise') may indicate the presence of **attributes**.

RULE 4: If a **noun phrase** ends with (**'number', 'no', 'code', 'date', 'type', 'volume', 'birth', 'id', 'identifier', 'name'**), this may indicate that it is an **attribute**. (Omar, Hanna, and McKevitt 2004)

RULE 5: If a **noun** is marked as potential **attribute** or potential **entity** it is most likely an **entity**. (Al-Safadi 2009)

RULE 6: **Verbs** may indicate **relationships**.

RULE 7: **Determiners** and **cardinals** may indicate **cardinalities**.

RULE 8: Certain words ('many', 'multiple', 'many', 'any', 'various', 'more') may indicate cardinalities.

RULE 9: List items may indicate attributes.

4.3.2 Rules for mapping ER modelling constructs

The following rules for identifying and relating modelling constructs have been implemented:

RULE 10: Attributes are likely to be mentioned in the same sentence as their entity. (Al-Safadi 2009)

RULE 11: If a sentence contains two entities, then the verb between the entities may indicate their relationship.

RULE 12: The determiner or cardinal before an entity in a relationship-identifying sentence may indicate the cardinality of the relationship. (Al-Safadi 2009)

RULE 13: The plural noun form may indicate a 'multiple' cardinality. (Tjoa and Berger 1993)

RULE 14: If no entity is mentioned but attributes are present then they likely belong to the entity mentioned in the previous sentence.

4.3.3 Transformation rules for model generation

In regard to the transformation, the ER implementation in the chosen modelling tool (Bee-Up) had to be considered. As the tool provides support for XML imports, the format and syntax constraints of the needed input file where considered as transformation rules.

As described by Bray et al. (1997), XML or "Extensible Markup Language", developed by the World Wide Web Consortium (W3C), is a general purpose markup language. Its use within the Bee-Up tool revolves around data communication between the tool itself and other applications. Overall, markup languages are used to annotate documents, either to simply structure a given document, to pass on instructions to applications on how certain text passages should be treated or to semantically enrich a given document through the use of labels. A widely used markup language is HTML for example, which comes with a given set of predefined semantics that define the appearance of a document or web page. XML on the other hand is a general purpose markup language, meaning that it does not come with predefined semantics, but instead allows its users to create their own tags and define the use of these tags themselves. Because of its interoperability and simple syntax, XML has become the standard for a variety of applications and is now frequently used for data communication between different systems.

The tags which can be used with the Bee-Up tool, thus the format and syntax constrains, are described in the Document Type Description (DTD) file for XML import/exports[10], developed by the Business Objectives Consulting (BOC) Group. As such, this specification file also has to be passed on along with the XML file in order for the import to work. The elements relevant for the ER import are hereby:

- General tags used for describing the system and model itself, namely <ADOXML>,<MODELS>, <MODEL>, <MODELATTRIBUTES>.
- The element used for declaring the modelling constructs such as entities, attributes and relations, namely <INSTANCE>.
- The elements used for creating the connecting lines between ER constructs, namely <CONNECTOR>, <FROM> and <TO>.
- The element which allows the specification of details about the above mentioned constructs such as position, denomination or key attribute, namely <ATTRIBUTE>.

The above mentioned DTD file describes the elements which can be used for tools based on ADOxx in general, thus is not only applicable to ER models. To specify which modelling language is to be used, amongst others, the model type needs to be declared in the <MODEL> element, e.g. <MODEL modeltype="ER model">. Furthermore specific classes should be used when defining an element. Examples of entity, attribute and relationship declarations and their respective class descriptions are given below.

Table 6 Examples of ER modelling construct declarations in XML

Entity declaration	<INSTANCE class="**Entity (ER)**" name="Supplier"> <ATTRIBUTE name="Position" type="STRING">NODE x:3.00cm y:5.00cm w:3.6cm h:1.2cm index:1</ATTRIBUTE> </INSTANCE>
Attribute declaration	<INSTANCE class="**Attribute (ER)**" name="Supplier ID"> <ATTRIBUTE name="Position" type="STRING">NODE x:3.00cm y:7.00cm w:3.6cm h:1.2cm index:5</ATTRIBUTE> <ATTRIBUTE name="Key attribute" type="ENUMERATION">no</ATTRIBUTE> <ATTRIBUTE name="Denomination" type="STRING">Supplier ID</ATTRIBUTE> </INSTANCE>

[10] The DTD file can be downloaded from https://github.com/LearnPAd/learnpad/blob/master/lp-ontology-recommender/src/test/resources/testdata/TitoloUnicoV4/adoxml31.dtd, accessed on 29 April 2018

Connector between the Entity and its Attribute	<CONNECTOR class="**has Attribute (ER**)"> <FROM instance="Supplier" class="Entity (ER)"></FROM> <TO instance="Supplier ID" class="Attribute (ER)"></TO> </CONNECTOR>
Relationship declaration	<INSTANCE class="**Relation (ER)**" name="supplies"> <ATTRIBUTE name="Position" type="STRING">NODE x:6.00cm y:2.00cm w:3.2cm h:1.8cm index:2</ATTRIBUTE> </INSTANCE>
Connector between the Relationship and the Entity	<CONNECTOR class="**Links (ER)**"> <FROM instance="supplies" class="Relation (ER)"></FROM> <TO instance="Supplier" class="Entity (ER)"></TO> <ATTRIBUTE name="Chen-Notation" type="ENUMERATION">1</ATTRIBUTE> </CONNECTOR>

4.4 Implementation of modelling approach

The third and final stage of the generic approach entails the implementation of the text mining – enabled modelling approach. Hereby a solution for each part of the process has to be developed. The snippet below, highlights the focus areas at each stage.

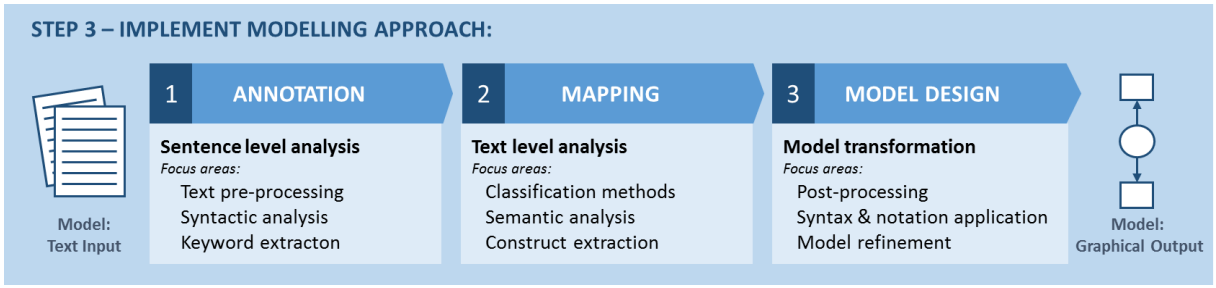

Figure 17 Step 3 - Implement modelling approach

The GUI (Graphical User Interface) has hereby been kept very simple. It has been built using the Tkinter module for Python and follows the model design process very closely while providing intermediate results between each step of the process. For simple implementations and while natural language processing capabilities are still limited, the generic concept recommends the separation of these steps, with the possibility of adjustments and manual intervention between each step as this has shown to improve the output at the end of the process. However, for tool implementations that strive towards complete automation, the steps could be combined or simply run in the background with the model output being immediately generated following the input of the textual specifications.

