

# Model-Dependent Software Evaluation of Text-Processing Tools

## Masterthesis

in fulfillment of the requirements for the degree of Master of Science (M.Sc.)  
in Information Systems

submitted by  
Thomas Kaspers

First Assessor: Dr. Michael Möhring  
Institute for Information Systems Research

Second Assessor: Prof. Dr. Klaus G. Troitzsch  
Institute for Information Systems Research

Koblenz, December 2014



## Declaration / Erklärung

I declare that this thesis presents work carried out by myself and does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university. To the best of my knowledge, it does not constitute any previous work published or written by another person except where due reference is made in the text. The submitted written version is equal to the version on the CD-ROM.

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

	Yes	No
	Ja	Nein
I agree to make this work available in the library Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I agree to the publishment of this work on the internet Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	<input checked="" type="checkbox"/>	<input type="checkbox"/>

.....  
(City, Date)

.....  
(Signature)



## Zusammenfassung

Unstrukturierte Textdokumente enthalten viele Informationen die heutzutage mit automatisierten Methoden extrahiert werden können. In dieser Arbeit wird ein Framework entwickelt mit dessen Hilfe ein Evaluationsschema zur Evaluation von Textverarbeitungstools erarbeitet wird. Das Evaluationsschema basiert auf der *Modelabhängigen Softwareevaluation* und der modelabhängige Teil basiert auf dem Verarbeitungsprozess der von dem *Conceptual Analysis Process* abgeleitet ist. Der *Conceptual Analysis Process* ist im Rahmen des GLODERS Projektes entwickelt wurden. GLODERS ist ein EU-Projekt mit dem Fokus ein IKT Modell zu entwickeln welches helfen soll Extortion Racket Systems besser zu verstehen. Im Rahmes des GLODERS Projektes wurden Gerichtsdokumente eines Falles in Deutschland zu Verfügung gestellt, die in dieser Arbeit die Datengrundlage stellen. Zum Schutz involvierter Personen sind die Daten anonymisiert. Mit dem entwickelten Schema werden dann sechs verschiedene Softwarelösungen im Bezug auf die automatisierte Verarbeitung von unstrukturierten Textdokumenten evaluiert.



# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Aim . . . . .	2
1.2	GLODERS . . . . .	3
1.3	Extortion Racketeering/Protection Racketeering . . . . .	4
1.4	Structure . . . . .	5
<b>2</b>	<b>The Case</b>	<b>6</b>
<b>3</b>	<b>Theoretical Background</b>	<b>9</b>
3.1	Content Analysis . . . . .	9
3.2	Information Extraction . . . . .	10
3.3	Methods . . . . .	11
3.3.1	Language Identification . . . . .	11
3.3.2	Tokenization . . . . .	12
3.3.3	Stemming . . . . .	12
3.3.4	POS Tagging . . . . .	13
3.3.5	Named Entity Recognition . . . . .	14
3.3.6	Co-Occurrence . . . . .	14
3.3.7	Pattern Matching . . . . .	15
3.4	CRISP-DM . . . . .	15
3.5	Social Network Analysis . . . . .	17
3.6	Software Evaluation . . . . .	18
3.7	Model-Dependent Software Evaluation . . . . .	18
<b>4</b>	<b>Text Processing Tools</b>	<b>20</b>
4.1	Selection Process . . . . .	20
4.2	The Tools . . . . .	20
<b>5</b>	<b>Analysis</b>	<b>23</b>
5.1	The Evaluation Scheme Framework . . . . .	23
5.2	The Process . . . . .	24
5.2.1	Data Preparation . . . . .	25
5.2.2	Concept Identification . . . . .	27
5.2.3	Relationship Identification . . . . .	29
5.2.4	Concept Network Analysis . . . . .	29
<b>6</b>	<b>Evaluation</b>	<b>31</b>
6.1	Evaluation Scheme . . . . .	31
6.1.1	Model-Independent Criteria . . . . .	31
6.1.2	Model-Dependent Criteria . . . . .	33
6.2	Model-Independent . . . . .	36

6.2.1	IBM SPSS Modeler . . . . .	36
6.2.2	WEKA . . . . .	37
6.2.3	RapidMiner . . . . .	38
6.2.4	KNIME . . . . .	40
6.2.5	AutoMap . . . . .	41
6.2.6	GATE Developer . . . . .	41
6.3	Model-Dependent . . . . .	42
6.3.1	Data Preparation . . . . .	42
6.3.2	Concept Identification . . . . .	50
6.3.3	Relationship Identification . . . . .	56
6.3.4	Concept Network Analysis . . . . .	60
6.4	Comparison . . . . .	67
<b>7</b>	<b>Conclusion and Outlook</b>	<b>69</b>

## List of Figures

1	GLODERS Logo . . . . .	3
2	Hierarchical Organization (left) and Criminal Network (right) . . . .	4
3	First Extortion (2011) . . . . .	6
4	Second Extortion (2012) . . . . .	7
5	The Case . . . . .	8
6	Krippendorff Framework . . . . .	9
7	CRISP-DM Reference Model according to [Chapman et al., 2000] by Kenneth Jensen . . . . .	16
8	Process Model for Software Evaluation according to Ebert and Dumsloff	19
9	Evaluation Scheme Framework . . . . .	23
10	The Analysis Process . . . . .	24
11	Subprocesses of the Analysis Process . . . . .	25
12	IBM SPSS Modeler Stream . . . . .	43
13	WEKA StringToWordVector Filter . . . . .	44
14	RapidMiner Data Preparation Process . . . . .	45
15	RapidMiner Process Documents Subprocess . . . . .	46
16	KNIME Data Preparation . . . . .	47
17	KNIME Anonymization Sub Process . . . . .	48
18	IBM SPSS Modeler List of Concepts . . . . .	51
19	WEKA Named Entity Recognition Confusion Matrix . . . . .	51
20	RapidMiner List with Named Entities by POS-Filter . . . . .	52
21	RapidMiner Named Entity Recognition Example . . . . .	52
22	KNIME Part-Of-Speech tagging . . . . .	53
23	KNIME Named Entity Recognition . . . . .	54
24	KNIME Dictionary tagging . . . . .	55
25	IBM SPSS Modeler Concept Categories . . . . .	57



26	KNIME Term Co-Occurrence . . . . .	58
27	KNIME Index Query for Pattern Matching . . . . .	59
28	IBM SPSS Modeler Concept Map for Suspect 1 . . . . .	61
29	IBM SPSS Modeler Category Network . . . . .	62
30	RapidMiner Document View . . . . .	63
31	KNIME Tag Cloud . . . . .	64
32	KNIME Tag Cloud Process . . . . .	64
33	KNIME Social Network Analysis Process . . . . .	65
34	KNIME Social Network . . . . .	65
35	MetaNetwork in ORA . . . . .	66
36	GATE Developer Identified Concepts . . . . .	67
37	Evaluation Scheme . . . . .	76
38	Evaluation Scheme (continued) . . . . .	77

## List of Tables

1	Tool Selection Criteria . . . . .	20
2	Selected Tools Overview . . . . .	21
3	Selected Tools Overview 2 (continued) . . . . .	21
4	Selected Tools Overview 3 (continued) . . . . .	22



## **Abstract**

In this work a framework is developed that is used to create an evaluation scheme for the evaluation of text processing tools. The evaluation scheme is developed using a model-dependent software evaluation approach and the focus of the model-dependent part is the text processing process which is derived from the *Conceptual Analysis Process* developed in the GLODERS project. As input data a German court document is used containing two incidents of extortion racketeering which happened in 2011 and 2012. The evaluation of six different tools shows that one tool offers great results for the given dataset when it is compared to manual results. It is able to identify and visualize relations between concepts without any additional manual work. Other tools also offer good results with minor drawbacks. The biggest drawback for some tools is the unavailability of models for the German language. They can perform automated tasks only on English documents. Nonetheless some tools can be enhanced by self-written code which allows users with development experience to apply additional methods.

# 1 Introduction

Extortion Racket Systems are widely spread all over the world. They vary in their appearance and structure. The best known group performing extortion is the Mafia. Other groups for example are motorcycle gangs like the Hells Angels or the Bandidos or the Japanese originating Yakuza. These groups are considered organized crime syndicates and they are known to participate in organized crime like protection racketeering or extortion.

In Germany extortion racketeering is carried out by national and foreign groups. Turkish (16.1%) as well as Russian and Vietnamese groups (9.7% each) play a major role [Bundeskriminalamt, 2008, p.82-92]. In Germany the perpetrators also include motorcycle gangs. In 2003 to 2005 0.1% of all reported offences in Germany had been classified as extortion offences. In absolute numbers the amount is close to 6000 reported extortion offences per year with a clear up rate of around 85% [Tangenberg, 2007]. In a study on extortion racketeering, performed in 2008 by the Joint Research Center on Transnational Crime (University of Trento and University of Milano in Italy), all European countries had been analyzed. The study rates the seriousness of extortion racketeering in Germany with medium whereas other countries like Italy are rated with a high value. The Italian situation with reference to extortion racketeering is one of the most complex, owing to the types of actors involved and the relationships they create with the victims of extortive demands [Impresa, 2007]. Organized extortion racketeering as an activity of organized crime is mostly concentrated in some southern regions of the country. It is generally employed by mafia-type organizations in order to gain better control over the territory at local level and infiltrate the legitimate economy. These two examples of European countries show that extortion racketeering is a common crime. When extortions are reported to the police the case is usually brought to court where all involved people are questioned to determine the suspect.

The European project GLODERS aims to develop an ICT model to understand Extortion Racket Systems. Within the scope of the GLODERS project data from court files qualify for further research to gain information out of raw court files in order to reveal structures automatically. In the last few years the amount of tools for text processing increased and big machine learning suites obtained extensions for text processing functionality. All these tools offer different amounts of text processing methods that are required to analyze structured and unstructured data, but there is no tool optimized to analyze data from extortion racket cases.

## 1.1 Aim

The aim of this work is to evaluate different text processing tools, which are previously selected, based on their ability to process extortion related data and to find a suitable tool. Three research questions are used to find answers to subgoals to achieve the main goal of this work.

1. What is the structure of the text processing workflow for extortion related data?
2. What are the evaluation criteria?
3. Which phases of the process are supported by which tool?

To answer the first question the *Conceptual Analysis Process* of the GLODERS project is used. The process is derived and methods are applied which fulfill and support the steps of the process. The second research question is based on the first one. The evaluation criteria are developed based on a framework, literature and the process resulting of the first research question. Afterwards six previously selected tools are evaluated using the scheme to determine which tools support which phases of the process and which tools are able to produce results that can be compared with results created by manual analysis.

## 1.2 GLODERS

The GLODERS (The Global Dynamics of Extortion Racket Systems) project is a European project funded by the European Union under the 7th framework program. The aim of the project is “to develop an ICT model for understanding Extortion Racket Systems, an aspect of the dynamics of the global finance system”<sup>1</sup> (Figure 1).



Figure 1: GLODERS Logo

The project started in October 2012 and is scheduled to be finished end of September 2015 after 3 years. The project is coordinated by the University of Surrey in the United Kingdom. The three other participants are the National Research Council in Rome, the University of Palermo and the University of Koblenz-Landau in Germany. In previous work extortion related court documents had been analyzed. Therefore a process was developed that will be the baseline of this work. Additionally a German case was made available for the project in the end of 2013 that will serve as the data input for this work. In chapter 3.7 the case is described in detail to give a short overview of the complexity.

---

<sup>1</sup><http://www.gloders.eu>

### 1.3 Extortion Racketeering/Protection Racketeering

Extortion racketeering and protection racketeering are criminal acts of obtaining money by extorting businesses or single persons. Extortion and protection racketeering is usually performed by organized crime groups. The difference between extortion racketeering and protection racketeering is the intention of the offender. In extortion racketeering the offender usually extorts the victim by threaten him with bodily harm, whereas in protection racketeering the offender offers the victim protection if the victim pays enough money to the offender. The offenders operate outside the area of the law. If a victim is not willing to pay, the offender may threaten the victim on a direct way which results in extortion racketeering. Extortion Racketeering exists all over the world. One group that is known for performing extortion and protection racketeering is the Mafia. Estimations show that 160 million Euro are paid by shops and businesses in the Palermo region as protection money per year [Pisa, 2008]. The amount paid by the shop or business depends on the kind of goods sold. 4 out of 5 Sicilian businesses pay protection money [Moore, 2007] which shows the huge size of protection racketeering especially in this region.

There are two different types of extortion racketeering, systemic and casual, linked to three main variables [TRANSCRIME, 2008]:

- The organizational structure of the criminal crime group that engages in extortion racketeering
- Its strong presence at local territorial level
- The victim-offender relationship

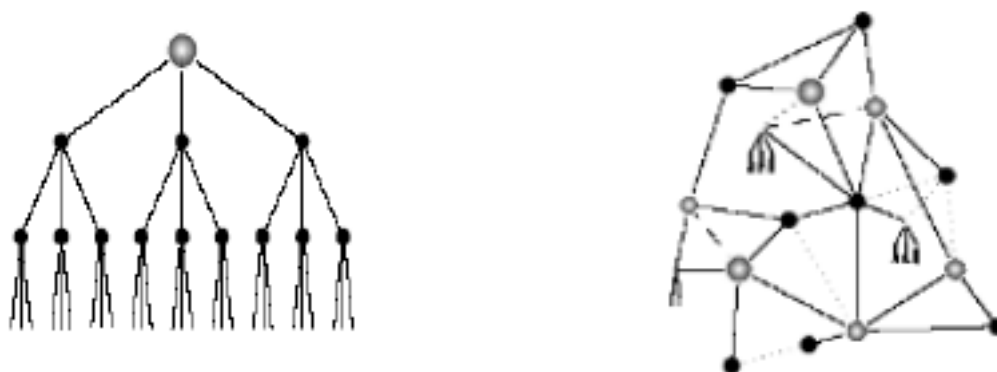


Figure 2: Hierarchical Organization (left) and Criminal Network (right)

There are two types of organizational structure that can be distinguished (Figure 2). The first one is a hierarchical organization. It is the most common form of organized criminal group identified in a study [TRANSCRIME, 2008]. It consists of a clear hierarchy with a single leader. The allocation of tasks inside the

organization is clear and there is usually a code of conduct, which is of course not officially recorded. The second type is a criminal network which usually consists of small numbers of individuals. Not every person inside a network is connected to each other, but they can be connected with the help of another individual [UNODC, 2008]. The territorial aspect can be differentiated between systemic extortion and casual extortion. In systemic extortion the extortion phenomenon is well rooted and well spread over a territory. Extortion is a part of criminal business and it is performed routinely. In casual extortion on the other hand the extortion is an episodic phenomenon and it is not spread over a territory. In this case extortion is not practiced routinely. The victim offender relationship can be separated in three categories - parasitic, symbiotic and predatory. In a parasitic relationship the perpetrator demands several payments over a long period of time, whereas in a symbiotic relationship the victim also receives an illicit benefit. Protection racketeering is a symbiotic relationship where the victim retrieves protection as benefit. The last category is the predatory relationship. In this case the extortive payment is demanded only once.

#### 1.4 Structure

The first chapter of this work briefly introduced the European project GLODERS, addressing the global dynamics of extortion racket systems, and the motivation for this work.

The second chapter contains a detailed description of the case which was made available for the project for research including a classification of the type of extortion previously described. The case is used as data basis for the evaluation of the text processing tools performed in a later chapter.

The third chapter provides the theoretical background of methods and technologies used in text processing and in the process of the evaluation as well as basic background knowledge about software evaluation.

To obtain a brief overview of the text processing tools, the fourth chapter then briefly introduces the tools after describing their selection process. The tools evaluated range from specialized text processing tools to machine learning tools with text processing extension and from open source to commercial software.

Afterwards the framework is developed to create the evaluation scheme which is then used for the evaluation. The *Conceptual Analysis Process* is derived which then serves as the base line for one part of the evaluation scheme.

In the sixth chapter the previously selected tools are then evaluated using the scheme. The result produced during the evaluation are then compared with the manually retrieved results.

The last chapter will then offer a guidance for a tool selection to process extortion related text documents and an outlook is provided containing thoughts about future developments.

## 2 The Case

In 2013 court documents to a case in Germany had been made available for the GLODERS project to research. The case is a small case covering two incidents of extortion racketeering involving two main suspects, two accomplices, two victims who own a business together and seven witnesses. The documents are protocols of testimonies and other reports covering the incidents of the case. Since the case is small it can be easily analyzed manually which offers a good way for comparison with results which are achieved while evaluating different text processing tools. In this chapter the case is described to give an overview. Both incidents and the overall relations are described. The analysis focuses on relations between persons. To protect any persons involved in the case all names and places are anonymized. The type of extortion covered in this case is more a casual extortion than a systemic one. The organizational structure identified during the research is a small criminal network. Both incidents are casual extortions. They occur two times in a time frame of a year and they are not practiced routinely. The victim-offender relationship for this case is predatory. There is no symbiotic or parasitic relationship since the extortion only occurred twice. If the extortion would go on for a longer time frame it could be considered as a parasitic relationship.

The first incident took place in 2011. The two main suspects (*Suspect 1* and *Suspect 2*) bodily harmed and extorted the proprietor (*Victim 1*) outside of the business. The suspects threaten the victim and the incident is not reported to the police. Two people (*Witness 5* and *Witness 7*) witness the extortion but they do not report it to the police at this point of time. Some days later the accomplice (*Suspect 3*) collects the money of the victim (Figure 3).

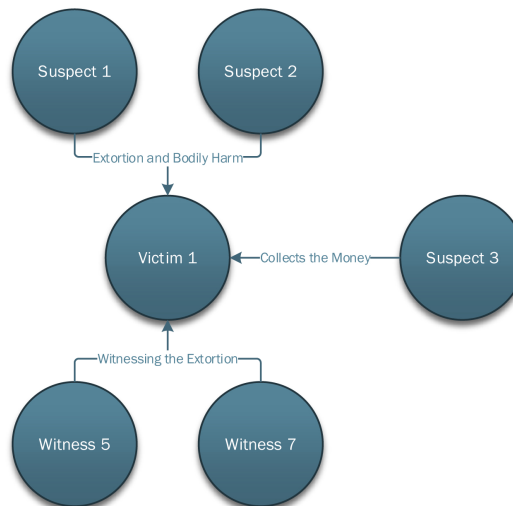


Figure 3: First Extortion (2011)



The second incident took place in 2012 and is a little bit more complex since more people are involved. Nobody was bodily harmed, but this time the incident was reported to the police. This time *Suspect 2* tries to extort *Victim 1* again. He refers to the previous extortion and bodily harm. The affiliate of *Victim 1* is on site during the conversation. The extortion focuses the business of both. Therefore he is referred as *Victim 2*. Nobody is bodily harmed but this time other people get to know of the extortion and motivate the victims to inform the police. The second accomplice is the mother of the second suspect who is the extorter in this case. She tries to intimidate the wife of victim 2 (*Witness 4*). Multiple relatives of the victims are involved and finally inform the police (Figure 4).

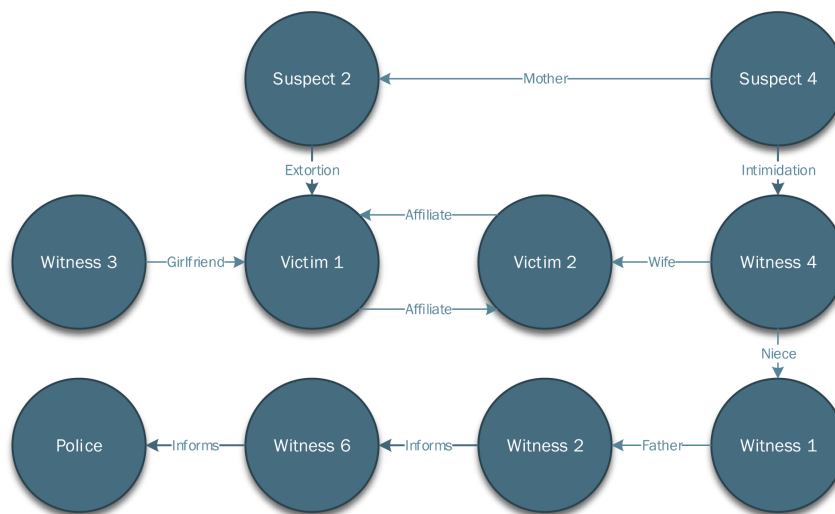


Figure 4: Second Extortion (2012)

Both incidents evolve around the main victim (*Victim 1*) and his small business (*Place 1*) which he runs together with *Victim 2*. The first extortion did not take place at the business itself. It was outside at a friend's place (*Place 2*). The extortion and bodily harm is witnessed by two witnesses who promise to not report the incident. After the second extortion both witnesses describe the first incident in their testimony. The second extortion is only performed by *Suspect 2*, who also participated in the first incident, and this time it also involved *Victim 2*. The first suspect is not involved in the second incident. More witnesses are involved in the second incident which lead to revealing both incidents to the police. The whole structure of both incidents combined shows that both extortions differ in size and it also shows a complex structure of relations between all involved people (Figure 5). Actions are visualized with solid directed edges and relations are visualized with dashed directed edges.

