Universität Koblenz-Landau
Campus Koblenz
Universitätsstraße 1
56016 Koblenz

UNIVERSITÄT KOBLENZ · LANDAU

Concept Network Extraction from Text

Method and Tools for the Research into Extortion Racket Systems

Master Thesis

zur Erlangung des akademischen Grades Master of Science (M.Sc.)
an der Universität Koblenz-Landau, Campus Koblenz,
Fachbereich Informatik, Studiengang Informationsmanagement

Eingereicht von:
Oliver Krukow
Steinweg 7
46459 Rees

Erstkorrektor:
Dr. Michael Möhring

Zweitkorrektor:
Prof. Dr. Klaus G. Troitzsch

Abgabedatum: 15. Oktober 2013

Zusammenfassung

Große Mengen qualitativer Daten machen die Verwendung computergestützter Verfahren bei deren Analyse unvermeidlich. In dieser Thesis werden Text Mining als disziplinübergreifender Ansatz, sowie die in den empirischen Sozialwissenschaften üblichen Methoden zur Analyse von schriftlichen Äußerungen vorgestellt. Auf Basis dessen wird ein Prozess der Extraktion von Konzeptnetzwerken aus Texten skizziert, und die Möglichkeiten des Einsatzes von Verfahren zur Verarbeitung natürlicher Sprachen aufgezeigt. Der Kern dieses Prozesses ist die Textverarbeitung, zu deren Durchführung Softwarelösungen die sowohl manuelles als auch automatisiertes Arbeiten unterstützen notwendig sind. Die Anforderungen an diese Werkzeuge werden unter Berücksichtigung des initiierenden Projektes GLODERS, welches sich der Erforschung von Schutzgelderpressung durchführenden Gruppierungen als Teil des globalen Finanzsystems widmet, beschrieben, und deren Erfüllung durch die zwei hervorstechendsten Kandidaten dargelegt. Die Lücke zwischen Theorie und Praxis wird durch die prototypische Anwendung der Methode unter Einbeziehung der beiden Lösungen an einem dem Projekt entspringenden Datensatz geschlossen.

Abstract

Large amounts of qualitativ data make the utilization of computer-assisted methods for their analysis inevitable. In this thesis Text Mining as an interdisciplinary approach, as well as the methods established in the empirical social sciences for analyzing written utterances are introduced. On this basis a process of extracting concept networks from texts is outlined and the possibilities of utilitzing natural language processing methods within are highlighted. The core of this process is text processing, to whose execution software solutions supporting manual as well as automated work are necessary. The requirements to be met by these solutions, against the background of the initiating project GLODERS, which is devoted to investigating extortion racket systems as part of the global financial system, are presented, and their fulfilment by the two most preeminent candidates reviewed. The gap between theory and pratical application is closed by a prototypical application of the method to a data set of the research project utilizing the two given software solutions.

Contents

List of Figures

List of Abbreviations

ACE	Automatic Content Extraction
AI	Artificial Intelligence
CAS	Content Analysis Software
CAQDAS	Computer-Assisted Qualitative Data Analysis Software
CETA	Computer-Assisted Evaluative Text Analysis
CSV	Comma Seperated Values
DM	Data Mining
ERSs	Extortion Racket Systems
GI	General Inquirer
IDE	Integrated Development Environment
IR	Information Retrieval
IE	Information Extraction
KDD	Knowledge Discovery in Databases
KDT	Knowledge Discovery in Texts
KWIC	Key-Word-in-Context
LCA	Linguistic Content Analysis
MDS	Multidimensional Scaling
MECA	Map Extraction, Comparison, and Analysis
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
NTA	Network Text Analysis
PAUM	Perceptron Algorithm with Uneven Margins

PLCA	Program for Linguistic Content Analysis
POS	Part-Of-Speech
SVM	Support Vector Machine
TM	Text Mining
W3C	World Wide Web Consortium

1 Introduction

1.1 Motivation

Extortion Racket Systems (ERSs) are organized groups which try to get money from someone using threat or force. The mafias are the most well known of these groupings, ahead of all the three big Italian mafias Cosa Nostra, 'Ndrangheta, and Camorra. Outside of Italy other mafias exist, such as the Japanese Yakuza, the Chinese Triads or the so called Russian Mafia and Albanian Mafia. Furthermore are ERSs also formed by other types of organisations, e.g. in Germany the biker clubs Hells Angels and Bandidos are said to pursue this scheme. All of these groups are a dominant authority in their root location, but also a considerable economic and financial force, flourishing in the shadows of the legitimate state and meddling in the population's affairs. These groups do not restrict their activities to the territory they control though, but instead act like dynamic enterprises and reinvest in transregional and international markets. Using the funds raised in their illegal activities, ERSs become influential participants in the global financial system.

Glodal **D**ynamics **o**f **E**xtortion **R**acket **S**ystems (GLODERS) is a research project, funded by the European Union's Seventh Framework Programme, which aims at understanding these systems from a global point of view. The universities of Surrey, Koblenz, and Palermo, and the National Research Council located in Rome partnered to form a interdisciplinary team, with distinct competencies in the social and computer sciences. Essential part of the project is the exploration of a considerable number of textual data, with the designated goal to establish theories and hypotheses rooted in the empirical material gathered.

Our intent is to bring texts together with Natural Language Processing (NLP) and Text Mining (TM), in a way that makes purposeful use of a data source that is naturally very effortful to harness because of its inherent intricacy. We want to demonstrate a user-guided and computationally-driven method that combines network analysis, which is a recently blooming field not only in the social sciences, with NLP. The desired result is an extraction and presentation of content from text, which allows true insight into the social phenomenon under study and scales well with the amount of data.

1.2 Objectives, Approach and Structure

The elaboration starts in section 2 with the foundations of text analysis as necessary within the given project setting, whose elucidation is our first major objective. We examine what text is, why we do not only want to, but also have to draw on text as a data source, and what makes text as a data source so special. Then we will introduce TM in the sense of Knowledge Discovery in Texts (KDT) and the closely related fields of NLP, Information Extraction (IE) and Information Retrieval (IR). In order to demonstrate the development of technqiues and tools used fo text analysis in the social sciences, we will discuss the two basic paradigms of the empirical social research, which leads us to the different interpretations of content analysis and the software solutions utilized in its execution throughout the years.

With this background we describe in section 3 how TM and network analysis can be combined to form a method that uses texts to gain and analyze concept networks, which is the second major objective. During this we will depict the procedure, solution approaches for the different phases of the procedure, and the state of the art in the fields of text processing we are benefiting from.

Objective three is covered by section 4, where we state the requirements for software solutions that the depicted process is necessitating against the background of the given social scientific research project for the central phase of the analysis, which is the processing of texts. Looking at two selected solutions from the NLP area, we are subsequently assessing their level of fulfilment.

Using these two solutions, we are demonstrating how the proposed method can be applied to a small-scale assignment under realistic conditions in section 5. This depicts the method's feasibility and gives the opportunity to assess the solutions' capabilites in a practice-oriented analysis, which is objective number four.

In section 6 we are describing our expectations on the evolution of the method of concept network extraction from text and the software solutions that are available conducting this type of analysis, while section 7 tops the thesis off with a review of the major outcomes of the conducted work.

2 Foundations of Computational Text Analysis

2.1 Using Texts as Data

Although one typically has a notion of what the concept 'text' implies, there is no generally accepted definition of it, and data analysts and social scientists alike barely bother to define it. Lacking the sophistication of text linguists' attempts, but sufficient for the given purpose, a text can be characterised as a system of utterances in one or more natural languages, created with a specific intention and written down using the respective alphabet.[1] The type of text is arbitrary. Moreover, our sole interest pertains the informative content of a text, and not its outward appearance or physical form. Text is deemed to be semi-structured, in contrast to the rigid and explicit structure of data contained in relational databases. A deceptive but existing assumption is that texts are unstructured, although they are usually rich in implicit structure [Sánchez et al. 2008, pp. 363–364].

The reasons for using texts as data for analysis purposes are many-faceted. One is that the majority of information available to humanity is stored in texts, and that the availability and accessibility of it has reached unprecedented dimensions with modern information technology and the advent of the World Wide Web. Text's exact share in overall information is subject to discussions, and while unverified estimates range between 80 and 90 per cent a scientifically profound inquiry is still missing [Grimes 2008]. A self-explanatory reason for the usage of texts as data in the social sciences is the impossibility or impracticality of methods of primary data collection, which potentially produce quantitative data, for certain research questions, and thereby the inevitability to rely on secondary data in form of texts. For example are information demands about ERSs scarcely to be satisfied conducting interviews with known members of the illicit organisations, impeded by the nature of the subject.

The disposability of large amounts of data presents the scientist with particular problems concerning their relevancy and validity. Out of the entirety of potentially available information a greater proportion will not fit the research question, and the remainders are uncertain to provide a full coverage of the social phenomenon

[1] A highly respected characterisation of what constitutes a text from a linguist's perspective can be found in [Beaugrande and Dressler 1981]

under study. In addition textual data are prone to bias and error. As a result immense importance is ascribed to data selection and cleansing prior to analysis.

Due to the sheer amount of information it is also imperative to strive for machine support in the process of analysis, since extensive manual work often is prohibitively expensive [Nasukawa and Nagano 2001, p. 973]. But text is written for people, and although there was boundless optimism inside the Artificial Intelligence (AI) community at the end of the bygone century, machines will not be able to understand natural language as humans do in the foreseeable future [Hearst 2003]. Understanding texts requires synchronous processing on different levels of language, namely morphology, lexical, syntactic, semantic, discourse, and pragmatic, and incorporates meta-information as well as knowledge about the world itself [Liddy 2007, pp. 2130–2134]. It is the countless ambiguities, similarities, and dependencies resolved in this process, which make it heretofore impossible to enable machines to even remotely understand a natural language. So processing is restricted to several subtasks, and an isolated task solved for one language may form a massive obstacle in another.

On the other hand electronic texts and their processing have their own wealth of variants and complexity: storage, access and exchange formats, character encodings, mark-up languages, embedded material, metadata and so forth. Those impediments can be dealt with using information science's state of knowledge, but they exacerbate any application of machine-supported text analysis.

2.2 Text Mining

2.2.1 Definition

Giving a precise definition and an elucidating description of Text Mining is an arduous task. Plethoras of specifications differentiate themselves in respect to the identifiers used for the field and the perspectives on it, which shape its primary goals and the role of related areas within.

The most common definition is grounded on interpreting TM from a Data Mining (DM) perspective, as part of the process of KDT as proposed by Kodratoff [1999, 2001], who adopted the term from Feldman and Dagan [1995]. Based on the definition of Knowledge Discovery in Databases (KDD) by Fayyad, Piatetsky-Shapiro, and Smyth [1996, p. 30], KDT can be defined as *the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in texts.* TM often succumbs to the same ambiguity as DM, where the term is confusingly used for the process of KDD as well as for the modelling step within this process, aiming at extracting patterns by application of algorithms [Hotho, Nürnberger, and Paaß 2005, p. 21]. The KDT process is *nontrivial* as it "goes beyond computing closed-form quantities" [Fayyad, Piatetsky-Shapiro, and Smyth 1996, p. 30]. The result of the process is *valid* in the sense that it can be maintained when expanding the underlying pool of data. It is *novel*, i.e. new in its form, at least to the system and the analyst responsible for it. The information gained consists of *patterns*, stemming from a plural of *texts*, denoting that they cannot be deduced from any single piece of data within the collection, but only from their conjunction. Them being *useful* and *understandable* states, that the information resulting from TM can be interpreted by a human in the given context of application and transformed into knowledge qualified to alter action. Not explicitly stated in the definition by Fayyad, Piatetsky-Shapiro, and Smyth [1996] is the indispensable machine support, since the amount of unstructured data to be analysed cannot be processed by a human due to time and cost constraints, which is in turn the main criterion for the application of TM.

2.2.2 Related Fields

KDT is an interdisciplinary field and in order to accomplish its objectives, it makes use of methods and algorithms inherited from KDD and DM, most notably from the fields statistics and machine learning, and augments this fund with techniques from the field of NLP and its two main areas of application, IR and IE (figure 1). The distinction made between KDT and these fields is often imprecise. It is functionally sharp, the respective implementations though resort to a mutual body of methods and algorithms, which can lead to a vast assortment of techniques, applications, and systems being aggregated under TM as an umbrella term [see e.g. Gupta and Lehal 2009].

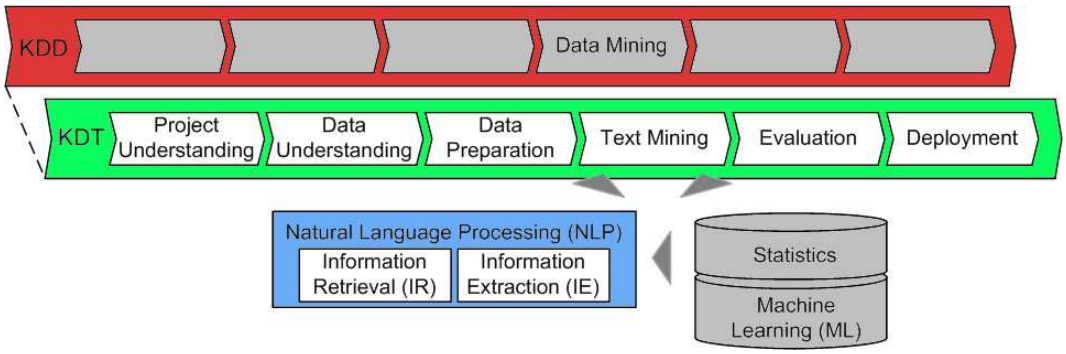

Figure 1: Text Mining and its connection to related areas.

Based on the CRISP-DM framework [Chapman et al. 2000].

NLP is a subfield of AI and offers a range of computational techniques for processing naturally occurring texts, i.e. human speech in oral or written form. Its ultimate goal is the human-like understanding and generation of speech, which would render TM redundant, but that goal is far from being accomplished. This scientific field has its roots in linguistics and is also titled applied computational linguistics [Liddy 2007; Stede 2008].

IR, also termed Information Access and Document Retrieval, is based on the notion of providing users exactly with the information that satisfies their information needs, which they state through natural language queries. The common metaphor for this problem is to look for needles in a haystack [Koll 2000], where it is known that the information exists, but does so by the side of many other

pieces of information [Hearst 1999, p. 3]. Solutions to this problem, as embedded in library systems and web search engines, effectively do not provide answers, but rather documents that correspond best to the key-words given by a user, based on the assumption that they are likely to contain the answer [Kodratoff 1999, p. 20]. Although expected to increase performance, only few implementations make use of deeper NLP [Liddy 2007]. IR technology can be utilized in the KDT process to collect and select data.

IE is concerned with analysing natural language texts to map contained pieces of information into a predefined, structured representation [Sánchez et al. 2008, pp. 3–4]. The aspired elements, word compositions or other parts of text, such as names, numbers, dates, and addresses, are extracted to meet a known information need. IE techniques are seldom applied in isolation, but rather as part of a range of applications where they are used to distil existent texts to tagged portions of the original [Liddy 2007]. IE technology is utilized in the KDT process to identify and extract key features of the data, which constitute the dimensions of analysis for subsequent modelling [cf. Feldman et al. 1999; Ben-Dov and Feldman 2005].

The use of NLP in KDT is often limited to shallow techniques, since the amount of texts to be processed constrains the complexity of algorithms, and semantic resources are often not available for the specific field of application [Rajman and Besançon 1997, p. 3].

2.2.3 Differentiation and Perspectives

Besides the presented perspective on TM from a DM standpoint, there are some academics who consider it to be a form of advanced IR, as it ”leaps from old-fashioned information retrieval to information and knowledge discovery” [Dörre, Gerstl, and Seiffert 1999, p. 398]. It is also interpreted as a form of IE for the purpose of advanced IR [Göser 1997, p. 3; Sullivan 2003, p. 99; Ananiadou et al. 2009, p. 1]. Drawing a distinction, the outcome of the IR process are documents to be read, while the results of TM are patterns, connections, and trends to be interpreted [Ben-Dov and Feldman 2005, p. 804]. IE is concerned with bringing structure to isolated pieces of information contained in texts, but without deriving new knowledge [Sánchez et al. 2008, p. 366], which by contrast is the stated aim of KDT.

In acknowledgement of the different perspectives, Mehler and Wolff [2005] differentiate two extremes which span the range of perspectives in terms of novelty of the knowledge gained. On the lower end of the spectrum are method-oriented perspectives, which see TM as a bundle of methods dealing with texts in order to enhance or substitute IE or IR, enabling humans to explore the information contained. On the upper end of the spectrum are knowledge-oriented perspectives, which demand knowledge creation and the direct discovery of "new trends and facts about the world itself" [Hearst 1999, p. 8] with only minimal human interference. The later extreme is also termed Intelligent Text Mining [Kroeze, Matthee, and Bothma 2003] or Text Knowledge Mining [Sánchez et al. 2008].

2.3 Empirical Social Research

2.3.1 The Need for Characterisation

Empirical social research is the systematic ascertainment and interpretation of social phenomenons, based on experience and data gained through observation or experiment. This science is dominated by two paradigms: quantitative and qualitative research. An adequate characterisation of the paradigms is a prerequisite for the work in hand, not only because empirical social research is the breeding ground for at least part of the methods and tools covered here. But also because someone entering the field of social scientific research from a text analysis perspective will constantly be confronted with the terms quantitative and qualitative. Unfortunately these terms are used ambiguously in the literature and among practitioners in the domain, referring not only to the paradigms underpinning empirical social research, but also to various components of a scientific endeavour, such as complete research plans, methodologies, methods, data, and tools.

The quantitative approach to empirical social research tries to equal the natural sciences and defines itself in this sense, while the qualitative approach is primarily defined in distinction from the former [Garz and Kraimer 1991, p. 1]. Attempts to define both approaches on basis of their methodology are insufficient up to this point [Wolf 2008, p. 7]. More often the distinctions of the approaches are established on different levels, with inclusion of both paradigms' extremes into the differentiation [Flick 2009, p. 24].