4.4.1 Text input

As a first step, the user is asked to enter text specifications. Because this is just an illustrative example, the tool works best when the text specifications follow a rather simple syntax, and some input constraints should be considered, e.g. the avoidance of quotation marks, apostrophes and other signs as the tool is not able to deal with those. Precision and recall of results are thus increased if each sentence in the text specifications is written in a declarative manner, if each entity or attribute is named explicitly and if sentences are written in a simple form, i.e. containing only one clause. A list of examples which give an indicator of the required simplicity of the text specifications can be found in Appendix A, with a result discussion of those examples being included in subchapter 4.5.

For the purpose of demonstrating the tool's functionalities in the following screenshots, the below example from Alechina (2008)'s lecture slides on ER modelling was taken:

"Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode. Each product has a single supplier. There is nothing to stop a supplier supplying many products."

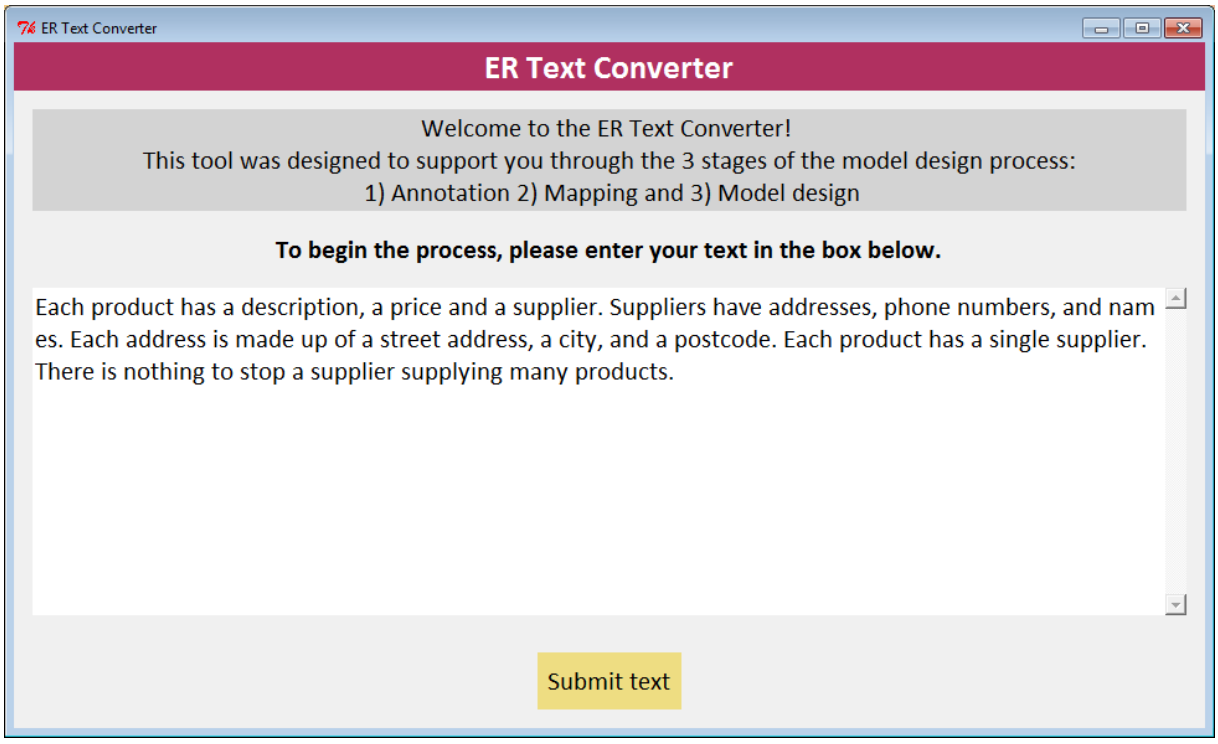

Figure 18 ER Text Converter - Text Input

4.4.2 Annotation

The annotation stage of the model design process deals with text comprehension and the identification of potentially useful information in respect of the modelling objective. This can be done in various forms. The ER Text Converter tool hereby does a visual annotation by highlighting potentially relevant text. Four colors are hereby used to identify potential entities (green), attributes (blue), relations (yellow) and cardinalities (grey). Relations hereby stand for relations between entities as well as attribute to entity links.

As outlined in the generic concept, this stage focuses on sentence level analysis, with the main part of the effort lying on text preprocessing and the testing of syntactic rules for each sentence. The preprocessing activities, hereby include sentence and word tokenization, part-of-speech (POS) tagging and lemmatization. For POS tagging the English language NLTK word tagger was used. The abbreviations are hereby based on the Penn Treebank tag set as described by Santorini (1990). Some of the used tags can be seen in the table below. [11]

Table 7 POS tags in the Stanford parser

POS	Word level
Adjective	JJ (adjective), JJR (comparative adj.), JJS (superlative adj.)
Adverb	RB (adverb), RBR (comparative adv.), RBS (superlative adv.)
Conjunction	CC (coordinating conj.), IN (preposition or subordinating conj.)
Interjection	UH (interjection)
Noun	NN (singular), NNS (plural), NNP (sing. proper), NNPS (pl. proper)
Preposition	IN (preposition or subordinating conj.)
Pronoun	PRP (personal pronoun), PRP$ (possessive pronoun)
Verb	VB (base form), VBD (past tense), VBG (gerund or pres. participle), VBN VBP, VBZ
Others	CD (cardinal number), DT (determiner), FW (foreign word), TO (to), EX (existential there), etc.

Once the input text is preprocessed, information extraction is performed based on the generic rules for identifying potential ER modelling elements described in section 4.3.1. Sentences are hereby viewed independently, as the main task at this stage is to highlight important information, not necessarily to relate it to one another. As a next step, lists are created for each of the four main

[11] For examples of words that fall under each tag see http://erwinkomen.ruhosting.nl/eng/2014_Longdale-Labels.htm, accessed on March, 4th 2018

categories with elements (words or word combinations) being appended to the list if they satisfy at least one rule, while not contradicting any of the other rules. These lists are then passed on to the highlighter, which highlights them in the next screen. A sample output of this process is presented in Figure 19 below. At this stage the user is only given the possibility to view the annotated text. Amendments to the annotation are not possible.

