Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

Andreas Kaufmann

MASTER THESIS

Using Natural Language Processing to Support Interview Analysis

Submitted on 05.05.2014

Supervisors: Prof. Dr. Dirk Riehle, M.B.A.
Ann Barcomb, Msc
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander University Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 05.05.2014

License

Erlangen, 05.05.2014

Acknowledgements

An integral part of our software implementation relies on the keyword extraction service provided by AlchemyAPI. I would like to express my sincere gratitude towards AlchemyAPI for allowing us to utilize their resources for our research.

A special thanks also to my supervisor Ann Barcomb, for her valuable advice and guidance throughout my research.

Abstract

In social sciences the process of interview analysis is often performed as a form of qualitative data analysis (QDA), to extract relevant information from transcribed interviews. A researcher performing such an analysis is usually assisted by Computer-Assisted Qualitative Data Analysis Software (CAQDAS), which provides a structured framework for coding documents. We expect that this process may significantly benefit from the application of Natural Language Processing (NLP).

This thesis provides an overview of potential applications for different NLP technologies. We applied a machine learning approach to automatically generate codings for an existing coding system. During a prototypical implementation of an autocoding algorithm we measured the impact of different NLP techniques on the autocoding process.

Keywords: QDA, NLP, Interview Analysis, CAQDAS, Coding, Autocoding

Zusammenfassung

Die Analyse von Interviews wird in den Sozialwissenschaften häufig zur qualitativen Datenanalyse (QDA) eingesetzt, um relevante Informationen von transkribierten Interviews zu extrahieren. Dieser Prozess wird üblicherweise durch Computer-Assisted Qualitative Data Analysis Software (CAQDAS) unterstützt, welche ein strukturiertes Umfeld zum codieren von Dokumenten bereitstellen. Wir erwarten, dass dieser Prozess durch den Einsatz von natürlichsprachlicher Textverarbeitung (NLP) entscheidend profitieren kann.

Diese Arbeit beinhaltet einen Überblick über protentiell anwendbare NLP-Methoden. Desweiteren wurde ein machine learning Ansatz verwendet um den Effekt verschiedener NLP Methoden auf unseren autocoding Algorithmus in einem experimentellen Umfeld zu dokumentieren.

Contents

List of Figures

List of Tables

List of Abbreviations

BNC	British National Corpus
CAQDAS	Computer-Assisted Qualitative Data Analysis Software
CPT	Concept-Positioning Test
DAP	Doubly-Anchored Pattern
IR	Information Retrieval
JAPE	Java Annotation Patterns Engine
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part-Of-Speech
QDA	Qualitative Data Analysis
REST	Representational State Transfer
SBD	Sentence Boundary Disambiguation
STT	Speech-To-Text
TF-IDF	Term Frequency – Inverse Document Frequency
VRS	Voice Recognition Software
WSD	Word Sense Disambiguation
XML	Extensible Markup Language

1. Introduction

1.1 Original Thesis Goals

The goal of this thesis is to assess the applicability of different NLP methods in a QDA environment.

Within this exploratory research field the thesis is focused on the assistance of the coding task within a grounded theory approach to QDA. Experimental results are presented using a prototypical autocoding algorithm leveraging existing tools and services.

Coding results for a set of interviews generated using different methods and settings are compared to codings performed by a human researcher.

2. Research Chapter

2.1 Introduction

2.1.1 Overview

In this paper we present a machine learning approach to thematic coding of interviews which were conducted as part of qualitative research. Our algorithm is easily adaptable to any qualitative research project relying on interviews without the need to construct a domain specific rule set. The semantic context of each answer will be determined by a series of NLP methods, and training data is used to tie semantic context to a coding system.

Central to our algorithm is the task of keyword extraction for which we used a commercial web service provided by AlchemyAPI[1]. We also measured the impact of different word conflation methods on our autocoding results.

2.1.2 Qualitative Data Analysis

Qualitative research methods form one of the main forms of inquiry in the social sciences, nursery, psychology and market research. A typical goal of qualitative research is to achieve a thorough understanding of behaviour, reasoning and opinion, through the analysis of data.

The most common approaches to gathering data for QDA are:

- Interviews
    - *Structured*, using a structured questionnaire.
    - *Semi-structured*, using a rough guideline and open ended questions.
    - *Unstructured* in-depth interviews, which only cover very few topics and are driven by the interviewee.
- Discussions and focus groups are frequently used for market research.
- An observational byproduct of normal behaviour, like an email discussion or a message log for instant messaging.

In most of these cases a textual representation of the data to be analyzed is either already present, or is created by the researchers as a transcript of an audio or video recording. Such a transcription facilitates documentation as well as easy processing and analysis through software packages which are frequently categorized as Computer-Assisted Qualitative Data Analysis Software (CAQDAS). This makes the task particularly suited for the application of NLP techniques.

An important part of the qualitative research process is the coding of the data. During this part of the process the text will be annotated to highlight the most insightful parts of the text, resulting in a coding system.

When working with a grounded theory approach the coding system will be further refined iteratively finding support for a theory, establishing new codes or combining or eliminating codes which are not sufficiently supported by the data.

1 http://www.alchemyapi.com/

2.1.3 Challenges of QDA

The process of coding is interpretive by design. This property is the source of three main difficulties that are frequently part of criticism of QDA. (Mays & Pope, 1995; Martin & Stenner, 2004; Neale, Allen & Coombes, 2005; De Ruyter & Scholl, 1998)

1. The research lacks *reproducibility*. Two qualified researchers may follow two different interpretations resulting in a significantly different theory.
2. The process is highly *biased*, and offers only the subjective impression of the researcher.
3. Qualitative research is often difficult to *generalize*, since the sample size is usually low and not statistically representative. Conclusions drawn from qualitative research are also said to be anecdotal.

We would also include the following:

4. The coding process lacks *transparency* and *traceability*. This correlates with the issue of *reproducibility*, but the context of a researchers reasoning for a specific coding is not only crucial for a reproduction of his work, but also for fully understanding it.

Some of these issues, specifically the *reproducibility*, *bias* and *traceability* may be addressed by partly automatizing the process through software. It is our expectation, that by applying NLP techniques and services, we will be able to speed up the process of coding while at the same time alleviating the issues of traceability and reproducibility.

2.1.4 NLP for QDA

The features of current state-of-the-art CAQDAS packages are mostly limited to systematic data management and querying of the text and codes. The use of computer assisted methods for such purposes has been widely accepted and is now common practice. Bong (2002) described the impact of CAQDAS packages on qualitative research as facilitating “the rigor of methodology and the transparency of method as manifested in one's audit trail that in essence constitutes research that is accountable, innovative and effective.” We believe that high level NLP including semantic analysis of the data may be integrated into such environments to extend on these virtues.

We see a wide range of applications for various NLP techniques, which we would categorize in four main categories for which NLP seems well suited:

1. Finding new evidence for an existing theory.
2. Creating, or modifying a theory by creating new codes, if there is sufficient evidence in the text.
3. Support the researcher's workflow by enhancing manual codings with additional information or offering new tools to search for relationships.
4. Voice recognition software may be used to assist the transcription of an interview from an audio recording.

An overview of this categorization of NLP techniques for QDA is provided in Table 1.

NLP for Interview Analysis (QDA)		
Task	Method	Tools
Find new evidence	Rule based	• Manually written rules tailored to specific codes. • Pattern analysis
	Corpus based (Machine learning)	• Keyword extraction • NER • Lexical databases • Statistical linguistics • Sentiment Analysis
Generate new codes	Machine learning	• Concept extraction • Keyword Extraction • Sentiment Analysis
Enhance manual codes	Generation of structured meta data	• Sentiment Analysis • NER
Facilitate transcriptions	Dictation	• Voice Recognition Software

Table 1: Overview: Tasks - Methods - Tools

In this work we will focus on finding new evidence for an existing theory using a ML approach. The data we use are transcribed and manually coded interviews .

2.1.4.1 Criticism of Automation in QDA

We realize that automation of the coding process is not an uncontroversial proposition. Qualitative methods are usually driven by subjectivity and creativity, which most kinds of standardization and certainly automation will impair.

Knoblauch (2013) criticized the tendency towards the automation of interpretation which is *"reinforced by various software programs that reduce the demanding and reflexive work of interpretation to the seemingly innocent process of coding and leave what must be interpreted to those who program software."*

Although we certainly agree that qualitative research should be subject to the researchers creativity and experience, we argue that both are not virtues that should be held above all measures when it comes to assessing the quality of a coding. As with all research results, one needs to be aware of how the coding was constructed, and approach it in an educated way in light of the used methodologies, which may also include a partially automated coding process.

Also we do not suggest, that the use of NLP techniques will replace the interpretative ingenuity of a qualified researcher, but performing content analysis though the use of NLP, and automatizing the early stages of the interpretative work by extracting new evidence for an existing theory and creating a thematic coding, may complement the researchers abilities, or assist in a way, that he may focus on the refinement of his theory.

To improve the acceptance of such techniques, we assume it is of particular importance, that the algorithm deriving the coding is transparent. Using open source software may inspire more trust, and we assume that a comprehensive explanation of the process on a more abstract level will also be critical to generating a wider acceptance in the research community.