2.3.2 The Contrasts

The quantitative approach starts from the premise that the characteristics of social phenomenons can be measured or counted and analysed statistically. It is deductive and aims to generate generalisable declarations about causal relationships within the population [Seipel and Rieker 2003, p. 13], which is its declared strength. The main quality criteria of quantitative research are objectivity, reliability, and validity [Brühl and Bruch 2006]. Therefore the process of research is linear, standardized and constructed in an attempt to control the conditions of the research and eliminate confounding factors [Kromrey 2007], to which also the influence of the researcher itself belongs.

Until the 1960s the quantitative approach dominated, but then the critique of it increasingly helped establish qualitative research as an equal practice [Steger 2003, pp. 6–7]. Critics argue that the characteristics and categories used in quantitative research do not live up to social reality, since they only deliver data within the predetermined scheme [Heinze 2001, p. 65]. Abstraction and abbreviation distort the created representations of the objects of study. Kromrey [2007, p. 540] even claims, that the practice of quantitative research shows, that reliable results can only be obtained if the research question is narrowed down so heavily, that it becomes meaningless.

Qualitative research covers a variety of approaches and methods [Steger 2003, p. 1], and attempts to structure and classify them are numerous [Creswell 2006, pp. 6–9]. They all share the tradition of inductive reasoning tracing back to Aristotle [Rost 2003, p. 9], aiming to develop new theory through research [Heinze 2001, p. 16].

While quantitative research tries to measure the known, qualitative approaches explore social reality and reveal connections [Kleining 1982, p. 227], claiming to do so less biased and more thoroughly [Heinze 2001, p. 65]. Guided by a wider research question, smaller samples are covered holistically and in detail [Wolf 2008, p. 7]. The research process is open, flexible, and circular, allowing the researcher both interaction with the object of study and subjective interpretations of the collected data.

A consistent and universally accepted conception of quality criteria of qualitative research is non-existent [Seipel and Rieker 2003, p. 131]. Steger [2003, p. 16] divides the advocated variants into four categories. Worth mentioning are the positivism point of view, whose representatives apply the same criteria as in quantitative research, and the post-positivsm point of view, which strictly denies the applicability of the very same criteria. The main critique of qualitative research is, that its findings are not generalizable, which is a basic requirement for scientific research. Qualitative researchers though would disagree, although not claiming that the investigated population or the situation of the examination are representative but rather concentrating on the specific case [Heinze 2001, pp. 44–45].

2.3.3 The Present State of Practice

The controversy between the two paradigms exceeded its culmination, and in the newer literature the contradiction is rebutted [Steger 2003, p. 3]. Forcing all endeavours in the empirical social sciences into a dichotomy does not meet the requirements of its practical application, and disregards existing commonalities [Heinze 2001, p. 28]. With the researcher's aim to orchestrate a selection of methods most adequate to the research object and question [Flick 2009, p. 33], a combination of qualitative and quantitative methods may do the task best [Flyvbjerg 2006, p. 242]. Theoretical approaches of this character are by now manifold as well, covered by umbrella terms such as triangulation or mixed method approaches. The maturity of this endeavour though is seen critically, as Flick [2009, p. 30] states that the development of "really integrated qualitative/quantitative methods of data collection or data analysis remains an unsolved problem". However one may decide to orchestrate the methods for his research, his maxim should be to execute his endeavour in adherence to the most basic principles of relevancy and rigor.

2.4 A Review of Computer-Assisted Text Analysis in the Social Sciences

2.4.1 Introduction

The objectives of text analysis applied for research within the social sciences are to describe, understand, and explain social behaviour, values, structures and norms [Alexa 1997, p. 4]. The forms of, and methodologies surrounding these analyses are numerous and hard to comprehend. Since there is no single approach which suits all kinds of text analysis, they all gained their right to exist within their original domain.

Content analysis is one of these approaches, suited not only for the analysis of primary but also secondary data [Harris 2001, p. 201]. The origin of content analysis as a method is accredited to Max Weber, who proclaimed its usage for the analysis of the content of communication in press [Mayring and Brunner 2009, p. 672]. Although content analysis is essentially just one of a multitude of methodologies, it became the prevailing method of text analysis within the social sciences throughout the years [Carley 1993, p. 77], and so in this domain the term is often used interchangeably with quantitative text analysis [Alexa 1997, p. 4]. At the same time content analysis has become a fuzzy area, incorporating manifold methods and techniques [Alexa 1997, p. 11]. This is reflected in the definitions of it: while Berelson [1952, p. 18] sharply defines content analysis as ”a research technique for the objective, systematic and quantitative description of the manifest content of communication”, Shapiro and Markoff [1997, p. 14], in a review of existing definitions 45 years later, propose a minimal definition of content analysis as ”any methodical measurement applied to text (or other symbolic material) for social science purposes”, thereby including qualitative approaches.

It is obvious that the quantitative-qualitative controversy, as introduced in the previous chapter, also impacted the shaping of content analysis. There are positions which claim that content analysis is strictly quantitative [see Silverman 1993, p. 59], those that make use of content analysis as a qualitative method [Mayring 2000], and those that acknowledge that the distinctions are blurring, so that ”the content analyst should use qualitative and quantitative methods to supplement each other” [Holsti 1969, p. 11].

In accordance with the epistemological development, two categories of software tools used to foster the analysis of texts can broadly be distinguished in the social sciences, and content analysis heavily influenced both. Qualitative oriented tools are referred to as Code-and-Retrieve Programs and Code-based Theory Builders or Annotation Aids, and constitute the majority of the tools nowadays commonly classified as Computer-Assisted Qualitative Data Analysis Software (CAQDAS), while quantitative oriented ones are referred to as Text Retrievers and Textbase Managers, Dictionary-based Content Analysers or simply Content Analysis Software (CAS) [Weitzman and Miles 1995; Evans 1996; Lowe 2003; Lewins and Silver 2009]. At the core of both categories' tools lies the expectation to augment the efficiency of the analysis process, but while CAQDAS mainly assists the manual work of the researcher, CAS seeks to automate at least part of the tasks [Alexa 1997, p. 8].

In order to reconstruct the development of text analysis software until the beginning of the last decade, we refer to the scheme as outlined in figure 2. Regarding the three quantitative approaches, "in a thematic text analysis one examines occurences of themes (or concepts), in a semantic text analysis the examination is of sentences (or clauses) in which themes are interrelated, and in a network text analysis it is of themes' and/or sentences' locations within networks of interrelated themes" [Roberts 1997, p. 3]. Züll and Landmann [2002, p. 10] subsume three areas of CAS, namely dictionary-based, co-occurence based, and network or relationship approaches, while Klein [1997, p. 355] splits software for quantitative content analysis into thematic and clause-based types, with a distinction within the later class between evaluative approaches and those that produce cognitive maps. There are further attempts to systematize software for text analysis, e.g. Weitzman and Miles [1995], which also include a broader range of tools. We knowingly disregard tools designed to partially automate the construction of dictionaries and grammars [Lowe 2003, p. 1], tools that offer features for textual exploration [Züll and Landmann 2002, p. 10], and text retrieval tools and word processors at this

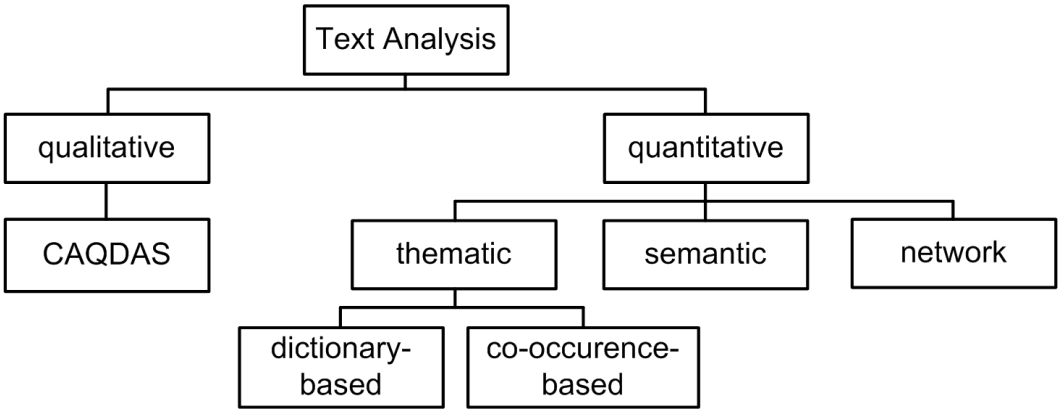

Figure 2: A schema of computer-assisted text analysis in the social sciences.

point. Compiling a comprehensive list of available software is beyond our ambition, so that reference is made only to outstanding examples[2].

2.4.2 CAQDAS

CAQDAS is defined by Lewins and Silver [2009, p. 3] as an umbrella term which includes wide ranges of packages, whose "general principles are concerned with taking a qualitative approach to qualitative data". Data initially almost exclusively was, and today still mostly is, texts, but increasingly other types, such as audio, images, and video, are also supported, which makes the packages multimedia compatible [Kuckartz and Rädiker 2010, p. 745]. The development began in the 1980s, with the most prominent representatives today being ATLAS.ti, NVivo, MAXQDA, and NUD*IST. The packages help users to manage and explore their data, to create hierarchic code schemes and to assign these codes to data segments. The user can carry out searches in, and make amendments and assign meta-data to the text. The process of interpretation is supported by supplying means to write memos and comments, by visualizing the results of coding in manifold ways, and by creating visual links between memos, codes, and texts, all within one environment. CAQDAS also aims to support the work of teams by mechanisms to export and merge projects [Lewins 2001; Zhang and Wildemuth 2009, p. 8; Kuckartz 2010;

[2]Besides the works cited in this section, the writings of Mohler and Frehsen [1989], Weber [1990], and Mehl [2006] provide further information on the large array of available software and their applications.

Schutt 2011, pp. 350–353]. By now the support of analyses is so extensive, that a discussion arose whether CAQDAS is still aligned to the principles of qualitative methodologies or already turned into a methodology in its own right, limiting the researchers' interpretive efforts when utilized without proper reflection [Atherton and Elsmore 2007; Kuckartz and Rädiker 2010, pp. 735–736].

2.4.3 Thematic Analysis

The development of software for quantitative content analysis began in the 1960s, with the 'Content Analysis Conference' hosted by The Annenberg School of Communications of the University of Philadelphia in November 1967, as a catalyser for many of the subsequent innovations. After a stagnation of the development in the 1970s and 1980s, it intensified in the 1990s due to increasing availability of access to computational facilities for researchers, caused by the spread of the personal desktop [Züll and Landmann 2002, p. 9].

The General Inquirer (GI), created by Philip Stone [Stone 1966], is considered to be forefather of CAS and prime example of the dictionary-based approaches. Beyond merely creating list of word (co-)occurrences, it is capable of classifying words, combinations of words (idioms), and sentences by means of a pre-defined dictionary. The GI can be used with any dictionary, but its possibilities often were equated with the utilisation of the Harvard III dictionary, which was part of the software. It also provides suffix-chopping, which is a simple form of stemming, and a routine based on hand-crafted rules to identify homographs and idioms before assigning tags, and thereby a sort of context-sensitive disambiguation [Psathas 1969].

A helpful distinction between CAQDAS and the dictionary-based approach of content analysis as a quantitative method can be made when looking at the process of coding or tagging, i.e. the creation of categories and the assignment of categories to text segments. This process is agreed to be central to all systematic qualitative approaches [Schutt 2011, p. 350; Kuckartz and Rädiker 2010, p. 739], and "it is acknowledged that content analysis stands or fails depending on the quality of its categorization scheme" [Alexa 1997, p. 16]. Those CAS, that follow a dictionary-based approach, support a linear process in which the first two steps, the a-priori

and deductive creation of categories, and the establishment of links between those categories and the manifest target content by creating a dictionary, are carried out manually, and the final step of assigning codes to content by means of the dictionary is automated [Wood 1980, p. 276]. This approach, which is also called a theory-driven or top-down approach, is also feasible using CAQDAS, but those software applications do not take over the assignment of codes. The common approach in using CAQDAS is to inductively derive categories from the manifest and latent content, reversing that linear process and running through it circularly, with all the tasks carried out by hand.

The dictionary-based approach still is well-liked, and over the years a range of tools has been developed, within different scientific fields for specific research questions, for numerous text types and with varying sophistication. Among them are TEXTPACK [Olsen 1989], INTEXT [Klein 1991], and also TACT [Hawthorne 1994], DICTION [Hart 1985], and LIWC [Pennebaker, Francis, and Booth 2001; Pennebaker et al. 2007]. Common features of these CAS packages are word lists or indexes, category frequency counts represented by matrices and means for their analysis and visualizations, calculation of vocabulary-based measures, export of results for further processing with statistical tools, Key-Word-in-Context (KWIC) lists, concordances, and internal dictionaries and thesaurus facilities [Evans 1996, p. 6; Lowe 2003, pp. 1–4; Lewins 2001, p. 304; Diefenbach 2001].

WORDS [Iker and Klein 1974] is the first noteworthy CAS, or rather a collection of program routines, that makes use of word co-occurrences to describe texts by clusters of associated words [Wood 1980, p. 277]. The approach is data-driven, which means that all semantic units used to describe texts stem from the texts themselves, so no prior construction of a dictionary is needed [Diefenbach 2001, p. 25]. The procedure used basically divides an input text into segments, i.e. paragraphs, sentences or a fixed number of words, determines the frequency of words within each segment and then creates an intercorrelation matrix by calculating the covariation between each and every word. Covariations range from -1 to 1, whereby a high negative value indicates the presence of one word associated with the absence of the other, whereas a high value positive value indicates the conjunct occurence of the two words [Alexa 1997, pp. 21–22]. The intercorrelation matrix "is then reduced by multivariate procedures (factor analysis, cluster analy-

sis, etc.) to locate, in a systematic fashion, the presence of common word groups" [Iker and Klein 1974, p. 430]. Visualization of the result is to be attained using external programs.

VBPro in combination with VBMap, which form the implementation of concept mapping [Miller and Riechert 1994], and CATPAC [Doerfel and Barnett 1996] have the same approach as WORDS. In addition to fixed-size windows, CATPAC is able to count co-occurrences in sliding windows, so that a word is associated to another word if it occurs within X words to the left or right of it [Evans 1996, p. 273]. A slight variation of VBPro is its use of the Chi-square statistic, and thereby relative frequency of words among texts, to determine which words are included in subsequent analysis, and cosine similarity to produce an intercorrelation matrix [Alexa 1997, p. 24]. VBPro uses the largest three vectors from this matrix to project the words into two-dimensional space, while CATPAC applies Multidimensional Scaling (MDS) in two or three dimensions for output and external visualisation.

2.4.4 Semantic Analysis

The semantic analysis of texts encodes not only the concepts, but also the relationships that link them, which are manifest in the grammatical relations of the natural language expression [Roberts and Popping 1993, p. 659]. A relationship, just like a concept, can be represented by a single word or a set of words. Coding these relationships is often achieved by applying semantic grammars and converting clauses into nuclear sentences or statements with an Subject-Verb-Valance-Object (S-V-V-O) form. In most cases the valence information is part of the verb component [Popping 2000, p. 29], so that the semantic grammar is reduced to a S-V-O form. Approaches that make use of this encoding method are brought forward by Franzosi [1990], who applies a semantic grammar consisting of subject/action/object units (semantic triplets) to newspaper reports, and Roberts [1989], whose Linguistic Content Analysis (LCA) codes the intended relations among words in various socio-temporal contexts. His approach thus is representational as opposed to instrumental analyses, where the researcher's theory is systematically applied to interpret the semantic of a text. Coding produced using LCA can be validated with the use of the Program for Linguistic Content Analysis (PLCA), which re-

constructs the clauses as coded [Roberts and Popping 1993]. The parsing required to apply semantic grammars can only be performed by humans [Franzosi 1990, p. 232], a certain extent of reliable automated coding can only be achieved when applied to texts with a very limited variety. Examples are a parser used for the Gottschalk-Gleser Method of Content Analysis, reported to operate with 60% accuracy [Wood 1980, p. 280], and a parser for the Kansas Event Data System (KEDS), reported to have a 90% rate of accuracy.

2.4.5 Semantic Network Analysis

The semantic analysis of texts subsumes the thematic approach, since the identification of themes is necessary before their relations can be established. Likewise semantic network analysis subsumes semantic analysis, and thereby also thematic analysis. A semantic network is assembled from statements extracted through semantic analysis, so that equivalent themes or concepts form the nodes and relations are equal to arcs [Roberts and Popping 1996, p. 658]. The result of a thematic analysis can be interpreted as a net without arcs, to which a semantic net can be reduced if required [Diesner and Carley 2010, p. 580]. Thematic and semantic analysis comprise necessary steps in semantic network analysis, but need not be conducted separately. Research in the field of semantic network analysis is mainly driven by the groups of Carley and van Cuilenburg [Popping 2003, p. 6].

Carley [1986, 1988, 1993] developed her map analysis approach to analyse the cognitive maps of participants in a sociological study at MIT. Within a semantic network as defined by her, concepts can be hierarchically classified or typed, but carry no meaning, except that they are connected to other concepts. According to Carley, relationships can have strength, directionality, sign and meaning. The approach is supported by the Map Extraction, Comparison, and Analysis (MECA) toolkit, which includes programs to assist in the definition of the network's characteristics and the coding of statements by means of dialogues tailored to the given definition of the network, to amend and complement compiled networks based on expert knowledge codified, and to calculate intersections, differences and unions of networks for comparison.

Cuilenburg, Kleinnijenhuis, and Ridder [1986, 1988] applied their approach, which is a generalization of Charles Osgood's Evaluation Assertion Analysis, to evaluative texts in communication science, i.e. journalistic articles. The relationships they use are simpler than Carley's, as they are all of one type or share one meaning, are always unidirectional, and use a combined indicator for sign and strength. Mathematical graph theory is used to reduce a sequence of statements to a single relation between two concepts, which transforms implicit into explicit links. Their Computer-Assisted Evaluative Text Analysis (CETA) 2 toolkit shows remarkable similarities to MECA, since it also assists the coding process in a dialogue-style, uses an inference engine to amend the compiled network, and supports analysis by calculating indices concerning the relationship between concepts.

Popping [2003] discusses some issues in the conception of semantic networks when used as knowledge graphs, namely implicit and exclusive knowledge, strength of relations, and unraveling of broad concepts. Diesner and Carley [2010, p. 512] state that relationships can be enhanced with further attributes, such as time or place of validity, and Popping [2003, p. 100] suggests to augment networks with inclusion or exclusion criteria for the sake of validly joining them.