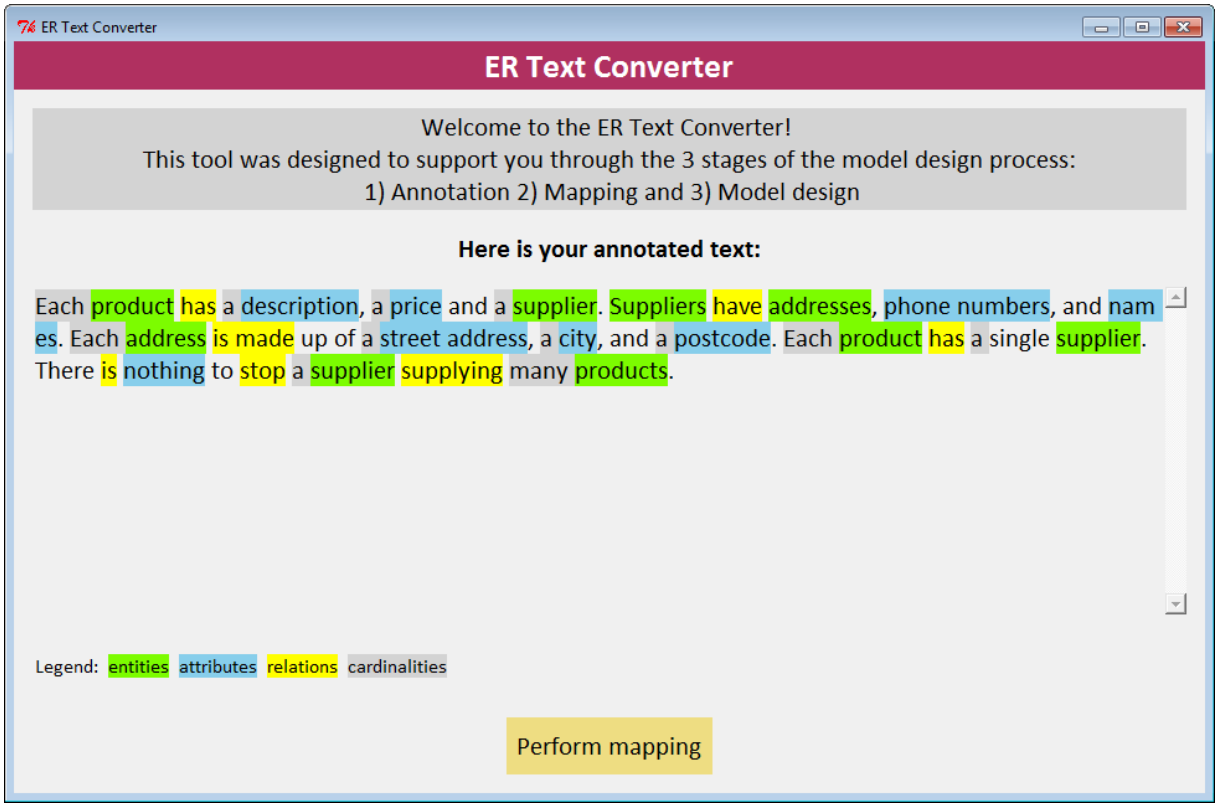

Figure 19 ER Text Converter - Annotation

4.4.3 Mapping

During the mapping stage, text level analysis is performed, as the text now has to be viewed in its entirety. While some semantic analysis could already be used in the annotation stage, it is a focus area at this stage given that the main tasks hereby lies in identifying all the entities, their corresponding attributes and putting the relations together with the correct cardinality indication. Thus, while in the first stage potential ER elements were identified, the mapping stage focuses on determining how they relate to one another. For this purpose, the specific rules for identifying ER modelling constructs defined in section 4.3.2 were utilized. The information extraction principle was similar to the one used in the annotation stage. Furthermore, the lemmatized version of entity and attribute names was chosen, as they should normally be displayed in singular form. If an entity, attribute or relationship name occurred twice then the next occurrence was renamed by appending a number to the end of the name, to ensure naming uniqueness. At this stage, a decision was made to allow the user to amend

the proposed mapping, thus to change the names of the elements, add missing elements or delete incorrectly identified ones. Hereby the following standard notation is required to enable further processing: *"ENTITY -> ATTRIBUTE, ATTRIBUTE, ..." and "RELATION: ENTITY - CARDINALITY, ENTITY – CARDINALITY".* Figure 20 below shows the sample output of this stage.

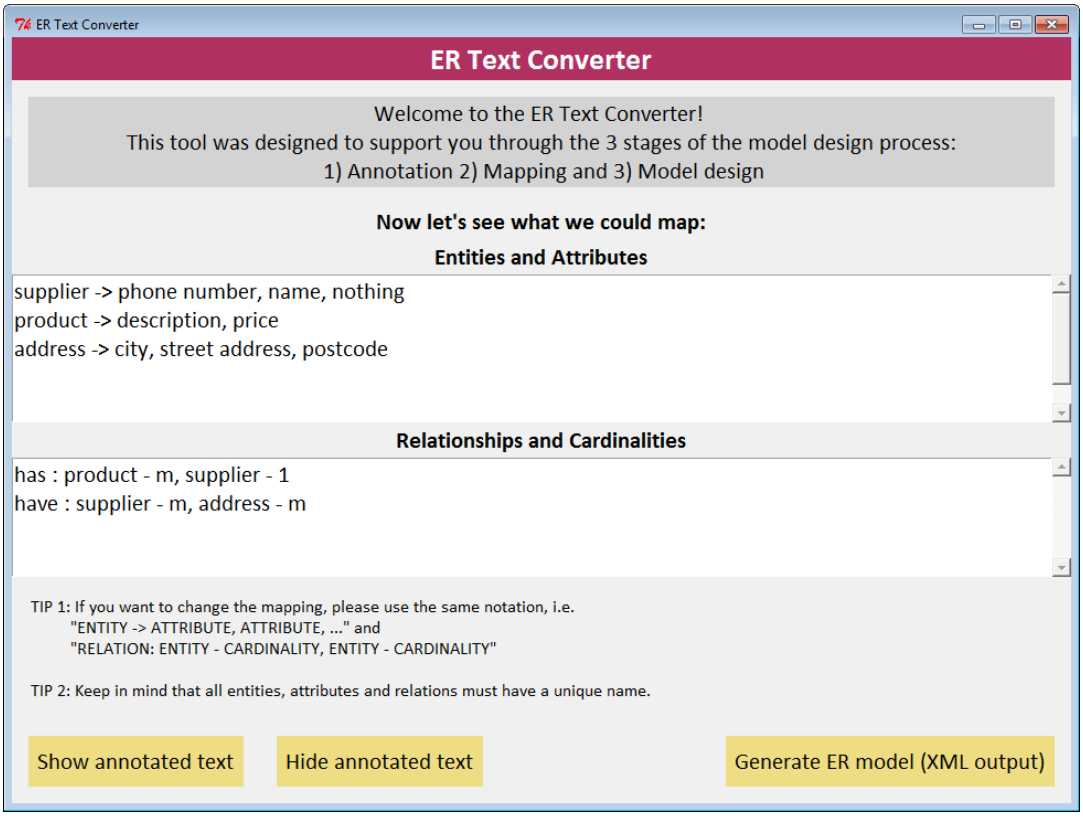

Figure 20 ER Text Converter – Mapping

Because at this stage the user has the possibility to adapt the proposed mapping, the previously annotated text can be called up and displayed to the right of the entry screen. This should make it easier to check and amend the mapping if needed. Figure 21 below depicts how this looks like.