2.2 Related Work

2.2.1 Application of NLP to QDA and Computer-Assisted Coding

While plenty research of existing CAQDAS tools, both comparative (Lewis & Silver 2009; Saillard 2011) and instructional (Welsh 2002), is available, there has been little research into the application of sophisticated NLP techniques for QDA in general.

Richards (2002) argued that the lack of an active and critical academic debate on the topic of qualitative computing is partially due to the fact that the new technologies that certainly are available are only assessed in terms of the old methods, and that software was perceived a mere tool to enhance fluidity and speed of the analysis process.

However, improving fluidity and speed are both desirable features for CAQDAS software. And software, and our approach for autocoding in particular, is well suited to support both. But we see other applications of NLP techniques that improve different aspects of the process, besides speed and fluidity.

Since Richards's investigation into the reasons for a lacking debate in 2002 a methods *revolution* is still not in sight, however a few new related publications emerged that show a continuing interest in this topic. Yu, Jannasch-Pennell and DiGangi (2011) investigated text mining and qualitative research, and claimed that several common principles make them epistimelogically compatible. Still, also in 2011 Yu et al. noted that, at the time, querying several research databases with "*text mining and qualitative*" yielded very few results.

Verspoor, Sanfilippo, Elmore & MacKerrow (2006) also provided a very brief overview into applicable NLP techniques for social science analysis.

2.2.1.1 Autocoding

The only published research performed regarding autocoding using high level NLP was performed by Crowston et al. (2010a). They presented a case study in which they built a rule-system to perform autocoding for 12 pre-defined codes on a set of instant messaging logs, and compared the results to a manually performed coding.

Crowston et al. (2010b) compared the preliminary results of their case study with the preliminary results of a ML approach using the same data. They concluded that both approaches show promise, but only case study using a rule-based approach was published in reasonable detail.

2.3 Research Question

Our goal is to determine how established NLP technologies and services may be applied to interview analysis. We are interested in a brief overview of the possibilities and want to assess their pertinence in our specific scenario.

Complementary to Crowston's research describing a rule-based approach we explore a machine learning methodology and are interested in how accurately an autocoding algorithm incorporating ready-available technologies can reproduce the results of a trained researcher.

We are further interested in identifying any specific characteristics of interviews that may provide useful information in this process, and investigate how the consideration of these features affects the autocoding algorithm.

2.4 Research Approach

2.4.1 Overview

To assess the applicability of high level NLP to the problem of autocoding we implemented a tool, written in Java, to acquire experimental results using two interview sets. The manual coding was performed in MAXQDA[2] and exported to XML file format.

Figure 1 shows an abstract overview over our algorithm.

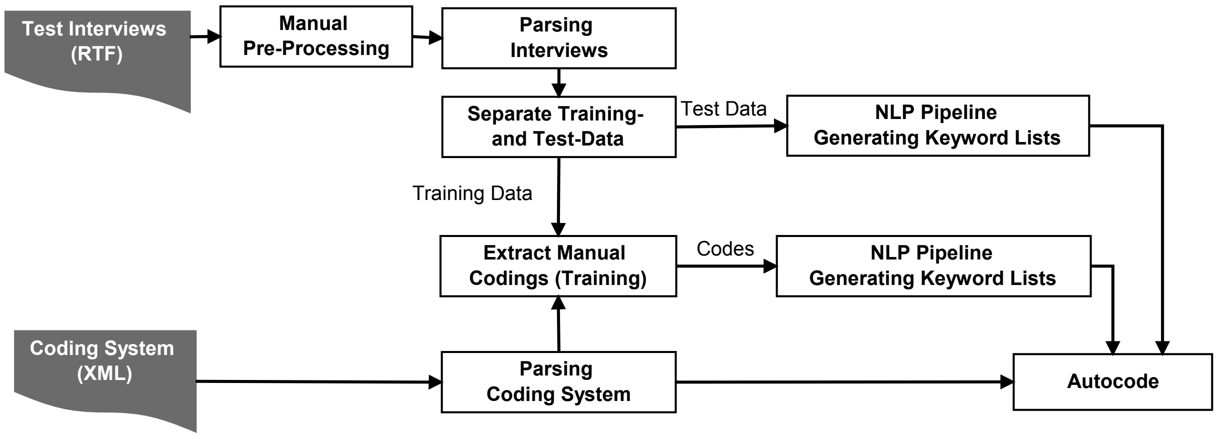
Test Interviews (RTF)
Manual Pre-Processing
Parsing Interviews
Separate Training- and Test-Data
Test Data
NLP Pipeline Generating Keyword Lists
Training Data
Extract Manual Codings (Training)
Codes
NLP Pipeline Generating Keyword Lists
Coding System (XML)
Parsing Coding System
Autocode

Figure 1: Algorithm Overview

To train our ML algorithm we divide the data into training and test data. Any machine learning method will require a certain minimum amount of data to perform well. Our method is not aimed at projects with a limited sample size of two or three interviews. Since our first interview set is comprised of only 6 interviews, each interview we code will use the other 5 as training data.

2 http://www.maxqda.de/

2.4.2 Data Preparation

Pre-processing of our interview data is necessary to preserve useful meta information before parsing the document. For this study the process is only partially automatized, since our data was not consistently formatted, which was solved by re-formatting manually into a unified format. When the text is subsequently parsed we can identify section headers, and differentiate between questions and answers.

While parsing the document we remove some metadata embedded within the text which is not useful and may skew the results of our autocoding methods. These include mentions of *[inaudible]* and time markers like *[00:??:???]* which exist in our text because one of our interview sets is transcribed from an audio recording of the original interview.

Further we generate statistics concerning the word frequencies relative to the current document and to all documents. Through user-settings our tool can then be configured to eliminate very common words, but still distinguish between words that are characteristic for one specific interview and words that are equally frequent in all documents.

2.4.3 NLP Processing Pipeline

2.4.3.1 Overview

Our NLP Pipeline shown in Figure 1 is presented with more detail in Figure 2.

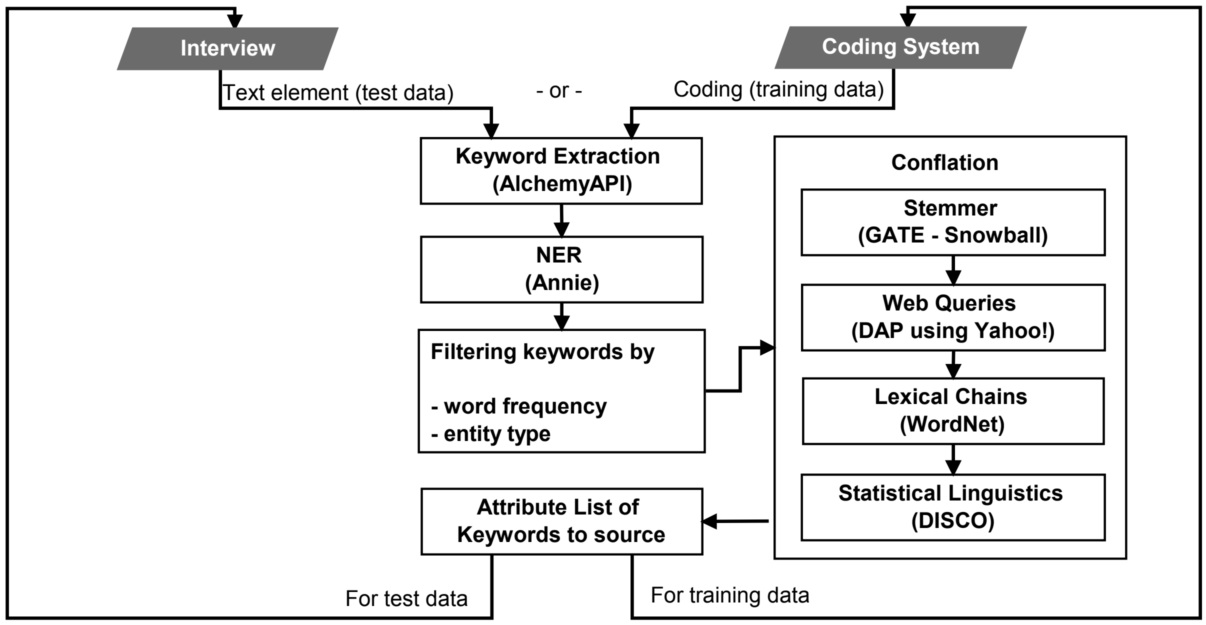

Figure 2: NLP Pipeline

For each code we look up the coded text parts within the training data and use the AlchemyAPI keyword extraction service on each coding. The keywords extracted from each coding will be assigned as semantic context to its corresponding code and can later be used to find more semantically similar text as new evidence for this particular code within our test data. The same process is used to determine the semantic context of each interview element.

2.4.3.2 Keyword Extraction

Keyword extraction is used to identify semantically significant words in a text. In Figure 3 we present an exemplary result of our data after keyword extraction has been performed by AlchemyAPI. Highlighted in red are the keywords extracted.