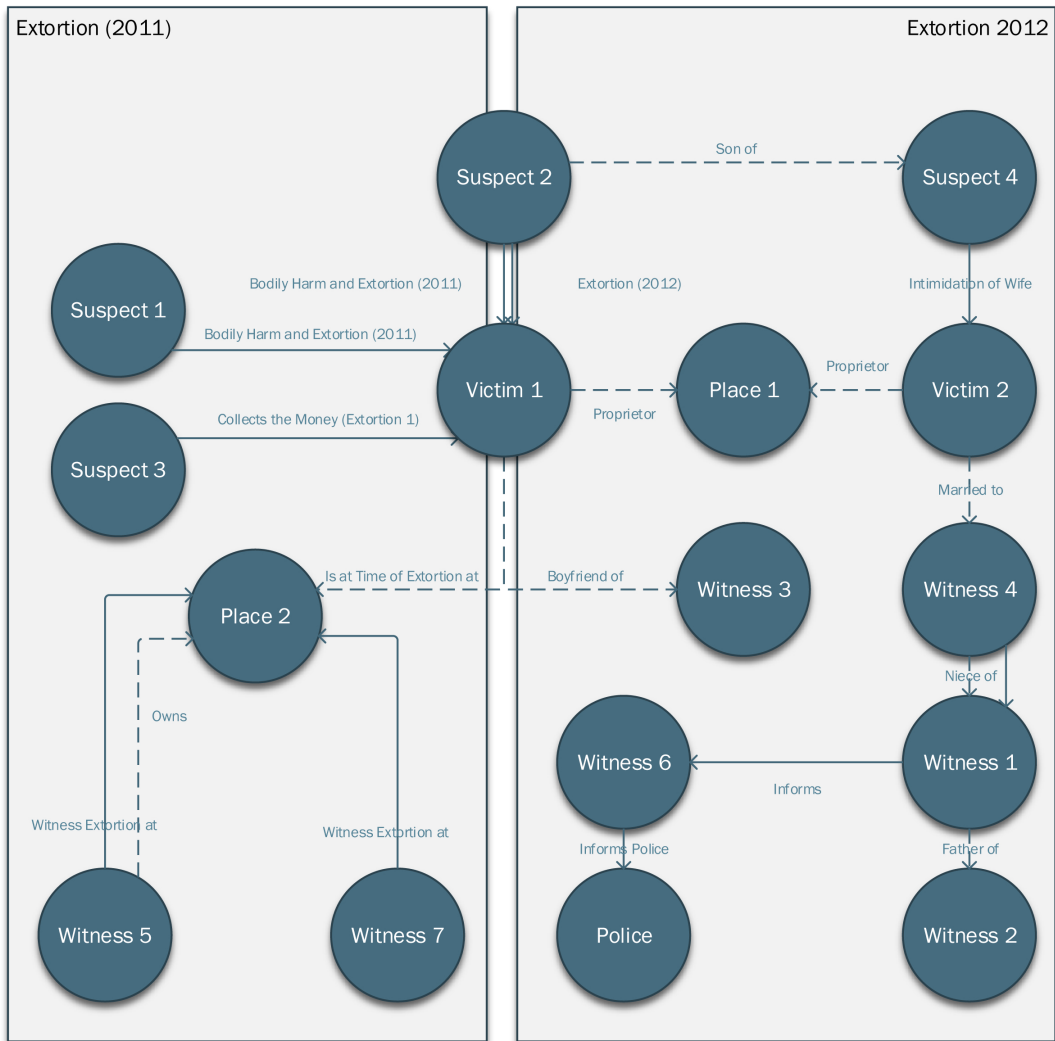


Figure 5: The Case

### 3 Theoretical Background

A simple sentence can have various meanings for different recipients. A human is able to use additional knowledge to interpret the sentence. A different perspective or background knowledge can lead to various actions since the pragmatism is based on the sentence and the background information. "The train arrives at 4:30pm." will inform a person that a train will be at 4:30pm at a specific location. For most people this information is useless unless they intent to pick someone up from this train or if they want to take the train themselves [Searle, 1969]. In this work text data is analyzed with different text processing tools to gain information about relations between named entities mentioned in the document. The different tools are evaluated based on a evaluation scheme that is developed with a model-dependent software evaluation framework. Therefore this chapter will give background knowledge of text processing, its methods and software evaluation.

#### 3.1 Content Analysis

In social sciences, Content Analysis is the general name for the methodology and technologies to analyze the content of messages [Holsti, 1969]. Basically any technique to make inferences by objectively and systematically identifying specified characteristics of messages is defined as Content Analysis. Content Analysis therefor is a research technique to make replicable and valid inferences from texts to the context of their use [Krippendorff, 2012]. Nowadays Content Analysis is often referred as *Text Processing*. The task of text processing is supported by systems and tools with either manual or automatically steps. Klaus Krippendorff invented a framework for *Content Analysis* (Figure 6). The general process in content analysis is to analyze data based on a specific research question. The text is analyzed to gain answers to the research question. Therefor methods are applied to infer these answers.

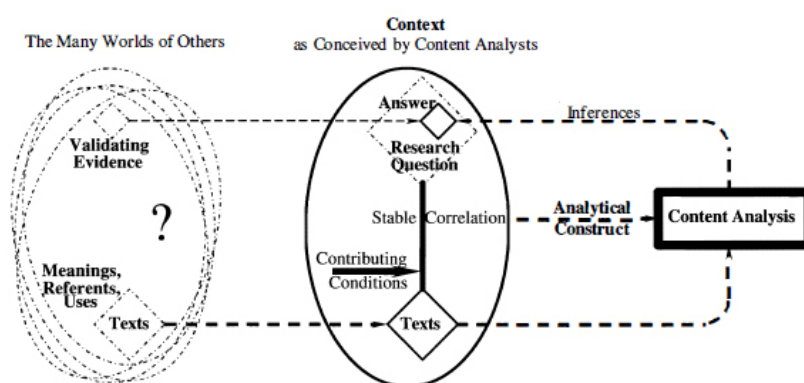


Figure 6: Krippendorff Framework

The Krippendorff framework differentiates between two views. The first view is the view of different people. These people may have knowledge about the information in the text. The second view is the view of the person who conducts the content analysis. This person has the text and wants to analyze it to gain information about a specific research question. The content analyst wants answers for his questions and therefore he performs the content analysis using different methods. Afterwards the answers can be validated with the evidence of the people of the first view.

The aim of the analysis is to determine the value of one or more theoretically interesting variables based on a message content [van Atteveldt, 2008]. Van Atteveldt differentiates between two common methods to conduct *Content Analysis*. The first one is *Thematic Content Analysis* which is a manual approach where human coders classify documents using a pre-defined categorisation scheme. The biggest drawback of this method is that human coding is expensive and human coders need to be extensively trained to achieve reliable coding [van Atteveldt, 2008]. The second alternative is called *Semantical Network Analysis* or *Relational Content Analysis* [Roberts, 1997]. Semantic Network Analysis first represents the content of the messages as network of objects. This network can be queried to answer the research question. The automated methodology to extract the information out of a text to create a network is called *Information Extraction*. This work focuses on a more automatic way of text processing whereby the first method of human coding is not used. The network analysis method is used, since it allows to create a network using automatic text processing methods which then can be queried by an analyst.

### 3.2 Information Extraction

Information Extraction (IE) is an automatically approach used for *Content Analysis*. *Information Extraction* describes the process of extracting information from actual text documents by computer at high speed [Wilks, 1997]. It can be confused with *Information Retrieval (IR)* which is used to select a relevant subset of documents from a larger set. IR retrieves texts from users in response to their queries, whereas IE processes texts into fixed format unambiguous data. IE can also exist on the same level as an IR system being used to automatically extract data on some topic of interest to a user from a corpus of texts and use this structured data as input to a spreadsheet or database [Smeaton, 1997]. The distinction between IE and IR is not totally clear everywhere and is a question of degree but usually the process starts with IR and in is succeeded by IE.

Most of the information in documents exist only in natural language form. Information Extraction helps to analyze the information by distilling a document into a more structured form in which individual facts are accessible [Grisham, 1997]. Huge documents with widely scattered information are reduced to a simple data base using different methods. The simple data base then can be used to further analysis of the data. Machine readable dictionaries are used to enrich the document with additional information and machine learning algorithms are used to derive structures and then adapt them to more data. Common methods used in IE are described in

the following section.

In 1997 when the topic of IE became of more interest it was seen as a new technology rather than a new idea since the idea of extracting information from data was not new. In 1964 papers were published with titles like "Text searching with templates" [Wilks, 1987]. The earliest effective IE work was that of Sager [Sager, 1981] within a medical domain, and constituting a long- running project combining surface syntax analysis and the use of templates [Wilks, 1997]. Around 1995 multiple IE projects were supported by the European Commission, covering industrial-backed applications or research projects. AVENTINUS for example is a project to enhance existing information systems for prevention and detection of offences by using IE methods. The project was used to underpin European police and government law-enforcement in the drugs field<sup>2</sup>.

### 3.3 Methods

Multiple methods are used in this work to extract information from unstructured documents which in this case are the court documents of the case described in chapter 2. The complexity of the methods differ and range from simple methods like *Language Identification* to more complex methods like *Named Entity Recognition*. The methods and their functionality are described in the following section.

#### 3.3.1 Language Identification

Language Identification is one of the simple tasks of Information Extraction [Grefenstette, 1995] and there are multiple approaches. One of the easiest approach is to rely on given meta data. This approach is quite inflexible since it requires the given language in the meta data of a document which is not always given. An approach for web pages is to simply rely on the domain name. In this case the assumption is made that a site on a *'de'* domain always provides text in German. This assumption has some flaws. For example countries with multiple languages like Belgium or Canada have one domain and the language of the content can differ. Nowadays most web pages offer multi language support which makes the approach not effective. Additionally since there are domains which are not bound to a country like *'ag'*. A more complex approach which is the most common approach for language identification is to identify the language based on given character sequences. Based on a dictionary containing common short words or common sequences the language of the document can be identified based on calculated probabilities from any large text [Grefenstette, 1997]. Common short words appearing in German texts are articles like *'der'*, *'die'* or *'das'*. English documents usually contain a huge amount of *'the'*. Every language has different common words that can be used to identify a language. Multiple common short words are used to identify a language to enhance the quality of identification. The advantage of this approach is that it does not rely on any given meta data or additional information.

---

<sup>2</sup>[http://cordis.europa.eu/project/rcn/34067\\_en.html](http://cordis.europa.eu/project/rcn/34067_en.html)

### 3.3.2 Tokenization

A tokenizer segments an input stream of data into an ordered sequence of tokens. Each token corresponds to an inflected word, a number, a punctuation mark, or other kind of unit to be passed on to subsequent natural language processing [Grefenstette, 1997]. The process of tokenization is language dependent and requires a model build for the specific language. In some languages a sequence of words needs to be considered as single token for further linguistic treatment. In English combined words are usually separated. '*Criminal Network*' for example is a sequence of two words that have to be seen as a single token. Another example for language specific cases is the use of separators inside words. In French the separator splits a word into two tokens whereas in English it usually combines a sequence of words belonging together. '*l'amour*' can be considered as two single tokens whereas '*won't*' should be considered as a single token. Depending on the algorithm used for tokenization punctuations are not included in the resulting list of tokens. For the algorithms it is important to identify sentence boundaries [Grefenstette and Tapanainen, 1994] since parser tools work on entire sentences. Besides the tokenization into word algorithms exist to tokenize an input stream into sentences, linguistic sentences or to tokenize it based on given regular expressions.

### 3.3.3 Stemming

Stemming is a method which is used to reduce words to their stem. A stem of a word is a form to which affixes can be attached [Sampson and Postal, 2005]. The words '*consistency*' and '*consistent*' both have the same stem which is '*consist*'. A stem is the part of a word that is common to all its inflected variants [Kroeger, 2005]. Stemming can be important since it helps to identify important words inside a document which appear in different versions. Nonetheless there is also a theory that shows that the quality can be reduced by stemming since words might be reduced to a wrong stem which can result in wrong classifications.

Algorithms used to reduce words to their stems are called *Stemming Algorithms* or stemmers. The first stemmer was developed in 1968 by Julie Beth Lovins [Lovins, 1968] and had huge impact on following stemming algorithms. Stemming is language dependent since every language appends different affixes to the stem. The first stemmers developed were always build for English words. In 1979 Martin Porter developed a stemming algorithm nowadays known as the *Porter Stemmer*<sup>3</sup>. The stemmer was released in 1980 and is still used in many applications. The stemmer was released as an official free software which allowed different implementations for multiple development languages. The *Snowball Stemmer*<sup>4</sup> is the stemmer which is suggested by Martin Porter since it derived from the Porter Stemmer and has a higher efficiency. The Snowball Stemmer is available for different languages

---

<sup>3</sup><http://tartarus.org/martin/PorterStemmer/>

<sup>4</sup><http://snowball.tartarus.org/algorithms/english/stemmer.html>

including German<sup>5</sup>, Italian and French. The algorithm used for different languages is similar but some rules are different. The algorithm for the German language for example includes more vowels and also special characters like 'ö' or 'Ä'. For multilingual documents multilingual stemmers can be used. They use morphological rules to find the best fit.

### 3.3.4 POS Tagging

Every word inside a sentence has a specific role and sentences usually follow language specific rules defining the syntax. A method used to identify the role of a word inside a sentence is called *Part-Of-Speech tagging*. Each word gets a part-of-speech symbol assigned using a model which is developed for the language. Syntax in languages differ which requires language dependent models trained for the specified syntax to enhance the sensitivity of the model. The tags assigned by the tagger depend on the model and the tagset which is used. For German tagging the *Stuttgart-Tübingen-Tagset (STTS)*<sup>6</sup> is used. English documents can be tagged using the *PennTreebank tagset*<sup>7</sup> developed by the University of Pennsylvania and French documents can be tagged using the *French PennTreebank* tagset which is derived from the PennTreebank tagset and provided by the National Center for Scientific Research of France.

A word can have different meanings depending on the context and the part inside the sentence. The word '*watches*' can be a noun or it can be a verb. The part-of-speech has to be determined by taking the rest of the sentence into account. In some languages nouns can be identified quite easily since they are mostly preceded by an article [Grefenstette, 1997].

A common method is to create a manually tagged corpus which then can be used to calculate probabilities about the frequency with which tag sequences are found [DeRose, 1988]. To store the probabilities for a word enough text needs to be tagged [Briscoe et al., 1994] to allow the creation of a *Hidden Markov Model* using the statistical information to disambiguate the parts-of-speech by calculating the most probable part through all the possible tags [Charniak, 1993]. Other alternative techniques are to train a tagger over untagged text so that the entropy of tag transitions is minimized [Cutting et al., 1992] or to use a manually tagged corpus to generate correction rules [Brill, 1992] or to create a set of rules describing which transitions are impossible [Voutilainen et al., 1992]. The common tagger build into the text processing tools which are evaluated in this work use models build with the Hidden Markov Model on text corpora for specific languages.

---

<sup>5</sup><http://snowball.tartarus.org/algorithms/german/stemmer.html>

<sup>6</sup><http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html>

<sup>7</sup><http://www.cis.upenn.edu/treebank/>

### 3.3.5 Named Entity Recognition

Named entities can be identified with *Part-of-Speech tagging*. Names appear frequently in many types of texts, and identifying and classifying them simplifies further processing. Names are important as argument values for many extraction tasks [Grisham, 1997]. *Named Entity Recognition* is a method that uses a combination of regular expressions stated in terms of parts-of-speech syntactic features and orthographic features like capitalization to further identify the type of named entity which can be 'Person', 'Location' or other custom types. Named Entity Recognition is language dependent and requires a model trained for a specific language to guarantee a high accuracy. In the beginning the highest performance was reached using hand coded patterns but the performance of systems which learn from annotated corpora has been improved since then and was a few percent below that of hand coded system in 1997 [Bikel et al., 1997]. *OpenNLP* for example provides a model built for English documents using an annotated corpora. Models for different languages are available and trained using annotated corpora for the specific language. In the early days simple methods were used to identify names of persons or companies. A company name could be identified by the type of the company which is usually attached to the name like 'Ltd.' or 'GmbH'. Names were identified by preceding titles, common first names or middle initials. Often a single person is referred to in different ways in a single document. Modern *Named Entity Recognition* is able to identify aliases and match them to the corresponding person with high accuracy.

### 3.3.6 Co-Occurrence

Words inside a document often relate to other words in a document. *Co-occurrence* describes a relation between two words. The level of co-occurrence can be adjusted to find more specific relations inside a document. The most general approach is co-occurrence in a document where words are assumed to be topically related [Momtazi et al., 2010]. The distance inside a document is irrelevant as is their order of appearance. Document co-occurrence has been successfully used in many NLP applications such as automatic thesaurus generation [Manning et al., 2008]. The most common approach which is also used by tools in this evaluation is co-occurrence in a sentence. It assumes that a word  $w$  is related to a word  $w'$  inside the same sentence. An even narrower approach is the co-occurrence in a window of text. It only considers terms in a window surrounding a word. Relations are based on their proximity in text [Krukow, 2013]. Usually a window of fixed size is moved along the text. The size of the window is a flexible parameter and can be changed. Previous research suggests different parameters. Seven is suggested by Diesner [Diesner, 2012] and five by Church and Hanks [Church and Hanks, 1990]. The last and most specific co-occurrence is the co-occurrence in a syntactic relationship. It is based on the sentence co-occurrence with the additional requirement that both words need to be related syntactically. For example objects related to the same topic.



### 3.3.7 Pattern Matching

Matching and searching elementary discrete structures arose in Computer Science and their relevance was expected to grow even further due to the increasing amount of digital available information [Apostolico and Galil, 1997]. *Pattern Matching* describes the process of checking a sequence of tokens to find a pattern which is defined in before. A pattern is usually defined using a regular expression. Pattern matching only returns string that match exactly the regular expression. *Named Entity Recognition* uses pattern matching to classify a named entity to a more detailed type. Pattern matching can be combined with Named Entity Recognition and Co-Occurrence to identify relations between two named entities and to define the type of their relation.

```
Suspect1[NE(PERSON)] extorts[VAFIN] Victim1[NE(PERSON)]
```

In specific the verb between two named entities can be used to describe the relation. In the example the two named entities are related by the verb between them. The words are tagged using the STTS tagset. A regular expression used to identify words like this require the Part-of-Speech tagging and Named Entity Recognition in previous steps. Regular expressions are complex and have to be defined manually. The rules have to be tagset dependent since different tagsets use different tags for same words. The German STTS tagset for example defines twelve different tags for verbs. The tags depend on the type and the time. It differentiates between different forms of a verb. The imperative for example (*VVIMP*) has a own tag since it has strong influence on the meaning of a sentence.

## 3.4 CRISP-DM

The *CRoss Industry Standard Process for Data Mining (CRISP-DM)* reference model was conceived in late 1996 [Chapman et al., 2000]. It was developed to provide a standard process model for data mining independent of a specific industry, tool or application. In 1997 the inventors formed a consortium and obtained funding from the European Commission. The CRISP-DM 1.0 version was presented in 2000 and in 2008 the construction of version 2.0 started [Marbán et al., 2009].

A life cycle of a typical data mining project consists of six phases which are the six phases which are part of the CRISP-DM reference model. Each phase interacts with another phase and can be bidirectional (Figure 7). The six phases of the process are *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation* and *Deployment*.

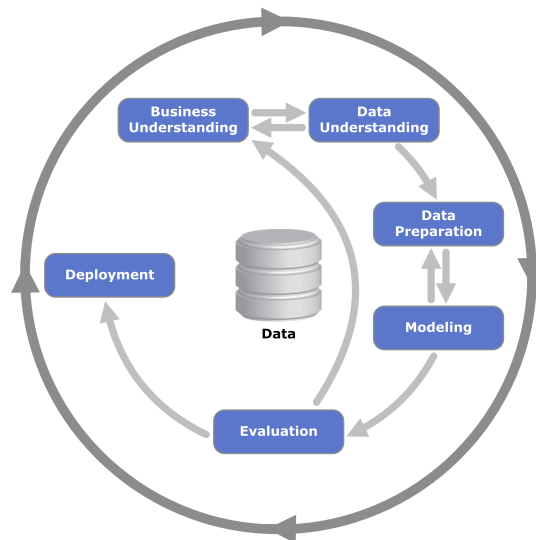


Figure 7: CRISP-DM Reference Model according to [Chapman et al., 2000] by Kenneth Jensen

The first phase is *Business Understanding*. It focuses on understanding the project objectives and requirements from a business perspective. This knowledge then can be used to identify the data mining problem and to develop a plan to achieve the objectives.

After the objectives are clear and the data mining problem is developed it is important to understand the data. Different data requires different methods to retrieve good results. Understanding the data can lead to changes of the data mining problem wherefore a bidirectional connection exists between the first two phases.

Missing values or outliers can lead to bad and flawed models. Therefore it is important to prepare the data and create a final dataset. In this phase missing values can be replaced using different methods and outliers can be normalized. The initial data can be reduced by attribute selection or the data can be cleaned using various other methods. Methods in this phase can be performed multiple times and are not bound to a specific order. Required methods depend on the data and the desired output.