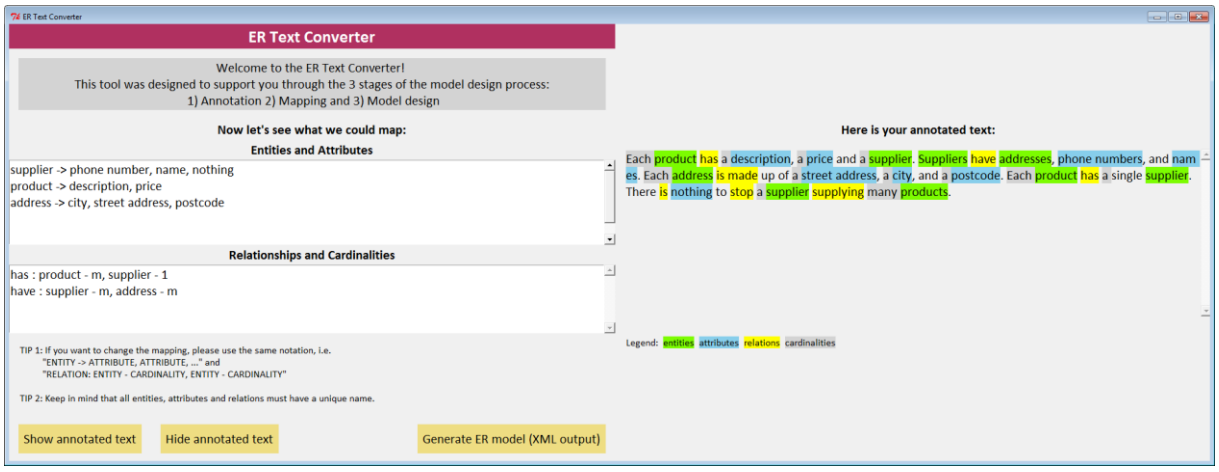

Figure 21 ER Text Converter - Mapping with annotated text

4.4.4 Model design

Finally, the model design stage transforms the mapped text into a format that can be imported into the modelling tool. Given the chosen environment of the ER Text Converter and the transformation rules applicable, this is an XML format. The notational rules for each construct are applied and the model is initially laid out in a way that relations are presented in the first row, entities in the second and all attributes belonging to an entities are presented below the entity. Entities, attributes and relations are hereby all specified using the <INSTANCE> tag and the connecting lines using the <CONNECTOR> tag. The initial positioning and size of an element is specified by using the <ATTRIBUTE name="Position"> tag that can be included with each <INSTANCE> declaration. Hereby the positioning is given as simple x and y coordinates, whereas the size of the element is given by specifying the width and length in the same tag. The size for each element has been kept identical for each construct type. To achieve a positioning that is not overlapping and allows for easy adjustments, the first row, which was specified by a fixed y value of 2 cm, was used for displaying the relations. The second row, fixed at a y value of 5 cm, was used for displaying the entities and any further rows, starting at a y value of 7 cm were used for displaying attributes. Attributes were hereby displayed below the corresponding entity, thus inheriting their entity's x value. Furthermore the canvas width and height was minimized. Hereby the width was minimized based on the total number of entities and the height was minimized based on the maximum number of attributes belonging to one entity respectively. At this stage, the user could still make changes to the XML file before saving it.

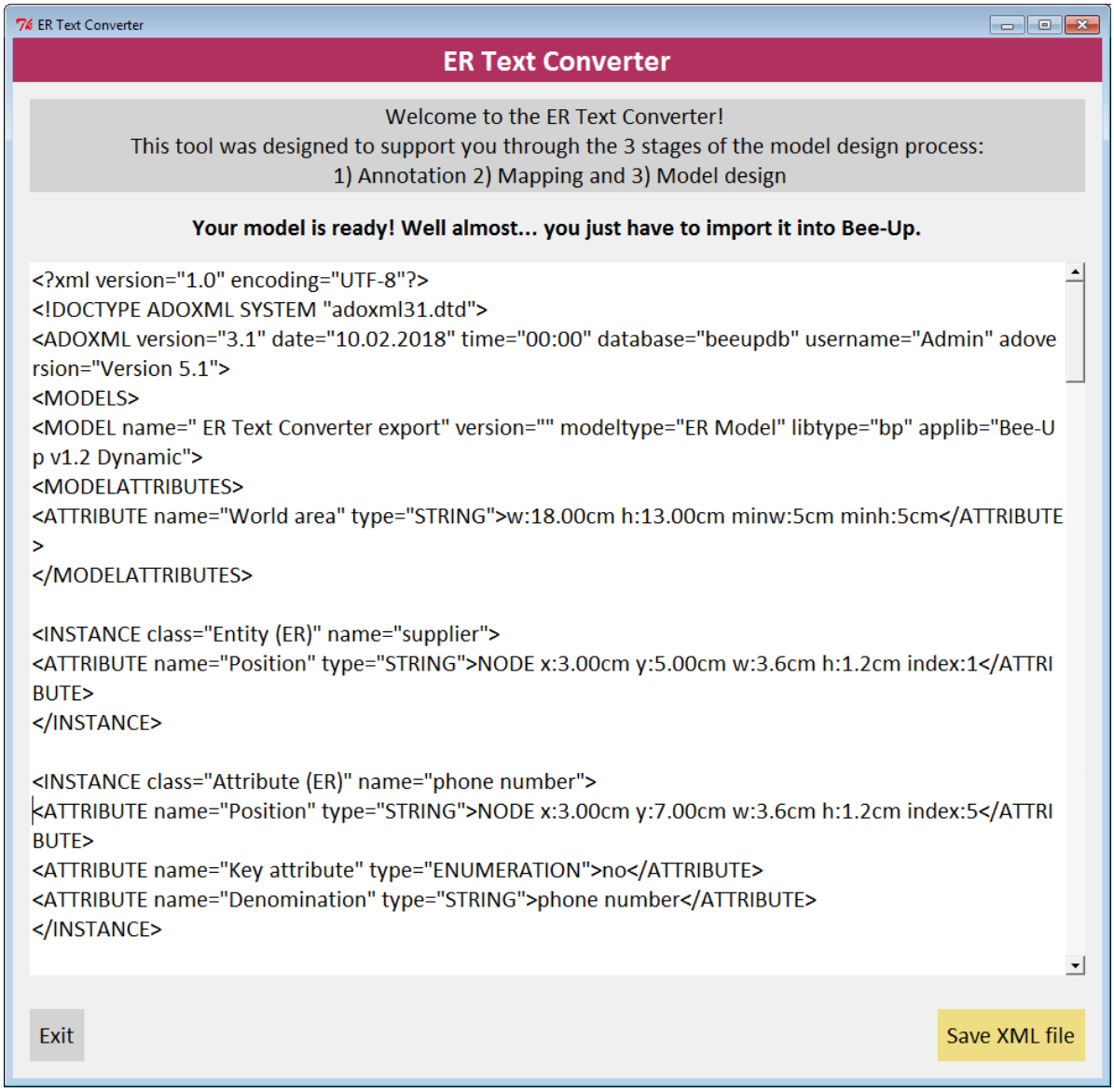

Figure 22 ER Text Converter - Model design (XML export)

4.4.5 Model output

Finally, the last stage is the import into the Bee-Up modelling toolkit. The output of the presented example is shown in Figure 23 below. Once the model is available in Bee-Up, the user can utilize all available functionalities to refine the model, add descriptions, rearrange the elements, generate images or SQL code and export the model in various formats.