The drawback of the semantic and semantic network approaches is that the coding of relationships is even harder to automate than the coding of concepts, and thus relies heavily on time-consuming manual coding [Cuilenburg, Kleinnijenhuis, and Ridder 1986, p. 90; Carley 1993, p. 102]. Facilitation of automatic coding can be achieved when relations are not semantically coded as described above, but with employing a proximal, or a temporal or sequential approach [Carley 1993, pp. 105–108]. The downside would be a loss of information on, or the coding of non-existent or even contradistinct relations.

2.4.6 Automation and Software Evolution

In general, quantitative methods are less elaborate than qualitative ones, and since they are automatable to a greater extent, they are potentially more useful for big amounts of texts. Perfect intercoder reliability is achieved if the coding is fully automated, but this is seldom the case for more sophisticated semantic and semantic

network approaches. And if so, the programs are restricted to one particular type of text and score below human capabilities in terms of accuracy.

The distinctions made between software packages supporting a certain methodology are rather artificial nowadays. Not only are the distinctions between the demonstrated types of quantitative analysis programs blurring since the sophisticated ones can benefit from their predecessors. The increasingly popular CAQDAS tools also blur the distinctions between the quantitative and qualitative oriented packages [Schutt 2011, p. 312]. Basic quantitative functionalities, such as word and category counts, co-occurrence matrices, and export of quantifiable results for statistical analysis, are incorporated by CAQDAS developers since they are wanted by the users for exploratory purposes [Kuckartz 2010, p. 250; Kuckartz and Rädiker 2010]. At the same time, the graphical depiction of semantic networks allows for a fairly qualitative interpretation of results initially designed for quantitative analysis.

3 Extracting Networks of Concepts from Text

3.1 The Process of Analysis

The outlined approach aims at extracting mentions of concepts and their relations from text, in order to enable a social scientist to investigate the phenomenon of ERSs by analysing those concepts and relations and the networks formed by them. It brings together network analysis, which, as depicted earlier, is the latest movement of text analysis in the social sciences, with NLP methods and tools, forming an approach which can be labeled KDT, since it aims to generate new knowledge from texts, making use of core IE techniques. In the remainder of this thesis, *a concept is an abstract entity that is manifest in a text as a linguistic item of arbitrary extent, being the author's expression of a single piece of the reality under investigation.* Examples of concept expressions in text, i. e. text-level concepts, are ”Luciano Varese”, ”restaurant Da Bruno” or ”demand a pizzo”.

The approach is a form of Network Text Analysis (NTA), but does not result in a purely semantic network, since the concepts can carry meaning beyond their connections to other concepts, which is contrary to the traits of a semantic network [Sowa 1992]. A five-phase procedure outlines the approach followed hereafter, designed to fill a data model as depicted in figure 3. The later sections concentrate on the three central phases of text processing, namely concept identification, relationship identification and reference reconciliation.

The five phases are:

1. Concept Ontology Development
   Define an ontology that captures the structure of knowledge as deemed relevant for the intended analysis. Describe the elements of the ontology and specify rules for mapping text-level apparitions to these elements.

2. Concept Identification
   Identify all text-level concepts assignable to one of the defined concept classes or attributes.

3. Relationship Identification
   Identify all relationships between text-level concept assignable to one of the defined concept class or attribute relationships.

4. Reference Reconciliation
   Consolidate identified concepts and assign a canonical representation for each set.

5. Concept Network Extraction and Analysis
   Extract all consolidated concepts and relationships and merge them to form a network. Analyse the network using visual depiction, statistical or graph-based measures.

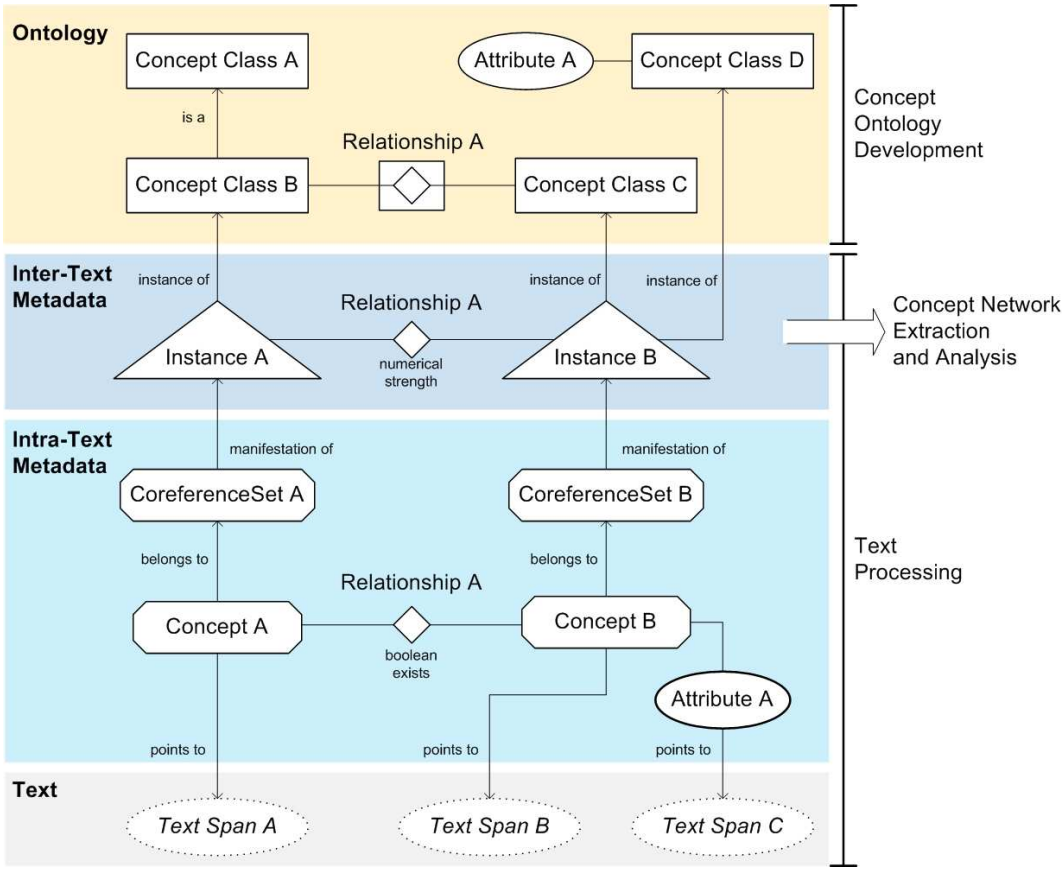

Figure 3: A data model of concept network extraction from text and the processing phases working on its segments.

Note that this outline assumes that both the research questions, and thereby the goals of the analysis, and the data are present in advance. Moreover the process is not strictly linear, but will rather involve iterating through it and falling back to previous phases until satisfactory results are achievable. All phases generate results which are valuable beyond being input to the succeeding phases. The main challenge is to efficiently and reliably extract the network while dealing with the shortage of appropriate ground truth to validate the extracted network [Diesner 2012, p. 7] and the knowledge gained form its analysis. Under these circumstances we try to shed a light on how to make use of computer support and automation in the given phases.

3.2 Concept Ontology Development

3.2.1 Ontology Structure

The first step in extracting and analysing networks of concepts from texts is to determine what exactly is to be extracted, dependent on the insights striven for. This implies creating an *ontology, which is an explicit and formal definition of a set of representational primitives, typically classes, attributes, and relationships, with which to model a domain of knowledge* [Gruber 2009]. To be more precise, we want to create a theoretical framework for a chunk of the total knowledge of a domain, built according to a current problem [Even and Enguehard 2003, p. 3], primarily as input for the later phases but also for the purpose of reaching consensus within a group about the task at hand.

Creation of an ontology might seem a trivial task, the decisions made in the process though are not to be treated lightly. The model of language resulting from the application of an ontology is inherently incorrect, but nevertheless an ontology must trade off expressive power against effort to spend for its application. Expressive power induces complexity, which can be broadly divided into two kinds: structural and contentual complexity.

Structural complexity is determined by the number of component types and components in modelling the ontology. Concept classes and the relationships between them are the essential basis of an ontology and correspond to nodes and edges in out final network. In order to reduce structural complexity, all relations

are to be defined on a pair of concept classes $\langle a, b \rangle$. Text-level concepts can be assigned to more than one concept class, but the conception of the ontology should aim to prevent such cross-classifications.

Hierarchical classification of concept classes increases complexity. Conceptually there is no limit to the number of levels that could be added to the hierarchies of concept classes for finer levels of analysis [Diesner and Carley 2005], due to the unlimited number of possible dimensions along which to develop subcategories [Chandrasekaran, Josephson, and Benjamins 1999, pp. 22–23]. Concept class hierarchies imply taxonomical relations between concept classes, such as is-a and part-of, in addition to the basic associative relationships that connect concepts across the tree structures [Stevens, Goble, and Bechhofer 2000, p. 400]. Note that the relations used here do not only describe the static condition of the domain, but also their dynamics, the possible interactions of concept classes, which is contrary to many perceptions of ontologies.

An attribute is another component type often used in ontologies, consisting of an attribute-value pair attached to concept classes, or infrequently also to relationships, adding factual detail to the respective component. Attributes and associative relationships are inherited by the subclass in a taxonomical relation, i. e. by the hyponym from its hypernym. Attributes are only useful for our purpose in so far, as they can be used to filter concepts and relationships when only a subset of the whole network identified is to be extracted for analysis.

There are further component types which can be used to design an ontology, such as axioms and cardinality of relations [cf. Stevens, Goble, and Bechhofer 2000, pp. 400–401], which are not elaborated here. For the given purpose structural complexity would needlessly increase by using additional component types.

While some decisions in modelling seem natural, some are ambiguous and offer a greater freedom of choice. For example, the fact |Person born-in Nation| could be modelled as a subclass of the concept class Person having the designated nationality, a directed relationship between the concept classes Person and Nation with the semantic born-in, or an attribute bornIn:Nation attached to the concept class Person. Those choices solely depend on the domain and the analysis one intends to use the extracted concept network for [Noy and McGuinness 2001]. For example, if we would want to calculate a graph-based measure, the second option

is preferable, but only if the relation born-in is useful for establishing a connection between a node representing the instance of the concept class Person and another node linked to the node representing the instance of the concept class Nation [Barthélemy, Chow, and Eliassi-Rad 2005]. Let us illustrate this with the following example: When Guiseppe and Toni are both born in Italy and this fact is qualified to establish a connection between them, then the relationship is suitable for a graph-based approach to analysis and the second option is a good fit, in contrast to the other two options with whom the connection would remain undetected. This is a reason why attributes are generally to be employed carefully when constructing an ontology for the purpose of extracting a concept network.

Contentual complexity is determined by the components' coverage of text. For each representational primitive, it finds expression in the number of lexical items making up its linguistic realisations, the variety of these items, and their coherence. It influences the clarity of identifying matches and determining their boundaries in text. Choosing a fitting granularity is key to establish an ontology which is expressive enough for the task at hand without evoking too much contentual complexity. While both structural and contentual complexity influence the cost of application, it is mainly the latter which determines what degree of automation is achievable in identifying the ontology's corresponding text-level instances. Some components might just be too general to be treated by an IE component or identified in any other automated manner.

3.2.2 Development Process

There are numerous methodologies used in ontology development studied in the field of ontology engineering [Fernández-López 1999; Dahlem and Hahn 2009]. Among the more mature ones are METHONTOLOGY [Fernández-López, Gómez-Pérez, and Juristo 1997] and the Unified Process for Ontology (UPON) [De Nicola, Missikoff, and Navigli 2009], but at the same time the more mature ones are rather rich in policy. Since we are presumably geared towards small-scale ontologies, an appropriate development process is to be established, avoiding any overbearing specifications. A suitable methodology can be derived from the guidelines by Noy and McGuinness [2001] and the skeletal methodology by Uschold and Gruninger

[1996]. The goals and the usage scenario of the ontology are prescribed by the research and the purpose of extracting a network of concepts from text.

The first step in the development therefore is the definition of the scope of the ontology, which is the major factor of influence on its complexity. It specifies what is to be included and what not, minimizing the amount of concepts to be analysed [Brusa, Caliusco, and Chiotti 2006, p. 8]. A set of informal competency questions, capturing requirements in natural language questions that the ontology must be able to answer, are a useful tool to assist in setting the right scope [Grüninger and Fox 1995].

The second step is re-use of ontologies. Existing ontologies can be refined, enhanced and integrated as a whole or partially to form the targeted ontology. It does not matter in which formalism an ontology exists, since translating an ontology from one formalism to another is usually not a difficult task [Noy and McGuinness 2001, p. 6]. However, with growing size of ontologies their re-use becomes a costly endeavour. Besides self produced ontologies for previous extraction tasks, ontology libraries accessible via the world wide web can provide utilisable material.

The third step is capturing the components, describing these elements unambiguously and specifying which text-level concepts qualify as instances of these components. Determining the necessary components can go from most abstract to concrete (top-down), from most concrete to abstract (bottom-up), or from most important to more abstract and more concrete (middle-out) [Uschold and Gruninger 1996, pp. 20–22]. It is advisable to follow the latter and start by listing key terms, then derive the concept classes and their hierarchy from it, and afterwards establish the attributes and relations and their respective properties. Determining the key components can follow a theory-driven, a data-driven, or a hybrid approach, which is the most apt variant and includes alternating between formulated ontology, theoretic demands and a set of texts. Description of the components are to be precise and comprehensible, which includes giving examples where appropriate, particularly of relatable text-level concepts. For the use of text analysis one may also incorporate synonyms and Part-Of-Speech (POS) tags or other NLP related information [Noy and McGuinness 2001, p. 6] already available at this phase in order to accelerate the identification process.

In the fourth step the ontology is to be coded, that is using a formal language to define what has been captured in a non-formal or semi-formal manner in step three. Two intertwined decisions must be made: on the primitives used to represent the ontology, also called meta-ontology [Uschold and Gruninger 1996], and on a representation language capable of supporting these primitives. We have already elaborated on the former in section 3.2.1. Considering the latter one can choose from a range of ontology languages, the most prominent at the time of writing being OWL 2, endorsed by the World Wide Web Consortium (W3C) for the semantic web and primarily exchanged in the format RDF/XML [W3C 2012]. All ontology representation languages are based on the formal system of a logic such as first-order logic or description logic [Yildiz 2007, p. 11]. Capturing and coding of the ontology can be carried out concurrently, especially for the given purpose and against the background of contemporary ontology editors, which allow definition and modification with the aid of visual representations. Thereby an intermediate representation is skipped in favour of a swifter development process.

The whole process goes along with a documentation of all the decisions and results and their rationale. Further common steps in ontology engineering would be to populate the ontology with instances and evaluate it by means of application. Both activities are postponed to later phases of concept network extraction and are thus not part of the ontology development phase.

3.3 Concept Identification

3.3.1 Introduction and Result Conservation

In the phase of concept identification the text-level concepts that refer to the defined concept classes and attributes are identified. In a strict sense, all of these ontology components must be mapped to the linguistic phrasings of their instances, but some components, especially concept superclasses, might initially only have been added for the sake of comprehension or reaching consensus. These are to be omitted in the identification phase, which makes the ontology used for identification a subpart of the prior defined structure. The result of identification can best be understood as mapping between the two types of ontology components and text-level concepts. Applied storage of this mapping can broadly be divided

into two categories: on the one hand detached from the texts (stand-off annotation), making reference to the positions within a text by means of an index, which allows exact identification of every character the text is made of. An example of this option is the pioneering TIPSTER format [Grishman et al. 1998]. And on the other hand by inserting inline markup (inline annotation), often using an XML or XML-like format which exactly encloses the expressions, such as the format defined in the TEI guidelines [Consortium 2013]. Since XML is a widespread standard format, this option has the advantage that the respective texts can be utilized by many software tools, and also text and annotations can be visualized with ease. However, XML essentially is a tree-based model, and thus tag cross-overs fail to adhere the standard [Cunningham, Maynard, Bontcheva, et al. 2013]. For example, the annotation

<Person>Bruno <Place>Perone's</Person> Palace</Place>

would constitute invalid XML. Moreover, if annotated elements are numerous, lengthy or multi-part, inline annotations are hard to keep track of, at least for human viewers. The more information is stored in combination with an annotation, e. g. related to the annotator or the annotation process, the heavier this observation weighs.

3.3.2 Text-level Identification

The approaches to concept identification range from fully manual to fully automatic, though realistic scenarios will be hybrid approaches, located somewhere in between the two extremes of the spectrum. The reasons for not solely utilizing fully automatic identification are simple: for some tasks the present state of technology is not capable of reaching acceptable performance, whilst for others the effort needed to reach acceptable performance is too big to be justifiable in consideration of the benefits derivable from the automation.

Proper improvement of the most rudimentary method of manual identification, that is using paper and pencil, can be achieved by utilizing ontology-based annotation tools [Vargas-Vera et al. 2001]. They provide uniform access to and storage of all the documents in the corpus, and help creating the annotations through knowledge of the underlying ontology and keeping them in a uniform format by

adhering to a given standard. They allow quick navigation through the corpus and offer refined search mechanisms.

Automation of needed identification is the core of a subfield of IE termed ontology-based or ontology-driven IE, in which ontologies are used to guide the process of identification, also called extraction, from semi-structured or unstructured natural language texts and to present the output of that process [Wimalasuriya and Dou 2010]. It is argued that the automated extractors used within this approach become part of the ontology [Wimalasuriya and Dou 2010, pp. 308–309]. We refrain from that idea to strictly separate status from process, although in some cases the step from precisely defining which text-level concepts match an ontology component to building an extractor is quite small.

In order to attain one or a composition of extractors for a given component of an ontology, one can either re-use existing extractors or build them by utilizing an adequate development tool. Finding an unalteredly reuseable extractor for any but the most general concept classes is unlikely and exacerbated by a possible mismatch between the original and the intended genre of text to be analysed. Since Named Entity Recognition (NER) is the most advanced form of IE, benefiting from the Message Understanding Conference (MUC) and Automatic Content Extraction (ACE) competitions, components which at least partially match these text-level concepts are most likely to be obtainable with existing extractors.