> *[...] first of all is to create **trust** to give them the **feeling** they are working on their **baby**, then we had some **measures** regarding **payment**, so we started to pay our key **people** more, sufficiently more than we pay **people** who we thought -- it might be not good but it's not **catastrophe** they leave us and other **measures** for **example** **Chinese** are very -- **let**'s say they have their **network** and you have to create a **network** within the **company** so you have to go out with them for **dinner** for -- you have to be interested in their **family issues** and so on, so creating **network**, creating **trust**, giving them the amount of **money** they -- not they want to have they always claim they want to have more but at least an amount of **money** they said okay, if I now leave I leave my **comfort zone** for **let**'s say 20% more -- it's -- will I do this, because Chinese have some **comfort zone**. And in China if you keep someone in the **company** who until he is 40 -- or **let**'s say 35 to 40 and you haven't -- and you want to keep him and there are no **big issues** in the **companies** between **people** and him, then he would say I would stay forever because you have to face the **fact** that -- I don't know if it's still the **case** in this -- in every **company** but in **China people** get very high **salary** at the beginning and our **competitors** in China start to reduce **salary**, when the **people** cross the 40 years border. So then you will not get more **money** -- if you have a **Chinese employer***

Figure 3: AlchemyAPI Results

This is an example of text which has been manually coded with the code *Turnover,* which we were able to replicate using our methodology. This example shows the significance of attributing actual semantic context to a code. Some very simple approaches of existing autocoding features in current CAQDAS tools support pattern matching for occurrences of a code's name within the text. In slightly more sophisticated mechanisms synonyms are detected as well.

In this instance such a technique would have lead to a false application of the code *Trust,* since the word trust occurs twice. We assume that the researcher who performed the manual coding did not apply the code *Trust* in this instance, because most codings for this code have the semantic context of trust or distrust of German- / US- Developers towards collaboration with a Chinese development team. So the code *Trust* has a more specific semantic context than the word trust which happens to occur in this part of the interview. In this example however the term trust occurs in the context of a motivational tool to encourage a good working morale.

Although the word trust is highlighted as a keyword, our algorithm detects that most of the text is about money, payment and salary, and because these concepts are closely connected to the code *Turnover*, our algorithm correctly applies the code *Turnover*.

2.4.3.3 Named Entity Recognition

The next step in our processing pipeline is to perform named entity recognition (NER) using the Annie pipeline, which is available as one of the core plugins for the GATE text processing framework. The entities we are interested in identifying are organizations, locations and persons. Our tool can be configured to include or ignore keywords labeled as one of these when determining which code is most likely to be applied for a text segment.

2.4.3.4 Conflation and Elimination

An important part of our autocoding algorithm are the different conflation methods. Through word conflation two words that have been conflated are treated as two instances of the same entity. This allows us to match semantically similar codes to a text, even though the vocabulary within the training data was different from the one used in the test data.

Conflation can be performed on derivations of the word, as well as semantic relationships from a lexical database. The more coarse grain our conflation methods are, the more information will usually be lost, but more connections between codes and text may be uncovered. The drawback of the lost information is that these new connections may be based on false assumptions. Conflating "China" with "china" draws a connection between porcelain and a country, which although there may be a etymological connection, spans a significant semantic distance.

All our coding methods have been tested in conjunction with conflation through lexical chains, stemming and taxonomies created through web queries.

For lexical chains we used WordNet[3], for stemming we chose the snowball stemmer integrated into the GATE framework[4], which is based on the Porter stemmer, and for conflation through web queries we used the Doubly-Anchored Pattern (DAP) (Kozereva, Riloff & Hovy, 2008) with the Yahoo! web-search. Implementation details on these methods are described in 3.2.1.2 Conflation.

The autocoding process is performed for all interview element types (Questions, Answers, Meta-Data). This is helpful because a question may sometimes give a more precise hint about the semantic context of the answer. Therefore if we identify a coding in a question we then increase the probability of the same code being applied to the following answer.

After each interview element has been processed this way, there is a high probability that too many codes have been assigned. To counter this effect we try to eliminate the least likely candidates among all assigned codings. Criteria for this selection are mainly the length of the text to be coded and the score of the assigned code. Codes assigned to questions or meta data will also be deleted during this step.

3 http://wordnet.princeton.edu/
4 http://gate.ac.uk/

2.4.4 Autocoding

The autocoding process is outlined in Figure 4

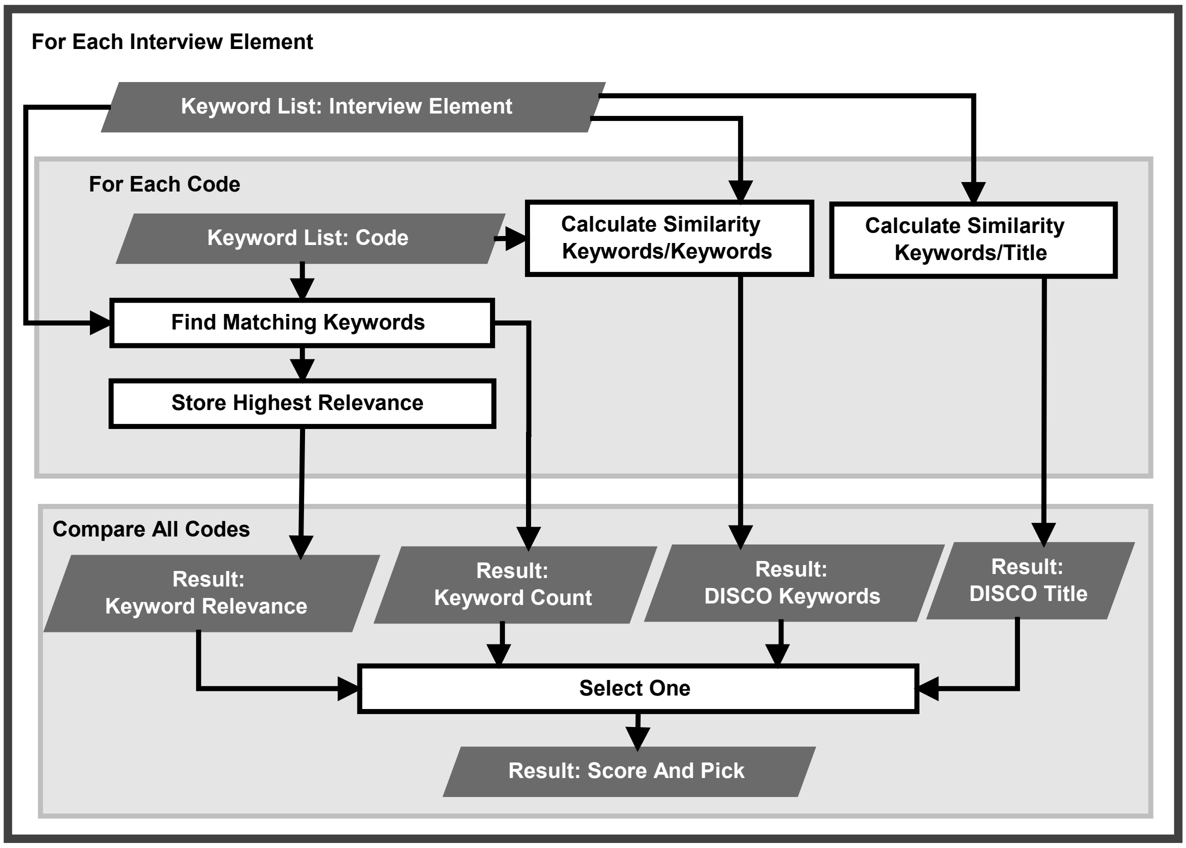

Figure 4: Autocoding

During our autocoding process we match each interview element with a code that is considered to be semantically closest to the content of the text. At this stage our data is pre-processed, parsed preserving meta-data, keywords were extracted from each interview element as well as for each coding of every code, and these keywords were filtered and conflated by user-settings.

Our coding algorithm now has two main input parameters:

1. An interview element, containing the actual text, meta data on which kind of text it is (Question / Answer / Meta) and a list of keywords associated with this text.
2. A set of codes (the coding system), each containing a set of codings and a list of keywords associated with this code.

The goal of our algorithm is to select the correct code from the coding system (2) to assign to the interview element (1) or to select none, if the algorithm determines the semantic similarity to any already coded text parts to be below a certain threshold. We set this threshold for each individual method low and eliminate more codings after the whole document has been processed. This method was chosen to utilize even weak semantic connections when gauging the agreement of our different methods.

2.4.4.1 Coding Methods

To calculate the semantic similarity of an interview element to a set of codings of each code within the training data, and thus to the code itself, we experimented with different methods and technologies, and ultimately implemented a weighting scheme to leverage the benefits of some of these techniques under different circumstances.

Keyword Count

Our first approach was to compare the list of keywords generated for the interview element to each list of keywords assigned to each code, and count the number of matching keywords. We would then normalize the count, to account for the fact that codes which have been assigned more frequently are more likely to contain matching keywords. Through the elimination of this factor we accounted for an disproportional skew towards the more common codes. We would then assign the code which had the highest score.

Keyword Relevance

The keyword extraction service we used provides a relevance score for each keyword, which we used to assign the code with the most relevant matching keyword. During our experimentation we found that this method profits, similar to `Keyword Count`, by factoring in the size of the text to code and amount of existing evidence of a code.