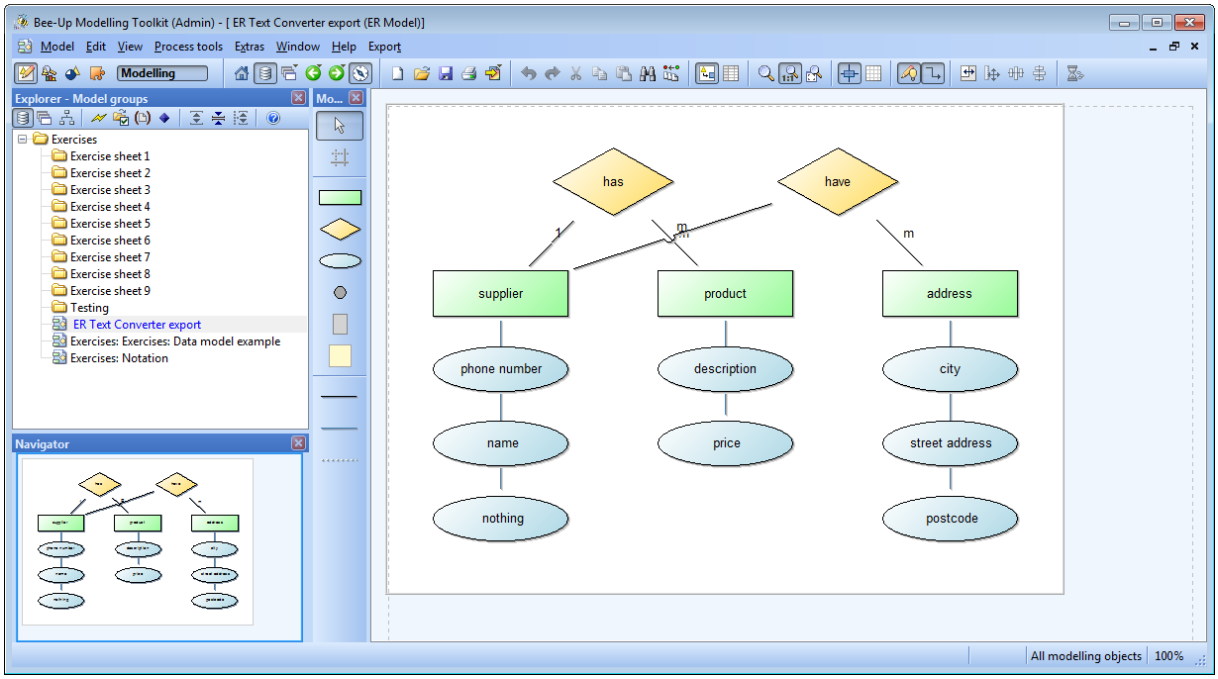

Figure 23 Sample model output after XML model import into Bee-Up modelling tool

4.5 Result evaluation

As Uduwela and Wijayarathna (2014)'s "*Survey On Tools and Systems to Generate ER Diagram from Requirement Specifications*" has pointed out evaluating and comparing the results of ER generating tools and system is rather difficult. A very common way to measure the quality of such tools is by looking at completeness measures, which are usually quoted in each study. Completeness measures hereby include precision and recall, which are calculated as follows:

$$\textbf{Precision} = \frac{\text{Number of correct items identified}}{\text{Total number of correct and incorrect items identified}}$$

$$\textbf{Recall} = \frac{\text{Number of correct items identified}}{\text{Total number of correct items}}$$

For both measures an ideal percentage of 100% is desired, whereas the principle applies "*the higher the better*". Precision indicates how many of the identified items are correct and thus represents a measure for the accuracy of the system. Thus, the higher the precision, the less there is that needs to be removed. Recall, on the other hand, shows how many items were correctly identified in relation to the items that should have been identified, thus giving a measure of completeness. Thus, the higher the recall, the less there is that needs to be added manually afterwards. While this is not always the case, frequently, improving one of those measures leads to worsening the other, thus one should think about which measure is of higher importance in respect to the application that one is trying to design.

To provide an example, if the correct entities for a model are "*class, teacher and student*", but a tool has identified "*class, teacher, desktop and database*" as entities, then the number of correct items identified is two, namely "*class and teacher*". The total number of correct and incorrect items identified is four, as it identified two correct and two incorrect entities. This makes the precision measure 50% (=2/4). If it had only identified "*class and teacher*" as entities, then even though it would not have identified all entities, it would have a precision of 100% for the entities it did identify. Regarding the recall measure for our original example, the number of correct items identified remains the same, namely two. The total number of correct items would in this case be three, as the ideal reference model has three entities. This makes the recall measure 66% (=2/3), meaning that 66% of all entities were identified by the tool.

The "ER Text Converter" tool has been tested against the eight database problems included in Annex A. Below is a listing of the completeness measures for those eight example descriptions in respect to entity, attribute and relationship identification. Success measures were hereby defined as "correctly identified" for entities, "correctly identified and attached to the correct entity" for attributes and "correctly identified as relation that exists between those entities, while ignoring the naming".

Table 8 Completeness results for sample texts in Annex A

Example	Entities		Attributes		Relationships	
	Precision	Recall	Precision	Recall	Precision	Recall
EX 1	100%	100%	88%	100%	100%	100%
EX 2	100%	100%	67%	86%	100%	100%
EX 3	60%	100%	63%	59%	50%	100%
EX 4	75%	75%	55%	46%	50%	33%
EX 5	100%	100%	n/a	n/a	100%	71%
EX 6	100%	100%	46%	67%	100%	75%
EX 7	38%	75%	7%	8%	20%	25%
EX 8	71%	77%	42%	53%	63%	42%

One has to keep in mind that this tool was developed in order to demonstrate how the generic concept could be put to practice, and while its precision and recall measures decrease with added specification complexity, one can say that the objective was achieved. As there is no or very limited indication on the nature of specifications used by other researchers to calculate the above mentioned measures, and most tools presented in other research papers are not publicly available, there is no benefit in comparing the results to other tools as they cannot be tested against the same dataset. The above measures have nevertheless been presented in order to show how such tools are usually evaluated and to provide an opportunity to discuss the tool's performance and identify areas for improvement.

Given that this illustrative example only included a very limited set of heuristics, the results seem to be very promising. Overall, the figures show that from the three categories, the tool's rules on entity recognition seem to be the strongest. The setback in entity precision that can be observed in example 7, is mainly due to words describing the system environment, such as “database”, “information” or “data”, being picked up erroneously. The inclusion of a rule that excludes such words from entity recognition has shown to be beneficial, as Omar, Hanna, and McKevitt (2004) or Btoush and Hammad (2015) have already demonstrated. However, the entity identification rules seem to be performing well for simpler examples. However, when considering attributes, the results show that particularly the rules for attributes recognition and mapping need to be refined as a lot of noise is still being picked up erroneously. This can be observed in specification 2 for instance, where the examples which are mentioned in brackets are inaccurately identified as attributes or in specification 7 where common abbreviations such as "e.g." or "etc." are marked as attributes. To avoid this, a stop word list should be introduced to remove such common words from the identified concepts. Moreover, the tool is not yet able to handle the assignment of attributes to relationships, which explains the low attribute precision and recall measures in specification 3. This is something that should undoubtedly be addressed in a more sophisticated version of the “ER Text Converter”. In regard to relationships, the tool mainly still seems to struggle with implied relationships. Finally, the results show that the tool performs better on examples with a simple sentence structure and where relationships are stated explicitly, such as examples 1, 2 or 5. Thus the consideration of rewriting more complex specifications to a simpler text form could improve results substantially. An example is an additional clause that has been included within one of the sentences in example 6, namely “*Each pizza is made of various ingredients (one ingredient has a unique name)*”. The clause has hereby been included in brackets rather than stated explicitly. While this may be common in natural langue, particularly in cases where one wants to convey something of lesser importance, the writing style could easily be improved and the complexity decreased by making the clause a standalone sentence. This would lead to an immediate improvement of the completeness measures and thus the tool’s performance as well.