When the final dataset is created it can be used as input for a various range of modeling techniques. The type of model applied can have impact on the structure of the dataset which then requires to run the data preparation phase once more. Different models can be used for the same problem.

After a model is built which is assumed to have a high quality based on the previous phases it is common to evaluate the result before the model is used in a productive environment. If the model fulfills the defined business objectives it can be deployed in the next phase. If this is not the case the process has to start over and the objectives have to be refined or the data has to be prepared different to fulfill all defined goals.

After validating a high quality model it can be deployed. The last phase of the process is used to deploy the model in a live environment or just to create a report based on the generated knowledge. In some cases it can be the creation of a whole automated data mining process for a organization.

In this work the model is used to underpin the selection of the criteria for the evaluation scheme. Especially the first phases are important and used to verify the importance of the *Data Preparation* phase. The CRISP-DM reference model also shows that the process is not necessarily linear.

### 3.5 Social Network Analysis

In *Social Network Analysis* relations between individual actors are analyzed using network theory. The origins of an approach to social structure explicitly using ideas of a *social network* are difficult to discern. In the 1930s the network thinking emerged as a distinct approach to social structure [Scott, 2011]. A social network is an explicit representation of the relationship between individual and groups in a community [Finin et al., 2005]. Relations in a network can be labeled with information about the strength of a relation between two actors but network analysts are also strongly interested in combining concepts and methods of social networks and semantic web to retrieve information about the communicative content exchanged by the related actors [Mika, 2007]. Semantic webs are used to represent semantic relations between concepts. Such vision of a *Semantic Social Network* depends on the availability of automated tools to extract and formalize information from unstructured documents [Cucchiarelli et al., 2012]. The human interaction would be restricted to post-editing. Over the past generations the public and academic interest in social networks grew rapidly [Knoke and Yang, 2008].

Graph theory originated in the mathematical investigations undertaken by Euler and provides a method for studying networks of all kinds [Scott, 2011]. The two main elements required for a network are nodes and edges between the nodes. In a social network nodes represent actors who may be individual natural persons or collectives such as informal groups and formal organizations [Knoke and Yang, 2008]. The edges represent a connection between a pair of nodes. They can also represent the flow of influence or resources in a social network and they can be assigned a value to represent the strength of their relation [Scott, 2011]. A node can have multiple relations to different nodes. In special cases the relations can be directed in which case the graph also can be displayed as matrices. Distances between nodes can be visualized using a dendrogram. In traditional social network analysis relationships are modeled as a function of quantifiable social interactions [Wasserman, 1994]. The desired network analysis for this work is a systematic network analysis based on indicators.

### 3.6 Software Evaluation

Evaluating software is a common process to identify a software suitable for a specific need or to guarantee a high quality software by evaluating the software during the development to find quality issues. In this case it can be compared to quality assurance or software assessment. In this work the software evaluation is performed from a user perspective to evaluate suitable tools for a specific need. The process of software evaluation is not standardized and can be performed in different ways. Evaluation schemes help to evaluate a software based on given criteria. Nonetheless software evaluation is a tricky balance between hard objectivity and the very subjective individual user experience. The user experience is usually a very subjective part since it strongly depends on the background knowledge of the user who is evaluating the software.

The Software Sustainable Institute (SSI)<sup>8</sup> has two guidelines for the evaluation process. Following these guidelines leads to a objective evaluation since the criteria are split into multiple sub criteria that can be evaluated using given values. Nonetheless the evaluation always stays subjective unless it can be completely evaluated using measurable values. It is common to use methods like the delphi methods to remove the subjectivity of an evaluation by combining multiple results of different expert evaluators. The two guidelines provided by the SSI are:

- *Criteria based Software Evaluation* and
- *Tutorial based Software Evaluation*

The criteria based evaluation uses a given scheme that can be applied for general software and the tutorial based evaluation focuses on the evaluation of a developers perspective. In this work the guideline for the criteria based evaluation is derived to create an evaluation scheme specified for text processing tools. The specific evaluation framework that is used in this work is the *Model-Dependent Software Evaluation* which is described in the next section in more detail.

### 3.7 Model-Dependent Software Evaluation

Model-dependent software evaluation is an approach introduced by Ebert and Dumslaff in 1993. It can be described as a software evaluation with the focus on specific aspects of the software which is evaluated. The base line for the model-dependent software evaluation is a model of the reality wherefore the software will be used [Winter et al., 1993]. Usually a modern software offers a lot of functionality which is not necessarily used by every user since software is not always developed for a specific task. The model-dependent software evaluation approach requires a model of a specific process to develop the criteria for the evaluation. Additionally some model-independent criteria known from the casual software evaluation are

---

<sup>8</sup><http://www.software.ac.uk>

used. These criteria and framework conditions are not model-dependent. For example criteria like the user interface, the supported platforms or the available documentation are not model-dependent.

The first step of the model-dependent software evaluation is the creation of the model. The model is developed by people with knowledge of the desired process and with the help of literature. The model is a representation of the process and is divided into the single important parts that will retrieve the focus during the evaluation. The model serves as base line for the development of the criteria for the model-dependent evaluation, which can be divided into qualitative and quantitative criteria. The criteria for the model-independent software evaluation parts are taken from literature since they are parts of a casual software evaluation. The model-independent part additionally includes framework conditions like the supported operating systems.

A software offers a specific performance spectrum, a user interface and framework conditions that have to be evaluated based on the criteria developed out of the model and the literature. Each part of the software is evaluated based on the criteria which is based on the model (Figure 8). This results in a software evaluation with the focus on a specific process.

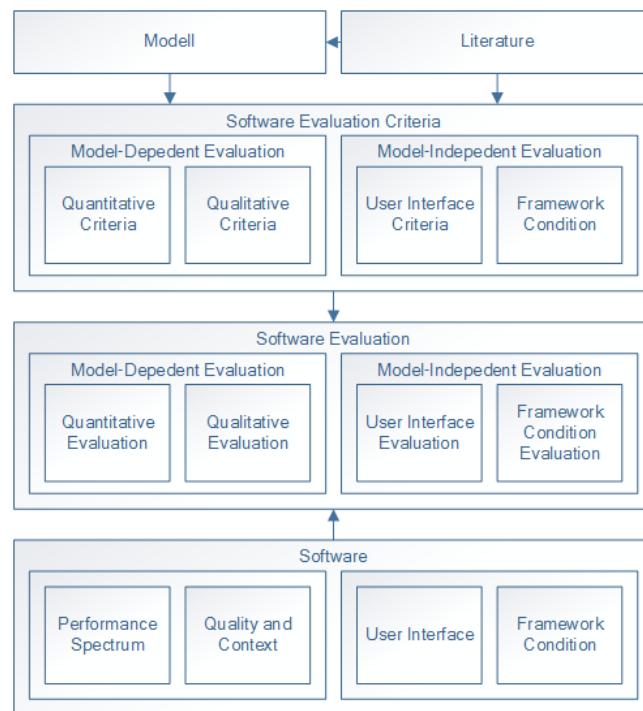


Figure 8: Process Model for Software Evaluation according to Ebert and Dumslaff

## 4 Text Processing Tools

The available amount of tools that can be used to analyze big data is huge. They range from specialized tools for specific areas to general tools with a huge amount of methods for different tasks. In this work a selection of this tools are evaluated to gain information of the suitability for extortion related data. In this section the selection process is described followed by a brief introduction to the selected tools.

### 4.1 Selection Process

For the evaluation six tools are selected that offer text processing methods. The selection of the tools is based on different factors. The first factor is the degree of specialization. This factor is the most important since it likely has the biggest impact on the results of the process. This is based on the fact that a tool which is specialized on text processing offers more in depth methods, whereas a general tool might offer more options for a common data analysis but less in depth methods. Another factor in the selection process is the availability and the licensing of the tools summarized as accessibility. Most tools are open source and free available, whereas some are commercial tools with special licensing for research purposes. The open source tool usually offer a huge amount of methods and can be extended by adding own snippets developed in a common programming language. To guarantee a good overview for each combination a tool is selected and evaluated (Table 1).

	<b>General</b>	<b>Specialized</b>
<b>Open Source</b>	WEKA RapidMiner Knime	AutoMap GATE Developer
<b>Commercial</b>	RapidMiner IBM SPSS Modeler	AutoMap

Table 1: Tool Selection Criteria

### 4.2 The Tools

The six tools selected for further analysis are WEKA, RapidMiner, Knime, IBM SPSS Modeler, AutoMap and GATE Developer. WEKA is a strong open source data mining tool developed by the University of Waikato in New Zealand (Table 2). Since WEKA is open source, it offers a lot of methods and functions to analyze

big amounts of data and it can be extended by writing classifiers in Java. RapidMiner and Knime are both open source data mining tools which offer text mining/processing extensions (Table 2 & 3). Both tools can be extended by self written nodes in Java. RapidMiner is available in different licensing options. For the analysis the free starter version is used since it offers the required functionality. The other options are only required for bigger data analysis and interoperability with other analysis tools. WEKA, RapidMiner and Knime are three tools that represent the General Open Source category. IBM SPSS Modeler is the only fully commercial tool selected since IBM offers licenses for educational purposes (Table 3). It is a big tool for data mining and offers many nodes for different data mining tasks. The text analysis extension allows the IBM SPSS Modeler to perform text processing methods.

The two tools that are specialized on natural language processing are AutoMap and GATE Developer (Table 4). AutoMap is a tool developed by the Carnegie Mellon University that can be used to reveal structure of social and organizational systems from texts. It works together with ORA, which is also developed by the Carnegie Mellon University, to visualize the structure. AutoMap can be freely used for research purposes and a commercial licensing of the software is possible. The last tool is GATE Developer, an open source tool specialized on the analysis of the human language.

	<b>WEKA</b>	<b>RapidMiner</b>
<b>Publisher</b>	University of Waikato	RapidMiner GmbH
<b>Country</b>	New Zealand	Germany
<b>Model</b>	Open Source	Open Source & Commercial
<b>Specialization</b>	General	General
<b>Version</b>	3.6.11	6

Table 2: Selected Tools Overview

	<b>KNIME</b>	<b>IBM SPSS Modeler</b>
<b>Publisher</b>	KNIME.com AG	IBM
<b>Country</b>	Switzerland	USA
<b>Model</b>	Open Source	Commercial
<b>Specialization</b>	General	General
<b>Version</b>	2.10	16

Table 3: Selected Tools Overview 2 (continued)

	<b>AutoMap</b>	<b>GATE Developer</b>
<b>Publisher</b>	Carnegie Mellon University	University of Sheffield
<b>Country</b>	USA	United Kingdom
<b>Model</b>	Open Source & Commercial	Open Source
<b>Specialization</b>	Specialized	Specialized
<b>Version</b>	3.0.10.36	8

Table 4: Selected Tools Overview 3 (continued)



## 5 Analysis

The model-dependent software evaluation approach depicted in 3.7 is divided into two parts. The first one is a general part and the second one is the specific part that is dependent on a model. In this case the model is the text processing process of court documents into a visualization of the structure of the persons involved in the case. In this chapter a framework for the evaluation scheme is developed and the process used for the model-dependent part is described in detail and the origin of this process is outlined.

### 5.1 The Evaluation Scheme Framework

The evaluation scheme is developed and described in the next chapter. For the development of the evaluation scheme a framework is used based on the *Model-Dependent Software Evaluation* framework introduced in chapter 3.7. The framework defines how the criteria of the evaluation scheme are selected. (Figure 9).

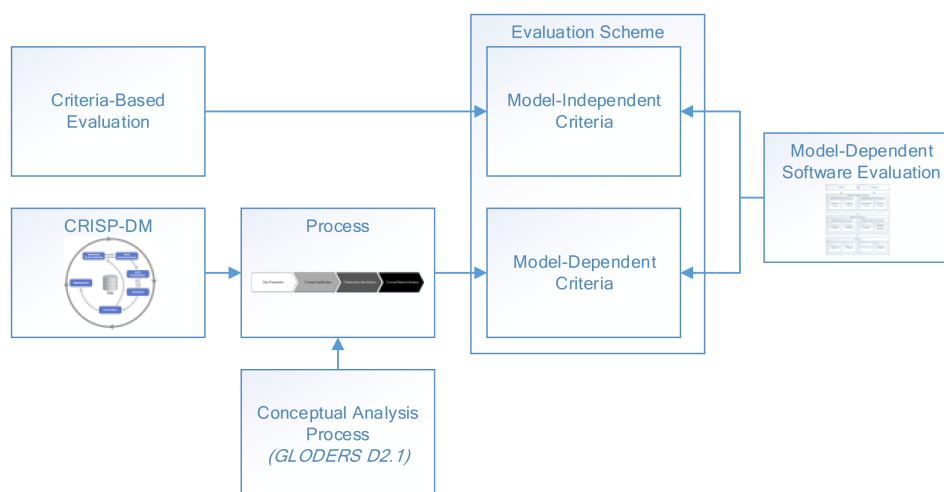


Figure 9: Evaluation Scheme Framework

The evaluation scheme is divided into a model-independent and a model-dependent part. The model-independent criteria are derived from the *Criteria-Based Evaluation* guideline of the *Software Sustainability Institute*<sup>9</sup>. The model-dependent criteria are based on a specific model. The model is the text processing process which is derived from the *Conceptual Analysis Process* developed in Deliverable 2.1<sup>10</sup> of the GLODERS project and the CRISP-DM reference model introduced in chapter 3.4 and described in chapter 5.2. The evaluation scheme criteria are defined and described in the next chapter.

<sup>9</sup><http://software.ac.uk/>

<sup>10</sup>[http://www.gloders.eu/images/Deliverables/GLODERS\\_D2-1.pdf](http://www.gloders.eu/images/Deliverables/GLODERS_D2-1.pdf)

## 5.2 The Process

To get a useful visual representation out of text files, these text files have to be processed. The aim is to generate a visualization of the social structure. Previous work done inside the GLODERS project introduced the *Conceptual Analysis Process* [GLODERS, 2013]. The *Conceptual Analysis Process* aims at extracting concepts and their relations from text. The Process is based on manual and automatic steps which are not strictly linear. The focus of this work is to perform quantitative analysis on unstructured text data. Therefore the *Conceptual Analysis Process* serves as baseline for the process developed in this work. The process introduced in Deliverable 2.1 of the GLODERS project consists of four main phases. *Concept Ontology Development*, *Concept Identification*, *Relationship Identification* and *Concept Network Analysis*. The results of each phase are used to improve the methods to achieve better results in a second run. The targeted process should be able to run nearly completely automatic. Therefore the process will contain the main phases defined in the GLODERS project with an additional data preparation phase in the beginning to prepare the input data. The CRISP-DM reference model defines a data mining process and the phases of the reference model explained in chapter 3.4 are used to verify the process derived from the *Conceptual Analysis Process* of the GLODERS project. The process developed is divided into four sub processes (Figure 10). Each subprocess has multiple methods that are used to transform the data and prepare it for the next subprocess. The methods inside the subprocesses can and in some cases have to be combined to achieve a result that can be used in the next subprocess. The first step is the *Data Preparation*, where the input data is transformed and prepared for further processing. The second step is the *Concept Identification*, where the concepts are defined and analyzed. In this step the data is enriched with information that is required to identify named entities. The third step is the *Relationship Identification*. In this step the relationships between the identified concepts are evaluated. The last step is the *Concept Network Identification* where the relationships discovered in the previous steps are visualized and modified.



Figure 10: The Analysis Process

The process derived from the Conceptual Analysis Process replaces the *Concept Ontology Development* phase with the *Data Preparation* phase since the aim is to analyze the data automatically without any manual work. It is the most important of the four subprocesses. It can't be carried out blindly [Pyle, 1999, p. 87] since all the following steps build on the outcome of the preparation steps. The data preparation phase offers the most methods of all phases of the process. All methods inside this

phase can be combined to create the best outcome that can be used in the following phase of the process. The three other phases of the process have less methods which are more specified then the general preparation methods of the first phase (Figure 11).

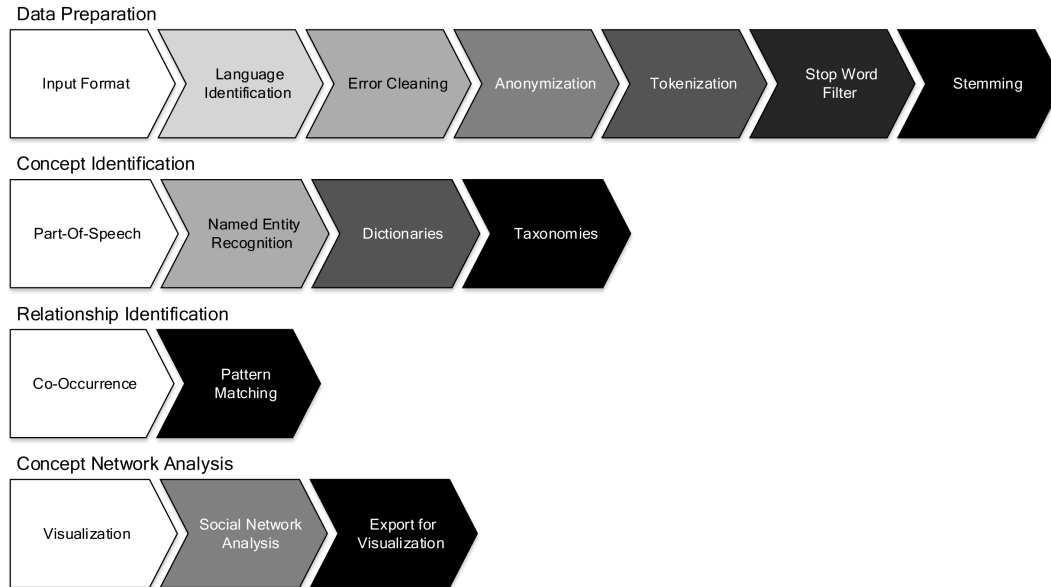


Figure 11: Subprocesses of the Analysis Process

### 5.2.1 Data Preparation

The most time consuming phase is the first phase of the process. In this phase multiple methods are used to remove unnecessary parts of the text and to enrich the text with additional information that helps in the following phase to retrieve better results.

The court documents which serve as input data for this evaluation are scanned copies in the .pdf-format. Before any tool is able to process these documents they need to be transformed into text files. This is done using an open source *Optical Character Recognition (OCR)* tool. Afterwards some additional manual preprocessing might be required to enhance the quality of the text files. The amount of manual preprocessing always depends on the quality of the raw input data. This part is not evaluated and not part of the process itself.

Since the input data for this work is unstructured text data from court documents some important data preparation methods are required and mandatory. Methods that are used in the first phase are *Input Format, Language Identification, Error Cleaning, Anonymization, Tokenization Stop Word Filter, and Stemming.*

## **Input Format**

After using optical character recognition the unstructured text data is available as flat file. Different tools support different data format as input. Most tools accept flat files, but some tools require files to be in a specified structure. The first step of the data preparation phase is to transform the data into the required input format. This step has to be performed manually if the tool does not support unstructured text files.

## **Language Identification**

The case used in this evaluation is completely in German, whereby there is no direct need for this phase since the result is already known. Nonetheless it is required for texts where the language of the input data differs since some text processing methods require language specific models.

## **Error Cleaning**

Error Cleaning is required in this process to enhance the quality of the desired output. Therefore it is important that the data which is used in the process is known. Errors can be results of the optical character recognition transformation. The errors can be handled by given dictionaries or by creating dictionaries manually for recognized errors in previous analysis phases. Typos of names can result in two different concepts discovered in the concept identification phase even if it should be a single concept. When errors like this are discovered when the result is reviewed the dictionary for error cleaning can be updated and the result of the next iteration of the process will enhance the quality of the result.

## **Anonymization**

Court documents and other documents that are used to identify relationships between concepts usually contain names of persons, locations and organizations which are the desired result of the analysis. If the results are just used for internal research anonymization is not necessary. If the data, results, or just part of the results should be published in any kind of way the names of persons, locations or organization might give insights on the input case for persons who are not allowed to get these information. In this evaluation the input data are court documents containing real names of persons and locations that need to be anonymized. In the case description in chapter 2 the names already have been anonymized. The same dictionary can be used for anonymization to ensure that the manual analysis delivers similar results to the results of the different tools.

## Tokenization

Unstructured data can be regarded as one single variable or token. Tokenization helps to split the data into smaller parts which increases the speed of the analysis. Tokenization usually varies between splits into paragraphs, sentences or single words. Paragraphs can be detected by line breaks whereas sentences can simply be detected by punctuation.

## Stop Word Filter

A sentence usually consists of multiple single words combined containing a specific message. Some words can be stripped from the sentence to reduce the amount of words without changing the meaning. Stop Word Filter are used to strip these words from the text to enhance the quality and to optimize the data for the concept identification and relationship identification phase. Tools might have language specific filter lists whereas other tools might not have filter lists at all. Dictionaries can be created to write these lists manually.

## Stemming

Words exist in many variations and usually have similar meanings. Every word consist of a root from which these variants derived. Stemming is used to reduce these variations to the root. This method helps to optimize concepts since possible duplicates are prevented. Stemming is language dependent and requires specific models for each language to deliver good results.