DISCO Title

We would like to be able to determine how similar a text is to parts of our training data, even when their vocabulary is completely disjunct. In one attempt to achieve this we employed the tool DISCO[5] by linguatools. To determine the semantic distance of two words DISCO uses statistical analysis on large corpora. We used DISCO in combination with the British National Corpus (BNC)[6]. DISCO calculates the similarity of two words either by counting and weighing the co-occurrences of the two words in a 3-word window within the training corpus or by measuring the distributional similarity (Kolb, 2009). We tried both measures in our context and then focused on the similarity by co-occurrences, because it performed better.

We calculated the similarities of each keyword assigned to an interview element to the name of each code. From these values we calculated a normalized score for each code and assigned the one with the highest score. A significant advantage of this approach is, that no training data is needed to perform the autocoding.

DISCO Keyword

We extended our first approach using DISCO, to include the training data. The similarity between each keyword of each code and each keyword in the text are calculated and normalized.

DISCO Inflation

In another approach to extend the vocabulary of a keyword list we added co-occurring words within the BNC extracted through DISCO, and subsequently applied the Keyword Count Method.

5 http://www.linguatools.de/disco/disco_en.html
6 http://www.natcorp.ox.ac.uk/

2.5 Used Data Sources

2.5.1 Interviews

For our research we focused on two different sets of interviews. Both were performed in English, were transcribed, and coded using MAXQDA.

2.5.1.1 German/Chinese and American/Chinese SWE Collaborations

Interview Set 1 is comprised of 6 unstructured interviews about the intercultural challenges that ensue when collaborating with Chinese software development teams from the perspective of German and US developers. Five of these interviews are available to us as word by word transcriptions, each between 15 and 28 pages long. For the sixth interview, our coded document is a two page summary. A total of 441 codings have been applied manually and there are 9 codes to be applied by our tool.

The original manual coding was performed by Zaghloul (2014). The coding style presented some challenges to us. We consider this data to be a relevant real-life test for our approach, because a viable autocoding approach should be able to adapt to different coding styles. The challenges, how we approached them, and the limitations that were revealed through them are discussed in 2.7.2 Limitations.

2.5.1.2 Life Satisfaction

Interview Set 2 is comprised of 11 structured interviews about general life satisfaction performed by ResearchTalk Inc. The data is available as an example project of MAXQDA. Each of these interviews is between 2 and 4 pages long, and a total of 315 codes have been applied. We ignored the code *Autocode: satisfaction* (37 occurrences) since it is applied to every occurrence of the word satisfaction, thus it is a trivial case we are not interested in.

We use the second data set to test if an approach that shows promise during our experimentation is successful only due to specifics of our data or indicates an improved process. This set was also used to determine the benefits of analyzing a structured interview.

2.5.2 Knowledge Bases

As knowledge resources we utilized the following sources

- WordNet as a lexical database to build lexical chains connecting keywords in our training data with keywords in out training data.
- British National Corpus (BNC) for statistical analysis, which contains 100 million English words from the late 20th century, both written and spoken (Aston & Burnard, 1998). We process this data using DISCO to retrieve the distributional similarity of two words (Kolb, 2008).
- Yahoo! web search for conflation based on the Double-Achored Pattern (DAP)

2.6 Research Results

2.6.1 Metrics

To measure the success of our autocoding algorithm we decided to use *recall* and *precision*. Recall describes the percentage of manual codes, that could be replicated by the algorithm.

$$Recall = \frac{|\{\text{manual codes}\} \cap \{\text{automated codes}\}|}{|\{\text{manual codes}\}|} \times 100\%$$

Precision measures the percentage of auto-coded codes that are correctly applied according to our manual codings.

$$Precision = \frac{|\{\text{manual codes}\} \cap \{\text{automated codes}\}|}{|\{\text{automated codes}\}|} \times 100\%$$

Both metrics are, to some extent, inversely correlated. For example: coding each interview element with all possible codes will yield 100% recall, but a very low precision value, while applying very few, but only correct, codes compared to the amount of manual codes will yield 100% precision and a very low recall value.

In our scenario we consider an ideal algorithm to be able to apply all codes that a human coder would, but would not apply any additional codes, thus yielding 100% for both recall and precision.

2.6.2 General Results

Table 2 shows the results of our autocoding algorithm applied to interview set 1. The coded documents were interview I01 and I02. For each, we used all other interviews as training data.

Code	**Recall (%)**	**Precision (%)**	**# Assigned**		**# training instances I03 - I06**
			manual	**autocode**	
Communication Culture	88,89	34,78	9	23	87
Tasks Coverage	66,67	33,33	3	6	31
Trust	53,33	57,14	15	14	41
Turnover	22,22	66,67	9	3	11
Time Difference	66,67	100	3	2	12
Project Examples	0	0	2	0	0
Corporate Strategy	0	0	7	0	0
Project/Product	100	100	0	0	22
Team Structure	100	0	0	1	12
Total	**45,83**	**45,83**	**48**	**48**	**216**

Table 2: Recall & Precision: Interview Set 1

The codes Project Examples and Corporate Strategies show an inherent drawback of a machine learning algorithm in conjunction with our test data. These codes appeared exclusively in interview I01. Therefore, when our data was divided into training and test data, there was no possibility for instances to be included in both.

Table 4 shows the the same results for the top level codes of interview set 2. The code Challenges is another example of a code that could never be replicated by a machine learning algorithm, because it occurred only in a single instance within the data.

Code	**Recall (%)**	**Precision (%)**	**# Assigned**	
			manual	**autocode**
Day-to-Day Issues	79,31%	67,64%	29	34
Interview Guideline Topics	70,00%	43,75%	40	64
Key Quotes	0,00%	0,00%	6	1
People	14,29%	33,33%	7	3
Challenges	0,00%	0,00%	1	0
Total	**62,65%**	**50,98%**	**83**	**102**

Table 3: Recall & Precision: Interview Set 2 (Top Level Coding)

Since the abstraction level in these top level codes were not the same as in interview set 1, we measured the performance for autocoding of the lower level codes. The only subcodes of the codes listed in Table 3 we did not consider were the interview guideline topics.

Code	**Recall (%)**	**Precision (%)**	**# Assigned**	
			manual	**autocode**
Emotions	33,33	33,33	6	6
Education	00,00	00,00	11	9
Interests	100	20,00	1	5
Money and Financial Issues	00,00	100	1	0
Religion and Spirituality	00,00	100	3	0
Significantly Positive	64,71	35,48	17	31
People	50,00	28,57	4	7
Friends	71,43	26,32	7	19
Parents	80,00	66,67	5	6
Partner	00,00	00,00	4	1
Siblings	100	100	0	0
Key Quotes	36,36	23,53	11	17
Pivotal Moments	00,00	100	1	0
Total	**69,05**	**29,71**	**42**	**101**

Table 4: Recall & Precision: Interview Set 2 (Deep Level Coding)

2.6.3 Variables and Methods for Semantic Matching

2.6.3.1 Keyword Frequencies

Figure 5 shows the results of our experimentation with the frequency threshold under which a keyword is considered during the autocoding. If a keyword is associated with more codes than the value of the threshold, then it will be ignored. So a low frequency threshold means, that keywords have to be more specific. It is an apparent trend, that our Keyword Count method performs better if we allow keywords to be associated with a higher number of codes, although the advantage slightly shrinks, if we allow a keyword to be associated with almost every code. On the other hand we observed, that methods that draw on each keyword's relevance, or its semantic meaning, show an opposite reaction to a change of the threshold variable. The DISCO Similarity method is unaffected by this variable, since it disregards the training data. Another baseline we added to compare the performance of our Keyword Count method is the method labeled Word Count. Word Count works exactly like Keyword Count, except that every word is treated as a keyword. We observed, that using a sophisticated keyword extraction method can double the number of correctly identified codings.

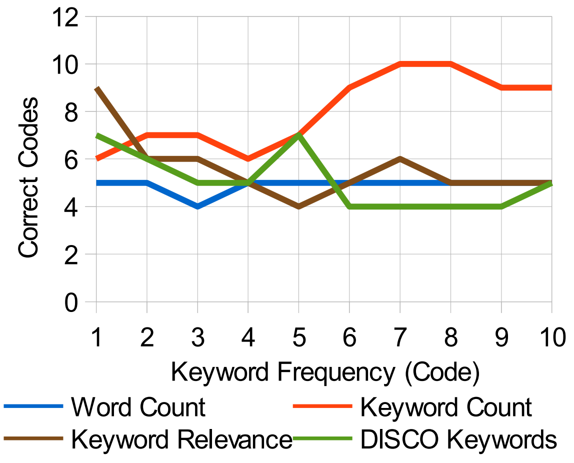

Figure 5: Keyword Frequency (Code)

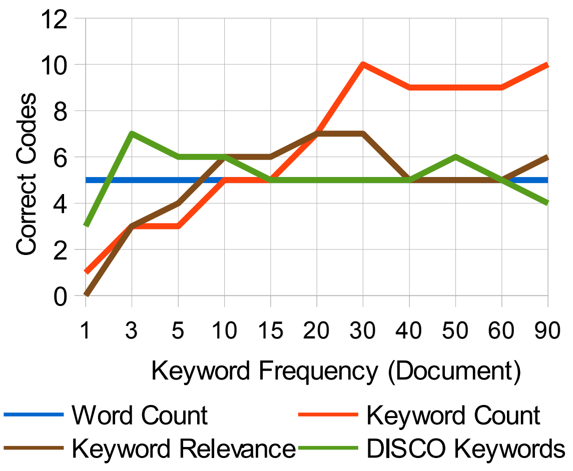

Figure 6: Keyword Frequency (Document)

We also measured the impact of limiting the frequency of a word in regards to its overall occurrence, which is presented in Figure 6. While we found the parameter for inter-code word frequencies to be a useful variable to tune the algorithm to a specific method, excluding a word because it is generally common in all documents did not have a positive impact in our scenario. In the test runs we measured for this criterion, the only method slightly benefiting from a lower threshold (between 3 and 10) was DISCO Keywords.