While not suitable for highly complex database design problems, given that the tool’s functionalities are limited to indicative examples and simple rule implementations, the tool could be used for generating models from simple requirement specifications. This could be used in two scenarios: either, users could be asked to describe their model using simple language, or the examples fed into the model would have to be of low complexity themselves. In the first scenario, while asking the user to create simple text specifications or requiring them to provide the specifications in a specified format is indeed shifting some of the selection and pre-processing tasks back to the user, it could still save the user considerable time and effort, particularly when it comes to model design, as the creation of the

elements, the attachment of the attributes, the linkages and relations between the entities and the overall model alignment would be done by the tool. This would allow less tech-savvy users to create models as well, given that providing simple text specifications is still far easier then creating an entire model from scratch. On the other hand, this tool could be suitable for database problems, which themselves possess a low complexity to begin with. To provide an example, the tool could be used for database problems as typically used in introductory university courses on database design, given that examples used in such courses are typically written in a simple and straight-forward manner. Undoubtedly, however, there are still many areas for improvement and more complex application areas would require an expansion of its rule set and functionalities. Should this tool be used as a basis for a more sophisticated version, then first of all the expansion of its rule set needs to be addressed. This includes more and refined rules for the already identified concepts, such as entities, attributes, relationships, links and cardinalities; but also rules for some ER concepts which were omitted so far, such as primary keys or different types of relations. Considering the results presented in Table 8, particularly the rules for identifying attributes could be revisited or refined. Additionally, while constraining the allowable syntax or sentence form of specifications is a method that improves results substantially, it should be used with care. Rewriting the specifications to a simpler form undoubtedly improves the precision and recall measures of the tool, however, ideally, the ability to deal with more complex sentences and ultimately unconstrained natural language input should be pursued. One could also consider incorporating such a step into the tool itself. This could be done by designing a function which first rewrites complicated texts into simpler ones, and then upon the confirmation of the user that this text and the information contained within it, is still semantically valid, moves on to annotate the simplified text. Particular attention should, in that regard, also be given to the inclusion of semantic rules. See Omar, Hanna, and Mc Kevitt (2006) for suggestions. Furthermore the use of lexicons or knowledge bases should be considered, as Uduwela and Wijayarathna (2014) point out this usually leads to an improvement of completeness measures. While there are many areas that could be improved further, e.g. allowing the manual amendment of the text annotation, incorporating text-input constraints that check whether the user input is in the correct format or even designing it as an incorporated feature in already existing modelling tools, this tool provides a good starting point for future efforts in that area. While it did not provide a complete solution to automated ER model generation, this chapter did provide an illustrative implementation of the concept, thus ultimately demonstrating its validity and usefulness. Considering that the tool's main objective was to demonstrate the usability of the generic concept, one could argue that this objective was achieved successfully.

5. Conclusion and future directions

The proposed generic text analysis approach for conceptual model generation provides a structured approach to utilizing text analytics for conceptual modelling regardless of domain, information extraction approach, modelling language or implementation specifics. These factors are rather seen as input variables which need to be defined at the beginning of the process and can be amended or extended as desired at a later stage, with clear indicators of how this influences the final implementation. While this work builds upon the work of previous researchers who have either applied text analytics to specific modelling languages, see Yue, Briand, and Labiche (2010), Friedrich, Mendling, and Puhlmann (2011) or Sintoris and Vergidis (2017), or who have presented very text mining centric approaches, such as Btoush and Hammad (2015), Sagar and Abirami (2014) and Montes et al. (2008), this approach provides a different perspective which may be more useful in domains where modelling-centric, multi-functional or scalable aspects are desired. The level of abstraction, cross- platform and cross- domain applicability, as well as its "model design process" viewpoint is thus what differentiates this work from others. Furthermore, while limited in its scope the example "ER Text Converter" tool implementation discussed in chapter 4, demonstrates how the theoretical concept can be put to practice.

Looking forward, this thesis proposes to view the generic concept as a potential starting point for the generation of specialized applications or the extension of already available modelling tools. While the illustrative example discussed in chapter 4 only focused on one modelling language and simple text specifications, the proposed concept provides a method how to easily extend these efforts to include different languages and more complex descriptions. The example was further presented as a separate application, however given the broad availability of various specialized as well as hybrid conceptual modelling tools, one could consider extending the functionality of such tools in the future by allowing natural language input. This concept could be used as a basis for building such functionalities. Additionally, the application of text analytics in different areas other than the model generation process itself should be explored further. Finished models still incorporate varying amounts of textual information which could also potentially be analyzed and processed with the help of various text analytic techniques. Ultimately, this thesis aims to serve as advocate for the utilization of text analytics in the conceptual modelling domain.

6. Appendices

6.1 Appendix A: List of ER Modelling examples used for evaluation

EX1 (Alechina 2008): "Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode. Each product has a single supplier. There is nothing to stop a supplier supplying many products."

EX2 (provided by University of Vienna): "Each aircraft has an International Registration Number, name, date of last maintenance, date of commissioning, belongs to a specific type and is assigned to a hangar. Each type of aircraft has a specific model name, passenger capacity and weight. A hangar is identified by a number and the capacity of the aircraft. Airlines hold certain shares (in %) of the aircraft. An airline is defined by a two-digit code (for example OS for Austrian Airlines, LH for Lufthansa, ...). Furthermore, the name, company headquarters and telephone number are recorded."

EX3 (Tahaghoghi and Williams 2006): "The university offers one or more programs. A program is made up of one or more courses. A student must enroll in a program. A student takes the courses that are part of her program. A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced. A course has a name, a course identifier, a credit point value, and the year it commenced. Students have one or more given names, a surname, a student identifier, a date of birth, and the year they first enrolled. We can treat all given names as a single object. When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade and a mark are recorded. Each course in a program is sequenced into a year and a semester."

EX4 (Tahaghoghi and Williams 2006): "The airline has one or more airplanes. An airplane has a model number, a unique registration number, and the capacity to take one or more passengers. An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time. Each flight is carried out by a single airplane. A passenger has given names, a surname, and a unique email address. A passenger can book a seat on a flight."

EX5 (Alechina 2008): "A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enroll in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students."

EX6 (provided by University of Vienna): “The pizza service offers different pizzas, which have a number and a name. Each pizza is made of various ingredients (one ingredient has a unique name). There is a customer file where customers are clearly identified by their telephone number, and the name and address are given. Each order comes from exactly one customer, includes any number of pizzas and is delivered by a driver who has a driver number and a name. The orders are identified by a sequential order number. On a tour, a driver can deliver multiple orders.”

EX7 (provided by University of Vienna): “A database for storing a private collection of books is required (yours or somebody else's). It should function both as an inventory of books as well as a help to find books to read. This database should focus on books and provide information surrounding those. Consider that books come in different editions (1st, 2nd etc.) and each edition is owned once at most. Some books are written by one author, others by many and others can also have editors. The existence of series (Dark Tower, Discworld etc.) being a collection of several books should also be considered in the data structure. Additionally the data structure should support finding books for a specific mood. It should also be possible to enter books that are not yet obtained, but are planned to be added to the collection, or in other words a wish list. The wish list should contain some extra data (e.g. date added, a price limit etc.) in addition to the normal information about books.”