### 5.2.2 Concept Identification

The second phase of the process is applied right after the data is preprocessed. The purpose of this phase is to identify the concepts laying behind the collection of strings in the raw data. These concepts are required for the third phase where the relationships between concepts are analyzed. In the Concept Identification phase there are multiple methods and approaches that can be used and combined. The approaches differ in the amount of the required manual work. The methods are language dependent since they require models trained on each language. Methods for in this phase are *Part-Of-Speech Tagging* and *Named Entity Recognition*. Additionally *Dictionaries* and *Taxonomies* can be used to enhance the concept identification.

#### Part-Of-Speech Tagging

The first method used in the *Concept Identification* phase is Part-Of-Speech tagging. It is used to detect named entities in the data which are the desired concepts. Part-Of-Speech tagging also tags each single token based on a specific tagset. These tagsets differ for different languages. The most common tagsets are the *Stuttgart-Tübingen Tagset (STTS)* for the German language and the *PENN Treebank Tagset* for English

texts. The Part-Of-Speech tags can also be used in relationship identification phase to help name a relationship. A verb for example can define the relationship between two concepts.

### **Named Entity Recognition**

The most common natural language processing method to identify a concept is the Named Entity Recognition (NER). It is a specialized form of Part-Of-Speech tagging since it focuses only on named entities. A named entity is a phrase that contains the name of a person, an organization or a location [Tjong Kim Sang and De Meulder, 2003]. Named entities are already identified by the Part-Of-Speech tagging, but the NER is more specific about the exact role in the sentence. Part-Of-Speech tagging defines a named entity as NE whereas Named-Entity-Recognition specifies them as Person, Location, Organization, Money, Date or Time.

*[Suspect A (PER)] threatens [Victim A (PER)] at [Place A (LOC)]*

In this example Named-Entity Recognition identified three named entities. Two persons and a location. Combined with previous data preparation methods a good starting point for a relationship identification is given by *Named Entity Recognition*.

### **Dictionaries**

The most simple method for Concept Identification are dictionaries. A dictionary contains the information that is used to identify a concept in the input data. Therefore the person who creates the dictionary needs to have background knowledge of the input data. Dictionaries require a lot of manual work and the size of a dictionary depends on the size of the input data. In many cases it is easier to use automatic methods like the *Named Entity Recognition* first to check how good the automatic methods perform without adding additional manual work. Missing entities can be then added with the help of dictionaries. The output data of the first phase can also be reviewed to generate a dictionary. A list of all discovered named entities during the Part-Of-Speech tagging can be used to create the dictionary without additional background knowledge of the input data.

### **Taxonomies**

Similar to dictionaries taxonomies can be helpful to clearly identify concepts. By using taxonomies the amount of duplicate concepts can be reduced. A common way is to use predefined taxonomy dictionaries or to manually create dictionaries containing taxonomies. For the manual creation the user requires background knowledge of the input data. Like previous methods the quality can be enhanced by updating the taxonomy dictionaries after duplicates have been recognized in previous runs.

When the taxonomy dictionaries are updated the relationship identification phase should deliver better results.

### 5.2.3 Relationship Identification

After concepts are identified in the data the identification of relationships between concepts are the aim of this phase. Two different methods are used to identify these relations. The first method is *Co-Occurrence* which is a automatic method. The second method is *Pattern Matching* which is a more manual approach.

#### Co-Occurrence

Co-Occurrence uses the assumption that a relationship exists between concepts if they occur inside a predefined range. If previous identified concepts occur inside a given range they are taken as relationship.

*[Suspect A (PER)] threatens [Victim A (PER)]*

In this example a relationship between Suspect A and Victim A would be identified. Co-Occurrence is not defining the type of the relationship.

#### Pattern Matching

Pattern Matching identifies relationship based on predefined rules. These rules are usually regular expressions and they have to be defined. A concept which is followed by a verb which is then again followed by a second concept would be an example for a rule. The verb in this case defines the relationship. Available rulesets can be used or they can be created manually. Part-Of-Speech tagging is required before this method to make the rules work. These rules also depend on the language of the input data since the syntax of a sentence can be different in different languages. Rules can be defined in different ways and using *Part-Of-Speech* tags is just one.

### 5.2.4 Concept Network Analysis

The last phase of the process is the phase where the identified relations are visualized. In this phase there are two methods that are used to fulfill this requirements. Additionally there is one important additional criteria which covers the possibility of data exportation. This step might be important if a tool is not supporting visualization itself. *Visualization* and *Social Network Analysis* are the methods that are reviewed for the tool itself.

#### Visualization

The first step of the *Concept Network Analysis* phase is to visualize the results of previews phases in any kind of way. The most simple way is a list of identified re-

lations. Better results are graphical outputs. A cluster map or a network connecting concepts are other more complex variants. The availability of visualization methods may vary between the tools.

### **Social Network Analysis**

Social Network Analysis is the best result for a visualization of the identified relations. In this case it is important that the results can be viewed and if it is possible to manually edit them. Manually work on the networks can be performed by users knowing the case to remove invalid relationships. The best case of the whole process is a network graph which contains all important relations that had been detected by the manual analysis.

### **Export for Visualization**

Tools that are not able to visualize the results of the *Relationship Identification* phase might be able to export the results to visualize them with a different software. Some tools might offer integrated interoperability with specialized tools for displaying relations whereas other tools might be able to export the data in a format that can be used with these tools as well.



## 6 Evaluation

After developing the process and the evaluation scheme framework the evaluation scheme itself is required to evaluate the different tools. The evaluation scheme is build using the evaluation scheme framework described in the previous chapter. The first part of the scheme is a model-independent part where the tools are evaluated based on criteria that are not dependent on the process. The second part is the model-dependent part that evaluates the tools based on criteria that is influenced by a specified model or process as described in chapter 3.7. After defining the evaluation scheme and describing the criteria the tools are evaluated based on the scheme. Subsequent to the evaluation the result of each tool is compared to the networks manually generated and depicted in chapter 2. The full evaluation table can be found in the appendix.

### 6.1 Evaluation Scheme

The evaluation scheme is developed using the model-dependent software evaluation model introduced in chapter 3.7. The focus of the evaluation scheme is on the model-dependent part. Nonetheless the model-independent part is considered as well. To enhance the clarity of the evaluation scheme the single criteria will be divided in main criteria with sub criteria. In most cases of the model-dependent criteria it will be a binary evaluation whether a tool supports a specific functionality or not. In some cases on the other hand it will be an evaluation about availability of specific attributes like models for different languages. The aim of this evaluation is to give an overview about the tools and their functionality in focus of the use for text processing extortion related data. The evaluation is aimed to support a future user to decide which tool fits the best for the specific purpose of text processing on extortion data. Therefore the evaluation is designed for a user perspective. The Software Sustainability Institute<sup>11</sup> differentiates between a user evaluation, a user-developer evaluation, a developer evaluation and member evaluations. The only one suitable is the user evaluation, since this evaluation focuses on the usability of the software as-is without the need of writing code [Jackson et al., 2011b].

#### 6.1.1 Model-Independent Criteria

The first part of the model-dependent software evaluation framework is the model-independent part. In this part the criteria are developed based on suggestions by the Software Sustainability Institute who derived the criteria from *ISO/IEC 9126-1 Software engineering - Product quality*<sup>12</sup> [Jackson et al., 2011a]. Three main criteria are used for the evaluation of the model.independent part. The criteria are *General*, *Usability* and *Sustainability*. The sub criteria are show in the following list and described in each subsection.

---

<sup>11</sup><http://www.software.ac.uk>

<sup>12</sup>[http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=22749](http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749)

- General
  - Publisher / Developer
  - Country
  - Extensions
  - Specialization
- Usability
  - User Interface
  - Documentation
  - Supported Operation System
- Sustainability
  - Support
  - License
  - Accessibility
  - Interoperability
  - Version

#### 6.1.1.1 General

The first criteria of the model-dependent part are summarized under the main criteria *General*. These criteria help to gain a brief overview about a tool. The first criteria shows the publisher or the developer of the tool followed by the country where the tool is developed. The third criteria is an important criteria for the evaluation since it lists extensions that are necessary to enable text processing capabilities for the tools that are more general and not specified for natural language processing processes. The last general criteria is the specialization of a tool. The tools are selected by two different aspects as described in chapter 4.1. This criteria captures the specialization aspect of the selection process. The second aspect is covered by the *Accessibility* criteria under the *Sustainability* main criteria.

#### 6.1.1.2 Usability

Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and specified context of use [ISO, 2008]. It is an important criteria in every user perspective evaluation since it has a huge impact on the user experience. For this evaluation scheme the *Usability* main criteria is split into three sub criteria. The first one is the *User Interface* which sums up the general user interface of a tool. Tools have different user interfaces that will be evaluated on a range from good to bad with the fact in mind that it is a user perspective

evaluation. Therefore aspects like error-handling, learnability and memorability are considered. The second criteria is the *Documentation* of the tool. The criteria is used to measure the quality of the documentation. The documentation of a tool may vary in detail and in completeness. The documentation can be available in different languages additionally to English which sometimes can be helpful even if the English documentation usually is more detailed. The last usability criteria is the *Supported Operation System* criteria. It is used to display on which operating system is supported by the tool.

### 6.1.1.3 Sustainability

The sustainability main criteria is the last criteria of the model-independent part of the evaluation scheme. The criteria summarizes criteria that show how sustainable a tool is. The first criteria is the *Support* for the tool. Some developer offer support for a tool, but for some tools the support is only available for commercial packages of the tool. The support itself can vary between boards supervised by the developers to phone or email support. The advantage of boards is that topics created by other users can already help to find a solution and that developers can be informed about errors in specific versions. These information then help to improve upcoming releases. The *License* of a tool is the second criteria. It helps to achieve an overview of the tool and to determine how the tool can be used or how it can be further developed. The third criteria is *Accessibility*, the second aspect of the tool selection process described in chapter 4.1. It is differed between open source and commercial tools. Open Source tools usually offer a greater user base which leads to faster development, whereas commercial software usually are better tested before a release. The *Interoperability* criteria is used to list other tools that can be combined with the tool evaluated to offer more possibilities for the user. The visualization of data is usually a part that is performed by additional tools. The last criteria is the *Version* of the tool that is evaluated. The version of a tool sometimes indicate how often a tool is released or updated. Commercial tools tend to have bigger release cycles then open source tools.

### 6.1.2 Model-Dependent Criteria

The model-dependent criteria are based on the process developed in chapter 5.2. Each main phase of the process is a main criteria in the evaluation scheme consisting of at least two sub criteria. These sub criteria are desired functionalities or methods of a text processing tools to support the text processing process. They are evaluated based on their availability and their scope. Some methods use models for specific languages to optimize the result. Not every tool offers models for every language. The availability of languages for different methods are part of this evaluation. For the evaluation court documents from a German case are used. Nonetheless the performance on English texts in general are evaluated as well. The main criteria and each sub criteria are listed below and then described in each sub section.

- Data Preparation
  - Input Format
  - Language Identification
  - Error Cleaning
  - Anonymization
  - Tokenization
  - Stop Word Filter
  - Stemming
- Concept Identification
  - Part-Of-Speech Tagging
  - Dictionaries
  - Taxonomies
  - Named Entity Recognition
- Relationship Identification
  - Co-Occurrence
  - Pattern Matching
- Concept Network Analysis
  - Visualization
  - Social Network Analysis
  - Export for Visualization

#### 6.1.2.1 Data Preparation

Data preparation is the first phase of the process and therefore the first main criterion of the model-dependent part of the evaluation scheme. The sub criteria of the first phase include functionalities and methods to prepare the data for the following steps. This phase contains the most sub criteria since there are many methods that are required to clear the data and enrich it with information for the following phases. The first sub criterion is the *Input Format*. Each tool supports different ways to import unstructured data. First of all the requirement is that the documents are available in a machine readable version. In the case used for the evaluation the data was allocated as scanned copies. These images had to be transformed into machine readable documents using an OCR tool. This step is not part of the evaluation since it always depends on the given input data. For some of the methods in further steps language dependent models are required to deliver valuable results. Therefore the

second criterion of the first phase is the *Language Identification* to determine the language of the input data. In some cases, especially when the input data is the result of transformation via OCR tools data cleaning steps are required. The third criterion is the *Error Cleaning*. Common OCR errors are handled in this step to clean the input data. In the input data for this evaluation some common errors in the spelling of suspect names occurred which lead to wrong relationships in further steps if they are not handled in the preparation phase. Since extortion related input data sometimes contains real names an anonymization step is required to secure all involved persons. The fourth criterion *Anonymization* is used to evaluate the possibilities to anonymize data inside a tool to ensure this security. This step is only required if the results are published in any kind of way. For this work it is important and the data is anonymized to match the description of the case in chapter 2. The last three criteria are methods of text processing tools to transform the input data into optimized structures for further processing. The availability of *Tokenization* allows the user to transform a document in single sentences or paragraphs to reduce part that is evaluated. *Stop Word Filter* help to delete unnecessary words from the document to speed up methods depending on the amount of words. The last criterion is *Stemming*. Stemming helps to reduce variants of words by cutting them down to the stem of a word. Stemming can be achieved by different methods and is language dependent. Due to this the available stemming methods and the available languages are evaluated for each tool.

### 6.1.2.2 Concept Identification

The second main criterion based on the process is *Concept Identification*. It contains sub criteria that are used to create concepts that can be used in the next phase to identify relationships between these concepts. In this work a concept can be a name of an involved person or a location. These parts of a text can be determined with different methods. The first criterion in the second phase is *Part-Of-Speech Tagging*. Part-Of-Speech tagging is dependent on the tagset which is used to tag the input data. These tagset are language dependent since languages differ in their structure. Therefore the criterion is based on the available tagsets and the supported languages. One structure that can be detected by Part-Of-Speech tagging are named entities. The second criterion is *Named Entity Recognition*. Named entity recognition works similar to Part-Of-Speech tagging but it is optimized to differ between named entities. Named entity recognition identifies persons, locations and organization based on a model. Like Part-Of-Speech tagging these models are build up for specific languages whereby the availability and the language support are evaluated with this criterion. Dictionaries can be used to support the quality of the named entity recognition. Therefore the third criterion are the availability of *Dictionaries*, or the support to create dictionaries. Similar to dictionaries *Taxonomies* can be used to enhance the quality of the concept identification. The availability of taxonomy support is the last criterion of the concept identification phase.

### 6.1.2.3 Relationship Identification

*Relationship Identification* is the third main criterion of the evaluation of the model-dependent part. It consists of two sub criteria which are based on text processing methods that can be used to identify relationships between concepts. The first criteria is *Co-Occurrence*. Co-Occurrence of terms can be measured in different ways. The main focus is the availability of this method since it is a key component of the process. Co-Occurrence can be completely automated and requires no user input, whereas the second criterion *Pattern Matching* defines relationship based on given pattern. These pattern can be automated by specified regular expressions but it is a manual work since there are no regular expression dictionaries for extortion related data. For both criteria the focus of the evaluation is the availability of at least one method. If they are supported, the quality of the method is evaluated for each tool.

### 6.1.2.4 Concept Network Analysis

The last main criterion is *Concept Network Analysis*. It is used to evaluate the last phase of the process in which the identified relationships are visualized and modified. The first sub criterion is *Visualization*. It is used to show if the tool is able to visualize the data of the previous step in any kind of usable way. The best way of visualization is a network of concepts with labeled relations. *Social Network Analysis* is the second criterion and it the best possible way of visualization. The criterion show if the tool is able to produce a network itself. If tools are not able to visualize previous identified relations the last criterion *Export for Visualization* shows if the data can be exported in any kind of way to visualize it using a different tool. If so the format and the tool for visualization are named.

## 6.2 Model-Independent

All six tools are developed for different purposes. As described in the selection process for the tools (chapter 4.1) they range from big machine learning suites to specific programs for natural language processing. WEKA is the most general tool whereas GATE Developer and AutoMap are natural language processing tools. All tools except the IBM SPSS Modeler are at least available in a free version. To gain access to text processing features some tools required to download additional extensions/plugins. In the following section each tool is evaluated using the model-independent part of the evaluation scheme. The full evaluation table can be found in the appendix.

### 6.2.1 IBM SPSS Modeler

IBM SPSS Modeler is the only commercial tool which is evaluated in this work. It was developed by Integral Solutions Limited under the name Clementine in 1994. Since then it was continuously developed and enriched with more functionalities. Since 1994 the name changed twice. In 2009 it was renamed to PASW Modeler. At

this point it was version 13 of the original Clementine tool that was designed as a consulting tool [Shearer, 1994]. In 2009 it was acquired by IBM, an American multinational technology and consulting corporation, and it was renamed to IBM SPSS Modeler. Since then the tool was released three more times in newer version. The current version released end of 2013 is version 16. "It is an extensive predictive analytics platform that is designed to bring predictive intelligence to decisions made by individuals, groups, systems and the enterprise."<sup>13</sup> IBM SPSS Modeler is a general data mining tool that offers many data preprocessing steps. The tool is available in different editions as desktop and as server configurations. The *Professional* edition is the basic version and requires the *Text Analytics* extension to enable text processing functionalities. The *Premium* edition of the IBM SPSS Modeler already includes this extension. IBM also offers a version for educational purposes. This edition is similar to the *Premium* edition and it is used for this evaluation.

The IBM SPSS Modeler is a node-based data mining tool. It offers nodes for different data mining tasks that can be combined by connecting the nodes. Nodes required for text processing are added with the *Text Analytics* extension. A workflow created in the Modeler is called a stream. Errors are displayed to the user when they occur in an extra window. IBM offers a detailed documentation for the Modeler. The documentation<sup>14</sup> covers everything from the installation to the usage of the basic data mining methods. The extensions are documented as well in great detail. The *Text Analytics* extensions documentation offers over 200 pages of detailed information about functionalities included in the extension. The whole documentation is available in twelve different languages. The Modeler is available for Windows, Linux and UNIX systems as desktop or server version. For this evaluation the Modeler was installed on a Windows 8 device as desktop version.

Since the IBM SPSS Modeler is a commercial tool the support is different then on most open source tools. IBM offers contract based support for paying customers. There is no official support forum where users can search for help. The only way besides searching in the documentation is the direct support by IBM or third party fora. The Modeler is sold as proprietary Software which is common for commercial tools. In this case the source code is not available and customizing functionalities can only be done in the scope offered by the modeler itself.

## 6.2.2 WEKA

WEKA (Waikato Environment for Knowledge Analysis is a collection of machine learning algorithms for data mining tasks [Hall et al., ]. It was developed by the Machine Learning Group at the University of Waikato in New Zealand. The development started in 1993 primarily designed as a tool for analyzing data from agricultural domains [Holmes et al., 1994]. This version was programmed in the Tool Command Language. In 1997 the development of the current Java version started. Today

---

<sup>13</sup><http://www-01.ibm.com/software/analytics/spss/products/modeler/>

<sup>14</sup><http://www-01.ibm.com/support/docview.wss?uid=swg27038316>

WEKA is a powerful suite combining multiple data mining and machine learning methods using the Java programming language.

Since WEKA is completely programmed in Java it is easy portable to different operating systems. It runs under Windows, Linux and UNIX operating systems. WEKA can be used using a simple user interface or by using the console. The WEKA Explorer is the user interface version and it is split into six registers which can be compared to the workflow of a process. The first register is for the data preprocessing, the next is for classification and the last register is for the visualization. All methods can be used by self-written Java applications which allow the user to use machine learning method inside a Java application by adding the WEKA libraries. The WEKA Explorer is the user interface version which is a simple user interface restricted on the main functionalities. All methods as well as the installation on different operating systems is well documented for each available version of WEKA<sup>15</sup>. It is officially only documented in English. Additionally version 3, which is the current main version of WEKA, is covered in the Book "Data Mining: Practical Machine Learning Tools and Techniques" [Witten et al., 2011] which is written as companion book. The current third edition is only available in English, Chinese and Korean. The first edition is also translated into German. To get deeper understanding of the functionalities of the mighty workbench the website offers tutorials for general data mining tasks.

WEKA is a free software available under the GNU General Public License in the current stable version 3.6.11 or the developer version 3.7.11. In this evaluation the stable version 3.6.9 is evaluated. One disadvantage of WEKA is the requirement of a specified input format. It is not possible to use unstructured text as input data without transforming it into the required *Attribute Relation File Format (ARFF)*. The data needs to be available as flat file or inside a SQL database since WEKA supports database connectivity. WEKA offers no direct support. Users can search a mailing list archive for threads where other users had similar problems. Since WEKA has a great userbase the mailing list archive consists of many threads that may help a user. If a problem is not tackled in any thread or documented in the documentation or the book the user can add a new thread and wait for other users to help finding a solution.

### 6.2.3 RapidMiner

RapidMiner is a data mining software developed by the Technical University of Dortmund. It was developed under the name Yet Another Learning Environment (YALE) in 2001 by the Artificial Intelligence department. In 2006 the main developers founded the company Rapid-I and continued the development of the tool. In 2007 YALE was renamed to RapidMiner. The company changed the name from Rapid-I to RapidMiner and it has offices in Boston, London and Dortmund where it originates. RapidMiner is a mighty tool for data mining and machine learning. To

---

<sup>15</sup><http://www.cs.waikato.ac.nz/ml/weka/documentation.html>



enable the full capabilities for text processing it offers two extensions bringing important methods required for a text processing workflow. The first extension is the *Text Processing* extension. It brings many methods for data preparation and data enrichment. *Information Extraction* is the second extension which adds methods mainly for relationship identification. The current version 2.0 of the *Information Extraction* extension is bugged which results in important methods not working. The documentation and the examples made with this extensions itself show that they should perform the required steps of the *Relationship Identification* phase.