IE methods and systems can be categorized along two dimensions: hand-coded or learning-based creation, and rule-based or statistical extraction. Hand-coded systems require a human to define and code the rules used for extraction, while for learning-based systems the human labour consists of annotating the text in a way that an algorithm can infer the correct solution of the extraction task from those labels. Hand-coded methods require skills in linguistics, programming and the domain of the extractor's application combined within one step. With learning-based methods these competences can be more easily separated, and the most laborious part of annotating the text can be done by less skilled individuals, which enables a more efficient scaling. For complex identification tasks still hundreds or thousands of examples are necessary [McCallum 2005, p. 53]. "Rule-based extraction methods are driven by hard predicates, whereas statistical methods make decisions based on a weighted sum of predicate firings" [Sarawagi 2008, p. 278]. In comparison to sta-

tistical methods rule-based methods form the approach with the longer tradition, they are easier to interpret and adapt and are proven to perform very well on certain tasks, e.g. dates can be recognised robustly as regular expressions. Statistical methods though work better on noisy data [Sarawagi 2008, pp. 278–279].

The basic decision, concerning the construction of an extractor, is to either manually code a rule-based system, or to annotate a set of texts to subsequently use them as training material for a rule-based or statistical system, dependant on the nature of the extraction problem and the data. However the choice, human work still is the bottleneck to creating performant IE systems. Performance usually is measured in terms of precision and recall (as defined in figure 4), although Wimalasuriya and Dou [2010] point out that these measures are debatable and thus reference alternatives.

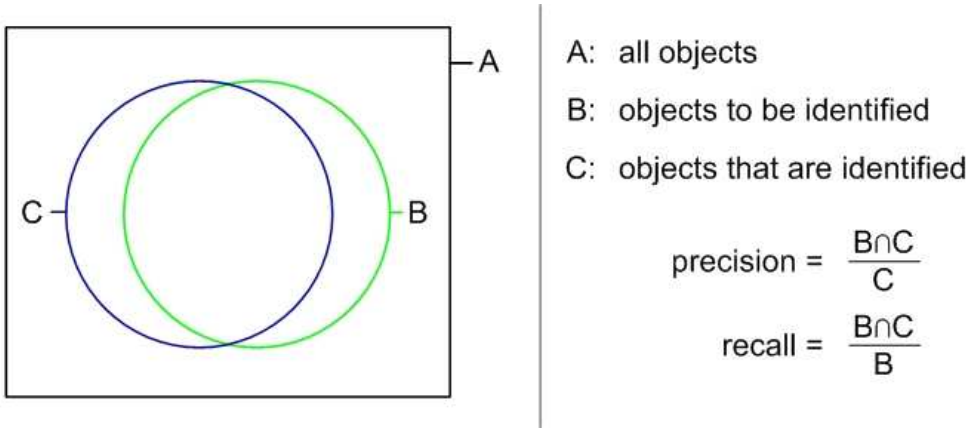

Figure 4: Definition of recall and precision for IE tasks.

There basically is a trade-off between precision and recall. With a very large corpus, such as the web, a system can be trimmed to reach high precision at the expense of recall, because the data likely contains duplicates and a single correct extraction is sufficient. However, given the nature of our sources we are unlikely to benefit from this circumstance.

Automated identification makes use of shallow NLP techniques, either as preprocessing techniques, analyzing and enriching the whole text, or during the extraction process, in which case only the contexts of the trigger items are treated [Nédellec and Nazarenko 2006, p. 5]. Among the usual routines are:

- chunking, which partitions texts into semantic units, such as sentences or paragraphs;
- tokenization, which splits the stream of characters into words;
- filtering, using a task-dependent stop-word list or token statistics;
- part-of-speech tagging;
- syntactic parsing, determining the syntatic structure of a sentence;
- morphological analysis, determining the root form of a word via lemmatizing or stemming.

The performance of these routines varies and impacts the performance of the eventual identification. Sources of knowledge or knowledge repositories are aiding the identification. Aids of this type are WordNet, handling the semantic of terms, gazetteer lists, containing instances matching specified parts of the ontology, or even the web, which can be queried for various reasons in the process of identification.

Hybrid solutions use automated methods to either identify some parts of the ontology independently, while other parts are subject to manual identification only, or to suggest solutions for the identification tasks which are then revised by a human expert. The second approach is also termed information highlighting and is the preferential solution to achieve optimal results. An ideal software environment does not only enable the human expert to interactively work on the identification, but also learns dynamically from the adjustments made for a continuous improvement of the automation. To our knowledge no such solution yet exists, due to the effort needed for an extractor to learn even small adjustments, especially with statistical approaches. A hybrid solution enables greater reuse of existing extractors, since slight missmatches can be corrected manually. The result of a given extractor can be refined and narrowed down to fit the definition of the ontology component, or multiple results can be combined. Admittedly it is hardly possible to determine the performance of a third party extractor on a particular set of data beforehand, just as to estimate the necessary level of performance needed to save effort by manually correcting an extractor's output instead of sole manual coding.

3.4 Relationship Identification

Relationship identification aims to discover the relationships existent between the established concepts as defined in the ontology, that is relations between the instances of classes and between those of classes and attributes. In terms of the concept network it is the linking of nodes to form edges.

A quite common approach to establish relations, due to the fact that it is easy to automate, is to use windowing, which connects concepts based on their proximity in text: they are linked if they co-occur within a set span of tokens, usually only within the borders of given chunks of text, e. g. sentences or paragraphs. This approach though does not only fail to satisfy the need for differentiated types of relationships, since it can only capture existent and non-existent. It has also been proven to generate non-acceptable rates of false positives, i. e. relations are established which are not existent according to a human understanding of text. Diesner [2012, pp. 63–83] demonstrates that with a window size of at least seven, meaning seven space separated tokens between the heads of the concepts to be related, an acceptable recall of more than 90 per cent is reached, but at the cost of a precision of only 10 per cent and less.

In IE relationship identification is considered an even harder problem than entity identification. Conceptually though the problem of deciding on a binary relation between given entities is the simplest relation identification task, and simpler than concept identification as it only requires a scalar prediction [Sarawagi 2008, p. 317]. Most relation extraction systems address this type of task [Bach and Badaskar 2007, p. 2]. Precisely we want to figure out, if in a chunk of text with two marked entities any of the relationships in a given set exists. State-of-the-art systems doing this type of supervised relation identification unsurprisingly use learning-based approaches [Bach and Badaskar 2007, p. 13].

These systems though are usually designed to identify relations only within sentences and not across them, despite a considerable share of all relations being of this type, as an analysis of Swampillai and Stevenson [2010] reveals. The difficulty of identifying inter-sentential relations is higher than in the case of intra-sentential relations, since syntactic parse trees constitute a weaker feature for the task of learning and a smaller fraction of positive examples are available for it [Swampillai

and Stevenson 2011]. The features used for relation identification are basically the same as for the previous mentioned NLP tasks.

In the latest evaluation of semantic relation identification in SemEval-2010, the best system reached a combined F_1 score, which is a measure combining precision and recall as defined in formula 1 to asses an identification result, of 82 per cent on nine different relations, which can be deemed more than satisfactory for practical application [Hendrickx et al. 2009]. Swampillai and Stevenson [2011] show that, dependant on the type of relation, inter-sentential relation detection can achieve comparable results, but generally is more likely to deviate downwards. Just like the the task of concept identification, relationship identification will most likely need the assistance of human experts to achieve useable results.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{1}$$

3.5 Reference Reconciliation

During reference reconciliation the exact mapping between text-level concepts and concept class instances is determined by identifying all text-level concepts that are manifestations of the same instance and consistently associating them to their unique identifier and canonical representation [Diesner 2012, p. 26]. The process is also termed reference resolution – amongst others, depeding on the application context [McCallum 2005, pp. 50–51] – and commonly said to consist of two separate tasks, coreference resolution and anaphora resolution [cf. Diesner 2012, p. 26]. The theoretical separation between these tasks though is only vague and they are increasingly carried out jointly in resolution systems, and so the terms have become virtually synonymous [Poesio, Ponzetto, and Versley 2010, pp. 4, 16–17]. Reference reconciliation comprises three core tasks, namely resolving the coreference of mentions of proper nouns, noun phrases and pronouns, but other types of expressions can also be included, such as date mentions which are to be canonicalized. Just like the previous text processing tasks it is inherently complex, so that even humans reached an agreement of only $F_1 \approx 0.8$ for the corpus of MUC 6 [Poesio, Ponzetto, and Versley 2010, p. 41]. For the given purpose reference reconciliation has not only to be carried out within, but possibly also across texts.

The common case is to have various different linguistic phrasings for one instance which must be identified, but a single linguistic expression can also be an expression of several different instances, which is a special difficulty of IE and seldomly considered by linguists concerned with reference reconciliation. It is to be carried out for all the previously identified concepts, and when refering expressions which are to be incorporated are not already identified, this has then to be completed in a separate step. This step is usually termed mention detection and basically another concept identification phase, which should be avoided in favor of identifying all relevant concepts in the first place. The result is intended to be sets or chains of corefering expressions which can be linked to one instance of the ontology's concept classes in each case.

The importance of reference reconciliation is highlighted in studies by Diesner and Carley [2009] and Diesner [2012], demonstrating that 60 per cent and more of the entities in their data are subject to the procedure. Therefore it is a crucial step in the extraction of concept networks, potentially increasing the amount of information available on truly distinct concepts.

The simplest automated solution for coreference resolution is a system based on string matching, the results to be expected though can clearly be judged as not reliable enough [Diesner 2012, pp. 48, 54; Nédellec and Nazarenko 2006, p. 10]. As a technique in NLP, reference reconciliation underwent the same shift from hand-coded to learning-based systems as other NLP tasks, relying on preprocessing techniques capable of reliably extracting features, external sources of knowledge and hand-annotated corpora [Poesio, Ponzetto, and Versley 2010, pp. 49–50].

The latest evaluation of coreference resolution systems in SemEval-2010 [Recasens et al. 2010] highlights the insufficient state-of-the-art in the field, since only six participants were able to deliver valid results and the two baselines – each entity in its own set and all entities in one set – turned out to be hart to beat by the systems. In the most realistic setting of this closed scenario a maximum of 73.9 per cent in F_1 could be reached. It also demonstrated that the lack of proper evaluation metrics and corpora annotated with coreference information hinders progress in the field. Most of the work has focused on news corpora, so it is unclear how the systems would behave in others domains, including the one we intend to target [Clark and González-Brenes 2008, p. 16]. Reference reconciliation is a topic of

ongoing research in NLP which is far from being solved, and at the moment suitable results cannot be achieved without human intervention [Diesner 2012, p. 186]. Common sense knowledge as used by humans seems to be the key to success.

3.6 Concept Network Extraction and Analysis

Technically speaking the extraction of networks is just transforming the data gathered in the previous steps into formats that are interpretable as networks, be it a set of matrixes or lists of nodes, edges and attributes. The theoretical decisions to be made are more profound though, reflecting the implications of the interweavement of research questions, ontology development, network extraction and network analysis.

Depending on the circumstances, different networks can be extracted, ranging from a simple uni-modal and uni-relational network to complex multi-modal and multi-relational networks. Nodes and edges can be valued, depicting their weight or probability, or not, which implies for edges to be reduced to binary edges. Unidirectional edges can be changed to bi- or non-directional edges, nodes and edges can be merged, and only a subset of the full network can be selected for analysis.

There are two major types of analysis which complement each other: (1) interactive visual analysis and (2) measurement-based analysis. During both types the analyst may adapt the extracted network again, which includes merging nodes and edges, removing isolated nodes and edges below a determined value treshold, filtering nodes dependant on attributes, and so forth. Turning attributes into nodes is also an option to be considered, but careful planing during the ontology development phase renders this unnecessary, as was explicitly discussed in subsection 3.2.

Interactive visual analysis is most useful with small networks. It uses color, shape and size of nodes and edges in combination with text to visualize information in two- or three-dimensional spaces and allows the user to manipulate the graph and its visualization in manifold ways. Essential for the visualization are high-quality graph drawing algorithms, which arrange the components in the Euclidean space so that certain conspicuities, such as subgroups and hubs, are easily detectable using this type of analysis.

Within measurement-based analysis statistical and graph-based measurements are computed and interpreted. Measurements are either calculated on the node level, such as Betweenness Centrality, Degree Centrality, and Clique Count, or on the network level, such as Density, Fragmentation, and Hierarchy [Hanneman and Riddle 2005]. Which measurements are calulatable depends on the network characteristics, and only for some cases existing measures and their interpretations are available. For most though, existing measures are to be reinterpreted, and for some even new measures have to be invented. Even for the comparatively well explored social network analysis, which exhibits over a hundred established measurements [Carley and Pfeffer 2012, p. 3], there are areas which are not well represented, such as multi-relational networks. For some networks there might be no statistical or graph-based measures which are meaningful, so that analyzing their visual representation is the only option.

4 Text Processing Solutions

4.1 Introduction

Carrying out the analysis process as described in the previous section requires the support of software tools in all phases, to enable users with varying capabilities in the processing of natural language texts to efficiently extract a concept network. A holistic solution, covering all phases and their diverse requirements, would be preferable but cannot be supplied by the market at present. More realistic is a segmentation into the three spheres of (1) ontology construction, (2) text processing, and (3) network analysis, each having their own set of tools with a narrower focus and specialised capabilities. Splitting the text processing part up further may be able to better leverage the native capabilities of tools in a focused area, but comes at the cost of dealing with additional data exchanges.

Following the requirements to be met by a text processing system against the background of the GLODERS research project are covered. The phase of ontology construction can be supported with a wide range of available tools, whose requirements are mainly driven by the expected complexity of the ontology, as outlined in section 3.2, and the integration with adjoining text processing systems. Since the project's requirements in terms of complexity are low, and none of the findable software systems displays capabilities to integrate with an applicable text processing system, a resulting catalogue of requirements would be too shallow to truly steer a selection process and is therefore omitted. Integration at this point of intersection is mainly determined by the text processing system's extensibility, to enable it to process common ontology serialisation formats as input in order to create the metadata metastructure, or its general capability of importing a metadata metastructure in a transparent format which can be generated from an ontology serialisation format using a custom converter in an intermediate step.

4.2 Requirements

4.2.1 Metadata Metastructure Definition

The process of analysis intends to extract a network from texts, but doing so initially requires enriching the texts with metadata from which the network then can be constructed in consecutive steps. The most basic requirement therefore is, that a software must provide a structure capable of mapping this metadata of a text. This includes metadata on a) successive parts, i.e. spans, of a text, which implicates means to unambiguously locate them, b) on relations between two predetermined spans, i.e. binary relations, within a text, and c) on sets of multiple corefering predetermined spans, i.e. coreference sets, within a text. And for the sake of processing texts metadata on a text as a whole, so-called document-level metadata, should be mappable as well.

The metadata objects must be associated with classes, which are relatable in a taxonomy. Classes must be able to carry primitive attributes, i.e. designators for which objects carry values of boolean, numerical, or nominal nature. Furthermore they should also be able to carry complex attributes, whose values consist of collections of primitive attributes, a single metadata object, or even collections of these. This is important since the metadata structure is not only used to map data as previously defined in the ontology, i.e. the outcome of the text processing step, but also for data that serves as in- and output for one of the intermediate steps of text processing.

Means to define and adapt the metadata metastructure according to the identified needs are of course a necessity.

4.2.2 Project Administration

For each analysis all associated data should be accessible in one central project container. The most important pieces of this data are the corpus, which is a distinct collection of all texts used in the project, and the metadata attached to it. The corpus should be organizable in a file tree or similar structure, easing the management of concurrent and extensive analysis projects. It would be favorable if the user could inspect any actions taken throughout a project's existence in detail, allowing him to explain and reproduce any analysis he has conducted, which is

important to adhere to the standards of scientific rigor. Other functionality should automatically store and restore any configurational change made to the software environment.

4.2.3 Further Metadata Functionality

For each piece of metadata information on its creation should be accessible, e.g. when it was created, if it was created manually or by an automated component, and, if applicable further, details on the processing step it was created by. This possibility further improves replicability. It should also be possible to organize metadata in sets, i.e. collections which exist separated in parallel, for the whole corpus or a subset of it. That would enable isolated analysis attempts and is important for the development and testing of automated analysis components. Besides creating and deleting these sets, it should be possible to merge them by creating the union, intersection, or difference of sets.

4.2.4 Text Import

The minimum requirement is the processability of text files residing in a local file system in plain text format. Character encodings to be accepted must include UTF-8 and UTF-16, and beyond that should include common western encodings, such as Windows-1252, ISO-8859-1 (latin1) or ISO-8859-15 (latin9). The correct character encoding of a file should be recognised and the file treated accordingly. Apart from a local file system, texts may also be fetchable from remote URLs via HTTP, HTTPS, FTP, and SFTP network protocols. Further storage formats for textual data, such as Rich Text Format, OpenDocument Format, Microsoft Word Binary File Format, Office Open XML, Hypertext Markup Language, and Extensible Hypertext Markup Language, should be supported by automatically converting them to plain text format. In that process existent formatting and structural and semantic markup should be transformed into metadata, accessible in the same way as the metadata created during analysis, as far as possible. Further preprocessing routines should remove redundant whitespaces and non-printing characters. This ensures, that for a wide range of sources any processing induced by a user is based

on a uniform representation, i.e. refined plain text plus a set of accessible metadata, independent of the type of the source.

4.2.5 Text and Metadata Visualization

Texts and their metadata must be visualizable and navigable in a graphical form. This implies that a text is inspectable in a continuous form, and optionally one page at a time. It must be possible to move from metadata to corresponding text passage by selecting pieces of metadata, and to graphically display the corresponding text span in the case of concepts, or spans and their connections in the case of binary relationships and coreference sets. The visualization should be applicable to a single metadata object, selected from a list or equivalent, to all objects belonging to a selected class, and to all objects belonging to a class and being subject to attribute restrictions as indicated by the user. Each metadata class may have its distinct, configurable style of visual display. At the same time moving from text to metadata should also be possible by selecting a point or span in the text and displaying all metadata objects that relate to that selection, plus their attributes. These visualization capabilities enable the user to intuitively inspect the current status of enrichment and the outcomes of previous processing steps.