EX8 (Karagiannis, Burzynski, and Miron 2017): “Participants at the summer school are either students or teachers. Each student registers for the NEMO Summer School providing, amongst others, their level of study (Bachelor, Master or PhD) and their field of study. Additionally each student provides her/his first name, last name, their country of provenience and e-mail address. Students attend courses during the summer school. Courses can be a lecture, a fundamentals exercise or application exercises. [The fundamental exercise is considered as one unit as it covers one topic, although it takes place in several sessions.] Each course has a title, is being given by one or more lecturers and takes places in a room. Every room has a name, a seating capacity, and technical equipment. Lectures and application exercises take place in a lecture hall, while fundamental exercises are conducted in PC-labs. Within the fundamentals exercise students are split in groups. Each group has a group number, a room (i.e. PC-lab) and a tutor. Teachers can be either lecturers or tutors. Each teacher has a first name, last name, host institution, and country.”

6.2 Appendix B Abstract English

Recent technological advances in natural language processing and data storage capabilities, have led to the increased utilization of text analytics in many different domains. This work discusses how text analytics can be utilized in the conceptual modelling domain. First of all, an overview of both domains is given. Then, the domains are combined and potential application areas of text analytics in conceptual modelling are discussed. This is followed by a detailed discourse on the use of text analytics for conceptual model generation, including the introduction of a platform- and domain- independent "generic text analysis approach for conceptual model generation", which is closely tied to the model design process itself. The purpose of the generic concept is to enable further research into this area and to ease the development of similar applications in the future. Furthermore, a listing of free tools which can be used for conceptual modelling or text analytics is included. Finally, the theoretical part of this thesis is supported by a comprehensive example that demonstrates how the generic concept can be put to practice. In this context, the "ER Text Converter", a tool that extracts Entity- Relationship modelling constructs from natural language text specifications is presented.

Note: The source code for the "ER Text Converter" is submitted on the enclosed CD.

6.3 Appendix C: Abstract German / Kurzfassung

Jüngste technologische Fortschritte in der maschinellen Verarbeitung natürlicher Sprache und der Datenspeicherung haben dazu geführt, dass Text Mining in vielen verschiedenen Bereichen verstärkt eingesetzt wird. In dieser Arbeit wird erläutert, wie Text Mining in der konzeptionellen Modellierung genutzt werden kann. Zunächst wird ein Überblick über beide Bereiche gegeben. Dann werden die Domänen kombiniert und mögliche Anwendungsgebiete werden diskutiert. Es folgt ein ausführlicher Diskurs über die Verwendung von Text Mining für die konzeptionelle Modellgenerierung, einschließlich der Beschreibung eines plattform- und domainunabhängigen "generischen Ansatzes für Text Mining zur konzeptionellen Modellgenerierung", welches sich stark an dem Vorgehen zur Modellbildung orientiert. Der Zweck des generischen Konzepts besteht darin, weitere Forschung in diesem Bereich zu unterstützen und die Entwicklung zukünftiger Anwendungen zu erleichtern. Darüber hinaus ist eine Auflistung von kostenlosen Tools enthalten, die entweder für konzeptionelle Modellierung oder für Text Mining verwendet werden können. Letztendlich, wird der theoretische Teil dieser Arbeit durch ein umfassendes Beispiel unterstützt, welches zeigt wie das generische Konzept in die Praxis umgesetzt werden kann. In diesem Kontext, wird der "ER Text Converter" vorgestellt, ein Tool, das Entity-Relationship-Modellierungskonstrukte aus Textspezifikationen extrahiert.

Hinweis: Der Quelltext für den "ER Text Converter" ist auf der beiliegenden CD hinterlegt.

7. Bibliography

Ackoff, Russell L. 1989. "From data to wisdom." *Journal of applied systems analysis* 16 (1):3-9.

Aggarwal, Charu C, and ChengXiang Zhai. 2012. *Mining text data*: Springer Science & Business Media.

Akilan, A. 2015. "Text mining: Challenges and future directions." Electronics and Communication Systems (ICECS), 2015 2nd International Conference on.

Al-Safadi, Lilac AE. 2009. "Natural Language Processing for Conceptual Modeling." *JDCTA* 3 (3):47-59.

Alder, Gaudenz, and David Benson. 2018. "About us." accessed 10 January 2018. https://about.draw.io/about-us/.

Alechina, Natasha. 2008. "Entity/Relationship Modelling - Database Systems Lecture 4." Last Modified 24 April 2008, accessed 1 April 2018. www.cs.nott.ac.uk/~psznza/G51DBS08/lecture4.pdf.

Allahyari, Mehdi, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. "A brief survey of text mining: Classification, clustering and extraction techniques." *arXiv preprint arXiv:1707.02919*.

Awad, Ahmed, Artem Polyvyanyy, and Mathias Weske. 2008. "Semantic querying of business process models." Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE.

Barker, Richard. 1990. *CASE method: entity relationship modelling*: Addison-Wesley Longman Publishing Co., Inc.

Bellinger, Gene, Durval Castro, and Anthony Mills. 2004. "Data, information, knowledge, and wisdom."

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*: " O'Reilly Media, Inc.".

Bögl, Andreas, Gustav Pomberger, Michael Schrefl, and Norbert Weber. 2014. Method and an apparatus for automatic semantic annotation of a process model. Google Patents.

Bögl, Andreas, Michael Schrefl, Gustav Pomberger, and Norbert Weber. 2008. "Semantic annotation of epc models in engineering domains to facilitate an automated identification of common modelling practices." International Conference on Enterprise Information Systems.

Bray, Tim, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. 1997. "Extensible markup language (XML)." *World Wide Web Journal* 2 (4):27-66.

Bridgwater, Adrian 2011. "French Model Specialist Modeliosoft Goes Open Source." Last Modified 11 October 2011, accessed 2 January 2018. http://www.drdobbs.com/open-source/french-model-specialist-modeliosoft-goes/231900564.

Btoush, Eman S, and Mustafa M Hammad. 2015. "Generating ER diagrams from requirement specifications based on natural language processing." *International Journal of Database Theory and Application* 8 (2):61-70.

Burzynski, Patrik. 2013. "Social Network for APIs." Dipl.-Ing., Wirtschaftsinformatik, University of Vienna.

Chen, Peter Pin-Shan. 1983. "English sentence structure and entity-relationship diagrams." *Information Sciences* 29 (2-3):127-149.

Chen, Peter Pin-Shan. 1988. "The entity-relationship model—toward a unified view of data." In *Readings in artificial intelligence and databases*, 98-111. Elsevier.

College, Butte. 2018. "The eight parts of speech." accessed February 24th, 2018. http://www.butte.edu/departments/cas/tipsheets/grammar/parts_of_speech.html.

Community, Apache OpenNLP Development. 2017. "Apache OpenNLP Developer Documentation." The Apache Software Foundation, accessed 8th January 2017. https://opennlp.apache.org/docs/1.8.4/manual/opennlp.html.

Cook, Diane, and Sajal Kumar Das. 2004. *Smart environments: Technology, protocols and applications*. Vol. 43: John Wiley & Sons.

Cunningham, Hamish, Valentin Tablan, Angus Roberts, and Kalina Bontcheva. 2013. "Getting more out of biomedical documents with GATE's full lifecycle open source text analytics." *PLoS computational biology* 9 (2):e1002854.