RapidMiner is a node-based tool with a user friendly user interface. Nodes can be connected to produce a workflow and warnings and errors are shown to the user to prevent nodes being combined that can not work together. In the option the user can active the expert mode which allows the user to change parameters that are usually not available. These parameters are not required but they can be used to optimize methods. These parameters should only be changed if the user knows what impact the change of the parameter has on the result. An icon inside the node represents the status of the node. A green dot show a successful execution and a green arrow indicates that the node is still executed. A yellow dot indicates that the node is not run yet and a red dot indicates that the connection is invalid, or that the node failed to execute. The user interface of RapidMiner is split into two main views. The first one shows the process and the second view contains the results. Previous results of the process can be viewed and inspected in the result view. Like all other tools RapidMiner offers a good documentation<sup>16</sup> where the installation process and the tool with each node are documented in English. Additionally tutorials and example workflows are available to allow a quick entry into the tool. It is based on Java and it runs on Windows, Linux and UNIX operating systems. Besides the client version RapidMiner offers a server version to make use of dedicated computing power for powerful analysis on big data.

RapidMiner is available with different licensing methods. The core and earlier versions of the software are available under the Affero General Public License whereas other editions are sold as proprietary software in different version with different additional services. One important difference is that the free versions are restricted to use a maximum of 1GB memory for computation which restricts the user to perform only analysis on small data. The free version is used to introduce new users to the tool. Some CPU-intensive algorithms can already run into the 1GB memory limit on small data sets. The current stable version of RapidMiner is version 6.1, released in October 2014. For the evaluation the free version 5.3 is used since it is available as open source. On the RapidMiner website a forum<sup>17</sup> is hosted which is used to offer support by users for users. Besides the forum the RapidMiner GmbH itself offers support for the commercial versions of the software.

---

<sup>16</sup><https://rapidminer.com/learning/>

<sup>17</sup><http://forum.rapid-i.com/>

## 6.2.4 KNIME

KNIME is the abbreviation for Konstanz Information Miner which is based on its development origin. In 2004 a team at University of Konstanz in Germany started the development of KNIME and in 2006 the first version was released. Today KNIME is managed by the KNIME.com AG based in Zurich, Switzerland. KNIME is a tool that supports data mining and machine learning functionalities. To enable the tool to support text processing it requires the *Text Processing* extension which can be added with the *Extension Manager*. The Extension Manager offers many extensions that add functionalities for different tasks since these extensions can be developed and provided by everyone for everyone. The *Text Processing* extension is part of the KNIME Community Contributions which are provided and maintained by various community developers. KNIME also offers commercial extension that can be acquired to increase the productivity. These extensions are not required for the purpose of this evaluation.

KNIME offers a node based user interface similar to RapidMiner and IBM SPSS Modeler. The nodes can be connected to generate a workflow of multiple methods. Nodes can only be combined when the output of the first node is compatible with the input of the second node to prevent errors before they occur. This helps the user to identify errors while preparing a workflow. Nonetheless errors can occur even when the input and output are compatible since only the meta information is checked. If the output of the first node contains invalid data the following node might fail. In this case the user get notified and the workflow is stopped. An error message is shown which helps the user to identify the error. When a node is run and succeeded without error a small traffic light beneath it shows a green light. Failed nodes are marked with a red light and a node which is not yet run is marked with a yellow light. An exclamation mark shows warnings. The output data of each node can be inspected to verify the results manually or to search for possible invalid data leading to an error in the following node. Every node offers parameters that can and in some cases have to be adjusted to perform the desired method. KNIME offers a detailed documentation<sup>18</sup> for every node. This documentation, which is available only in English, can be viewed online or in the tool itself by clicking on a node. The documentation shows a description of the node itself and a description about parameters and the input and output ports. Additionally KNIME offers a starting guide which helps new users to set up KNIME and to get used to the interface. Since it is based on Java it is available for Windows, Linux and UNIX operating systems. It is build on the eclipse<sup>19</sup> framework which is a common framework for Java development.

The documentation of the nodes is detailed but in some cases users might require additional help. Therefore KNIME offers a forum on their website where users search for threads with a similar problem. New topics can be created and other

---

<sup>18</sup><http://tech.knime.org/documentation>

<sup>19</sup><https://eclipse.org/>

users or members of the KNIME development team can help to find a solution. In this work the forum was helpful since a member of the KNIME development team found a missing German model for a required method. KNIME is an open source software available under the GNU General Public License which is one reason for the availability of the extensions and the continuous development. The current version is 2.10, which stable release was in July 2014. In this evaluation version 2.9.4 is used since not all extensions already support in version 2.10.

### 6.2.5 AutoMap

AutoMap is a text mining tool developed by the Center for Computational Analysis of Social and Organizational Systems (CASOS) at the Carnegie Mellon University in Pittsburgh, Pennsylvania, USA. AutoMap offers Network Text Analysis methods to extract information from one or more unstructured texts as input data. Since AutoMap is specialized on *Natural Language Processing* it does not require additional plugins or extensions.

The user interface of AutoMap is different from the previous node based user-interfaces of KNIME or RapidMiner. The main window is used to visualize the text and the methods are applied by selecting them in the menu bar at the top. It is not possible to create a workflow. The user has to perform every method by hand in sequence. AutoMap offers a version of each text after each method is applied which can be helpful to spot the changes. The CASOS offers a User Guide<sup>20</sup> which is regularly updated. AutoMap is the only tool in this evaluation which is only available for Windows operating systems.

AutoMap was developed for research purposes. Due to this it is free available for research purposes only. Since the commercial interest in the product grew it can be acquired for commercial use as well. The commercial version offers contract based support whereas the free version has no official support. AutoMap is able to generate files as output that can be used by ORA, another tool developed by CASOS of the Carnegie Mellon University, to visualize and modify relationships between concepts. The current version available which is also used for this evaluation is version 3.0.10.36.

### 6.2.6 GATE Developer

GATE Developer is an integrated development environment for language processing components [Cunningham et al., 2013]. It is part of the GATE software family which includes GATE Developer, GATE Teamware, GATE Mimir and GATE Embedded. GATE is the abbreviation for *General Architecture for Text Engineering*. The development of GATE started in 1995 as part of an project on *Large Scale Information Extraction* lead by the Engineering and Physical Science Research Council (EPSRC)<sup>21</sup>. It is now continuously developed and maintained by a research group of the

---

<sup>20</sup><http://www.casos.cs.cmu.edu/publications/papers/CMU-ISR-13-105.pdf>

<sup>21</sup><http://www.epsrc.ac.uk/>

University of Sheffield in the United Kingdom. Like AutoMap it is a tool specialized on text processing.

The user interface of the GATE developer is simple and can be compared to the user interface of AutoMap. In the main window the text is displayed. Besides the user interface all methods can also be called and used by self-written Java applications. GATE Developer offers *Processing Resources* which are the methods that are applied in form of a workflow. After running the workflow the main windows allows to highlight detected parts in the text. Errors are shown as Java exceptions which are not always clear. For new users GATE is offering tutorials and a good documentation to make the entry simple. The structure of GATE developer is not that easy to understand at a first glance. The tutorials and the documentation provided on the webpage<sup>22</sup> help to install and to use GATE Developer. The documentation is only available in English like all other evaluated tools except IBM SPSS Modeler which offers 12 different languages. The user guide is available for the last versions and it is always updated with new releases. GATE Developer is available for Windows, Linux and UNIX operating systems. The early versions had been only available for Linux, but since version 2.2, which was released in 2003, it is available for Windows and UNIX operating systems as well.

The user guide offers a good documentation of the functionalities of the GATE Developer. Users who run into problems can search in the FAQ, the documentation or in the mailing list archive. Besides these options the University of Sheffield also offers contracts to provide support or customization. Additionally professional services can be acquired by key partners of GATE. GATE Developer is currently available in version 8.1 as free software under the GNU Lesser General Public License 3.0. In the evaluation the stable version 8.0 is used. The development team released newer versions annually in the past.

### 6.3 Model-Dependent

Since the input documents for this work are written in German the evaluation will take this into account. Nonetheless different available languages are mentioned to give an overview of the tools. Each phase of the process is described for each tool. Example workflows are described and results of the workflows are visualized.

#### 6.3.1 Data Preparation

The most important phase in the process is the data preparation phase since it contains the steps which prepare the data for further steps. Each tool offers more or less the same amount of methods for data preparation but they differ in the way the user can combine them. Some methods are only capable to perform good on text for the English language, but most methods offer models for multiple common languages like English, German, Spanish or Italian. Some tools offer more options for the user

---

<sup>22</sup><https://gate.ac.uk/releases/gate-8.0-build4825-ALL/doc/tao/split.html>

to have impact on the results by giving customizable parameters. Some tools offer nodes for each single data preparation method which allows the user to change the order of some steps, whereas some tools only offer one node which includes all methods and the user can not change the order.

### 6.3.1.1 IBM SPSS Modeler

The *Text Analytics* extensions adds the required functionality to IBM SPSS Modeler to analyze unstructured data. The whole process covering all four phases is realised with a two node stream (Figure 12). The first node is the *Var. File* node which loads a document into the modeler. IBM SPSS Modeler also offers other nodes to import data from different data sources. It can import data from other IBM tools like IBM Cognos, Excel or from external databases. The second node is the *Text Mining* node which supports the full text processing process. When the *Text Mining* node is inserted for the first time all required packages are automatically installed.



Figure 12: IBM SPSS Modeler Stream

The *Text Mining* node has built in data preparation methods that can be adjusted with some parameters. Other parameters are hidden in the *Resource Editor*. This editor contains information about *Language Identification*, *Language Handling* and *Non-linguistic Entities* for seven languages including English, German, French and Italian. If the *Text Mining* node is used with default values it will assume that the text is in English. The node offers the *Text Language* parameter in the *Model* tab. When the parameter is set to 'All' the node will identify the language based on a script placed in the *Resource Editor*. The language is determined based on the first 10000 characters and the languages are checked based on an order that can be changed in the editor. This helps to speed up the process for large data sets. A fallback language can be set for the case that a language is not identified.

*Fuzzy Grouping* is used to remove common spelling errors in the text. Misspelled names or abbreviations of names are not fixed by this method. Names can be grouped by selecting a parameter in the *Expert* tab. This reduces the amount of identified persons by *Named Entity Recognition* since it removes duplicates. Exceptions for the *Fuzzy Grouping* algorithm can be added to a list in the *Resource Editor*.

IBM SPSS Modeler offers a *Anonymization* node which can be used to anonymize the whole document. This node is not suited for the process since it does not allow to anonymize single tokens. The *Text Mining* node also includes no option to use a dictionary to change tokens.

*Tokenization* and *Stop Word Filter* are included in the *Text Mining* node. The models and lists that are used are based on the language which is identified in before. These methods are performed automatically and can not be opted out or changed by a parameter. The filter which is used for stop words is build in and can not be edited. IBM SPSS Modeler uses linguistic tokens for the *Tokenization* process which allows concepts consisting of words.

The *Text Mining* node does not offer a parameter for *Stemming* and the documentation gives no indication if stemming is used at all. The results show that similar words are identified as the same concept based on the *Fuzzy Grouping* method which is also used to remove common spelling errors.

### 6.3.1.2 WEKA

WEKA's strength is the preprocessing of data since it offers a lot of methods. WEKA requires a specific input format which is called the *Attribut-Relation-File Format (ARFF)*. The file format is simple and an unstructured text file can be converted easily by hand. The file contains a relation name, a list of attributes and the type of the attribute which for example can be numerical or a string. Afterwards the data is defined. For classification the last attribute is always the class attribute. In the first step the attributes are a single attribute containing the whole text as string.

For the preprocessing a filter is used. When the ARFF file is loaded into the WEKA Explorer only filters are shown that can be applied on this kind of data. The *StringToWordVector* filter is used to perform the methods of the *Data Preparation* phase. The filter contains the methods and offers parameters to adjust these. By default the filter uses only *WordTokenizer* based on pre-defined delimiters (Figure 13). WEKA offers no *Language Identification* and all build in methods only support English models by default. The filter offers the possibility to select a stemmer and a word list. One of four stemmers included in WEKA is the *Snowball Stemmer*. The language of the stemmer can not be changed. To use a stemmer for a different language a stemmer has to be added to WEKA which is using a different model. WEKA only allows to use a build in stop word list which can not be modified. *Error Cleaning* and *Anonymization* are not supported by WEKA and have to be performed manually when converting the input data into the required format.

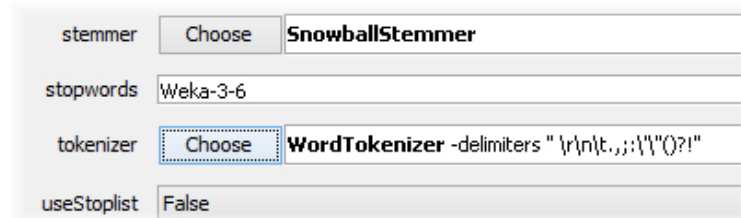


Figure 13: WEKA StringToWordVector Filter

The result of the preprocessing is a new ARFF File with a different structure. Every single word is now an attribute and the data contains the frequency of the word inside the whole document. This filter is usually used to classify multiple documents into a class based on occurring words. The missing tokenization into sentences reduce the possibilities for further work with WEKA. It is possible to build filters yourself in Java which would allow the whole process to be supported by WEKA.

### 6.3.1.3 RapidMiner

RapidMiner is a node based data mining tool and it always starts with an input interface and ends with a result node. RapidMiner offers multiple nodes that can be connected to produce a workflow which is processed on execution. For the *Data Preparation* phase it offers all important methods. The input files can be read with the *Read Documents Files* node which is included in the Text Processing extension. In this node the directory of the input files can be selected. Additionally the file extension can be setup whereby all files with a specific extension are taken as input out of the given directory. Files with a different extension are not read. With a parameter all structural information like xml or html tags can be removed while importing the files into RapidMiner. The encoding of the data can be selected to guarantee the correct encoding. The default setting for the encoding is on automatic. With this setting the encoding is determined based on the input data. This works fine for the input data used in this evaluation, but selecting the proper encoding is guaranteed by selecting it from approximately 50 different encodings.

After the data is imported into RapidMiner it can be preprocessed. For this purpose RapidMiner offers the *Process Documents* node (Figure 14). This node takes the input data, processes it with data preprocessing methods provided by the *Text Processing* extension and creates an word vector as output data. The node can be expanded and the preprocessing can be developed as sub process. Inside this node all methods of the *Data Preparation* phase are performed (Figure 15). Additionally the data is already enriched with Part-Of-Speech tags which is described in more detail in the *Concept Identification* phase. The *Process Documents* node accepts different inputs. It accepts word lists or one or more documents.

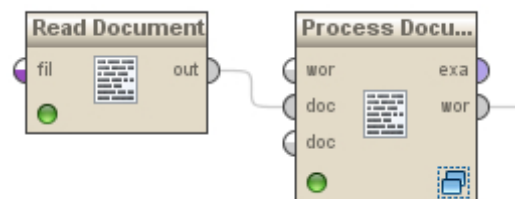


Figure 14: RapidMiner Data Preparation Process

*Language Identification* is not supported in RapidMiner. The user has to set the language for each method where a language specific model is required. *Error Cleaning* and *Anonymization* are performed with a single node. The *Replace Token* node replaces tokens based on a list of regular expressions manually provided by the user. The list can be edited inside RapidMiner which is good for small lists. The usage of regular expressions is bad since they are complex to define for each word. It is not possible to import a dictionary that can be used to replace strings.

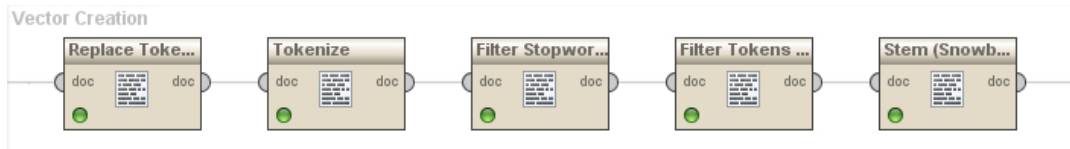


Figure 15: RapidMiner Process Documents Subprocess

The document can be split into tokens using the *Tokenize* node. It tokenizes the document based on a mode selected by the user. It can tokenize based on specific characters, regular expressions, linguistic sentences or linguistic tokens. The linguistic sentences and tokens are available for English, German and a generic Asian language. The linguistic sentence tokenization with the German language is selected in this evaluation.

The next method is the filter for stop words. Therefore RapidMiner offers the *Filter Stopwords* nodes which are available for English, French, Czech and Arabic language. Additionally a node exists which allows the use of a dictionary containing the stop words. The node for the German filter uses a build in standard stop word list which can be switched to a build in sentiment stop word list. The node requires the execution of the *Tokenize* node in a previous step.

The last method in the *Data Preparation* phase is stemming. The *Text Processing* extension includes seven different stemming nodes. The *Stem (Snowball)* is the most comprehensive stemmer using the Snowball language. It can perform stemming for sixteen different languages, including German, English, French and Italian. For German texts the *Stem (German)* node can be used as well which is a stemmer for German texts using a simple stemming algorithm. RapidMiner also offers the *Stem (Dictionary)* node which allows stemming based on a pattern matching rule. This node requires a lot of work and it is not recommended since stemming nodes are available supporting the required language.

#### 6.3.1.4 KNIME

The *Data Preparation* phase in KNIME is implemented in seven different nodes (Figure 16). The nodes need to follow a specific order to work properly. To anonymize the data a node is used multiple times in a row which is outsourced into a sub process to maintain clarity.



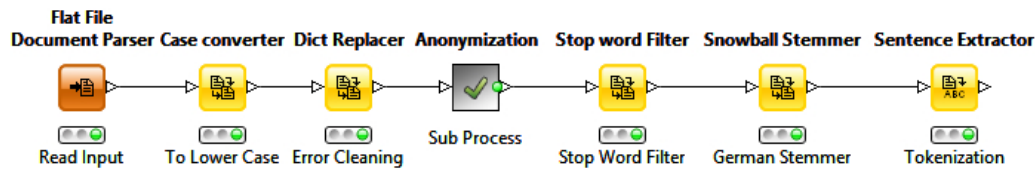


Figure 16: KNIME Data Preparation

The first node that is required for the workflow is the *Flat File Document Parser*. It is used to read the input data and it is the start for the workflow. KNIME is able to read multiple input formats. Word documents can be read with the *Word Parser* node or PDF files can be read with the *PDF Parser* node. These files need to be readable. For example the scanned copies of the court documents can not be read by the *PDF Parser* since they are image PDF files. The *Flat File Document Parser* node offers two important parameters. The first one is the directory which contains the input files, and the second one is the charset. It allows to import multiple files from a single directory.

The next step in the data preparation workflow is the identification of the language. The *Text Processing* extension itself offers no node for this task, but the *Community Contributions* contain a Java snippet for *Language Identification* which identifies the language for each sentence.

KNIME offers a *Dict Replacer* node which is used for the next phases of the process. Based on a dictionary with name value pairs it replaces words inside the document. In the first step common OCR errors are replaced. Therefore a dictionary is required which contains a list of common errors. Afterwards the *Dict Replacer* node is also used to anonymize the data by replacing names of persons, locations and organizations based on an additional dictionary containing a list of the names that have to be removed. The dictionaries can be optimized by a user who knows the data and all involved persons. Additionally other nodes in KNIME help to optimize the dictionaries. *Named Entity Recognition* which is used in the *Concept Identification* phase helps to identify named entities in the data. If names are found that are not yet replaced the dictionaries can be edited and the workflow can be run again. This is a common approach described in the CRISP-DM reference model. Building up the dictionaries for *Anonymization* and *Error Cleaning* is an iterative approach. The simple structure of the dictionaries used by the *Dict Replacer* node force the user to chain them if a string of two words should be anonymized into a single string (Figure 17). The string "FIRSTNAME LASTNAME" is replaced by the string "FIRST-NAME\_LASTNAME" by the first node. The second node then replaces it with the string "SUSPECT1". A *Case Converter* node can be used to reduce the complexity of the dictionaries by requiring only lower or upper case words in the dictionary.

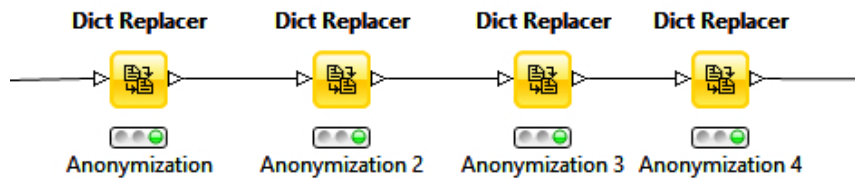


Figure 17: KNIME Anonymization Sub Process

The *Sentence Extractor* node extracts each sentence out of the input data and returns them as data table with an additional row containing the amount of words in the sentence. KNIME offers no node for *Tokenization* of paragraphs, but the sentence level is the best for the following methods. This node is not required for some methods in the following phases since these nodes have internal tokenization steps, but it is required for the *Named Entity Recognition* based on dictionaries in the *Concept Identification* phase and the *Pattern Matching* in the *Relationship Identification* phase.