4.2.6 Manual Metadata Handling

Another mandatory functionality is the support of manual creation of metadata by means of the graphical visualization of a text. This implies that a span in a text can be marked and then a piece of metadata, related to that span, can be created or an existing one deleted or adapted. For binary relations and coreferences within and across texts as well as for document-level metadata, similar simple capabilities to create, adapt, and delete them manually should exist. Deletion and adaption should not only be possible for single pieces of metadata, but also for groups according to class membership or attribute value. If the concept of a metadata set is implemented, they should also be creatable and deletable manually. These functionalities allow the user to enhance and refine his ontology and metadata metastructure by exercising it on the data, to develop ground truth data for the development of automated analysis components, to correct the outcome of auto-

mated analysis component, and to take over analyses which are not sufficiently or efficiently automatable.

4.2.7 Embedded Analysis Functionality

Exclusion

Parts of texts should be excludable from analysis, which means that no piece of metadata should be allowed to relate to them and automated analysis components should ignore them during their processing. This allows the user to exclude parts which may distorte or needlessly exacerbate the analysis without altering the text, allowing for a better comprehensibility than deletion would and preserving the original context. This exclusion should be enforceable by selecting the corresponding parts in the graphical visualization of texts and through callable routines. These routines should allow for an exclusion of lines containing less than a determinable threshold of characters, and lines which only consist of non-alphabetic and non-alphanumeric characters. In official documents, parts which exhibit these traits are frequently bearing content of administrative or bureaucratic function and are thus not relevant for the analysis.

Regular Processing

As the language of a text obviously influences its analysis and should be taken into account by most if not all automated analysis components, the system should have a functionality to detect a text's language for all the presumably western languages which might be used in one of the sources of GLODERS. Texts complying to these languages' writing systems should also be enrichable with metadata on spans indicating their composing elements, which are tokens, sentences and paragraphs. In the case of tokens several different types should also be differentiated, for example space tokens, punctuation tokens, and numeric or alphabetic tokens. The delimiters, which are used to identify separate tokens, should be configurable, for instance a hyphen may or may not used as a delimiter. These are important functionalities due to many analysis techniques presupposing the correct identification of the mentioned building blocks of language.

Advanced Processing

Metadata for spans in text should be creatable using regular expressions. Regular expressions trace back to Kleene [1951], are now widely used in programming, whereby among the many derivatives some slightly varying syntaxes exist, and are very efficient at identifying patterns such as dates, amounts of money or license plate numbers. Supplying metadata classes, optionally in conjunction with attribute values to assign, and regular expressions, one metadata object is created for every pattern matched. Another functionality required to create metadata for spans is the application of dictionaries. Dictionaries are basically lists of terms, and for every expression in a text that matches one of these terms, a piece of metadata for its span is created, with class and attributes to assign to be configured using the dictionary applicator. Single and multi word dictionaries should be supported, providable in plain text or comma-separated value format, stored using one of the common encodings mentioned in subsection 4.2.4. Preferrable application strategies to be supported are with or without considering capitalization, and matching all entries or only the longest one to a given span. And for multi word dictionaries also with allowing strictly full or also partial matches, and with strict or arbitrary order of words. Further should dictionaries not only be applicable to the tokens of text, but also to the values of string attributes of a metadata class, e.g. to match a tokens' stem which is given by such an attribute. Embedded may also be functionality to create metadata for spans determining the tokens' part of speech and stem or root word, and metadata for spans indicating common named entities, such as persons, organisations and locations, in the expected languages.

Dictionaries and metadata determining the tokens' part of speech should also be utilizable as negative filters, i.e. as exclusion mechanisms.

Concerning metadata creation for relationships a windowing functionality is desirable, which establishes relationships between spans based on their proximity in text. What should be configurable are the metadata classes of spans which are allowed to relate, the maximum number of tokens between two spans to be related, and the type of context unit in which the two spans must coexist, for example a sentence, a paragraph or none. Metadata for coreference sets are desirably

creatable by using string and substring matching, string edit distance, also termed Levenshtein distance [3], and the proximity of spans in text.

These advanced processing capabilities would lay the groundwork for an efficient extraction of concept networks from texts.

4.2.8 Analysis Component Development

The possibility to develop custom analysis components is imperative for a solution, since the embedded analysis functionality is unlikely to cover all efficiently automatable processing steps arising in the course of a project. These components should be able to operate on a single or a set of texts and their metadata, in order to add, change or delete metadata or to produce some form of result from it. Supporting this development in a high-level programming language, desirably incorporating object orientation, would allow a user to extensively manipulate metadata and produce results in the framework provided by the software. Since text processing solutions in their original state are unlikely to produce a result format that is readable by any network analysis software, providing this functionality is arguably the only way to avoid developing and utilizing customized converters in another intermediate step. For the manipulation of metadata a rule-based script language would be the preferred means of knowledge-based analysis component development. Providing this option enables a faster creation of analysis components in comparison with high-level programming, due to a steeper learning curve and substantially less lines of code needed, and is a feasible option even for an inexpert user. As a third possibility the development of rule-based or statistical analyzers using machine-learning may be supported, whose advantages and disadvantages have been examined in brief in section 3.3.2. The features utilisable by the machine-learning functionality should be allowed to be of boolean, numerical and nominal type, and of flat or tree-structured form. Such a function should allow the selection of features to incorporate from a basic list of features, which reflect the text and its metadata available in the predetermined ground truth data. The selection process should then be supported by capabilities to analyse the features' contribution to prediction quality and its caused processing effort. Analysis com-

[3] Tracing back to Vladimir Levenshtein's work on correcting transmission failures of binary words [Levenshtein 1966].

ponents should be testable and evaluable on texts or samples of them, both through a manual inspection of the outcomes and through comparing them against a given benchmark in form of ground truth data. The benchmark test should, dependent on the type of metadata, allow an evaluation of the component in terms of precision, recall and F_1 score, and supply feedback on the respective full and partial matches. The previously mentioned metadata sets are an important function for this test, since they allow the metadata manipulation carried out in a testrun to not affect the current status of the analysis by creating a separate testset. Moreover should analysis components be assemblable to compound components, by whom they are applied consecutively, in order to be able to break down more complex analysis problems into smaller steps and to reuse partial solutions already available.

4.2.9 Combined Analysis Usage

The embedded analysis functionalities should be includable into compound components just like the self-developed ones. Applying the functionality of an analysis component or compound component to a corpus, or a subset of it, and its metadata, should then require only determining potential configuration parameters. It should also be possible to apply the same component with different configurations within a project, and to access configurations previously defined in that context.

4.2.10 Data Export

Project containers should be exportable and importable to exchange them between users and to transfer them between systems. This involves the corpus and its metadata, and may also include data such as the metadata metastructure, the software environment configuration and the recording of actions taken within the project. The same should also be possible for any self-developed components, and therewith their configurations used within a project should become part of the transferable project container. Texts with visualizations of metadata on spans denoting the respective classes should be exportable in a common textual format or as XML documents with inline annotations, in order to be viewable with widely preexist-

ing software such as a text editor or a browser. This allows a major part of the text processing result to be disseminated to and evaluation by a wider audience.

4.2.11 Non-Functional Requirements

Besides the requirements stated, a suitable software solution shall also fulfil some non-functional ones, which are less influenced by the general process of extracting networks of concepts from text, and more by the conditions under which the GLODERS project has to operate. These have implications on several areas.

Performance
The number of texts which can be analysed in a project should be arbitrary, but rather than their total number the length of any single document is of importance to the performance aspect of the system. Since documents with hundreds of pages in length are expectable, memory efficiency is to be favored over runtime efficiency. A common desktop PC, with between three and seven gigabyte of memory available to the software tool, must be able to handle the whole text and thousands of pieces of metadata simultaneously in order to conduct an analysis. The runtime in contrast is non-critical, as the total number of documents and the deadlines to be met leave sufficient time for analysis. If there are unused hardware resources available, texts should nevertheless be analyseable in parallel to make the most of these.

Usability
All functionality that the text processing solution has to offer should be accessible through a graphical user interface, that allows giving commands with a mouse and hotkeys, which are especially valueable for swiftly creating metadata through the graphical visualisation of texts. Further should there be one single point of access for all the functionality, so the user is not forced to switch between different parts of the system while working. The system should allow the user to choose the language of the graphical user interface: while English is mandatory, German and Italian would be favorable.

Besides accessing the functionality, learning how to use it for the goals of the analysis is the most important part of usability. Learning how to set up the system

from scratch, and how to use all the contained functionality, with the exception of the one offered for the development of custom analysis components, are the two major steps on the way to getting acquainted with the system. The effort spendable to acquire the knowledge needed to make use of the functionality for the development of analysis components mary vary hugely, since previous knowledge on the part of the user and the spectrum of functionality offered greatly affect this measure. Influence on the learnability is wielded by any refering documentation, guidelines, examples and similar material, as well as support provided by the official support system of the supplier or independent sources. High availability and quality of these resources are obviously beneficial.

Further Non-functional Requirements
It is of utmost importance that no data is transfered to a third party, neither for processing nor for storage, and so any component of the system that is accessible outside of the local machine, and therefore potentially by a third party, must be secured accordingly. Moreover should no unexpected or erroneous user behaviour result in a software or a system crash, and it must not result in any form of data loss. Platform wise the system must support Microsoft Windows XP and upward, and should support Mac OS X and the major Linux distributions for personal machines, in each case both for 32-bit and 64-bit architectures. Concerning the licensing and the connected pricing of the system, a software that is free for academic purposes is preferred to not burden the available financial budget.

4.3 Candidates

From the huge variety of solutions promising to deliver text mining or text analysis, only those that can be labeled as NLP development workbenches offer the possiblity to create customized, automated text processing components with the necessary complexity. Therefore this is the spectrum examined to identify the most qualified candidates. The subsequent section is looking at how two software solutions, which are estimated to be most suitable for the task, meet the stated requirements. It has to be noted that only functionality that is either contained within the supplied installation or available as a plugin from the developer's web-

site has been considered for the review. One major advantage of the two reviewed solutions over others though is that there is quite some functionality offered by third parties which can be utilized as well.

The solutions assessed here are the Apache UIMA framework and its available set of tools, henceforth referred to as UIMA, as well as at the GATE framework and GATE Developer, the main graphical tool for utilising it, henceforth referred to as GATE [4].

GATE is chosen since it is the earliest member of the range of solutions examined, developed as a research tool for the NLP community to enable the development, testing and sharing of analysis components. It is widely adopted in this field and still constantly augmented. UIMA has a stronger focus on being deployed in productive environments to embed NLP tasks within a wider application. It has attracted our interest due to it being utilized in one of the most impressive computer systems doing natural language processing: IBM Watson.

4.4 Suitability

4.4.1 Metadata Metastructure Definition

UIMA possesses the concept of type systems, which are explicit metadata class definitions. Only metadata corresponding to these definitions can be added to a text. The type systems enable hierarchical ordering on the metadata classes, and the definition of concept classes, as subclasses of the standard class Annotation, which has pointers to a text span, and relationships and coreference sets, as subclasses of the rootclass TOP. It is possible to define the wanted simple and complex attributes on these classes, which allows one to define binary relations, as a class with two attributes each holding an Annotation object, and coreference chains, as a class with an attribute holding a list of type Annotation.

GATE by default does not enforce explicit metadata class definitions and does not support hierarchical ordering on the classes. Just like UIMA it has a built-in class Annotation, which can map concept classes by providing pointers into the

[4] More precisely, we are assessing the Apache UIMA Java framework version 2.4.0 and its tools, and GATE Developer version 7.1. as available in May 2013. Attainable from http://uima.apache.org, respectively http://gate.ac.uk.

text, and allows to define simple and complex attributes on its classes. Beyond that GATE also has a built-in construct to map coreference chains, which is a document-level attribute named matchesAnnot carrying a list, in which all coreference chains are contained as lists of metadata objects. Relationships must be mapped in the same way as with UIMA, as classes with two complex attributes.

None of the two solutions has a predefined metadata class for relations, while both provide tools to create a metadata metastructure and support document-level metadata. The value range of simple attributes for both solutions corresponds to the set of primitive Java data types.

4.4.2 Project Administration

UIMA does not have project container. It also does not offer anything which compares to the concept of a corpus, the user rather is expected to provide the data to be analyzed from the exterior every time an analysis is conducted. Any tool working within the framework is responsible for storing and restoring its configuration, the ones shipped with the framework partially do so.

GATE Developer does not have the concept of a project container as well, but it offers corpora to collect the texts for an analysis and the metadata attached to them within the software. These corpora don't offer any way to structure their content, all texts are managed in one list. The software automatically stores its configuration on exiting, and restores it on starting.

Both solutions offer only rudimentary capabilities to inspect the actions taken during analysis, as they only log errors arising from automatic components. Making sure that the results are reproduceable solely lies within the users responsibility, leaving it up to him to document the process and save any material used.

4.4.3 Further Metadata Functionality

Both solutions do not supply any data on the creation of metadata objects on their own. A user has to make sure that he adds any information on the creation wanted as attributes to the object, and that their values are filled in at the time of creation, either by the human annotator or the automated processing component.

Both solutions enable the user to organize his metadata in sets, or views as it is named in UIMA. For GATE though this is limited to metadata for concepts, all other metadata, i.e. metadata for relationships and coreference sets, are rooted at the document level and cannot be managed in sets.

Neither the UIMA tools nor GATE Developer offer capabilities to merge metadata sets.

4.4.4 Text Import

UIMA has the concept of file readers, which enables users to create their own text import components from scratch, and also expects them to do so. This explains why only a reader able to handle plain text files residing in the local file system is delivered with the system, as part of its examples package and its Document Analyzer tool.

GATE takes over the task of importing text and therefore supports a variety of formats. Besides plain text, also HTML and XML, some email, PDF, Microsoft Office and Open Office formats are supported. Unfortunately the creators don't specify which ones of the later exactly are supported, which leaves the user to the principle of trial and error. Files can only be imported from the file system, not from remote URLs. During importing, existing inline markup is transformed into the internal representation, further preprocessing to achieve a clean text is not executed.

Encodings supported by both GATE and UIMA are those supported by Java, which include UTF-8 and UTF-16 and a wide range of other encodings, covering virtually all character sets of the present.

4.4.5 Text and Metadata Visualization

UIMA possesses three text and metadata viewers, each with rather limited capabilites: CAS Editor, which is a plugin for the Eclipse IDE, and Annotation Viewer and CAS Visual Debugger, which are lean stand-alone tools.

The CAS Editor is capable of displaying text and all related concepts. It is possible to highlight all text spans belonging to concepts of selected certain concept classes, while the style of the display is customizable. One can select a concept

from a list and move to the corresponding text span and display its attributes, and also move from text span to the respective concept. There is no support for displaying or visualising metadata on relationships, coreference sets or the document as a whole.

The Annotation Viewer enables the visualization of text and metadata within a Java tool, or as HTML or XML output which can then be inspected via a browser. The Java tool subsumes the capabilities of the other two more static options. It offers capabilities to inspect concepts by selecting a point in text for which all related concepts are displayed with their attributes, and by highlighting all text spans which belong to selectable concept classes. The style of highlighting is customizable.

The CAS Visual Debugger enables a user to inspect a structured list of all metadata objects attached to a text and their attributes ordered by metadata classes. For all concepts the user can move to the respective text span. In a separate window all text spans belonging to one single concept class can be highlighted. The style of the highlighting can be customized.

GATE has an integrated text and metadata viewer, which makes it possible to inspect a concept and its attributes by selecting the corresponding text span, and also to go from concept to the respective text span by selecting the metadata object in a list, which results in the viewer jumping to the span. It is also possible to highlight all text spans belonging to a certain concept class by selecting the classes from a list. The style of displaying is customizable. However, it is not possible to only display the text span of a single concept or of all concepts belonging to a class and being subject to attribute restrictions. One can inspect to which coreference set a concept belongs by selecting the respective span in the text, and display all the text spans of concepts belonging to a selected coreference set. Document-level metadata is also being displayed. Displaying to which relationships a concept belongs, or enabling the user to navigate from a relationship to the involved concepts is not part of GATE's skillset, just like showing attribute values for relationships and coreference sets is not.

All bespoken viewers are not capable of displaying texts pagewise, but only as continuous text.

The majority of the text and metadata viewers are not capable of displaying and visualizing relationships, but there is a workaround to this worth remarking. Creating relationships as metadata objects of the class Annotation or a subclass of it, allows for both solutions to abuse the respective viewers' capabilities to visualize concepts in text to also visualize relationships, as spans stretching from the beginning of the first to the end of the second relationship. This is an imprecise technique and does not truly reflect the semantic of a relationship, but relaxes the lack of capabilities in this area to a certain degree.

4.4.6 Manual Metadata Handling

Manually creating, altering or deleting metadata within UIMA is only possible with CAS Editor. It offers capabilities to create, alter and delete concepts by marking or selecting text spans or selecting concepts from a list. Deleting concepts does not work for whole classes or groups determined by attribute restrictions. It is also not possible to manually create, alter or delete metadata sets, relationships and coreference chains. Document-level metadata though can be handled manually. Using the previously mentioned workaround for displaying relationships, they can at least be deleted by selecting the respective text span or list entry.

GATE's integrated text and metadata viewer enables a user to manually create concepts by marking text spans, and to alter or delete them using the same dialogue window popping up after text selection, or by selecting a concept from a list. It is possible to delete metadata sets and also all concepts belonging to a class. Deleting a group of concepts determined by attribute restrictions on the other hand is not possible. Coreference chains can be created by assigning a concept to a new chain and deleted using a list of them, and existing concepts can be added to or removed from a coreference chain by selecting the respective text span. Document-level metadata can be created using two text input lines for designator and value. It is not possible to manually create, alter or delete relationships in a proper way with GATE. Just like with UIMA's CAS Editor, the workaround for relationships allows a user to delete them.

4.4.7 Embedded Analysis Functionality

Exclusion

Neither UIMA nor GATE have capabilities to exclude parts of a text from an analysis. With GATE parts can altered or deleted, either by automatic components or using the embedded text viewer. With UIMA this not envisaged as well, which is a consequence of not having integrated corpora to store texts.