2.6.3.2 Named Entities

In an attempt to slightly tailor the parameters of the algorithm more closely to our semantic context, we experimented with removing specific entities that are possibly too common in our context. However, the results of measuring the impact of named entity keywords showed, overall, the opposite trend.

In Figure 7 & Figure 8 we measured the recall of our different autocoding strategies while including none of the entity types, only organizations, only locations, and both locations and organizations.

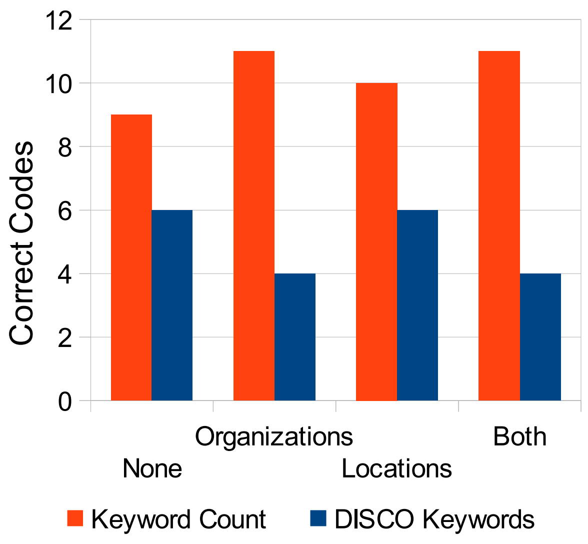

Figure 8: Impact of NER - I01

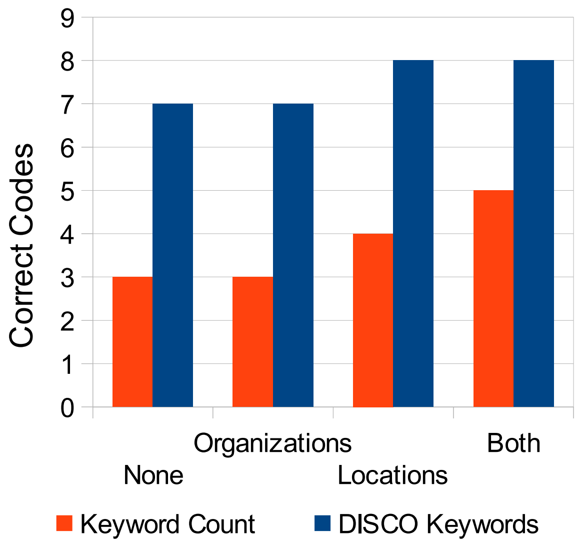

Figure 7: Impact of NER - I02

The methods Keyword Count and DISCO Keywords were the only methods affected, so all other methods are excluded from the f igures above.

2.6.3.3 Stemming

In Figure 21 we present the impact of stemming of the keywords on our autocoding algorithm. The results show a significant variance, but aggregated we see a slight advantage for including stemming. This method is clearly not suited for DISCO Similarity, but improved recall for all other methods by 9.23%.

Special attention has to be given to the impact of stemming on the DISCO Keywords method. The data is inconclusive, since it shows indications for both significant improvements for interviews I03 and I04, and negative effects for interviews I01, I02 and I05. The inconclusiveness at this point is not unexpected, since for this method the stemmed keywords have to be identified by DISCO to calculate their statistical semantic distance. Since the stem does not necessarily have to be a correct English word, some may no longer be identified by DISCO after stemming, which would explain the detrimental effect. In some cases stemming may also show the opposite effect, when a specific derivation of a word was not present in the analyzed corpus. When disregarding the results for DISCO Keywords due to its volatility, the advantage of using stemming is only 7,47%, but with less variance.

The error bars show the range of our test results when the method was only applied to interviews for which it showed improvements, or only to those where the method had a detrimental effect on the results.

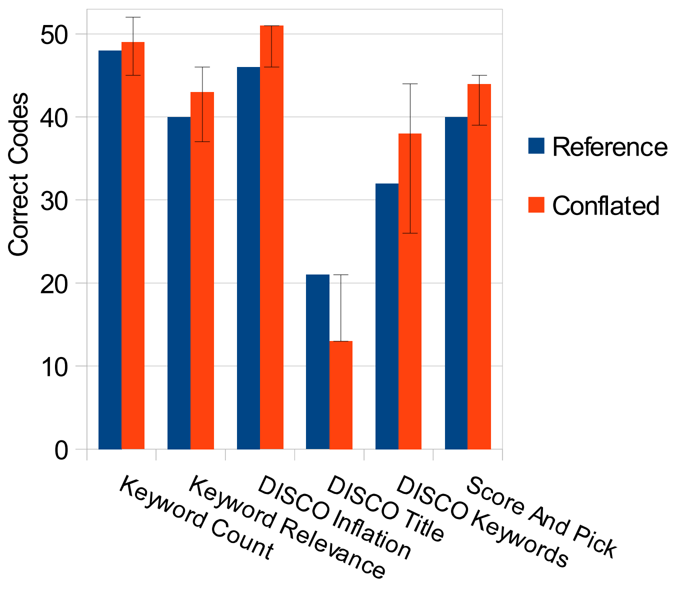

Figure 9: Stemming

2.6.3.4 Web Queries

For our conflation approach through web queries we utilized Yahoo! web search. Figure 10 shows the results, when we used this technique exclusively on the code title, and Figure 11 shows the results of conflating all keywords through web queries.

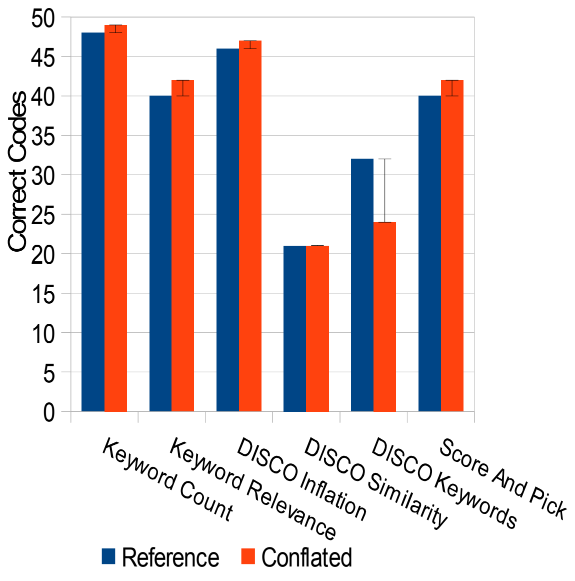

Figure 10: Web Queries (Title)

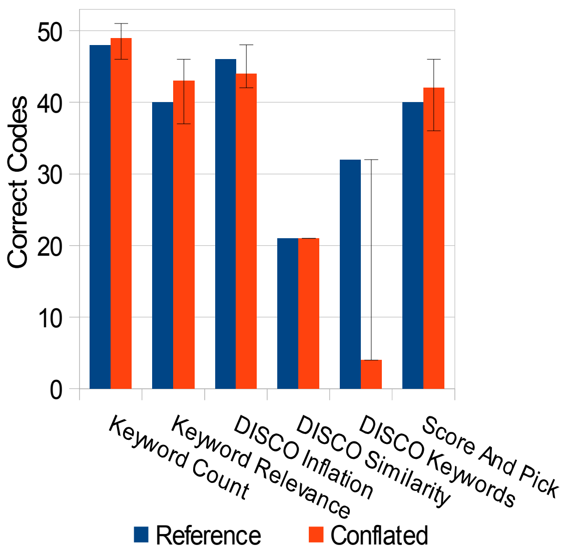

Figure 11: Web Queries (Keywords)

2.6.3.5 WordNet

For Figure 23 and Figure 13 we conflated all keywords with its hyponyms, hypernyms, meronyms, holonyms and derivationally different forms. While this conflation alone does not provide a significant improvement, it showed promising results for interviews that were already processed using stemming and web queries further improving performance by up to 17.8%. This effect is in line with an extensive series of experiments using other combinations of WordNet relationships.