Stop words are removed with the *Stop Word Filter* node. If not configured the node is using a build in list for the English language. In this case the stop word list is not case sensitive. The node offers three parameters which can be changed to use a user specific list, change the language of the build in list and to make the filter case sensitive. KNIME offers build in stop word filter lists for many languages including English, German, French and Italian. The user-specific list needs to be a single flat file containing a stop word in each column.

For *Stemming* the *Text Processing* extension of KNIME offers three different nodes. They differ in the algorithm which is used for the stemming process and the languages supported. The *Kuhlen Stemmer* is using the Kuhlen stemming algorithm and it is trained only for English words. The second stemmer is the *Porter Stemmer* using the Porter stemming algorithm and similar to the *Kuhlen Stemmer* it is also only trained for English words. The third stemmer node is the *Snowball Stemmer*. It uses the Snowball stemmer library<sup>23</sup> and it can be used for sixteen different languages including German, English, French, Italian, Dutch and Russian. Since this node is the only stemming node supporting the German language it is used to stem the text.

### 6.3.1.5 AutoMap

AutoMap offers different ways to import text. The first one is to import one or multiple text file from the harddrive. The second one is to extract the text from a web page. In this case AutoMap removes the meta information of the file. When the documents are imported the encoding and the text direction have to be selected. AutoMap is developed to work on English documents whereby most preprocess-

<sup>23</sup><http://snowball.tartarus.org/>

ing methods are only available for English documents. This is also the reason why AutoMap does not offer any *Language Identification* method.

Common errors can be fixed using the *Fix Common Typos* method. It works based on an English dictionary containing common typos of English words. On German texts this method shows no useful changes. *Anonymization* of names is not possible. The only way to anonymize names is by changing them manually in before. AutoMap automatically tokenizes the document into words when the concepts are created. Many methods are offered for text preparation. They can be applied to remove numbers, noise verbs or symbols. Like the *Fix Common Typos* method they are also only developed for English documents.

For *Stemming* the user can select between three different stemming algorithms. *KStemmer*, *Porter Stemmer* and *Lex Stemmer*. They use different algorithms and different models. Since version 2.6.1, which was released in 2006, the *Porter Stemmer* supports new languages including German, French and Italian.

### 6.3.1.6 GATE Developer

The structure of GATE Developer can be compared to AutoMap. Methods are called *Processing Resources* and the documents are called *Language Resources*. A workflow is called *Application* and it consists of chained *Processing Resources* working on *Language Resources*. The system used by GATE Developer is called ANNIE, a Nearly-New Information Extraction System. Processing Resources for ANNIE are available by default. ANNIE works on English documents and for documents in different languages additional plugins are required which can be added using the *Plugin Manager* of GATE Developer.

GATE Developer can import different kind of text files as *Language Resource*. A text is loaded into GATE by creating a new GATE document or a GATE corpus. A corpus can contain multiple GATE documents and a GATE document is a normal text file. The creation of a GATE corpus allows to run the workflow on multiple documents inside the corpus at the same time. A corpus can be populated by selecting a directory and defining the extensions that will be imported. Additionally the encoding can be manually defined.

For *Language Identification* the *TextCat Language Identification* plugin can be used. It offers the *TextCat Language Identification* processing resource which can be the start of a workflow. The result of this method can then be used to determine which method is applied next. This helps to build a workflow for documents where the input language is unknown. The method returns a single language for the whole document. A document consisting of multiple different languages will return the language which is detected with the highest confidence.

*Anonymization*, *Error Cleaning* and *Stop Word Filter* are not supported automatically by any processing resource. It has to be performed manually or with a different tool before the document is imported into GATE Developer.

*Tokenization* is performed by the *ANNIE English Tokenizer* and the *ANNIE Sen-*

*tence Splitter*. Both processing resources are designed for English documents and currently no processing resources for different languages are available in the plugin manager. Nonetheless the methods can be applied on German texts since the methods are quite simple and the language does not really affect the tokenizer.

For stemming different stemming methods are available as plugin. The *Porter Stemmer* can be used for English documents and the *Snowball Stemmer* can be used for multiple other languages. The Snowball stemmer supports 11 different European languages including English, German, Italian and French.

### 6.3.2 Concept Identification

The main purpose of the *Concept Identification* is to identify the underlying concepts in unstructured data. The tools that are evaluated fulfill this task in different ways. The second phase of the process is compared in this section for each of the six tools.

#### 6.3.2.1 IBM SPSS Modeler

The methods of the *Concept Identification* phase are included inside the *Text Mining* node. *Part-Of-Speech tagging* and *Named Entity Recognition* are performed after the *Data Preparation* methods are executed. The user has no influence on the order in which the methods are executed. Concepts can not be identified with *Dictionaries* or *Taxonomies* since the node encapsulates the methods and does not allow additional changes of the data set.

The model used for *Part-Of-Speech tagging* and *Named Entity Recognition* are custom models. The tag set used for *Part-Of-Speech tagging* depends on the identified language. The tags used are described in the documentation and in the *Resource Editor*. The custom tag set used for the tagging is more simple than the tag sets provided in other tools. It consists of 11 tags whereas the STTS tag set which is used for German *Part-Of-Speech tagging* by different tools consists of more than 30 tags. Special models exist for seven different languages including English, German, French and Italian. For other languages a generic model is used.

*Named Entity Recognition* classifies named entities into different types. The most common type is '*Unknown*' which is assigned as the default type. Other types identified in the evaluation input data are '*Person*', '*Location*', '*Organization*', '*Date*' and '*url*'. 1462 concepts are identified. The main persons involved in the case are identified as concept with the type *Person*. Some persons appear as duplicates. On successful execution of the *Text Mining* node the *Interactive Workbench* is opened. This view shows a list of all identified concepts and their type (Figure 18). The concepts can be filtered by type or by the minimum amount of appearance in the document. Additional custom types can be added by the user which than can be manually allocated to a concept.

		72 (27%)	57 (5%)	<Unknown>
Suspect 2		69 (2%)	51 (5%)	<Person>
polizei		55 (1%)	50 (5%)	<Unknown>
das geld		55 (1%)	38 (3%)	<Unknown>
auf frage		53 (1%)	53 (5%)	<Unknown>
Suspect 1		53 (1%)	46 (4%)	<Person>
Victim 1		52 (1%)	50 (5%)	<Person>
beschuldigten		52 (1%)	42 (4%)	<Unknown>
geschädigten		42 (1%)	34 (3%)	<Unknown>
Victim 2		40 (1%)	35 (3%)	<Person>
schulbeordnunge		35 (1%)	30 (3%)	<Unknown>

Figure 18: IBM SPSS Modeler List of Concepts

### 6.3.2.2 WEKA

The strength of WEKA is classifying documents. Therefore it supports many methods used for data preprocessing. WEKA itself offers no build in filter for *Concept Identification*. Part-Of-Speech tagging can be performed using the TagHelper<sup>24</sup> developed by the Computer Science department of Carnegie Mellon University. The TagHelper is a tool for WEKA which allows detection of Part-Of-Speech bigrams [Rose et al., 2007].

*Named Entity Recognition* in WEKA is very complex and requires annotation of the input data set. The *CoNLL-2003 Shared Task*<sup>25</sup> corpus is used as trainings data since it is annotated in German and in English. The data set is splitted into trainings and test data for classification. Afterwards the input document can be classified using different algorithms. Algorithms that can be used are *Support Vector Machines* (LibSVM or SMO), *Maximum Entropy*, *Boosting approaches* (AdaBoostM1) or *Decision Trees* (J48). The results of the algorithm is visualized with a confusion matrix showing how the words have been classified (Figure 19).

```

=== Confusion Matrix ===
      a      b      c      d      e      f      g      h      i  <-- classified as
44795  45    57    45   115    0    0    0    0 | a = 0
 561  1212   62    19   135    0    0    0    0 | b = I-PER
 388   225   623   23    78    0    0    0    0 | c = I-LOC
 642   39    46   346   53    0    0    0    0 | d = I-MISC
 719   267   261   49   821    0    0    0    0 | e = I-ORG
 9     0     0     5     1    0    0    0    0 | f = B-MISC
 2     0     0     0     1    0    0    0    0 | g = B-LOC
 0     0     0     0     0    0    0    0    0 | h = B-ORG
 0     0     0     0     1    0    0    0    0 | i = B-PER

```

Figure 19: WEKA Named Entity Recognition Confusion Matrix

WEKA offers no method for *Dictionaries* or *Taxonomies*. These have to be manu-

<sup>24</sup><http://www.cs.cmu.edu/~cprose/TagHelper.html>

<sup>25</sup><http://www.cnts.ua.ac.be/conll2003/ner/>

ally tagged in the training data set. It is also possible to create an own training set. This would require a lot of work since a training set needs to contain many examples. For small training sets WEKA offers the option to use cross validation on the training set instead of requiring huge training and test sets.

### 6.3.2.3 RapidMiner

The text is already enriched with *Part-Of-Speech tags* in the *Process Documents* sub process. The *Filter Token (by POS Tags)* node allows to filter tokens based on their tag. The node supports only English or German texts using the STTS tagset for German and the Penn Treebank tagset for English texts. The filter helps to identify the concept by reducing it to identified named entities (NE). The filter allows to filter for specific tags or to invert the filter to remove all tokens with a specified tag. The results are displayed in a data table containing all tokens tagged as named entity (Figure 20). The main persons of the case are identified and the occurrences also show that the main suspects and the main victim occur the most as expected.

Attribute Name	Total Occurences ▼
Victim 1	123
Suspect 1	98
Suspect 2	74
...	...

Figure 20: RapidMiner List with Named Entities by POS-Filter

*Named Entity Recognition* (NER) besides the variant using the *Filter Token (by POS Tags)* can not be performed by a different node of the *Text Processing* extension. The *Information Extraction* extension offers node to perform *Named Entity Recognition* in a modular system that can be easily adjusted. The extension is provided from the department of Artificial Intelligence at the Technische Universität Dortmund. The Plugin offers multiple nodes to perform NER and also examples how the workflow is created. Unfortunately the extension is not working on the current version used in this evaluation. The examples<sup>26</sup> trained on an older version of the extension show that in theory the methods offer good results (Figure 21). Nonetheless the nodes can not be used in this evaluation.

label	id	confidence(...)	confidence(...)	confidence(...)	confidence(...)	prediction(...)	word
PER	16	0	0	0	0	PER	Chris
O	17	0	0	0	0	O	goes
O	18	0	0	0	0	O	to
LOC	19	0	0	0	0	O	Hamburg
O	20	0	0	0	0	O	.

Figure 21: RapidMiner Named Entity Recognition Example

<sup>26</sup><http://sourceforge.net/projects/ieplugin4rm/>

RapidMiner has no *Dictionaries* or *Taxonomies* nodes that can be used to create concepts manually. The only way to import data is using a import data node and read a document of an .csv-file. This manually created concepts can then be identified in the main document.

### 6.3.2.4 KNIME

KNIME offers different nodes for *Concept Identification* and it allows to perform all methods of this phase. Not all methods are supported for German, but they all are supported for the English language. The methods in this phase do not need to be combined in a chain to retrieve good results. Some nodes output the original data which can be used by further nodes to find relations. The *Column Filter* node is used to specify a specific column for the following node containing only the required data. Otherwise nodes would output error messages since they do not now which column contains the input data.

The first node is the *Part-Of-Speech* tagger. KNIME offers multiple tagger nodes for different purposes. Besides common Part-Of-Speech tagging it offers tagger for specific areas like chemical named entities or simple dictionary tagger. For Part-Of-Speech tagging the *Stanford tagger* node is used to enrich the data with Part-Of-Speech tags. The models used by the tagger are of the Stanford NLP Group<sup>27</sup> and are applicable for German, English and French. Each language is using a different tag set which is trained on the specific language. The STTS tagset is used for German texts, the French Treebank tagset is used for French texts and the Penn Treebank tagset is used for English texts. The *Stanford tagger* node offers different models for each language which differ in speed and accuracy. Since the input data which is used for the evaluation is quite small the most accurate model is chosen and it finished in less then a minute. The *STTS Filter* node is used to filter the data for tags of the STTS tagset for German language (Figure 22). The node allows to filter the input data based on tags selected in the parameters. Filter nodes are also offered for the other tag sets. The *Stanford tagger* node needs to be applied right after the input node.

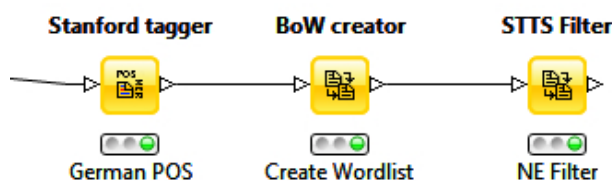


Figure 22: KNIME Part-Of-Speech tagging

Another tagger offered by KNIME is the *OpenNLP NE tagger* node. This node allows to specify named entities in the document as persons, locations and organi-

<sup>27</sup><http://nlp.stanford.edu/software/tagger.shtml>

zations. It is specialized for *Named Entity Recognition*. The disadvantage of this node is that it is not available with an German model. It can be used on a German text, but the quality of the results are not comparable to the quality of the results on an English text. On default the node is configured with a model for the English language. Additional models for different languages for the *OpenNLP NE tagger* node are developed by the Apache Software Foundation<sup>28</sup>. It is also possible to create a model yourself. This requires expertise and a good and large training set. After enriching the data the *Standard Named Entity Filter* node can be used to display all discovered named entities in the data (Figure 23). To filter the data it needs to be transformed with the *BoW creator* node which creates a bag of words. This is a list of each word and a reference to the position in the sentence and the position in the whole document. The results produced by the NE tagger show the main persons involved in the case. Nonetheless some concepts are identified which are not expected to be concepts. This is due to the use of the English model.

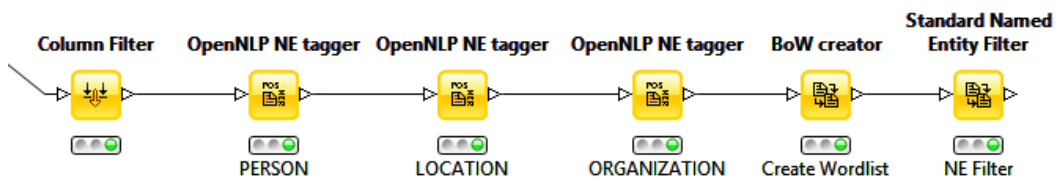


Figure 23: KNIME Named Entity Recognition

A *Dictionary tagger* node can be used to enrich the data based on words and tags defined in a dictionary. This is used as a work around for the German language since there is no model for the *Open NLP NE tagger* node. The creation of the dictionary requires some manual work and it needs to be optimized when additional named entities are discovered. The dictionaries are attached to the *Dictionary tagger* node after importing them with the *File Reader* node. Filtering the results of the *Stanford tagger* for named entities is a good starting point for the creation of a dictionary. The dictionary for this tagger have to contain a token each row. The tag itself is defined in the tagger. This results in single dictionaries for each tag. To enrich the data with PERSON, LOCATION and ORGANIZATION tag three *Dictionary tagger* nodes and three dictionaries are required (Figure 24). The *OpenNLP NE tagger* and the *Dictionary tagger* can be combined to achieve the most accurate results. In this case the *Dictionary tagger* node is used to add concepts, which are not discovered by the automatic detection of the *OpenNLP NE tagger* node.

<sup>28</sup><http://opennlp.sourceforge.net/models-1.5/>



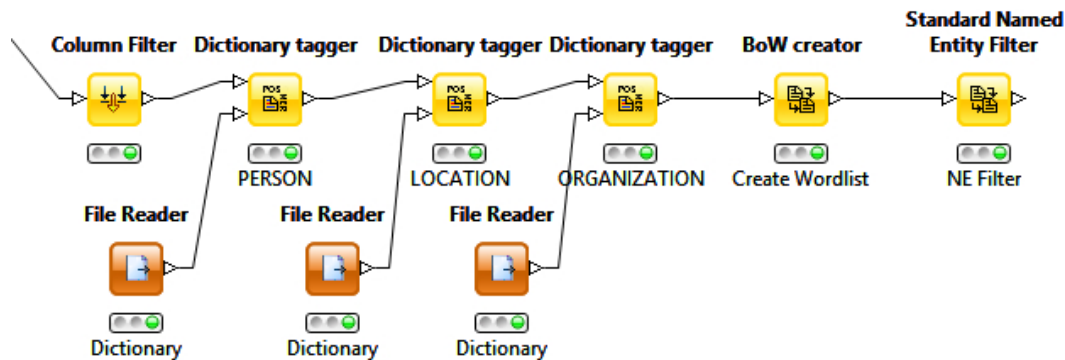


Figure 24: KNIME Dictionary tagging

KNIME supports *Taxonomies* only with the *Dict Replacer* node in the Data Preparation phase. Afterwards the other nodes previously described in this section can be used.

### 6.3.2.5 AutoMap

In the *Concept Identification* phase all automatic methods of AutoMap work with models trained for English text documents. There is no support for different languages in the current version for *Part-Of-Speech tagging* and *Named Entity Recognition*.

AutoMap offers four methods for *Part-Of-Speech tagging* in the *Generate* tab. The first one is to perform the tagging using the default parameters. The second method is to generate an attribute list. The third method is to extract only all verbs and the last is to extract only all nouns. The *Part-Of-Speech tagging* is based on the Hidden Markov Model which is a statistical model. It can be considered as the simplest *Dynamic Bayesian Network* [Carley et al., 2013]. The tags are assigned using the Penn Treebank tagset which is the common tagset for English documents.

Named entities can be extracted using the *Named Entities* method in the *Generate* tab. It extracts the named entities into a list. The list is a .csv file with three columns. The first one contains the entity, the second one the frequency of the entity and the last one the number of texts. After extracting the named entities a generalized thesauri can be created based on that list. For the German document the *Named Entity Recognition* is not suited. The quality of the result is bad. Most of the persons are identified, but additionally many capitalized words are detected as persons. For example the token '*Sachverhalt* (transl.: facts)' is detected as a person. This can be traced back to the fact that simple methods assume capitalized words to be named entities in the English language.

Since the automatic creation of a thesauri is not working properly for German text documents the thesauri needs to be created manually. A thesauri for English texts can be created automatically using multiple methods provided by AutoMap

which differ in the purpose of the thesauri. The structure of a thesauri is simple. It consists of four columns. The first one contains the original concept, the second one the transformed concept, the third one the type and the last one the name. Automatic created thesauri usually assign no name. The type for the automatic created thesauri of the named entity thesauri is *'agent'*.

```
"conceptFrom", "conceptTo", "metaOntology", "metaName"
```

The structure of the thesauri can be used to support *Dictionaries* and *Taxonomies*. AutoMap offers methods in the *Procedure* tab to automatically enhance the thesauri. Duplicates and circular logic can be detected and removed.

### 6.3.2.6 GATE Developer

The *Concept Identification* phase is supported by different processing resources in GATE Developer. By default the *ANNIE POS Tagger* is available to enrich the document with Part-Of-Speech tags based on the Penn Treebank tagset. This tagger is developed for English documents only. For other languages plugins are offered for different languages. For German the *Lang\_German* plugin offers processing resources to perform Part-Of-Speech tagging using a *TreeTagger*. It is assumed that the STTS tagset is used by the processing resource, but it is not documented. Other plugins including methods for different languages are *Lang\_French*, *Lang\_Hindi* and *Lang\_Italian*. The quality of these processing resources differ since they are developed by different contributors and mostly derived from the original processing resource.

*ANNIE Gazetteer*, *ANNIE NE Transducer* and *ANNIE OrthoMatcher* are processing resources that can be used for *Named Entity Recognition* on English documents. The plugins previously imported for Part-Of-Speech tagging also include processing resources for *Named Entity Recognition*. The types for the annotation can be defined in the parameters of the single processing resource. By default *'Persons'*, *'Locations'*, *'Organizations'* and *'Dates'* are annotated.

The *ANNIE Gazetteer* is using a list containing names of entities which can be described as a *Dictionary*. The list used by the Gazetteer can be edited manually. It is named *'lists.def'* and can be found in the GATE developer directory. GATE Developer does not support *Taxonomies*. These have to be edited manually before the file is imported.

### 6.3.3 Relationship Identification

The *Relationship Identification* phase consists of methods used to identify relationships between the previous identified concepts. In this section the capability of these methods are evaluated for each tool.

### 6.3.3.1 IBM SPSS Modeler

Relations between the concepts identified during the *Concept Identification* are also developed with the *Text Mining* node. The relations between concepts are based on similarity and co-occurrence. Concepts are automatically categorized using linguistic techniques. In the advanced settings for the category building the user has the option to specify the linguistic techniques. For example the user can choose which grouping techniques are used. On default the categories are grouped using *Concept Root Derivation* and *Concept Inclusion*. *Co-Occurrence* is not set as default. Additionally specific category pairs can be defined which should not be paired together. The user has the option to build categories using term frequencies instead of linguistic techniques. The case data did not show a huge difference in the categories for both methods. A category is based on a concept and contains one or multiple concepts. A category can also contain a sub category containing more specialized concepts (Figure 25). The amount minimum subcategories can be defined in the advanced options for the category building.