Regular Processing

UIMA offers language detection only through a wrapper for the web service of the provider Alchemy, which is a very comprehensive solution but in conflict with the data security requirements. Three tokenizers are available for UIMA: Whitespace Tokenizer, which always uses whitespace and punctuation to separate words, Offset Tokenizer, which is part of the Concept Mapper addon and can be configured in terms of the delimiters used, and JFlex Lexer which is part of the UIMA Ruta addon and distinguishes a variety of different token types arranged in a tree which specializes downward. These tokenizers are not specialised on any language, which makes them useable on all western languages at the cost of inaccuracies. Sentences can be detected using Whitespace Tokenizer, and paragraphs using UIMA Ruta Plain Text Annotator.

GATE comes with two components for language identification: the plugins Language Identification and LingPipe Language Identifier. Both are able to distinguish 15 european languages, the two language sets are not congruent though. The standard tokenizer within GATE is the Unicode Tokenizer, which produces multiple token types and is not specialized on any language. A specialisation of it for English is part of the ANNIE plugin. Furthermore there are OpenNLP Tokenizer, which differentiates between tokens and spaces, and LingPipe Tokeniser, which only detects words, both are not customizable. For splitting sentences there are four options available for general western texts: ANNIE Sentence Splitter, RegEx Sentence Splitter, LingPipe Sentence Splitter, and OpenNLP Sentence Splitter. A possibility to detect paragraphs seems to be missing though.

Advanced Processing

Within UIMA utilizing regular expressions to create concepts is possible using the

Regular Expression Annotator addon, for which rules for matching regular expressions to concept classes can be supplied as XML files. For the application of dictionaries two options exist: Dictionary Annotator and Concept Mapper. Dictionary Annotator works with dictionaries in a custom XML format, but this can be generated from plain text files containing one entry per line. It allows for single and multi word dictionaries, can apply them case sensitive or not, and besides text also to string attributes of concepts. Concept Mapper also requires its dicionaries in a custom XML format, but there is no possibility to transform simple plain text files to this format. In addition to the functionality of the Dictionary Annotator it can be used to add attributes to the concepts, and it can be configured to match all entries or only the longest one, strictly full or also partial occurrences, and with strict or arbitrary order of words. Part of speech tagging for English and German is possible with the Hidden Markov Model Tagger. With the Snowball Annotator one can produce stems for currently twenty European languages, inlcuding English, German, Dutch and Italian. Named entity identification is possible only by using wrappers for the web services of Calais or Alchemy, and windowing or coreference functionality is not part of the package.

Within GATE regular expressions can be used to create concepts by utilizing the integrated text and metadata viewer, supplying one expression and subsequently annotating all matching text spans. Moreover they can be created by using Simple Regexp Annotator, for which regular expression can be applied via rules supplied in a custom format file. For dictionary application an abundance of tools exists. The standard GATE gazetteer can handle single or multi word entries supplied as plain text files, and does partial or full matching and applies only the longest or all matching entries to a given span. Its drawback is, that it always creates the same concept class with two string attributes which are intended to reflect its meaning. The Hash Gazetteer is a reimplementation of the standard one for extensive dictionaries and can also be configured to match its entries case sensitive or not. The Flexible Gazetter allows all dictionaries complying to the GATE interface to be applied to string attributes of concepts. The BWP Gazetter extends the standard gazetteer with the ability to match using Levenshtein's Distance to handle noise and error in text. The Extended Gazetteer 2 extends the standard gazetteer so that the concept classes created can be determined and matching can be ad-

justed more detailed. The Feature Gazetteer can solely match string attributes of concepts, but in return is capable of also adapting or removing existing concepts relating to a matched span. Furthermore there are dictionary tools which transform parts of ontologies into dictionaries and apply them: the Gazetteer LKB does this for the names of instances, the Onto Root Gazetteer for the names of classes and instances, and for string properties, and the Apolda Ontology Annotator for two specified annotation properties of classes and instances. For part of speech tagging a user can utilize the LingPipe POS Tagger for English and Bulgarian, the RASP2 POS Tagger for English, and the Tree Tagger for English, German, Spanish, French, Italian and Bulgarian. The GATE Morphological Analyzer and the RASP2 Morphological Analyzer both produce lemmas for English, while the Durm Lemmatizer does the same for German. The Snowball stemmer is also part of GATE, with twelve of the currently twenty languages supported being available in the installation. Named entity identification is done for English by OPEN NLP NER and LingPipe NER, and by the respective language specific NER plugins for German and French. There is no component for windowing within GATE, and coreference matching is restricted to the ANNIE OrthoMatcher, which handles only specific named entities, preferable in English.

4.4.8 Analysis Component Development

Analysis components which operate on a single or a set of texts can be developed using the Java programming language within the framework provided by UIMA. Another option is to utilize the Bean Scripting Framework Annotator and write components in one of its supported scripting languages, including JavaScript and Phython, to create small scale components. UIMA Ruta is a rule-based script language which works on the text and its existing concepts to create, adapt or delete concepts. An integral part of it is the usage of regular expressions on the condition part, and in order to enhance the built-in functionality on the action part it is also possible to execute Java code with it. There is no machine learning package included which would allow a user to train its custom components. Inspection of the outcomes of an analysis component is possible only manually via the text and

metadata viewers already described, there is no option to benchmark it against a given ground truth and recieve standardized measures.

Within GATE analysis components can operate solely on a single text, and the development is restricted to Java as a high-level programming language. Other languages can only be used if a connector to the resulting component is supplied as a Java archive. As a rule-based script language GATE offers JAPE, which is very similar to UIMA Ruta and also offers the usage of Java on the action part. However, it is less powerful on the condition part and, leaving the option of Java code aside, also on the action part. Machine learning can be utilized with the Learning plugin, which supports the learning of chunks for concept classes, and relations as needed for relationships and coreference resolution, although the later two require more extensive post-processing to bring the metadata into the desired form. Algorithms available are a Support Vector Machine (SVM), a Perceptron Algorithm with Uneven Margins (PAUM), and the Naive Bayes, K nearest neighbour and C4.5 decision tree as implemented in the Weka toolkit[5]. The includable features may only be of nominal type and flat structure, and their determination cannot be done within the GATE environment but must be done fully manually by creating a custom XML configuration file. Feedback on the features' contribution to prediction quality is only available for the SVM with a linear kernel, the respective impact on processing time is not made evident. A manual inspection of the outcome of a component is possible using GATE's integrated text and metadata viewer, and the Annotation Diff tool enables an assessment of analysis components which create concepts against a ground truth. Precision, recall and F_1 score are displayable, with partial matches included or excluded, and all full and partial matches, misses and false positives are inspectable.

4.4.9 Combined Analysis Usage

With UIMA analysis components can be aggregated to compound components, and those again can make up more aggregated components, so that an aggregated component may consist of a component tree with arbitray depth. This is true for both existing and self-developed components, and one component can be used

[5]Weka is a collection of algorithms and tools for machine learning and data mining developed by the University of Waikato, www.cs.waikato.ac.nz/ml/weka/

within an application multiple times with distinct configurations. The current configuration is stored, previous ones though are not accessible if not saved by the user.

GATE enables a user to combine existing and self-developed components into applications, whereby a component can be utilized multiple times with distinct configurations as well. But it offers only one level of aggregation, so a compound cannot be part of a compound itself, and the user has to decide at the time of the creation of the compound if he is going to apply it to a single text or a whole corpus. Application to a subset of a corpus is not possible. Configuration details are only stored for the current setup, in order to have a history a user has to export the configurations manually.

4.4.10 Data Export

Both solutions do not have a project container which could be exported, and UIMA also does not have corpora. Within GATE datastores, which contain texts plus their metadata, can be serialized and then exchange as folders. Within UIMA the metadata metastructure, i.e. the type system, is exchangeable. Both are not capable of exporting any configuration of the environment or recordings of action, which both don't posess.

With UIMA components and applications composed from them can be packaged to form a PEAR (Processing engine archive) file using the PEAR Generation Wizard, which is an Eclipse plugin. These archives can then in turn be imported using the PEAR Installer. With the Annotation Viewer tool the results of an analysis can be exportet as XML file.

With GATE developed components cannot be exported, they can only be exchanged in form of the raw files making them up. Complete applications composed from components though can easily be exported and imported with one click, and the same is true for configuration details of such an application. Texts can be exported as XML files with inline annotations, so that concepts can be visualized.

4.4.11 Non-Functional Requirements

Performance

Performance can only be compared on the task level, and our limited experiences are by no means a comprehensive assessment. In our tests, both solutions were able to handle an analysis with components of moderate complexity, applied to a text of 600.000 characters, which corresponds to rougly 200.000 words or 450 DIN A4 pages, producing more than 100.000 pieces of metadata, with 3 gigabyte RAM. Judging the speed of processing GATE seems to be ahead because of the more efficient execution of its rule-based script language. However, depending on the task and the components used this can vary hugely, with both solutions taking only seconds up to hours depending on the scenario. While both frameworks possess the ability to handle parallel processing, the bespoken tools, delivered to work with them, always apply serial processing and therefore do not make the most of multiprocessor machines.

Usability

With UIMA the universal, single point of access is the Eclipse platform, while with GATE a user has to switch from GATE Developer to an Integrated Development Environment (IDE) of his choice to develop components using the Java programming language. Eclipse as the point of access though is far less comfortable than GATE Developer since it is a multi-purpose environment. not tailored to the task and far more complicated for any user who is not acquainted with it. GATE Developer is slim and the functions are neatly arranged. Both are operatable using a mouse and without command line, except when developing custom components of course, and both solutions' manual metadata editors support hotkeys to accelerate manual work. Both solutions' interfaces are solely available in English.

GATE is easy to setup, as it only requires executing the supplied installer. Setting up UIMA within Eclipse is more complicated, since it requires installing Eclipse, the UIMA Eclipse plugins, and the Eclipse Modeling Framework, and importing the UIMA examples to register the available tools. The supplied manuals are sufficient in both cases. Using the embedded analysis functionality is easy to learn for GATE, since the amount of configuration to be done by a user is low and the components'

documentations are extensive. UIMA is more demanding at this point, but in return custom component development using Java is easier to learn with it than with GATE, due to the examples provided by the developers. All things considered, both solutions are accompanied by enough material to independently learn how to make use of them. Since both solutions are available free of charge the developers' support is limited and users mostly have to rely on the independant communities for assistance, where GATE seems to have rallied the larger one.

Further Non-functional Requirements

The data security requirements render it impossible to use components which are provided by third parties as a web service, and therefore limit especially UIMA's capabilities. Besides that all analyses of the two solutions run on the local machine, and since both are implemented in Java they run on any OS that is supported by Java 6 or later, including Windows, Mac OS X and Linux. While UIMA operated stable, GATE has the shown the tendency to crash after a couple of consecutive analyses without restarting the application. This behaviour points towards a memory leak, the specific cause though could not be detected. Both solutions are free of charge for academic usage.

5 Prototypical Application

5.1 Source Data and Concept Ontology

The method is prototypically applied to a data set provided by the Dutch law enforcement agencies, covering their investigation into a criminal group carrying out extortion, among other criminal activities, in the wider area of Amsterdam during the 1990s and early 2000s. It is a collection comprising records of interogations, officers' field reports and commentaries, and reports on economic activities of the group members and associated businesses, which amount to a total of 193.000 words written mainly in Dutch.

One part of the scientific interest in this case concerns the involved actors. That is, how many persons are involved, who are they key players in the scheme, and what kind of structure does the criminal group in itself adopt? Coming closer to answers to these questions is the goal of the analysis, and the scope is narrowly set on persons as the concepts of interest. The ontology defined for our analysis is simple and does only model a fraction of the subject under investigation, since developing and applying a full-fledged one would go far beyong the scope of our experiment. So it consists of a concrete concept class Person and a bi-directional relationship links-with from Person to Person. Looking at the data source, we can identify two types of text-level concepts that clearly qualify as manifestation of the concept class Person, that is people's full names and forms which are at least partially anonymized, which flow into the definition of the concept class. The relationship finds its textual manifestations wherever an interaction between two text-level concepts of class Person is ascertained.

5.2 Concept Identification

Both text processing solutions utilized do not have ready-made capabilities to identify person entities for Dutch, so the automated component has to be self-developed. Due to the lack of annotated training material and the demonstrably good capabilities of hand-coded IE analyzers for this purpose, this approach is the one chosen. The identification proceeds in three steps: first identify all spans equating forenames by applying a dictionary composed from World Wide Web

sources, second identify all spans equating to shorthands that match the pattern used for anonymization, and third apply a set of rules that extends the identified spans to the full person mentions if applicable.

Evaluating the quality of concept identification using a manually annotated sample and the tools provided by the solutions proves, that the approach works reasonably well. Both solutions' dictionary components and their rule-based script languages in combination with regular expressions are up to the task, but due to the differences in these languages the results are only approximately identical. It is to be noted that in the given example the execution of GATE's rule-based language was considerably faster, by a factor of five.

5.3 Relationship Identification

The defined type of relationship is rather vague, so that even human annotators are likely to show relatively little inter-annotator agreement. Due to the known inadequacies of NLP techniques at this point, the only available option for automation is to use the windowing approach as discussed in section 3, despite of its shortcomings. We follow the recommendation by Diesner [2012] of a maximum of six tokens between two concepts to relate them, but since there is no agreement on what exactly constitutes a token, the result of this approach varies with the implemented conception. Figure 5 depicts how relationships are formed using windowing on a sentence in which tokens and concepts are already identified. As there is no built-in component for the task in either of the solutions, we are forced to develop this component as well, which necessitates programming using Java within the respective framework. The tokenizing though is done beforehand by an existing component, and here UIMA's configurable tokenizers offer an advantage over GATE's less accessible and transparent tokenizers.

5.4 Reference Reconciliation

Automatically matching concepts of the class Person to identify those that are manifestations of the same instance can be done using the ANNIE OrthoMatcher component within GATE, while for UIMA the component has to be self-developed. The OrthoMatcher possesses a feature that seeks to match nicknames to forenames

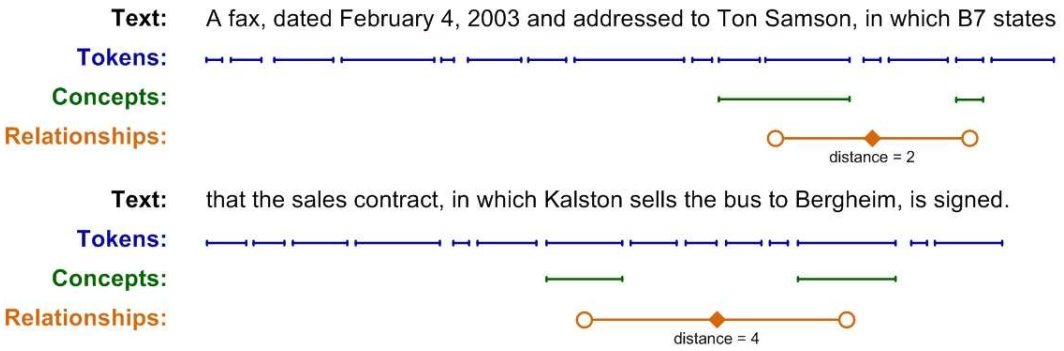

Figure 5: An example of the windowing technique.

steered by a dictionary, and beyond that uses string matching. Its exact inner workings though are not transparent due to its scarce documentation.

Evaluating the reference reconciliation shows, that the OrthoMatcher frequently erroneously identifies persons which share the same surename to be matches. For example, with the included dictionary 'Christine Simmons' and 'Chris Simmons' are matched, even though a human reader would notice that these concepts refer to different entities with different sexes. Since splitting coreference sets manually is far more effortful than joining them, an approach that favors precision over recall is preferable. This is the reason for the custom component developed for UIMA, which does automatic reference reconciliation, to follow a minimalist approach. It aims to only match Person concepts whose strings are equivalent when compared to each other case insensitive, so that 'PETER PARKER' and 'Peter Parker' are matched, errors as described above by contrast are avoided.

Reference reconciliation is further hampered by a special type of noise in the data in the given example, which is false anonymization conducted by the authority handing over the data, assigning the same shorthand to person mentions which are obviously distinct. These interferences stress that reference reconciliation in particular has to be done very carefully with a mixture of automatic and manual labour, even though anaphoric resolution, which further complicates the task, is not considered in this prototype.

5.5 Network Extraction and Analysis

Extracting the network is conducted by a custom Java component for both solutions, which transform the generated metadata into a further processable format. Since *ORA[6] is the target application, the produced format is a table in a Comma Seperated Values (CSV) format, with the canonical identifiers of the coreference sets as the designators of both lines and columns, and the numbers of occurrences of the relationship between the indicated instances, interpreted as its strength, as the values of the table elements.

Before the network can be analyzed it has to be correctect by adapting the resulting matrix since the integrated text and metadata viewer and editor functionality of the text processing solutions do not allow for an efficient manual correction. Since the processing done using UIMA is more transparent and produces less false positives in the task of reference reconciliation than its counterpart, we are limiting our subsequent description to the network gained from it. After automatic processing the source data contains metadata on 11.000 concepts, 666 coreference sets, and 6.455 relationships. The network consists of 666 nodes and 1.456 links after removing the symmetric ones.

207 nodes have to be removed, mainly due to incorrectly identified forenames that correspond to common dutch words such as dan or den, because of spelling errors, or names of persons that are part of a street or company name. After this procedure 459 nodes and 772 links remain. Then nodes have to be merged due to the minimalist approach during automatic reference reconciliation, finally resulting in a network of 246 nodes and 246 links.

The visualization of the full network is difficult to approach for analysis, but using a spring layout and the possibities of interactive analysis some instances are revealed which are highly connected and therefore likely to play a key role in the scheme. These nodes can also be identified using measures of centrality (figure 6). The results of the analysis correspond with the analyst's impression based on reading the material.

[6] *ORA is a network analysis tool developed by the Center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University, http://www.casos.cs.cmu.edu/projects/ora/.