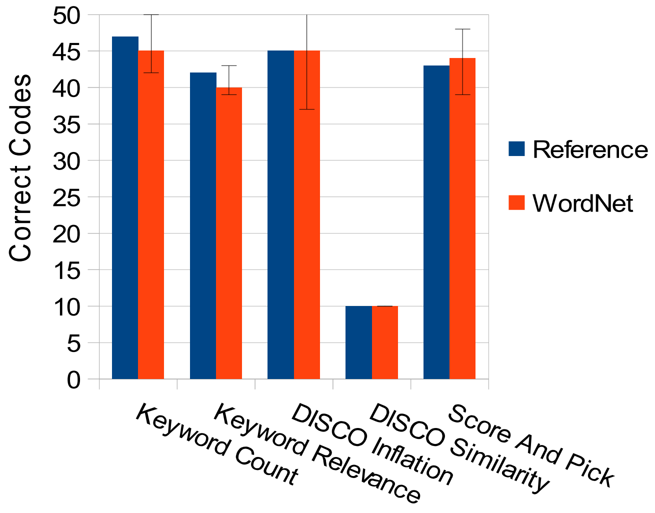

Figure 12: WordNet Conflation

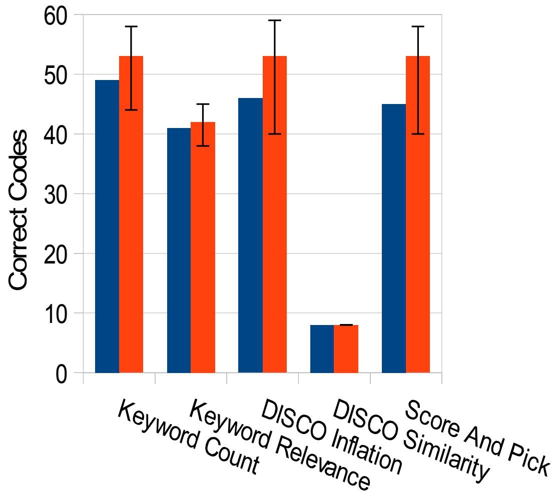

Figure 13: WordNet with Stemming & DAP

2.6.3.6 Consideration of Question-Answer Sequence

To utilize the structure of an interview we considered the codes which were calculated for a question when coding the corresponding answer. The results are shown in Figure 14: Utilization of Q&A Sequence. For a code which was calculated for a question to be considered in the following answer we required that 3 out of our 5 methods were in agreement. Depending on this criterion the results varied, showing that interviews I01 and I02 would benefit from different settings than interviews I03, I04 and I05. Overall a small, but significantly positive trend was observed using different criteria.

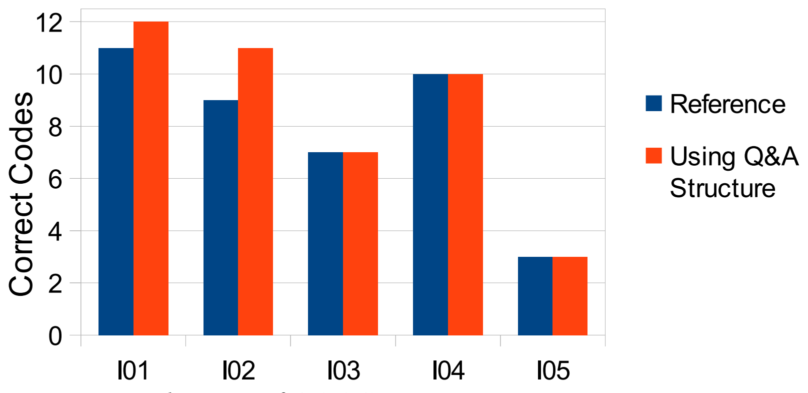

Figure 14: Utilization of Q&A Sequence

2.7 Results Discussion

2.7.1 Results interpretation

It may be argued that our metric for a successful algorithm is not ideal for measuring the quality of the coding. A different coding is not necessarily better or worse, and we did find instances, where the applied auto-coding made sense subjectively, but still did not match the human coding. Such an instance would negatively impact our precision metric, but it may be regarded as a successful categorization of the semantic context of the text. It is entirely possible, that the algorithm finds new evidence for a code, that the human coder did not. This would be a desirable effect of using auto-coding, besides improving the speed and fluency of the coding process. So in our efforts to replicate human behaviour it is important to realize, that replicating human error is not in the researchers interest. But due to its simplicity, its reasonable conclusiveness and significance to our research question, the replication of the human coder remained our gold standard. A detailed discussion of intercoder agreement in QDA can be found in Krippendorff (2011). For this research however, we kept the standard metrics for IR.

For interview set 2 it was particularly apparent that certain codes like *Friends* and *Relationships* are difficult to distinguish, because both concepts were often mentioned in the same paragraphs, although they were not both coded. In such instances our algorithm's selection was frequently different from the one the researcher performing the manual coding made. Subjectively we faced a similar difficulty like the algorithm when trying to recreate the original coding in these instances manually.

2.7.2 Limitations

During our research process we identified some critical limitations to autocoding in general and our machine learning approach specifically. An analysis of our test data revealed that inconsistencies of the manual coding hindered the automatic processing of the results. There was, for example, no homogenous coding system in place for all interviews within data set 1. Instead there were similarly worded codes for each of the interviews. So before we were able to process the data further we had to unify this heterogeneous set of codes to one coding system. It may be argued, that these kind of inconsistencies stem from bad coding practices, and therefore are not relevant to the applicability of our autocoding approach. However, these kind of bad practices are not prevented by using current CAQDAS packages. This may not be the most common problem, but we suggest that it is exemplary for the fact, that any autocoding software which relies on a machine learning algorithm would require training data that was generated using some standardized method. Researchers may have their reasons for deviating from standard best practices and may reject any process that forces them to adjust their way of coding to conform to standards. Further, the data has to be available in a standardized format. Although a standardized XML representation would be desirable, as discussed by Murr (2000), no such OpenQD standard has been widely used. Still, most CAQDAS packages support an XML export, and the problem may thus be solved technologically by implementing an adapter for those XML structures, to convert them for further processing as we have done in our research.

Another limitation we realized through analysis of our test data was the code hierarchy. Although it may be possible to handle code hierarchies with a more sophisticated approach to autocoding, we did not consider hierarchies in this research. To compensate for this we modified our data set 1 by flattening the hierarchy and mapped every occurrence of a coding to one of the 9 top-level codes. For set 2 we also performed autocoding using the exact code structure of the coding system, but we did not utilize the hierarchy in our algorithm. Also the granularity of possible codings was significantly simplified. Our algorithm would only assign one code to a complete paragraph, while the the manual codings could start and end at any position within the text, which would be extremely difficult to replicate.

A third observation is related to a general limitation of any machine learning algorithm. Since the algorithm requires some minimal amount of training data it is obvious that any code, which only has very few occurrences within the training data is very hard to be detected by a machine learning algorithm.

2.8 Conclusions

We have assessed the applicability of a wide range of NLP methods in an experimental autocoding system for both in-depth interviews and structured interviews. In our setup we achieved 45,83% recall and precision for our unstructured interview set and 62%-69% recall and 29,71%-50,98% precision on the structured interviews depending of the abstraction level of the codes. We have observed beneficial effect for using word conflation through stemming, DAP, and lexical chains. Also helpful was the consideration of the question-answer sequence of interviews.

No positive effect was observed for using word frequencies related to the documents, and filtering keywords by their entity type.

Our research demonstrates the exceptional adaptability of a machine learning approach for autocoding. Although our final results show a significantly lower coding agreement compared to a rule-based approach, the effort of adapting the algorithm to a specific interview are far lower. This feature is also supported by the easy integration of the AlchemyAPI keyword extraction service.

Based on our experimentations with AlchemyAPI, general-purpose keyword extraction methods available perform well in this environment with a consistent level of abstraction. We see little need to re-invent the wheel for an application of this technology to QDA.

Overall we are convinced that using high-level NLP in a machine learning approach to autocoding shows promise for a real-life application and that our results warrant further research on the subject.

3. Elaboration of Research Chapter

3.1 Qualitative Data Analysis

3.1.1 Basic Principles of Qualitative Data Analysis

Qualitative research is a methodology that seeks to describe, understand and even predict human behaviour. It is a particularly well suited method to investigate the reasoning of participants and trying to understand *what*, *why* and *how* they think, act or experience a phenomenon. Therefore the main application of this kind of research is typically sociology and market research.

The research question for qualitative research is generally decidedly different from quantitative research. While quantitative studies usually aim to answer very specific questions by means of statistical analysis of large test-data (hypothesis-testing), qualitative research is usually much more open. The goal is to build a theory, that emerges and will be refined during the research process.

This difference is described in detail by Auerbach & Silverstein (2003), who labeled quantitative research as *hypothesis-testing* research and qualitative research as *hypothesis-generating* research.

After an area of interest is identified the researcher has to choose a method for gathering data. The most common approach, which we examine in our experimentations is to use in-depth interviews. Other methods, that we will not present in detail include focus groups, which are frequently used for market research, observation as an outsider, participation and immersion where the researcher attempts to replicate the experiences first hand by surrounding himself in the setting he wishes to study.

3.1.2 Coding

The resulting theory is built through an in depth analysis of the gathered data. A vital step in the analysis process is called coding. Strauss (1987, p.27) claims that "*The excellence of the research rests in large part on the excellence of the coding"*. A code is a conceptual label, linked to a particular pattern of thought, action or behaviour. A high quality coding should feature codes that do not remain a mere description of the labeled pattern. To achieve this distinction it is necessary to provide context to each code. Our test data for example includes the code *Trust* but the code is not as general to include every occurrence of trust in the data. Instead the code contains implicit context concerning of who trusts whom, as well as a rough limitation for the subject of the trust. In manual codings this context is frequently provided through memos.

In this work the term *code* will always describe the abstract concept, while the term *coding* will describe a concrete manifestation of this concept within the coded text.
A *coding-system* denotes a, typically hierarchically structured, set of codes which builds the basis for the theory.