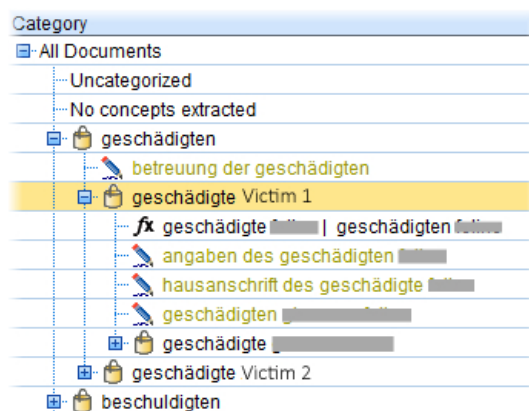


Figure 25: IBM SPSS Modeler Concept Categories

The categories automatically created out of the concepts based on similarity can be manually edited. Sub categories can be moved to the top level and new categories can be created and descriptors inside a category can be moved into a different category. This process is similar to qualitative analysis where text is coded. IBM SPSS Modeler offers the possibility to inspect a category and see which other category occurs in context with it. The category 'Victim' occurs often close to the category 'Suspect'. These categories are the biggest categories for the German case documents.

*Pattern Matching* is not supported for *Relationship Identification* but it is not necessary since the automatically identification using co-occurrence delivers a good quantity of valid relations between main concepts.

### 6.3.3.2 WEKA

WEKA offers many methods for *Data Preparation*. *Concept Identification* can be achieved using additional methods that are developed by third parties but for *Co-Occurrence* and *Pattern Matching* there is no external tool. The user has the opportunity to use a self created model. In this work the workflow of WEKA ends with the *Concept Identification* phase since creating a model for *Relationship Identification* using *Co-Occurrence* or *Pattern Matching* requires background knowledge of a user. For this evaluation the focus is on a user perspective evaluation which means that tools are evaluated from the perspective of a potential user. Developing own models is not part of this definition.

### 6.3.3.3 RapidMiner

RapidMiner has no nodes for any *Relationship Identification* method. The *Information Extraction* extension offer nodes for *Relation Extraction*. These nodes require the *Named Entity Recognition* results of the *Concept Identification* phase. The problem is that these results do not exist since the plugin is not working on the current version of RapidMiner. The nodes for this task use composite kernel methods which evolved of the tree kernel method [Jungermann, 2009].

### 6.3.3.4 KNIME

Relationships can be identified in KNIME with different methods which require different previous methods. *Co-Occurrence* can be realized with the *Term Co-Occurrence counter* node and *Pattern Matching* can be realized by querying a table consisting of sentences with expressions including the identified concepts.

The *Term Co-Occurrence counter* can be applied after the *Standard Named Entity Filter* node which is the last node of the *Concept Identification* phase (Figure 26). The node identifies co-occurring concepts. The co-occurrence level shows if the concepts occur inside a document, inside a sentence or if they occur as neighbours. The level can be adjusted in the parameters of the node. Additional parameters can be adjusted to optimize the node for larger input data. For the small data set the default configuration is used and the co-occurrence level is set to sentences. This method reveals a relation between two concepts, but the type of relationship is not revealed.

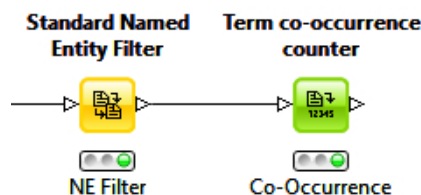


Figure 26: KNIME Term Co-Occurrence

*Pattern Matching* is more complex but the user has the opportunity to search for specific relations. The *Pattern Matching* process requires the combination of multiple nodes and it is applied after the *Tokenization* method which is the *Sentence Extractor* node. First of all an index of the table needs to be created using the *Table Indexer* node. In the parameters the columns needs to be selected which should be indexed. After the *Sentence Extractor* node the table contains a SENTENCE column which is selected in the parameters of the *Table Indexer* node. When the index is created for the table the *Index Query* node can be used to search for a specific pattern inside the table. The query is defined in the parameters where the rows and the operator for the query are selected as well. The query language used in this node is a custom query language described in the documentation of the node. The '\*' character can be used as a wildcard and each line is a single query which is combined with the operator selected in the parameters. The concepts can be used from the dictionary used for the anonymization. A query with the parameter 'OR' and the following input result in a list of sentences where either a suspect and a victim, a suspect and a witness or a victim and a witness are mentioned. The sentences resulting from the query show the main interactions between the main persons of the case.

```
SUSPECT* AND VICTIM*
SUSPECT* AND WITNESS*
VICTIM* AND WITNESS*
```

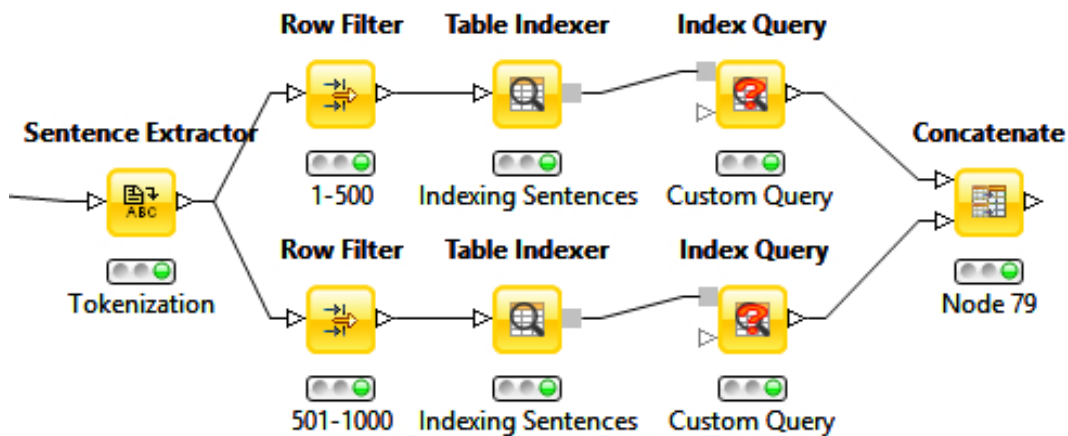


Figure 27: KNIME Index Query for Pattern Matching

The *Table Indexer* node takes a lot of computation which results in bad performance. To optimize the performance the input table is split into multiple smaller tables using the *Row Filter* node (Figure 27). For each smaller table a index is created and the table is queried. Afterwards the resulting tables containing the sentences

are concatenated using the *Concatenate* node which allows to merge two data tables. The disadvantage of *Pattern Matching* is that the query needs to be defined manually whereas the *Term Co-Occurrence counter* node performs automatically. The advantage on the other hand is that the type of relation can be better identified since the whole sentence is given as result.

#### 6.3.3.5 AutoMap

Relationships between concepts are detected based on the thesauri which is automatically or manually generated in the previous phase. Therefore the thesauri is applied on the imported text using the *Apply Generalization Thesauri* under *Preprocess > Text Refinement*. Besides the thesauri generated in the previous phase some standard thesauri are offered by AutoMap. These thesauri are optimized for locations, agents or organizations. When applying the thesauri the user can select if he wants to use the thesauri content only. This reduces the text and speeds up the process of the network creation in the next step.

In this step the relations between the concepts included in the thesauri are identified. For the German document used in this evaluation 298 relations and 249 concepts are identified. In this evaluation the thesauri is created automatically and shows many valuable relations. Due to the automatic creation many concepts without relations are identified. Nonetheless most of the important relations between the main persons are identified. A manually created thesauri is expected to achieve better results. But the creation of a good thesauri requires a lot of manual work and knowledge about the document. The common way is to generate a thesauri automatically and to review and edit it.

#### 6.3.3.6 GATE Developer

GATE Developer offers no plugins for automated *Relationship Identification*. Manual methods like the windowing technique can be applied to identify relations between previously identified concepts. Since GATE Developer can be enhanced by user developed plugins it is possible to develop a plugin using *Co-Occurrence* or *Pattern Matching*.

### 6.3.4 Concept Network Analysis

The last phase of the process is the *Concept Network Analysis* phase. In this section the tools are described how far they are able to visualize the output of previous phases in any kind of way. The best way is a visualization as a network diagram of concepts connected based on their relations. Some tools already delivered unsatisfying results in previous phases which results in short descriptions of their ability to visualize networks.

### 6.3.4.1 IBM SPSS Modeler

IBMs *Interactive Workbench* of the *Text Mining* node already offers a visual output for the user where the user can experiment with the concepts. Concepts map can be created for each identified concept. A concept map shows relations between concepts based on their similarity (Figure 28). The degree of the similarity is indicated by the edge. The concept maps include all concept types by default. The focus of this evaluation is on the relations between persons and locations. Therefore the map can be filtered. The concept map for *Suspect 1* does not show a relation to a location. This is based on the fact that only 13 concepts with the type location are identified. These concepts occur with a maximum of 8 times and usually are the place of birth or the current address of a person.

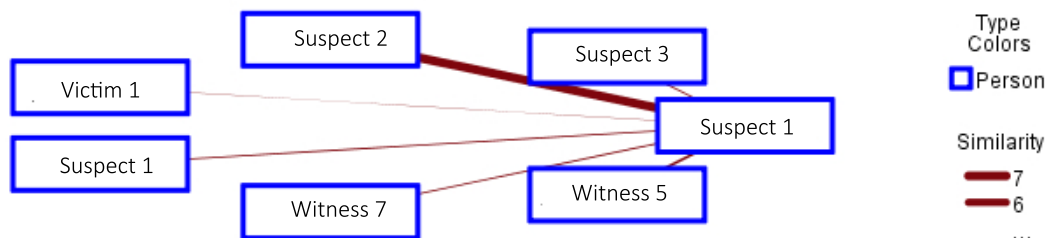


Figure 28: IBM SPSS Modeler Concept Map for Suspect 1

The network (Figure 28) shows the identified relations of *Suspect 1*. One relation is to himself which is a result of a duplicate concept due to variable names in the document. The strongest relation is with *Suspect 2* who is the other main suspect of the case. *Victim 1* and *Suspect 3* also show a relation to *Suspect 1* and two witnesses are also related. The witnesses are related since their names occur in the testimony close to the suspects name. *Suspect 3* shows a smaller similarity to *Suspect 1*. *Suspect 3* is only a bystander in this case and he is not mentioned that often in the case documents. The concept map automatically created reflects the results that have been achieved manually.

Besides the concept map category networks can be created. A network shows how categories relate to each other (Figure 29). The network diagram shows that '*Suspects (transl.: 'beschuldigten')*' and '*Victims (transl.: 'geschädigten')*' have a strong relationship based on their co-occurrence. The diagram is reduced on top level categories to maintain visibility. Nodes can be edited and moved around and the structure of the network can be changed based on four different network templates.

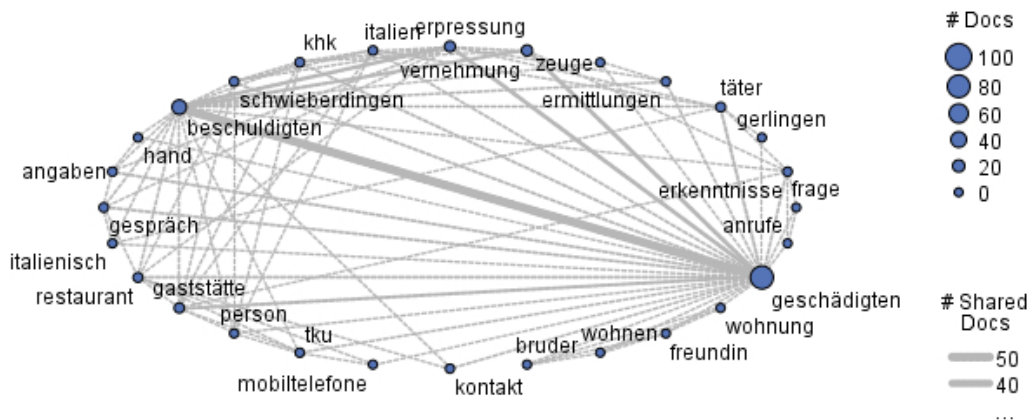


Figure 29: IBM SPSS Modeler Category Network

The visualizations can be exported as images and a model can be generated using the category and concept list. The model can be used in new streams to classify new documents.

#### 6.3.4.2 WEKA

The workflow of the process in WEKA already ended in the *Relationship Identification* phase. Therefore the methods in this section are only based on theoretical possibilities. WEKA offers a register tab for visualization which is usually used to visualize clusters or distributions of values. After preprocessing the *Term Frequency* can be visualized. The graphs created can be exported as images. Additionally to the graphical visualizations WEKA offers matrices that show the results of a classifier that had been applied to the data. A confusion matrix showing the result of the *Named Entity Recognition* is shown in Figure 19. The internal format of WEKA is always the ARFF format. After preprocessing the preprocessed file can be exported as ARFF file. This file then can be used again without applying additional preprocessing filters again. Models created with the training and test dataset can be saved as model file or the Java code for the model can be exported. This allows the user to implement the created model into a Java application. A Java application with a visualization library for example can be used to visualize the recognized named entities.

#### 6.3.4.3 RapidMiner

The first two phases of the process are supported well by methods of RapidMiner. Relations could not be identified in the third phase due to missing nodes. RapidMiner is able to visualize results and to export or store them. Every node offers the opportunity to inspect the outcome. Therefore the process needs to be stopped at



the node by setting a breakpoint. The breakpoint can be set before or after the execution of a node. At the end of a process the results are always displayed for the user. The way of visualization depends on the last node used in the workflow. The most common way are data tables showing the results. For documents the visualization is a view split into two parts (Figure 30). One part is showing the original document and the second part is showing the preprocessed one. After the *Tokenize* node the tokens are marked in different colors to show where the *Tokenizer* split the sentences.

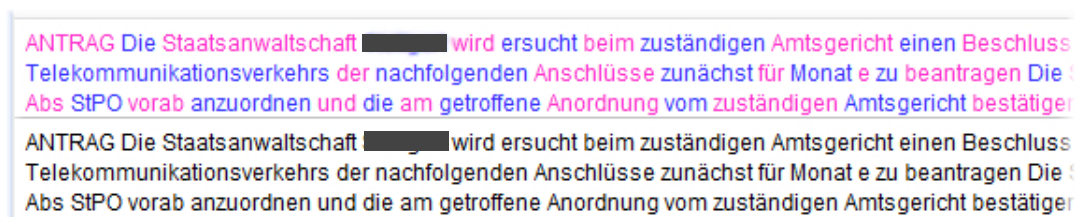


Figure 30: RapidMiner Document View

RapidMiner itself has no nodes to create a network diagram or to visualize relations in any kind of way. It only offers many nodes to export the data into different data formats. These formats range from simple flat files to Excel or Access data formats. RapidMiner also offers nodes to write results into a database or to update a database entry. Models created in RapidMiner can be exported as XML file or as binary which then can be used in a Java application.

#### 6.3.4.4 KNIME

KNIME is able to visualize the results in different ways. First of all it offers a *Document Viewer* node which can be applied after each node to visualize the current document with all preprocessing steps. The node is helpful while configuring the nodes in the *Data Preparation* phase. Additionally the output of every node can be inspected by the user after it is run. The view for the results depends on the output and usually it is a data table. The data tables can be sorted by each column in ascending or descending order. *Maximum Distance Seperable (MDS)* matrices can be used to visualize distance in term co-occurrence using distance aware tag clouds [Adä et al., 2010]. KNIME offers nodes to visualize these tag clouds. The most simple way is to visualize the amount of co-occurrences between concepts by changing the size of the concept. Two concepts that occur often will be displayed with a greater font than concepts that occur less (Figure 31). In this case the concepts *Suspect 1* and *Suspect 2* co-occur the most inside the document which reflects their relation in the case. The second most co-occurrence is between *Suspect 2* and *Victim 1*. This relation is strong since it is one of the few relations that appear in both incidents of the case.

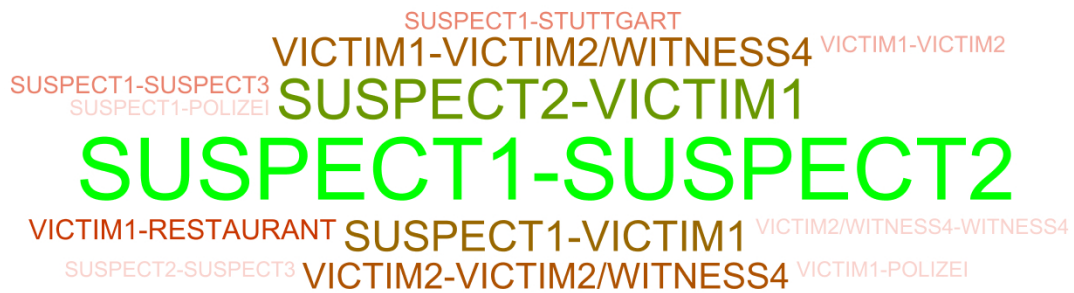


Figure 31: KNIME Tag Cloud

The tag cloud is generated based on the results of the *Term Co-Occurrence counter* node. For visualization it requires some additional nodes that prepare the data by merging columns, removing tags and adding colors (Figure 32). The nodes are part of the basic data mining methods of KNIME.

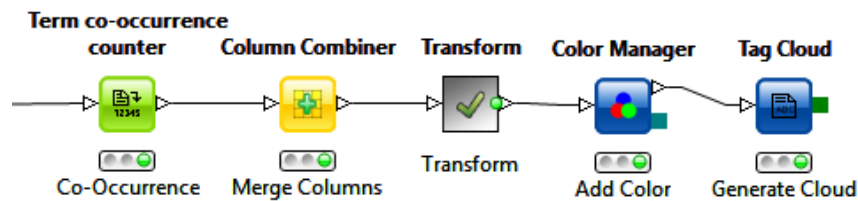


Figure 32: KNIME Tag Cloud Process

The results of the *Pattern Matching* method can be displayed as data table. The result of this method are sentences containing a relationship between two concepts. The results contain more information about the type of relation between two concepts, but it would require many transformation steps to develop a network out of it. A translated example revealed by *Pattern Matching* is displayed as a row in a table and looks like the following.

SUSPECT1 went to the restaurant of VICTIM1 and VICTIM2

KNIME has nodes to create a *Social Network*. The input for the network has to be created manually in form of dictionaries or the results of the *Term Co-Occurrence counter* can be transformed to fit the desired input of the network nodes. First of all an empty network is created using the *Network Creator* node. The *Object Inserter* node is then used to insert the nodes and edges provided at the second input. Afterwards the *Network Viewer* node visualizes the network (Figure 33). Parameters of this node can be changed to allow optical changes or to add weights to the edges based on a third column.

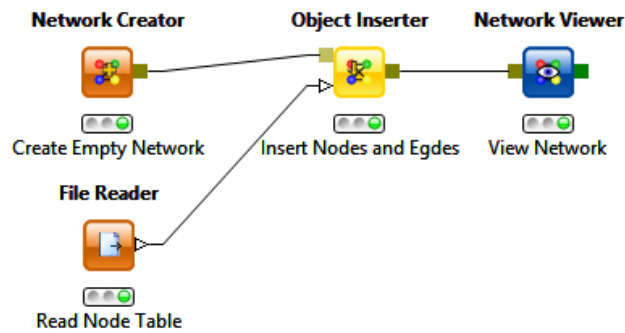


Figure 33: KNIME Social Network Analysis Process

The result of the *Social Network Analysis* offers an insight in the relations between the discovered concepts. The weights give insights on the strength of the relationship, but the nodes offer no insight on the type of relationship between the concepts. For this social network the number of sentence co-occurrences define the weights of the edges. The network (Figure 34) shows similarity to the manually created network in chapter 2. All relations between the main persons are represented. The network is created with the *Network Creator* node and it can not be modified.



Figure 34: KNIME Social Network

KNIME offers multiple nodes to export the data and write it into different file formats. Besides writing results into flat files or saving images of the created networks it can store results in a database.

#### 6.3.4.5 AutoMap

AutoMap is able to create a meta network containing nodes representing a concept and relations represented by edges. A network is created using the *MetaNetwork DyNetML (Per Text)* method under *Generate > MetaNetwork*. To create the network the previous developed thesauri is required. AutoMap then creates a xml-file containing the structure of the network.

The texts on which the selected methods are applied are always shown in the main window. Nonetheless the window does not show concepts or relations between concepts in this window. A XML-viewer provided by AutoMap can be used to inspect the xml-file which contains the nodes and their relations. AutoMap itself offers no possibility besides this xml-viewer to view the results or to modify it. The format used by AutoMap is the DyNetML language, a xml-derived language. It is an xml based interchange language for relational data.

ORA is a tool developed by the same developers as AutoMap and its purpose is to visualize networks provided in the DyNetML language format. The network created in AutoMap can be visualized using ORA which is available as 180-day free trial version (Figure 35). Meta information of the network are shown when the DyNetML xml-file is loaded into ORA. ORA allows to modify the network. Relations can be hidden or displayed differently based on a given confidence value. Different layouts can be selected to visualize the network in a specific way. A *MetaNetwork Designer* can be used to create a MetaNetwork manually.