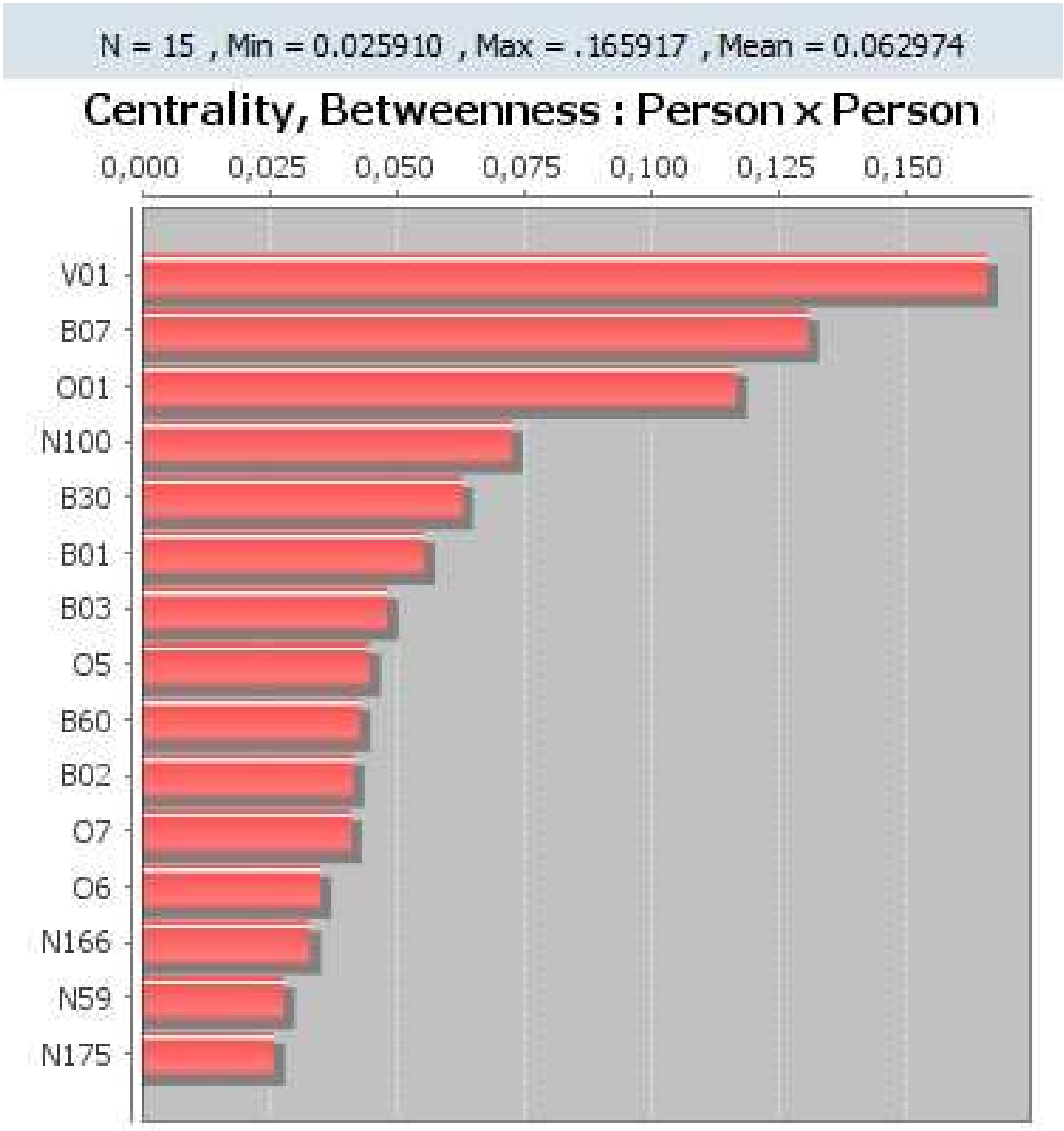

Figure 6: The 15 most central Person instances in terms of betweenness. Names are anonymized according to the following scheme: B = Contact, O = Offender, V = Victim, N = Not specified.

In order to examine the structure of the criminal group the respective subset of the extracted network is singled out. The given instances can hardly be allocated to the necessary subclasses of the concept class Person, which are **C**ontact, **O**ffender, **V**ictim, and **N**ot specified, using automatic text processing techniques, so this a task carried out manually on the instance level as well. The subnetwork (figure 7) consists of only 9 nodes and 13 links, but is more than 21 times denser than the full network (subnet: 0.3611, full: 0.0168), although it has two isolates. Here visual analysis reveals that the structure of the subnet highly relies on the instance O1, who is connected to all but the two isolates and is part of the strongest links, while all other links are weaker. Therefore we are intepreting the social group as reliant on one central figure, who is likely to be the person in authority, while all others are on an equal level with only little interaction. A verification of this claim remains yet to come.

5.6 Assessment

The prototypical application reveales that both UIMA and GATE have their advantages and drawbacks and eventually complement each other instead of being substitutes. Nevertheless GATE seems to be more fitting for the purpose, as it is more suited for manual work on the text, which is a key part of the process of concept network extraction from text, due to its superior annotation usability and functionality. This is of importance since the automated methods of text processing, especially if not up to the state of the art in NLP, which is likely for almost all self-developed components, leave a lot of room for deviations, and therefore necessitate manual input and human assessment in every phase. Still both text processing solutions unveil deficits in this particular area beyond the handling of spans in text. The other two major advantages of GATE, its considerably greater number of ready-made components offered and its built-in machine learning capabilities, do not play a central role in this application, but will likely impact a lot of other concept network extraction analyses.

For the conducted analysis, source data authored in a language more popular in the realm ofs NLP and IE, and which does not suffer from additional noise added through partial and incorrect anonymizations, would allow for higher quality text

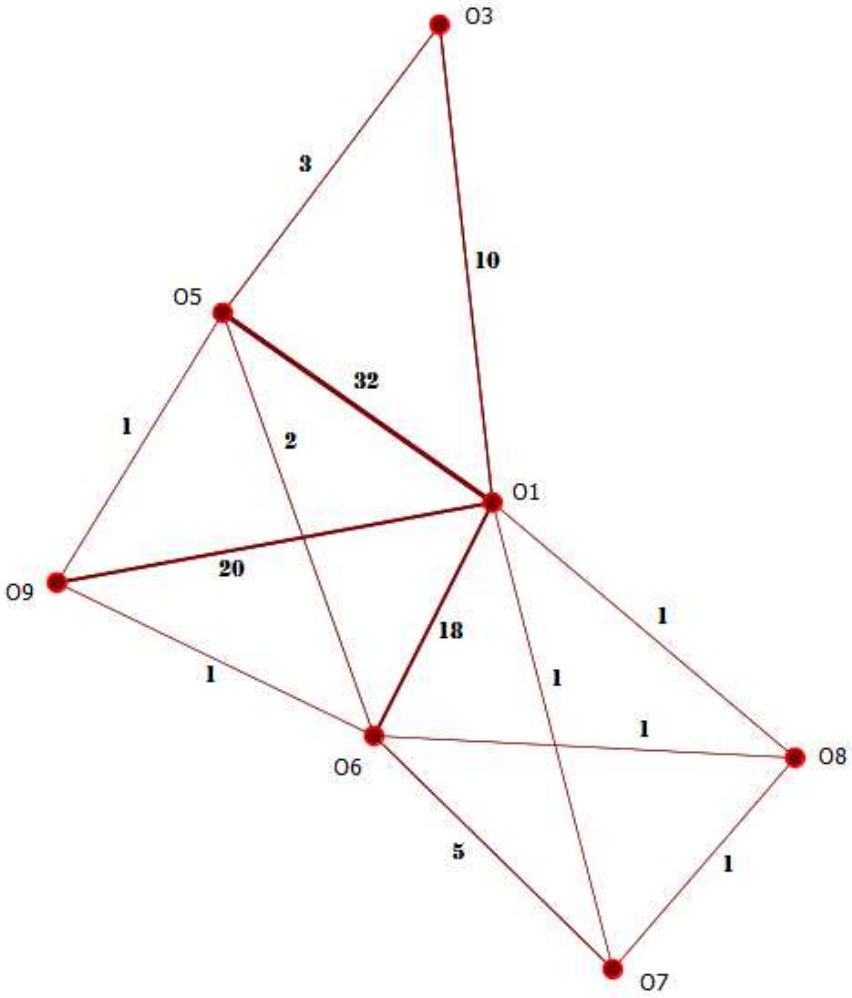

Figure 7: The Offender subnet without isolate nodes, link width corresponds to link value.

processing. To really benefit from automation the amount of data to be processed must be considerably higher than in this example. Especially for more complex concepts and relationships, reaching the point where the effort needed for automated and manual analysis are comparable will likely need millions of words as data.

In consideration of the amount of data processed though, the resulting offender subnetwork, which is at the heart of the research question the analysis was initially conducted for, is based on a relatively small amount of observations. It could be enhanced by incorporating anaphora resolution, but this is a very challenging task for both humans and NLP and far from being adequately solvable by a novice in computational linguistics. This result raises the question if there is a fit between research question, source material and analysis method. In other words, if the text does contain information on the desired topic, and if so, if the described method is capable of extracting it adequately. These questions can not be answered ultimately, but are rather subject to constant verification by the researcher bringing the pieces together for his research at hand.

6 Outlook

When accepting the premise that texts can be understood and analyzed as networks of concepts or terms, utilizing advances in TM and NLP technologies and methods to extract such networks in order to gain knowledge about the world itself, that is reflected in the authors' written expressions of their point of view on it, seems a natural and promising attempt. Research in NLP though is split up into several subfields, and in none of them a considerable advance could be achieved during the last decade, so that the possibilities are still far behind the expectations raised when the field emerged.

In order to benefit from computational approaches to textual analysis a common ground, which allows to tie automated and manual analysis together into a seamless process, is necessary. Manual work is inevitable to adequately process texts, which are so complex that the established opinion is, that we will not be able to fully delegate their creation and reception to machines until we are able to understand and mimic the entire human brain, for the purpose of concept network extraction. So one prospect is the enhancement of the standard NLP platforms with facilities that feature improved integration of manual analysis. The Argo platform[7] is a promising step into the direction of merging computer-driven and human analysis for high-quality text processing. But even though it is still in beta status at the time of this writing, it fails to acknowledge that there is a need for functionality to add metadata to more than just a whole text or spans in it, that is such constructs as relationships and coreference sets. Comprehensive and user-friendly functionalities in these areas are needed to roll out a process of concept network extraction on a larger scale.

If a common ground for a computer-driven processing of texts can be established or further interoperability between existing solutions reached, more researchers, companies, and governmental institutions may be induced to share their achievements in the myriads of possible NLP tasks. This would be especially beneficial for languages other than english, which is the language primarily focused by NLP researchers and software framework developers, and therefore has the greatest

[7] The Argo workbench is developed by the National Centre for Text Mining at the University of Manchester, http://argo.nactem.ac.uk

number of applicable tools. In the current state the individual strengths of the dozens of text processing solutions available, be it commercial or free software, are of limited value due to their existence in isolation and lack of interchangeability of results.

An increased availability of components dealing with the tasks of complex relationship detection and coreference resolution for different concept classes, which both occupy considerable interest in the NLP research but are yet underrepresented in the available software solutions, would greatly benefit the method, but are less likely to be pursued due to their limited applicability. The enormous effort needed for even small achievements in NLP will ultimately prevent a lot of useful analyses.

Apart from that, the soundness of the described method of concept network extraction from texts has to be tested in further analysis projects, with different research questions involving more complex ontologies and various types of texts in various languages, requiring additional network analysis methods and measures, to evaluate if and what insights can be gained in different settings. Only this way the considerable effort of the method can be justified within the context of scientific projects.

7 Conclusion

When textual data is an important source of knowledge about a social scientific phenomenon, and the amount of the data to process is too enormous to be handled by humans reading the material, automated methods are inevitable. The established methods of computer-assisted text analysis in the social sciences though have severe limitations in their applicability, and so the enrichment of the semantic network approach with achievements from NLP is a natural attempt to go beyond these boundaries. That incorporating these techniques have impacts on flexibility, validity, reliability and transparency in favor of pragmatism is unavoidable but to be limited by a well-structured approach, a thorough analyst, and proper software support.

Building on the existing work on semantic network analysis as introduced, we have outlined a process of concept network extraction from text that qualifies as a KDT method under the given definition, and is applicable to knowledge demands beyond social scientific research. In the course of this we have shown how to make use of computer-support in the various phases, and to what extend state-of-the-art NLP techniques might substitute manual work in text processing.

To approach implementation we have examined the requirements of text processing solutions to fulfil the described tasks at the core of the process, and how the two solutions, which we consider to be currently preeminent, are meeting these. In combination with the prototypical analysis conducted, these insights are demonstrating how to close the gap between the theoretic construct and its practical application, mixing computer-driven with manual analysis to acquire relevant results.

There are still general doubts about whether a text can be distilled to networks of concept representing the mental map of the author, and how the analysis of these networks can reveal truth about the reality the text is a witness of. All models of language applied by automated content analysis are inherently incorrect. Again, answering the general question of if the networks extracted and the information gained from them are valid is a task for other scientific endeavours, which may benefit from the advances into automation pursued in this work. In any case, the

method will neither eliminate the need for careful thought by researchers, nor remove the necessity of reading texts to fully absorb the information wanted.

References

Alexa, Melina. 1997. *Computer-assisted text analysis methodology in the social sciences.* ZUMA-Arbeitsbericht 97/07. Zentrum für Umfragen, Methoden und Analysen (ZUMA).

Ananiadou, S. et al. 2009. „Supporting Frame Analysis using Text Mining". In: *Proceedings of the 5th International Conference on e-Social Science.*

Atherton, Andrew and Peter Elsmore. 2007. „Structuring qualitative enquiry in management and organization research : A dialogue on the merits of using software for qualitative data analysis". In: *Qualitative Research in Organizations and Management: An International Journal* 2 [1], pp. 62–77.

Bach, Nguyen and Sameer Badaskar. 2007. „A Review of Relation Extraction". In: *Literature review for Language and Statistics II.* url: http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf [visited on 05/22/2013].

Barthélemy, Marc, Edmond Chow, and Tina Eliassi-Rad. 2005. „Knowledge representation issues in semantic graphs for relationship detection". In: *AAAI Spring Symposium*, p. 91.

Beaugrande, Robert-Alain de and Wolfgang Ulrich Dressler. 1981. *Einführung in die Textlinguistik (Konzepte der Sprach- und Literaturwissenschaft).* de Gruyter.

Ben-Dov, Moty and Ronen Feldman. 2005. „Text Mining and Information Extraction". In: *The Data Mining and Knowledge Discovery Handbook.* Ed. by Oded Maimon and Lior Rokach. Springer. Chap. 38, pp. 801–831.

Berelson, Bernard. 1952. *Content Analysis in Communication Research.* 1st edition. Free Press.

Brühl, Rofl and Sabrina Bruch. 2006. *Einheitliche Gütekriterien in der empirischen Forschung? Objektivität, Reliabilität und Validität in der Diskussion.* ESCP-EAP Working Paper 20. ESCP-EAP Europäische Wirtschaftshochschule Berlin.

Brusa, Graciela, Ma Laura Caliusco, and Omar Chiotti. 2006. „A process for building a domain ontology: an experience in developing a government budgetary ontology". In: *Proceedings of the second Australasian workshop on Advances in ontologies.* Vol. 72. Australian Computer Society, Inc., pp. 7–15.

Carley, Kathleen M. 1986. „An approach for relating social structure to cognitive structure“. In: *The Journal of Mathematical Sociology* 12.2, pp. 137–189.

— 1988. „Formalizing the Social Expert's Knowledge“. In: *Sociological Methods & Research* 17.2, pp. 165–232.

— 1993. „Coding Choices for Textual Analysis: A Comparison of Content Analysis and Map Analysis“. In: *Sociological Methodology* Volume 23, pp. 75–126.

Carley, Kathleen M. and Jürgen Pfeffer. 2012. „Dynamic Network Analysis (DNA) and ORA“. In: *Proceedings of the 2nd International Conference on Cross-Cultural Decision Making: Focus 2012*. San Francisco, California.

Chandrasekaran, Balakrishnan, John R Josephson, and V Richard Benjamins. 1999. „What are ontologies, and why do we need them?“ In: *Intelligent Systems and Their Applications, IEEE* 14.1, pp. 20–26.

Chapman, Pete et al. 2000. *CRISP-DM 1.0. Step-by-step data mining guide*. Tech. rep. SPSS. url: http://www.the-modeling-agency.com/crisp-dm.pdf [visited on 01/21/2013].

Clark, Jonathan H. and José P. González-Brenes. 2008. *Coreference Resolution: Current Trends and Future Directions*. url: http://www.cs.cmu.edu/~jhclark/pubs/clark_gonzalez_coreference.pdf [visited on 05/20/2013].

Consortium, TEI. 2013. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Technical report. Charlottesville, Virginia: Text Encoding Initiative Consortium. url: http://www.tei-c.org/Guidelines/P5/ [visited on 04/17/2013].

Creswell, John W. 2006. *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. 2nd edition. Sage Publications, Inc.

Cuilenburg, Jan J. van, Jan Kleinnijenhuis, and Jan A. de Ridder. 1986. „A Theory of Evaluative Discourse: Towards a Graph Theory of Journalistic Texts“. In: *European Journal of Communication* 1.1, pp. 65–96.

— 1988. „Artificial intelligence and content analysis“. In: *Quality and Quantity* 22 [1], pp. 65–97.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva, et al. 2013. *Developing Language Processing Components with GATE Version 7*. User Guide. The Uni-

versity of Sheffield, Department of Computer Science. url: http://gate.ac.uk/userguide [visited on 04/15/2013].

Dahlem, Nikolai and Axel Hahn. 2009. „User-Friendly Ontology Creation Methodologies - A Survey". In: *Proceedings of the Fifteenth Americas Conference on Information Systems*. AMCIS.

De Nicola, Antonio, Michele Missikoff, and Roberto Navigli. 2009. „A software engineering approach to ontology building". In: *Information Systems* 34 [2], pp. 258–275.

Diefenbach, Donald L. 2001. „Historical Foundations of Computer-Assisted Content Analysis". In: *Theory, Method, and Practice in Computer Content Analysis*. Ed. by Mark D. West. Vol. 16. Progress in Communication Science Series. Greenwood Publishing Group.

Diesner, Jana. 2012. „Uncovering and Managing the Impact of Methodological Choices for the Computational Construction of Socio-Technical Networks from Texts". Paper 194. dissertation. Carnegie Mellon University. url: http://repository.cmu.edu/dissertations.

Diesner, Jana and Kathleen M Carley. 2005. „Revealing social structure from texts: meta-matrix text analysis as a novel method for network text analysis". In: *Causal mapping for information systems and technology research: Approaches, advances, and illustrations*, pp. 81–108.