3.1.3 Grounded Theory

Amongst the variety of methodologies for QDA, we focus on the grounded theory approach, since our main goal is concerned with coding techniques which is one of the most pivotal features for theory building using grounded theory.

The grounded theory methodology was first developed by Glaser and Strauss (1967). In their original publication grounded theory was not categorized as a qualitative research method. Instead it was suggested, that any data could be analyzed with a grounded theory approach. Later, this was part of a critical debate between both founders of the methodology.

Besides the divide between Glaser's and Strauss's approach, there are many variations of grounded theory today and most are applied in the context of qualitative data analysis.

The main idea of this approach is, that the theory is rooted (*grounded*) exclusively in the data, and is not influenced by theoretical preconceptions, such as an in-depth literature review, as it is common with other methods for qualitative data analysis. In the following we describe some core features defining a grounded theory approach (Glaser & Strauss, 1967; Charmaz 2006)

Concurrent data collection and analysis

Using grounded theory, a researcher would roughly identify an area of interest and, if he chooses interviews as the tool for data collection, construct an initial interview guideline. While conducting the first series of interviews this initial guideline should evolve as an initial theory emerges, thus the data collection and analysis process are interconnected and should be performed simultaneously.

Theoretical sensitivity: Constructing codes and categories from data instead of preconceptions

The researcher should start the process with as few preconceived concepts as possible in an attempt to identify what is significant to the interviewee. This concept seems particularly well embodied by a computer assisted process, since the algorithm is neither biased by experience, nor does it hold any preconceptions. On the other hand we experienced during our brief experimentations on the subject of automated code creation, that the construction of new codes using machine learning algorithms may yet be limited by the current state of NLP capabilities. This remains to be subject to future research.

Refining the theory iteratively in each iteration of data collection and analysis

Constructing the theory should be an iterative process repeating the gathering of new data and its concurrent analysis until no new codes emerge, and the existing code system is no longer subject to substantial change.

Writing memos, to define the properties of codes and document their relationships

This is an important aspect for the documentation the research process. Our interview set about intercultural challenges, for example, did not include any memos, which made it difficult to understand the coder's intentions in several cases, and left us guessing with nothing but our own interpretation of the data. Our second interview set on the other hand featured memos, which provided a much clearer understanding of the coding system, and revealed information on the relationships between codes, that would otherwise have remained hidden.

Ideally the information contained in such memos would contribute to an autocoding algorithm. Since one of our main sets of test data did not contain any memos we did not investigate this possibility further. However, based on analysis on the memo data available to us, it may be possible to use NLP techniques to extract some of the context, particularly relationships between two codes. Typically when both codes are mentioned in a text, state-of-the-art functional part-of-speech classifiers are capable to extract the relationship between the two.

Use of theoretical sampling

The selection of what data to gather should be sensitive to how the theory evolves. After each iteration of data collection and analysis it should be re-evaluated what data should be collected next. Practical considerations however frequently limit the choice of who to interview next. On first glance this aspect of grounded theory seems incompatible with an autocoding technique that requires a considerable amount of data to work. However, using an autocoding algorithm to confirm existing theories by finding new evidence or rejecting codes due to a lack of evidence during each iteration step seems like a reasonable task and is what we tried to accomplish in our experiments.

Literature review is performed after an independent analysis

Many less rigid approaches to grounded theory accept that some form of initial research is often necessary to narrow the direction of the study. Limited resources, such as time and access to data sources, may also influence the decision how open a study will be to unexpected theories emerging.

3.1.3.1 Objectivistic vs. Constructivistic Approach

Another dimension that divides researchers conducting studies using grounded theory is an objectivistic approach versus a constructivist approach. Although many authors acknowledge the distinction between objectivistic and constructivistic grounded theory, Glaser (2002) categorized constructivist grounded theory as a misnomer and argued that if constructivist data existed at all, it would only constitute a very small part of the data that is used in conjunction with grounded theory. Others like Denzin & Lincoln (2005) call for the return of constructivistic elements, arguing that *"A constructivist approach emphasizes the studied phenomenon rather than the methods of studying it"*

In a constructivist approach the resulting theory is not grounded exclusively in the data, but the researcher himself is an active part in its creation. Denzin & Lincoln (2003) described causalities in constructivist grounded theory as suggestive, incomplete and indeterminate.

Objectivistic grounded theory on the other hand takes on an epistemological perspective. It embodies a positivistic philosophy, meaning it is rooted in the belief of the existence of an external truth, that may be analyzed and discovered using scientific methods. Researchers adhering to this methodology aim to discover a theory that reflects the external reality and is reliable, testable, and reproducible by following the same scientific guidelines. One example of this approach including instructional guidelines was formulated by Strauss & Corbin (1990), although in a revised 2nd edition published in 1998 they were less prescriptive. Ultimately an objectivistic theorist does not only seek understanding of the studied phenomenon, but also strives to derive predictions from the generated theory.

Although we do not dismiss the constructionist approach to be irrelevant, we realize that approaching grounded theory from a computer science perspective, may only support the objectivistic theorist. Using NLP techniques extends the more rigid scientific methodology of this approach well, and it certainly contributes to the goal of objectivistic research to be verifiable and reproducible.

3.1.4 Interview Analysis

Most of the relevant sociological research methods apply to QDA in general, independent of the used method for data collection. Some factors that impact a ML based autocoding approach, however, are specific to the type of data being collected.

Two factors vital to any IR system are the amount of available data and the type of text, its linguistic structure and the vocabulary that is used. We will address both issues in the following subchapters.

3.1.4.1 Sample Size

One of the important factors for our research, that is directly impacted by the type of data being processed, is the sample size of the data our algorithm should process.

If general applicability for typical coding tasks is the goal, then a viable algorithm should be capable to perform autocoding on a typical amount of data. To this effect we performed a literature review to evaluate what may be considered typical in this context.

In most literature on QDA, and as described in the previous chapters, the collection of new data, and thus the conduction of new interviews should continue until theoretical saturation is achieved. Although this criterion is frequently presented as a milestone at which data collection should conclude, the description of this concept frequently remains vague, without any guidelines to measure saturation.

Guest, Bunce & Johnson (2006) studied the question of how many interviews should be considered enough to reach theoretical saturation. They conducted a field study for which they conducted 60 interviews and correlated the number of interviews over the course of the research with the number of new codes that emerged after an analysis of the gathered data.

They showed that 73% of the final codes were already established after analysis of the first 6 interviews, and 92% were established after the second round of analysis with additional 6 interviews. From the remaining 8 rounds of analysis the remaining 8% emerged, of which over a third was identified in the third round of analysis with a total of 18 interviews.

Another significant correlation they presented were the the changes to the code definitions to the number of interviews conducted. The results showed that the large majority of changes to the definitions were performed during the second round of analysis, when the 7th-12th were added to the data collection. At this stage 58% of the total definition changes were made. The last change was made after 36 interviews, although 3 new codes emerged through analysis of interviews 37 – 60.

Finally they used Cronbach's alpha measure to show that the thematic significance of the codes identified early in the research process tend to be the most significant.

An overview of the problem of data saturation is also presented by Francis et al. (2010). They propose starting with an *initial analysis sample*, and a *stopping criterion,* which is the number of consecutive interviews that have to be analysed following the *initial analysis sample*, without new themes or shared ideas emerging from the data. Both measures have to be defined a-priori, depending on the complexity of the studied phenomenon. Since both measures are domain dependent, a generalization is difficult to make, but conducting a field test with this criterion they started with an *initial analysis sample* of 10 and a *stopping criterion* of 3, which resulted in data saturation after the 17th interview. They conclude, that the 10 + 3 rule for their stopping criterion may be regarded as a reasonable start criterion if there no specific indications on the required sample size within the data.

3.1.4.2 Transcription

It is important to keep in mind, that the main concern of any computer assisted QDA process is not to simply process text, but to assist the study of social phenomenon. The actual data source is the interview itself and it is crucial to be aware of which features of this data source are available for text processing in the transcripts, which may be missing, and how accurately and detailed they convey the information contained in the interview.

As Bucholtz (2000) noted "A reflexive transcription practice, as part of a reflexive discourse analysis, requires awareness and acknowledgment of the limitations of one's own transcriptional choices". This statement also applies to the limitations, as well as opportunities, any chosen transcription method will imply for the possibilities of further processing through NLP technologies, like we employ for an autocoding algorithm. To this effect we studied the common characteristics of interview transcripts relevant to NLP, as well as the impact of specific transcription choices available to the qualitative researcher.

An interviewer taking notes during the interview is commonly considered too much of a distraction for the interviewer and usually leads to unnecessary bias due to the interviewer's familiarity with the data. From this observation arose the need for transcriptions as a form of translating information captured in audio or audio-visual form into a textual representation for further processing and analysis. Besides letting all participants of an interview focus on the actual discussions, audio recordings serve an additional purpose for verification of the data veracity and the rigor of the research process (Halcomb & Davidson, 2006).

There are no commonly accepted guidelines for transcriptions, although some research on transcription guidelines has been performed. Particularly we would like to mention the publications by McLellan, MacQueen & Neidig (2002) and Davidson (2009).