Fan, Weiguo, Linda Wallace, Stephanie Rich, and Zhongju Zhang. 2006. "Tapping the power of text mining." *Communications of the ACM* 49 (9):76-82.

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. "From data mining to knowledge discovery in databases." *AI magazine* 17 (3):37.

Friedrich, Fabian, Jan Mendling, and Frank Puhlmann. 2011. "Process model generation from natural language text." Advanced Information Systems Engineering.

George, Harry. 2002. "Dia Tutorial." Last Modified 2 June 2002, accessed 10 January 2018. http://www.seanet.com/~hgg9140/comp/diatut/all/all.html.

Grimes, Seth. 2008. "Unstructured data and the 80 percent rule." *Carabridge Bridgepoints*.

Gupta, Vishal, and Gurpreet S Lehal. 2009. "A survey of text mining techniques and applications." *Journal of emerging technologies in web intelligence* 1 (1):60-76.

Han, Jiawei, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*: Elsevier.

Harel, David, and Bernhard Rumpe. 2004. "Meaningful modeling: what's the semantics of" semantics"?" *Computer* 37 (10):64-72.

Hensgen, Paul. 2013. "Umbrello UML Modeller Handbook." accessed 3 January 2018. https://docs.kde.org/trunk4/en/kdesdk/umbrello/index.html.

Hofmann, Markus, and Andrew Chisholm. 2016. *Text Mining and Visualization: Case Studies Using Open-source Tools*. Vol. 40: CRC Press.

Hogenboom, Frederik, Flavius Frasincar, and Uzay Kaymak. 2010. "An overview of approaches to extract information from natural language corpora." *Information Foraging Lab*:69.

Hornsby, Kathleen Stewart, and Naicong Li. 2009. "Conceptual framework for modeling dynamic paths from natural language expressions." *Transactions in GIS* 13 (s1):27-45.

Hotho, Andreas, Andreas Nürnberger, and Gerhard Paaß. 2005. "A brief survey of text mining." Ldv Forum.

Ingersoll, Grant. 2015. "5 open source tools for taming text." Opensource.com, Last Modified 26 December 2017. https://opensource.com/business/15/7/five-open-source-nlp-tools.

Karagiannis, Dimitris, Robert Andrei Buchmann, Patrik Burzynski, Ulrich Reimer, and Michael Walch. 2016. "Fundamental Conceptual Modeling Languages in OMiLAB." In *Domain-Specific Conceptual Modeling*, 3-30. Springer.

Karagiannis, Dimitris, Patrik Burzynski, and Elena-Teodora Miron. 2017. "The "IMKER" Case Study - Practice with the Bee-Up tool." Last Modified 06 August 2017, accessed 12 December 2017. http://vienna.omilab.org/repo/files/Bee-Up/The_IMKER_Case_Study.pdf.

Karagiannis, Dimitris, and Harald Kühn. 2002. "Metamodelling platforms." EC-Web, vol. 2455, p. 182.

Kasemsap, Kijpokin. 2017. "Text mining: Current trends and applications." *Web data mining and the development of knowledge-based decision support systems*:338-358.

Kaur, Arvinder, and Deepti Chopra. 2016. "Comparison of text mining tools." Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2016 5th International Conference on, pp. 186-192. IEEE, 2016.

Kühne, Thomas. 2006. "Matters of (meta-) modeling." *Software and Systems Modeling* 5 (4):369-385.

Levi, Primo. 1984. "The Periodic Table (1975)." *Trans. R. Rosenthal. New York: Schocken Books*.

Manning, Christopher D, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. "The stanford corenlp natural language processing toolkit." In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp. 55-60. 2014.

Molzberger, Lukas. 2017. "About Aika." accessed 8th January 2017. http://www.aika-software.org/index.html.

Montes, Azucena, Hasdai Pacheco, Hugo Estrada, and Oscar Pastor. 2008. "Conceptual model generation from requirements model: A natural language processing approach." In

International Conference on Application of Natural Language to Information Systems, pp. 325-326. Springer, Berlin, Heidelberg, 2008.

Mylopoulos, John. 1992. "Conceptual modelling and Telos." *Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development, New York: John Wiley & Sons*:49-68.

Omar, Nazlia, JRP Hanna, and Paul McKevitt. 2004. "Heuristic-based entity-relationship modelling through natural language processing." In Artificial intelligence and cognitive science conference (aics), pp. 302-313. Artificial Intelligence Association of Ireland (AIAI), 2004.

Omar, Nazlia, P Hanna, and P Mc Kevitt. 2006. "Semantic analysis in the automation of ER modelling through natural language processing." In Computing & Informatics, 2006. ICOCI'06. International Conference on, pp. 1-5. IEEE, 2006.

Research, University of Cambridge. 2013. "Our ambiguous world of words." University of Cambridge Research, accessed 12th November 2017. http://www.cam.ac.uk/research/features/our-ambiguous-world-of-words.

Rocha, Henrique, and Ricardo Terra. 2013. "TerraER–an Academic Tool for ER Modeling." *Methods and Tools* 1 (3):38-41.

Sagar, Vidhu Bhala R Vidya, and S Abirami. 2014. "Conceptual modeling of natural language functional requirements." *Journal of Systems and Software* 88:25-41.

Santorini, Beatrice. 1990. "Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision)." *Technical Reports (CIS)*:570.

Seidewitz, Edwin. 2003. "What models mean." *IEEE software* 20 (5):26-32.

Shwartz, Steven P, and C Schank Roger. 1987. *Applied natural language processing*: JSTOR.

Sintoris, Konstantinos, and Kostas Vergidis. 2017. "Extracting Business Process Models Using Natural Language Processing (NLP) Techniques." Business Informatics (CBI), 2017 IEEE 19th Conference on.

Stachowiak, Herbert. 1973. "Allgemeine Modelltheorie."

Sukanya, M, and S Biruntha. 2012. "Techniques on text mining." In Advanced Communication Control and Computing Technologies (ICACCCT), 2012 IEEE International Conference on, pp. 269-271. IEEE, 2012.

Swanson, Don R. 1987. "Two medical literatures that are logically but not bibliographically connected." *Journal of the American Society for Information Science* 38 (4):228.

Tahaghoghi, Seyed MM, and Hugh E Williams. 2006. *Learning MySQL: Get a Handle on Your Data*: " O'Reilly Media, Inc.".

Tjoa, A Min, and Linda Berger. 1993. "Transformation of requirement specifications expressed in natural language into an EER model." In International Conference on Conceptual Modeling, pp. 206-217. Springer, Berlin, Heidelberg, 1993.

Uduwela, Wasana C, and Gamini Wijayarathna. 2014. "Survey on tools and systems to generate ER diagram from system requirement specification." In Industrial Engineering and Engineering Management (IEEM), 2014 IEEE International Conference on, pp. 370-373. IEEE, 2014.

Wand, Yair, David E Monarchi, Jeffrey Parsons, and Carson C Woo. 1995. "Theoretical foundations for conceptual modelling in information systems development." *Decision Support Systems* 15 (4):285-304.

Yue, Tao, L Briand, and Yvan Labiche. 2010. *Automatically deriving a UML analysis model from a use case model*: Carleton University.

Yue, Tao, Lionel C Briand, and Yvan Labiche. 2011. "A systematic review of transformation approaches between user requirements and analysis models." *Requirements Engineering* 16 (2):75-99.