Diesner, Jana and Kathleen M. Carley. 2009. „He says, she says. Pat says, Tricia says. How much reference resolution matters for entity extraction, relation extraction, and social network analysis". In: *IEEE Symposium on Computational Intelligence in Security and Defense Applications*.

Diesner, Jana and Kathleen M. Carley. 2010. „Extraktion relationaler Daten aus Texten". In: *Handbuch Netzwerkforschung*. Ed. by Christian Stegbauer and Roger Häußling. VS Verlag für Sozialwissenschaften, pp. 507–521. isbn: 978-3-531-15808-2.

Doerfel, Marya L. and George A. Barnett. 1996. „The Use of CATPAC for Text Analysis". In: *Field Methods* 8 [2], pp. 4–7.

Dörre, Jochen, Peter Gerstl, and Roland Seiffert. 1999. „Text mining: finding nuggets in mountains of textual data". In: *Proceedings of the fifth ACM SIGKDD inter-*

national conference on knowledge discovery and data mining. KDD '99. ACM, pp. 398–401.

Evans, W. 1996. „Computer-Supported Content Analysis: Trends, Tools, and Techniques". In: *Social Science Computer Review* 14 [3], pp. 269–279.

Even, Fabrice and Chantal Enguehard. 2003. „Specific Domain Model Building for Information Extraction from poor quality corpus". In: *Ontologies and Information Extraction*. Ed. by Amalia Todirascu and Vincenzo Pallotta. Bucarest: EUROLAN 2003, pp. 3–9.

Fayyad, Usama M., Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. „The KDD process for extracting useful knowledge from volumes of data". In: *Communications of The ACM* 39 [11], pp. 27–34.

Feldman, Ronen and Ido Dagan. 1995. „Knowledge Discovery in Textual Databases (KDT)". In: *Knowledge Discovery and Data Mining*, pp. 112–117.

Feldman, Ronen et al. 1999. „Text Mining via Information Extraction". In: *Principles of Data Mining and Knowledge Discovery*. Vol. 1704. Lecture Notes in Computer Science, pp. 165–173.

Fernández-López, Mariano. 1999. „Overview of Methodologies for Building Ontologies". In: *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*. CEUR Publications, pp. 4.1–4.13.

Fernández-López, Mariano, Asunción Gómez-Pérez, and Natalia Juristo. 1997. *Methontology: from ontological art towards ontological engineering*. Technical Report. American Asociation for Artificial Intelligence.

Flick, Uwe. 2009. *An Introduction to Qualitative Research*. Fourth Edition. SAGE Publications Ltd.

Flyvbjerg, Bent. 2006. „Five misunderstandings about case-study research". In: *Qualitative Inquiry* 34.2, pp. 219–245.

Franzosi, R. 1990. „Computer-Assisted Coding of Textual Data: An Application to Semantic Grammars". In: *Sociological Methods & Research* 19 [2], pp. 225–257.

Garz, Detlef and Klaus Kraimer. 1991. „Qualitativ-empirische Sozialforschung im Aufbruch". In: Garz, Detlef. *Qualitativ-empirische Sozialforschung*. Ed. by Klaus Kraimer. Westdeutscher Verlag.

Grimes, Seth. 2008. *Unstructured Data and the 80 Percent Rule.* url: http://www.clarabridge.com/default.aspx?tabid=137&ModuleID=635&ArticleID=551 [visited on 01/29/2013].

Grishman, Ralph et al. 1998. *TIPSTER Text Architecture Design Version 3.1.* Technical report. National Institute of Standards and Technology (NIST). url: http://www-nlpir.nist.gov/related_projects/tipster/download.htm [visited on 04/15/2013].

Grüninger, Michael and Mark S. Fox. 1995. „Methodology for the Design and Evaluation of Ontologies". In: *Workshop on Basic Ontological Issues in Knowledge Sharing at the International Joint Conference on Artificial Intelligence (IJCAI95).*

Gruber, Tom. 2009. „Ontology". In: *Encyclopedia of Database Systems.* Ed. by Ling Liu and M. Tamer Özsu. Springer-Verlag. url: http://tomgruber.org/writing/ontology-definition-2007.htm.

Göser, Sebastian. 1997. „Inhaltsbasiertes Information Retrieval: Die TextMining-Technologie". In: *LDV Forum* 14.1, pp. 48–52.

Gupta, Vishal and Gurpreet Lehal. 2009. „A Survey of Text Mining Techniques and Applications". In: *Journal of Emerging Technologies in Web Intelligence* 1.1, pp. 60–76.

Hanneman, Robert A. and Mark Riddle. 2005. *Introduction to social network methods.* University of California, Riverside. url: http://faculty.ucr.edu/~hanneman/.

Harris, Howard. 2001. „Content Analysis of Secondary Data: A Study of Courage in Managerial Decision Making". In: *Journal of Business Ethics* 34 [3-4], pp. 191–208.

Hart, Ronald P. 1985. „Systematic analysis of political discourse: The development of DICTION". In: *Political communication yearbook* 1984. Ed. by Keith R. Sanders, Lynda Lee Kaid, and Dan Nimmo, pp. 97–134.

Hawthorne, Mark. 1994. „The computer in literary analysis: UsingTACT with students". In: *Computers and the Humanities* 28 [1], pp. 19–27.

Hearst, Marti A. 1999. „Untangling text data mining". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics.* ACL '99. Association for Computational Linguistics, pp. 3–10.

Hearst, Marti A. 2003. *What Is Text Mining?* url: http://people.ischool.berkeley.edu/~hearst/text-mining.html [visited on 01/30/2013].

Heinze, Thomas. 2001. *Qualitative Sozialforschung. Einführung, Methodologie und Forschungspraxis.* Oldenbourg.

Hendrickx, Iris et al. 2009. „Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals". In: *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions.* Association for Computational Linguistics, pp. 94–99.

Holsti, Ole R. 1969. *Content Analysis for the Social Sciences and Humanities.* Addison-Wesley.

Hotho, Andreas, Andreas Nürnberger, and Gerhard Paaß. 2005. „A Brief Survey of Text Mining". In: *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology* 20.1, pp. 19–62.

Iker, Howard P. and Robert H. Klein. 1974. „Words: A computer system for the analysis of content". In: *Behavior Research Methods* 6 [4], pp. 430–438.

Kleene, Stephen Cole. 1951. *Representation of Events in Nerve Nets and Finite Automata.* Tech. rep. Santa Monica, California: RAND Corporation.

Klein, Harald. 1991. „INTEXT/PC: A Program Package for the Analysis of Texts in the Humanities and Social Sciences". In: *Literary and Linguistic Computing* 6 [2], pp. 108–111.

— 1997. „Classification of Text Analysis Software". In: *Classification and Knowledge Organization.* Ed. by Rüdiger Klar and Otto Opitz. Studies in classification, data analysis, and knowledge organization. Springer, pp. 355–362.

Kleining, Gerhard. 1982. „Umriss zu einer Methodologie qualitativer Sozialforschung". In: *Kölner Zeitschrift für Soziologie und Sozialpsychologie* 34.2, pp. 224–253.

Kodratoff, Yves. 1999. „Knowledge discovery in texts: A definition, and applications". In: *Foundations of Intelligent Systems.* Ed. by Zbigniew W. Ra? And Andrzej Skowron. Vol. 1609. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 16–29.

— 2001. „Comparing Machine Learning and Knowledge Discovery in DataBases: An Application to Knowledge Discovery in Texts". In: *Machine Learning and*

Its Applications. Ed. by Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos. Vol. 2049. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–21.

Koll, Matthew. 2000. „Track 3: Information Retrieval". In: *Bulletin of the American Society for Information Science and Technology* 26.2, pp. 16–18.

Kroeze, Jan H., Machdel C. Matthee, and Theo J. D. Bothma. 2003. „Differentiating data- and text-mining terminology". In: *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*. SAICSIT '03. South African Institute for Computer Scientists and Information Technologists, pp. 93–101.

Kromrey, Helmut. 2007. *Empirische Sozialforschung: Modelle und Methoden der Datenerhebung und Datenauswertung*. 11. Auflage. Stuttgart: Lucius & Lucius.

Kuckartz, Udo. 2010. *Einführung in die computergestützte Analyse qualitativer Daten*. 3.Auflage. VS Verlag für Sozialwissenschaften.

Kuckartz, Udo and Stefan Rädiker. 2010. „Computergestützte Analyse (CAQDAS)". In: *Handbuch Qualitative Forschung in der Psychologie*. Ed. by Günter Mey and Katja Mruck. VS Verlag für Sozialwissenschaften, pp. 734–750.

Levenshtein, Vladimir Iosifovich. 1966. „Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet Physics-Doklady* 10.8, pp. 707–710.

Lewins, Ann. 2001. „CAQDAS: Computer Assisted Qualitative Data Analysis". In: *Researching social life*. Ed. by Nigel Gilbert. Sage Publications Ltd, pp. 302–323.

Lewins, Ann and Christina Silver. 2009. *Choosing a CAQDAS Package*. QUIC Working Paper 1. CAQDAS Networking Project and Qualitative Innovations in CAQDAS Project.

Liddy, Elizabeth D. 2007. „Natural Language Processing". In: *Encyclopedia of Library and Information Science, Second Edition*. Taylor & Francis. Chap. 271, pp. 2126–2136.

Lowe, Will. 2003. *Content Analysis Software: A Review*. Technical Report. Weatherhead Center for International Affairs, Harvard University.

Mayring, Philipp. 2000. „Qualitative Content Analysis“. In: *Forum: Qualitative Social Research* 1.2.

Mayring, Philipp and Eva Brunner. 2009. „Qualitative Inhaltsanalyse“. In: *Qualitative Marktforschung. Konzepte, Methoden, Analysen.* Ed. by Hartmut H. Holzmüller. Gabler, pp. 543–501.

McCallum, Andrew. 11/2005. „Information Extraction: Distilling Structured Data from Unstructured Text“. In: *Queue* 3.9, pp. 48–57.

Mehl, Matthias R. 2006. „Quantitative text analysis“. In: *Handbook of multimethod measurement in psychology.* Ed. by Michael Eid and Ed Diener. American Psychological Association, pp. 141–156.

Mehler, Alexander and Christian Wolff. 2005. „Einleitung: Perspektiven und Positionen des Text Mining“. In: *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology* 20.1, pp. 1–18.

Miller, M. Mark and Bonnie P. Riechert. 1994. *Identifying Themes Via Concept Mapping: A New Method of Content Analysis.* Paper. Annual Meeting of the Association for Education in Journalism and Mass Communication, Atlanta.

Mohler, P.P. and K. Frehsen. 1989. *Computerunterstützte Inhaltsanalyse: Grundzüge und Auswahlbibliographie zu neueren Anwendungen.* ZUMA-Arbeitsbericht 89/09. Zentrum für Umfragen, Methoden und Analysen (ZUMA).

Nasukawa, Tetsuya and Tohru Nagano. 2001. „Text analysis and knowledge mining system“. In: *IBM Systems Journal* 40 [4], pp. 967–984.

Nédellec, Claire and Adeline Nazarenko. 2006. „Ontologies and information extraction“. In: *arXiv preprint cs/0609137.*

Noy, Natalya F. and Deborah L. McGuinness. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology.*

Olsen, Mark. 1989. „TEXTPACK V: Text Analysis Utilities for the Personal Computer“. In: *Computers and the Humanities* 23.

Pennebaker, James W., Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count (LIWC): LIWC2001.* LIWC Manual. url: http://homepage.psy.utexas.edu/homepage/faculty/pennebaker/reprints/LIWC2001.pdf [visited on 09/01/2013].

Pennebaker, James W. et al. 2007. *The Development and Psychometric Properties of LIWC2007*. LIWC Manual. LIWC.net. url: http://homepage.psy.utexas.edu/HomePage/Class/Psy301/Pennebaker/HRtraining/liwc2007_languagemanual.pdf [visited on 09/01/2013].

Poesio, Massimo, Simone Paolo Ponzetto, and Yannick Versley. 2010. „Computational Models of Anaphora Resolution: A Survey". url: http://wwwusers.di.uniroma1.it/~ponzetto/pubs/poesio10a.pdf [visited on 05/16/2013].

Popping, Roel. 2000. *Computer-Assisted Text Analysis (New Technologies for Social Research series)*. SAGE Publications Ltd.

— 2003. „Knowledge Graphs and Network Text Analysis". In: *Social Science Information* 42.1, pp. 91–106.

Psathas, George. 1969. „The general inquirer: Useful or not?" In: *Computers and the Humanities* 3 [3], pp. 163–174.

Rajman, Martin and Romaric Besançon. 1997. „Text Mining: Natural Language techniques and Text Mining applications". In: *IFIP Working Conference on Database Semantics*.

Recasens, Marta et al. 2010. „SemEval-2010 task 1: Coreference resolution in multiple languages". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. SemEval '10. Los Angeles, California: Association for Computational Linguistics, pp. 1–8.

Roberts, Carl W. 1989. „Other Than Counting Words: A Linguistic Approach to Content Analysis". In: *Social Forces* 68.1, pp. 147–177.

— 1997. „Introduction". In: *Text Analysis for the Social Sciences: Methods for Drawing Statistical Inferences from Texts and Transcripts*. Ed. by Carl W. Roberts. Lawrence Erlbaum Associates, pp. 1–8.

Roberts, Carl W. and Roel Popping. 1993. „Computer-supported Content Analysis: Some Recent Developments". In: *Social Science Computer Review* 11.3, pp. 283–291.

— 1996. „Themes, syntax and other necessary steps in the network analysis of texts: a research paper". In: *Social Science Information* 35.4, pp. 657–665.

Rost, Jürgen. 2003. „Zeitgeist und Moden empirischer Analysemethoden“. In: *Forum Qualitative Sozialforschung* 4.2. url: http://www.qualitative-research.net/index.php/fqs/article/view/723 [visited on 12/08/2012].

Sarawagi, Sunita. 2008. „Information extraction“. In: *Foundations and trends in databases* 1.3, pp. 261–377.

Schutt, Russell K. 2011. *Investigating the Social World: The Process and Practice of Research*. 7th edition. Sage Publications, Inc.

Seipel, Christian and Peter Rieker. 2003. *Integrative Sozialforschung. Konzepte und Methoden der qualitativen und quantitativen empirischen Forschung*. 1. Auflage. Beltz Juventa.

Shapiro, Gilbert and John Markoff. 1997. „A Matter of Definition“. In: *Text Analysis for the Social Sciences: Methods for Drawing Statistical Inferences from Texts and Transcripts*. Ed. by Carl W. Roberts. Lawrence Erlbaum Associates, pp. 9–31.

Silverman, David. 1993. *Interpreting Qualitative Data*. 1st edition. Sage.

Sánchez, Daniel et al. 2008. „Text Knowledge Mining: An Alternative to Text Data Mining“. In: *IEEE International Conference on Data Mining*, pp. 664–672.

Sowa, John F. 1992. *Semantic Networks*. url: http://www.jfsowa.com/pubs/semnet.htm.

Stede, Manfred. 2008. „Computerlinguistik und Textanalyse“. In: *Textlinguistik: 15 Einführungen*. Ed. by Nina Janich. Narr. Chap. 15, pp. 333–351.

Steger, Thomas. 2003. *Einführung in die qualitative Sozialforschung*. Schriften zur Organisationswissenschaft 1. Professur für Organisation und Arbeitswissenschaft, TU Chemnitz.

Stevens, Robert, Carole A Goble, and Sean Bechhofer. 2000. „Ontology-based knowledge representation for bioinformatics“. In: *Briefings in bioinformatics* 1.4, pp. 398–414.

Stone, Philip J. 1966. *The general inquirer: a computer approach to content analysis*. M.I.T. Press.

Sullivan, Dan. 2003. „Text Mining in Business intelligence“. In: *Business Intelligence in the Digital Economy: Opportunities, Limitations and Risks*. Ed. by Mahesh Raisinghani. Idea Group Publishing. Chap. VI, pp. 98–111.

Swampillai, Kumutha and Mark Stevenson. 2010. „Inter-sentential relations in information extraction corpora“. In: *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, pp. 2637–2641.

— 2011. „Extracting Relations Within and Across Sentences“. In: *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pp. 25–32.

Uschold, Mike and Michael Gruninger. 1996. „Ontologies: Principles, Methods and Applications“. In: *Knowledge engineering review* 11.2, pp. 93–136.

Vargas-Vera, Maria et al. 2001. „Knowledge Extraction by using an Ontology-based Annotation Tool“. In: *International Conference on Knowledge Capture*.

W3C. 2012. *OWL 2 Web Ontology Language Primer (Second Edition)*. url: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/ [visited on 04/08/2013].

Weber, Robert Philip. 1990. „Basic content analysis“. In: *Sage University Paper Series on Quantitative Applications in the Social Sciences* 49.

Weitzman, E.A. and M.B. Miles. 1995. *Computer programs for qualitative data analysis: a software sourcebook*. A software sourcebook. Sage Publications.

Wimalasuriya, Daya C. and Dejing Dou. 2010. „Ontology-based information extraction: An introduction and a survey of current approaches“. In: *Journal of Information Science* 36 [3], pp. 306–323.

Wolf, Sabrina. 2008. „Quantitativ vs. qualitativ: der Methodenstreit in der empirischen Sozialforschung“. Bachelor Thesis. Universität Augsburg. url: http://websquare.imb-uni-augsburg.de/2007-08/2 [visited on 12/07/2012].

Wood, Michael. 1980. „Alternatives and Options in Computer Content Analysis“. In: *Social Science Research* 9.3, pp. 273–286.

Yildiz, Burcu. 2007. „Ontology-Driven Information Extraction“. PhD thesis. Vienna University of Technology, Faculty of Informatics.

Zhang, Yan and Barbara M. Wildemuth. 2009. „Qualitative analysis of content“. In: *Applications of Social Research Methods to Questions in Information and Library Science*. Ed. by Barbara M. Wildemuth. Libraries Unlimited, pp. 308–319.

Züll, Cornelia and Juliane Landmann. 2002. *Computerunterstützte Inhaltsanalyse: Literaturbericht zu neueren Anwendungen.* ZUMA-Methodenbericht 20/02. Zentrum für Umfragen, Methoden und Analysen (ZUMA).

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).

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☒	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☒	☐

Rees, den 26. September 2013

Oliver Krukow