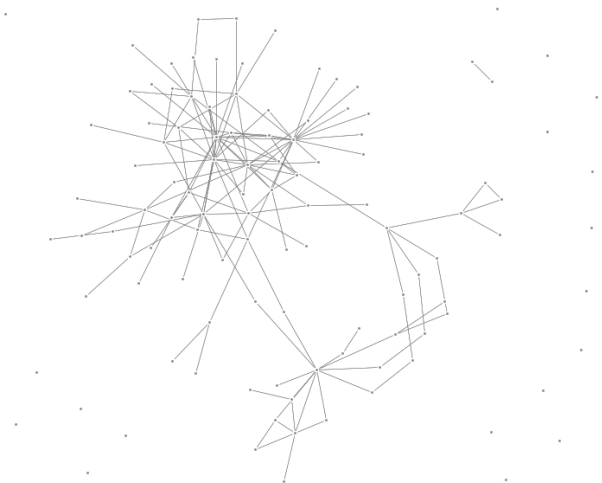


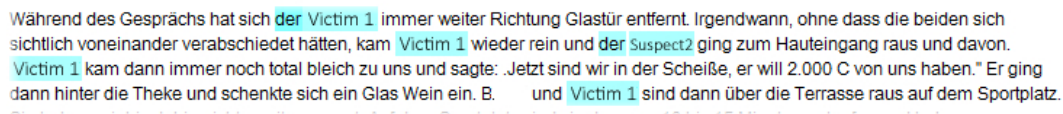
Figure 35: MetaNetwork in ORA

The network visualized with ORA shows the main concepts in the center with relationships to other main concepts. Due to the automatic creation on English language the result is not perfect. Some relations are between a single concept which is represented as multiple single concepts. Also German stop words are identified as concepts with a huge number of relations. This is due to the fact that AutoMap

just supports *Stop Word Filter* for English documents. Nonetheless relations between the four main persons are identified. These concepts also show a huge number of relations. Many concepts are represented without any relation. Compared to other networks the network visualized in ORA is much more complex and contains many wrong concepts.

#### 6.3.4.6 GATE Developer

GATE Developer visualizes changes inside the main window. After running the process, the document used for the process can be opened in the main window. The workflow takes a little bit more than one minute for the German document. The view allows the user to inspect the document. *Annotation Sets* can be selected to highlight detected entities (Figure 36). The German *Named Entity Recognition* included in the *Lang\_German* plugin identifies the major named entities, but also misses some. The English processing resource applied on the German text also identifies some of the main named entities, but also classifies many words as named entities which are none.



Während des Gesprächs hat sich der Victim 1 immer weiter Richtung Glastür entfernt. Irgendwann, ohne dass die beiden sich sichtlich voneinander verabschiedet hätten, kam Victim 1 wieder rein und der Suspect2 ging zum Haupteingang raus und davon. Victim 1 kam dann immer noch total bleich zu uns und sagte: „Jetzt sind wir in der Scheiße, er will 2.000 C von uns haben.“ Er ging dann hinter die Theke und schenkte sich ein Glas Wein ein. B. und Victim 1 sind dann über die Terrasse raus auf dem Sportplatz.

Figure 36: GATE Developer Identified Concepts

A network can not be visualized since there is no plugin for *Relationship Identification* which leads to a network of concepts. GATE Developer allows to save the current application state of a specific workflow. The workflow can also be exported for *GATEcloud*, a cloudbased GATE solution.

## 6.4 Comparison

All six tools are able to perform methods of the analysis process developed in chapter 5.2, but not all of them are able to perform the whole process. Each tool has advantages and disadvantages and only three tools are able to visualize relations between identified concepts. All tools support English documents and most of them also offer models and methods for different common languages including German, French and Italian. All tools, except AutoMap, run on Windows, Linux and UNIX operating systems. AutoMap and ORA are only available for Windows. The general usability of all tools differ a little bit. Most of the tools are quite easy to learn and to use whereas WEKA requires the user to have deeper knowledge of each single methods. Node-based tools like IBM SPSS Modeler, RapidMiner and KNIME offer a great user interface which allows new users to handle the tool quite fast. GATE Developer and AutoMap are not that intuitive and require the user to work close with the documentation. The documentation is great for all of the tools and at least

available in English. IBM SPSS Modeler offers a comprehensive documentation in sixteen different languages.

The quality of the results produced by the tools differ. KNIME and IBM SPSS Modeler offer the best results when compared to the manual results, but KNIME requires a lot of manual work. IBM SPSS Modeler is able to produce good results nearly completely automatic. AutoMap is the third tool that is able to visualize relations, but the quality suffers from the automatic generated thesauri, which is only recommended for English documents.

The three networks developed with the tools differ and are not equal to the networks manually developed. The reason is that the case covers two incidents that happened in a time frame of one year. The networks automatically developed show relations between concepts for both cases combined and the manual developed networks show each case. Both cases combined (Figure 5) are very complex and the automatically developed networks miss some of the relations between persons that are passively mentioned in the text. For example 'Victim 2 is married to Witness 4' is not represented in any of the automatic developed networks. In the text it is only mentioned when the victim refers to his wife. Nonetheless most of the important relations are included in all networks.

## 7 Conclusion and Outlook

In this work a framework to evaluate text processing tool was developed and six different tools for data mining, machine learning and natural language processing were evaluated. Therefore the process was derived from the *Conceptual Analysis Process* of the GLODERS project and text mining methods were identified required to perform each phase of the process. This process then was used to create an evaluation scheme using the *Model-Dependent Software Evaluation* framework.

The evaluation of the tools based on a German court document shows that still a lot of manual work is required for all tools to automatically generate a useful network which can be compared to the manual results. IBM SPSS Modeler offers the most comprehensive network without any manual work. KNIME offers a good network after some manual work which is required since the model for German NER is not available. AutoMap is good for English documents, but for German documents it is not recommended since it requires a lot of manual work to create a thesauri. WEKA, RapidMiner and GATE are not suited for automated text processing on German documents since they currently lack the required methods in the *Relationship Identification* and *Concept Network Analysis* phase.

The evaluation scheme developed can be applied for additional text processing tools to identify if they are suited for the process and the scheme can help a user to decide which tool he might use for a given purpose. Tools like GATE Developer, RapidMiner, KNIME and WEKA have the advantage that they can be enhanced by users with development experience by writing plugins or small code snippets. All these tools are developed in Java which is a common development language. GATE Developer has the advantage that it already offers many official plugins provided by trusted developers.

For German documents IBM SPSS Modeler is suggested since it allows to visualize relations between concepts without any additional manual work. The downside of IBM SPSS Modeler is that it is a commercial tool and all other tools are at least available in a non-commercial version. KNIME can be used to develop a network with some manual work for German documents since it lacks the German OpenNLP model for *Named Entity Recognition*. For English documents KNIME allows to create a workflow that is nearly completely automatic. The creation of the workflow is much more complex than the creation of the workflow in IBM SPSS Modeler. AutoMap is the third tool which can be used on English documents. The quality for documents in different languages depends on the amount on manual work put into the creation of a thesauri.

All three research questions are answered in this work. The first research question is answered in chapter 5.2. The second question is answered in chapter 6.1 and the last question is answered in the whole chapter 6 but it is summarized in chapter 6.4. In this work the evaluation and the conclusion focuses on an evaluation of the users perspective. The results may totally differ when the scope of the user is enhanced to a user who is able to develop additional code snippets. Tools like

WEKA and GATE Developer can deliver great results when the required methods are added.

Possible future work would be the expansion of the evaluation scheme for an evaluation from a different perspective. Especially the extendability of a software tool can have a huge impact on the results. The focus of this evaluation was to evaluate the tools based on a German document. This results in tools supporting the German language to deliver better results. Documents are available in many different language whereby a future work could also focus on an language independent workflow taking the *Language Identification* results into account.

Which tool a user is going to select for future work depends on the user and on the documents that the user wants to analyze. This work showed that for German documents the IBM SPSS Modeler is the most suited and that for English documents IBM SPSS Modeler, KNIME and AutoMap deliver good results. RapidMiner, WEKA and GATE Developer can be used, but they don't support the full process from *Data Preparation* to *Visualization*. Besides the functionality the user can take the model-independent facts into account to see which tool fits the best based on accessibility, usability and sustainability.



## References

- [Adă et al., 2010] Adă, I., Thiel, K., and Berthold, M. R. (2010). Distance Aware Tag Clouds.
- [Apostolico and Galil, 1997] Apostolico, A. and Galil, Z. (1997). Pattern Matching Algorithms. *Oxford University Press*.
- [Bikel et al., 1997] Bikel, D., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: a high-performance learning name-finder. *Fifth Applied Natural Language Processing Conf.*
- [Brill, 1992] Brill, E. (1992). A simple Rule-Based part of speech tagger. *Proceedings of the Third conference on Applied Natural Language Processing*.
- [Briscoe et al., 1994] Briscoe, T., Grefenstette, G., Padró, L., and Serai, I. (1994). Hybrid techniques for training hmm part-of-speech tagger. *Technical Report MLTT-007*.
- [Bundeskriminalamt, 2008] Bundeskriminalamt (2008). Trends in Organised Crime. *Organised Crime Situation*.
- [Carley et al., 2013] Carley, K. M., Columbus, D., and Landwehr, P. (2013). AutoMap User's Guide 2013.
- [Chapman et al., 2000] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinhartz, T., Shearer, C., and Wirth, R. (2000). CRISP-DM 1.0 Step-by-step data mining guides. *CRISP-DM Consortium*.
- [Charniak, 1993] Charniak, E. (1993). Statistical Language Learning. *MIT Press*.
- [Church and Hanks, 1990] Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*.
- [Cucchiarelli et al., 2012] Cucchiarelli, A., D'Antonio, F., and Velardi, P. (2012). Semantically interconnected social networks. *Social Network Analysis*.
- [Cunningham et al., 2013] Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. *Computational Biology*.
- [Cutting et al., 1992] Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992). A practical part-of-speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*.
- [DeRose, 1988] DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*.

- [Diesner, 2012] Diesner, J. (2012). Uncovering and Managing the Impact of Methodological Choices for the Computational Construction of Socio-Technical Networks from Texts. *Carnegie Mellon University - Dissertation*.
- [Finin et al., 2005] Finin, T., Ding, L., Zhou, L., and Joshi, A. (2005). Social networking on the semantic web. *Learning Organization, The*.
- [GLODERS, 2013] GLODERS (2013). D2.1 Analysis of content and structure of data sources leading into a shared ontology. *GLODERS Deliverables*.
- [Grefenstette, 1995] Grefenstette, G. (1995). Comparing two language identification schemes. *Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data*.
- [Grefenstette, 1997] Grefenstette, G. (1997). Short Query Linguistic Expansion Techniques: Palliating One-Word Queries by Providing Intermediate Structure to Text. *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*.
- [Grefenstette and Tapanainen, 1994] Grefenstette, G. and Tapanainen, P. (1994). What is a word, what is a sentence? Problems of tokenization. *3rd Conference on Computational Lexicography and Text Research*.
- [Grisham, 1997] Grisham, R. (1997). Information Extraction: Techniques and Challenges. *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*.
- [Hall et al., ] Hall, M., Frank, E., Witten, I., Holmes, G., Pfahringer, B., and Reutemann, P. *The WEKA Data Mining Software: An Update*, volume 11. SIGKDD Explorations.
- [Holmes et al., 1994] Holmes, G., Donik, A., and Witten, I. (1994). Weka: A machine learning workbench. *Proc Second Australia and New Zealand Conference on Intelligent Information Systems*.
- [Holsti, 1969] Holsti, O. (1969). Content analysis for the social sciences and humanities.
- [Impresa, 2007] Impresa, S. (2007). Decimo Rapporto. *Le mani della criminalita sulle imprese*.
- [ISO, 2008] ISO (2008). Ergonomic requirements for office work with visual display terminals (VDTs)– Part 11: Guidance on usability. *International Organization for Standardization*.
- [Jackson et al., 2011a] Jackson, M., Crouch, C., and Baxter, R. (2011a). Software Evaluation: Criteria-based Assessment. *Software Sustainability Institute Guides*.

- [Jackson et al., 2011b] Jackson, M., Crouch, C., and Baxter, R. (2011b). Software Evaluation: Tutorial-based Assessment. *Software Sustainability Institute Guides*.
- [Jungermann, 2009] Jungermann, F. (2009). Information Extraction with Rapid-Miner.
- [Knoke and Yang, 2008] Knoke, D. and Yang, S. (2008). *Social network analysis*.
- [Krippendorff, 2012] Krippendorff, K. (2012). Content analysis: An introduction to its methodology.
- [Kroeger, 2005] Kroeger, P. (2005). Analyzing grammar. *Cambridge University Press*.
- [Krukow, 2013] Krukow, O. (2013). Concept Network Extraction from Text - Methods and Tools for the Research into Extortion Racket Systems.
- [Lovins, 1968] Lovins, J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*.
- [Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval. *Cambridge University Press*.
- [Marbán et al., 2009] Marbán, ., Mariscal, G., and Segovia, J. (2009). A Data Mining & Knowledge Discovery Process Model. *Data Mining and Knowledge Discovery in Real Life Applications*.
- [Mika, 2007] Mika, P. (2007). Social networks and the semantic web. *Semantic web and beyond*.
- [Momtazi et al., 2010] Momtazi, S., Khudanpur, S., and Klakow, D. (2010). A Comparative Study of Word-Co-occurrence for Term Clustering in Language Model-based Sentence Retrieval. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of ACL*.
- [Moore, 2007] Moore, M. (2007). Italy's biggest business: the Mafia. *The Daily Telegraph*.
- [Pisa, 2008] Pisa, N. (2008). Mafia free supermarket defies mob extortion. *The Daily Telegraph*.
- [Pyle, 1999] Pyle, D. (1999). *Data Preparation for Data Mining*.
- [Roberts, 1997] Roberts, C. W. (1997). Methods for Drawing Statistical Inferences from Texts and Transcripts. *Text Analysis for the Social Sciences*.
- [Rose et al., 2007] Rose, C. P., Wang, Y., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., and Fischer, F. (2007). Analyzing Collaborative Learning Processes Automatically: Exploiting the Advances of Computational Linguistics in Computer-Supported Collaborative Learning. *International Journal of Computer Supported Collaborative Learning*.

- [Sager, 1981] Sager, N. (1981). *Natural Language Information Processing*.
- [Sampson and Postal, 2005] Sampson, G. and Postal, P. M. (2005). The 'language instinct' debate.
- [Scott, 2011] Scott, J. (2011). *Social network analysis: developments, advances, and prospects. Social network analysis and mining*.
- [Searle, 1969] Searle, J. (1969). *Speech acts: An essay in the philosophy of language*.
- [Shearer, 1994] Shearer, C. (1994). Mining the data-lode. *Times Higher Education*.
- [Smeaton, 1997] Smeaton, A. F. (1997). Information Retrieval: Still Butting Heads with Natural Language Processing? *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*.
- [Tangenberg, 2007] Tangenberg, J. (2007). Seminar on Counteraction of Extortion Racketeering. *Seminar on counteraction of Extortion Rackets*.
- [Tjong Kim Sang and De Meulder, 2003] Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4, CONLL '03*, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [TRANSCRIME, 2008] TRANSCRIME (2008). The need for an Instrument to Combat Activities of Organised Crime. *Study on Extortion Racketeering*.
- [UNODC, 2008] UNODC (2008). Good Practices for the Protection of Witnesses. *Criminal Proceedings Involving Organized Crime*.
- [van Atteveldt, 2008] van Atteveldt, W. (2008). Semantic network analysis: Techniques for extracting, representing, and querying media content.
- [Voutilainen et al., 1992] Voutilainen, A., Heikkilä, J., and Antilla, A. (1992). A lexicon and constraint grammar of english. *Proceedings of the Fourteenth International Conference on Computational Linguistics*.
- [Wasserman, 1994] Wasserman, S. (1994). *Social network analysis: Methods and applications*.
- [Wilks, 1987] Wilks, Y. (1987). Text searching with templates. *Technical Report ML 162*.
- [Wilks, 1997] Wilks, Y. (1997). Information Extraction as a Core Language Technology. *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*.

[Winter et al., 1993] Winter, A., Ebert, J., Dumsclaff, U., and Mertesacker, M. (1993). Ein Vorgehensmodell zur Software-Evaluation. *Universität Koblenz- . . . .*

[Witten et al., 2011] Witten, I., Frank, E., and Hall, M. (2011). *Data Mining - Practical Machine Learning Tools and Techniques*.

# Appendix

## Evaluation Table

Phase	Main Criteria	Sub Criteria	IBM SPSS Modeler	WEKA	RapidMiner
General	General	Publisher/Developer	IBM	University of Waikato	RapidMiner GmbH
		Country	USA	New Zealand	Germany
	Usability	Extensions	Text Analytics	General (ML)	Text Processing, Information Extraction
		Specialization	General (DM)	Console or Simple UI	General (DM & ML)
	Sustainability & Maintainability	User Interface	Good Node-based	Good	Good Node based
		Documentation	Good, Enormous	Good	Good
		Supported OS	Windows, Linux, UNIX	Windows, Linux, UNIX	Windows, Linux, UNIX
		Support	Contract based Support	Mailing List Archive	Forum
		License	Proprietary Software	GNU General Public License	AGPL / Proprietary
		Accessibility	Commercial	Free Software	Free & Commercial
Data Preparation	Input Format	Interoperability	IBM Cognos	3.6.11 (3.7.11)	5.3 (6)
		Version			
	Language Identification	Flat Files / XLS / XML	Build In	ARFF	Flat Files
		Build In	Fuzzy Grouping		Replace Token
	Error Cleaning				Replace Token
					Sentence / Token
	Anonymization		Text Mining	Word	Build In / Custom
			Fuzzy Grouping	Snowball	Snowball
	Stop Word Filter		Text Mining	Build In	Build In / Custom
			Fuzzy Grouping	Snowball	Snowball
Stemming		EN / DE / FR / IT (7)	EN	EN / DE / FR / IT (16)	
		Tagsets	TagHelper	STSS / Penn Treebank	
Concept Identification	Part-Of-Speech Tagging	Languages	EN / DE / FR / IT + Generic	EN / DE / ES / CN	
		Models	Custom	LibSVM	
	Named Entity Recognition	Languages	EN / DE / FR / IT + Generic	EN / DE	
		Dictionaries			
Relationship Identification	Co-Occurrence	Taxonomies	Similarity		
		Pattern Matching			
	Visualization	Category Network	TF-IDF	Data Tables	
Concept Network Analysis	Social Network Analysis	Concept Map	ARFF / Java Models	Flat Files / Database / Binary	
	Export for Visualization	Format Tool			

Figure 37: Evaluation Scheme

Phase	Main Criteria	Sub Criteria	KNIME	AutoMap	GATE Developer
General	General	Publisher/Developer	KNIME.com AG	Carnegie Mellon University	University of Sheffield
		Country	Switzerland	USA	United Kingdom
	Usability	Extensions	Text Processing	Specialized (NLP)	Multiple Plugins
		Specialization	General (DM & ML)	Specialized (NLP)	Specialized (NLP)
	Sustainability & Maintainability	User Interface	Good Node based	Medium	Medium
		Documentation	Good	Good	Good
	Support	Supported OS	Windows, Linux, UNIX	Windows	Windows, Linux, UNIX
		License	Forum	Contract based	Mailing List / Contract
	Interoperability	Accessability	GNU General Public License	Research Only	GNU Lesser General Public Licence 3.0
		Version	Free Software	Free & Commercial	Free Software
Data Preparation	Input Format	Flat Files / Word / PDF	2.1	3.0.10.36	8
		Language Identification	Java Snippet	Text Files / Web	Text Files
	Error Cleaning	Dict Replacer	Fix Common Typos	TextCat Language Identification	
	Anonymization	Dict Replacer	Word	Sentence	
	Tokenization	Sentence	Text Preparation	Snowball / Porter	
	Stop Word Filter	Build In / Custom	KStemmer / Porter / Lex	EN / DE / FR / IT (11)	
	Stemming	Snowball	EN / DE / FR / IT (16)	EN / DE / FR / IT	EN / DE / FR / IT (11)
		Tagsets	STSS / Penn Treebank	Penn Treebank	Penn Treebank / Custom
	Part-Of-Speech Tagging	Langauges	EN / DE / FR	EN	EN / DE
		Models	OpenNLP	Custom	OrthoMatcher / Custom
Named Entity Recognition	Langauges	EN	EN	EN / DE	
	Dictionaries	Dict tagger	Thesaurus	Gazeteer	
Taxonomies	Dict Replacer	Thesaurus	Thesaurus		
	Co-Occurrence	Term Co-Occurrence	Apply Thesauri		
Relationship Identification	Pattern Matching	Index Queries			
	Visualization	Data Tables / MDS Cloud	Text / XML	Text Overlay	
Concept Network Analysis	Social Network Analysis	Network Nodes	XML	GATECloud	
	Export for Visualization	Flat Files / Database	ORA		
Format Tool	Format				
	Tool				

Figure 38: Evaluation Scheme (continued)