Bucholtz (2000) made a notable distinction between what she called *naturalized transcriptions* and *denaturalized transcriptions.* Naturalized transcriptions favour features of written discourse and thus apply conventions of formal written language to the transcript, such as correct punctuation and structuring the text according to layout conventions. Such information is subjectively interpreted from the audio recording and added to the transcript. Denaturalized transcriptions on the other hand disregard some conventions of written text to retain certain features of the original spoken word. Oral details that may be included in a denaturalized transcription include stuttering or hesitation fillers like *"err...", "ahh"* and acknowledging, encouraging or denying sound like *"uh-hum",* etc.

3.1.4.2.1 Selectiveness and Possible Features of Spoken Language for Transciptions

It is a widely acknowledged fact, that large parts of communication are non-verbal. Although it is difficult to quantify the portion of information that is contained in non-verbal expressions in any conversation, one of the most frequently cited statistics on this subject is the "7%-38%-55% rule". This rule attributes 7% of the information to the words, 38% to other vocal attributes and 55% to nonverbal communication. These percentages are based on two studies conducted by Albert Mehrabian (Mehrabian & Ferris, 1967; Mehrabian & Wiener, 1967). Even though these results should not be generalized, since the studies were conducted specifically on messages on feelings and attitudes, they show the amount of information that may possibly be lost under certain circumstances. Challenging the notion, that nonverbal communication carries the most information value, Rimé (1982) studied the elimination of visible behaviour from social interactions, and concluded that the abundance of nonverbal behaviour can be explained mostly due to its assistance to the speech encoding process.

Regardless of the quantification of the information value, it is reasonable to postulate that transformation, or translation as described by ten Have (2007), from a recording to a textual representation of the interview, will never preserve all information present in the original recording. The transcription process is therefore labeled as *selective*. Information on the original recording like timing, accentuation, intonation etc. may be selectively transcribed or omitted.

Depending on the selectivity transcripts may contain meta information that goes beyond a simple word-by-word transcription of what was said, but may also contain information on how it was said.

The following list of features is not necessarily complete, but gives an overview of the features of language which have been studied with regard to the inclusion in transcripts.

- Accentuation or intonation may be included. Rhetorical devices such as irony may often only be distinguished from a sincere statement through intonation
- If the interview was video taped, other non-linguistic features like facial expressions may also be part of a transcription
- Poland & Pederson (1998) also investigated the possibility of including silence in transcriptions. It is reasonable, that the intention behind certain expressions in the

spoken language may only be fully comprehended if attributed with phonetic information, or pauses made for special emphasis on a particular part of a sentence

- Jefferson (1983) studied the transcription of laughter
- Gaps and pauses
- Overlapping talk

There are three common reasons why a feature is ignored during transcription:

1. Overly complex transcripts tend to be more difficult to read (Ochs, 1979; Seale & Silverman, 1997)
2. The transcription of additional features occupies a significant amount of usually limited time resources. Britten (1995) quantified the effort required for transcription by stating that “each hour's worth of interview can take six or seven hours to transcribe, depending on the quality of the tape”
3. MacLean, Lynne, Meyer and Estable (2004) suggested, that the inclusion of every encouraging “uh-hum” in the transcripts would make the interviewer appear less articulate than he might actually be. Regardless of the actual value of “uh-hum”s for the analysis, which may or may not encourage or otherwise influence the interviewee, this is a reminder that personal complacency may play a role when the researcher performs the transcription of his own talk.

Besides these features, that are selectively left out in a transcription, there are also features, that are not commonly represented in text form. An audio recording might, for instance, feature overlapping speech from multiple participants of an interview, while text usually has a defined reading direction which leads to a strict order of all sentences. Simple text formation like italic, bold or capitalized formatting, would not suffice in this case but a more elaborate notation would be required to represent a measure for time for this scenario. Some suggestions on notation for features like overlapping talk, gaps and pauses, breathing and laughing have been defined by Hutchby and Wooffitt (2008)

Due to the selectivity of the process, it might be difficult to distinguish some of the approaches to transcribing from the actual analysis. There are varying opinions in the literature concerning the possibility of delegating the work of transcribing a recorded interview to a professional typist. McCracken (1988) advocated such a delegation and Holloway & Wheeler (2013) advised to weigh the convenience and time benefit against the cost of hiring a professional typist. Others like Hutchby and Wooffitt (2008) on the other hand claim, that this is not standard practice, and advise researchers to perform the transcription themselves as an integral part of the analysis. MacLean et al. (2004) suggest that the transcription may be performed by a professional transciptionist, who should be regarded as an integral part of the research team.

In any case, but of particular importance if the transcription is performed by someone other than the researcher, the transcription should introduce as little bias as possible, which may easily lead to a misinterpretation of a non-linguistic feature.

Another problem might rise through ambiguities of certain non-linguistic features that have to be interpreted in a specific context.

The loss of information generated by transcription is not necessarily harmful to the QDA process. Contrarily, since overly elaborate transcriptions are difficult to read and to analyze, Ochs (1979, p44) even suggested that "a more useful transcript is a more selective one". The usefulness of such data, however, may have to be re-evaluated in light of the new possibilities of computer assisted text processing using NLP. These capabilities simply were not available at the time, when Ochs claimed this very general statement.

It should be noted, that the inclusion of emotions during the coding stage are frequently considered. Saldana (2012) called such methods affective coding methods, aiming at the inclusion of subjective data such as emotions, values and judgments of participants in the coding system.

The importance of emotions in the context of QDA is also highlighted by Corbin and Strauss (2008) "One can't separate emotion from action; they are part of the same flow of events, one leading into the other". Following this line of thought an interesting aspect of coding with emotions is, for example, to examine the relationships between emotions and actions that trigger these emotions.

3.1.4.2.2 Common Characteristics

Despite the selectivity of the transcription process, some characteristics will be embodied in every interview transcript.

It is a widely accepted position that interview transcripts should be verbatim descriptions of what happened in the interview (McCracken, 1988; Poland, 1995), although Halcomb & Davidson (2006) challenged this view in their publication titled *Is verbatim transcription of interview data always necessary?*.

Fasick (1977) suggested that due to the costs associated with transcriptions, it would be more effective to analyze the tape recordings themselves and claimed that this practice would even improve the quality of the research. This viewpoint, however, is not shared by the majority of the literature. The second argument against a verbatim transcription put forth by Halcomb & Davidson (2006) is that the complexity of the transcription process is prune to human error. While this may be true, the same can be said about coding or preparing and conducting the interview.

One of the shared characteristics of verbatim transcripts is the use of informal language. This type of language has distinct characteristics, clearly separating it from formal written language. Some of these characteristics that are relevant to our NLP task include:

- Sentences do not necessarily have to maintain correct grammar, which severely impacts some NLP techniques like POS tagging.
- Abbreviations may be used that are unfamiliar to most formal dictionaries.
- Contractions are frequently used, which may have to be split into separate words to be correctly identified.

3.1.4.2.3 Impact of NLP

Some of the text characteristic present challenges to certain NLP methods, however many NLP methods are successfully applied for IR on social media, which may share many of these characteristics.

Concerning the selectivity we believe that many of the features, that are omitted to retain easy readability for the researcher may be included as meta data in a transcription that is intended to be processed using computer assisted analysis.

Certain attributes of the text may, for example, be annotated or formatted during the transcription stage in a way that is not visible in the final representation of the text, which is analyzed by the researcher. These parts may be transcribed, and when the information has been parsed as meta-data into a CAQDAS environment empowered by NLP technology, the information may be stripped from the document for a more readable representation. Generally speaking we suggest, that the data of the transcription does not necessarily have to be equal to the visual representation of the transcription. This distinction has not been made in previous research on transcriptions.

For all these meta data options a formal notation would have to be defined. This would require significant additional effort once for every research project as well as for each transcription. Besides this burden on the required effort, this would be a feasible approach, and if the definition of the notation is formalized in a format that is easily processed computationally, it would be possible to integrate this information in an autocoding algorithm using NLP techniques. To limit the amount of effort the researcher should limit the number of characteristics being transcribed by assessing, which may have significant impact on the research topic and which may be ignored. This conscious decision has always been necessary in a well structured QDA process. However we suggest that readability should not be a factor in this decision any more, due to the easily adaptable representation of text data and meta data in a CAQDAS environment.

The effort of incorporating a configurable notation system for emotions, pauses and emphasis is also significant but would only have to be implemented once in a CAQDAS environment to be used in multiple research projects. To our knowledge such tools are currently not present in any CAQDAS package.

For NLP processing it is also necessary for the transcripts to be in the same format. This concerns both the file format and the layout. The former is usually automatically satisfied due to the widespread use of CAQDAS packages, but the latter is not necessarily enforced through the use of CAQDAS packages and is of particular importance for structured interviews.

3.2 Natural Language Processing

One of the central challenges when performing qualitative data analysis is to comprehend the true meaning of words in their respective context. Maykut & Morehouse (1994) wrote about the relevance of language to qualitative research:

"We create our world with words. We explain ourselves with words. We defend and hide ourselves with words. The task of the qualitative researcher is to find patterns within those words (and actions) and to present those patterns for others to inspect while at the same time staying as close to the construction of the world as the participants originally experienced" Maykut & Morehouse (1994)

Natural language processing (NLP) is a subdomain of artificial intelligence and as an interdisciplinary field, it is combining linguistic knowledge with computer science.

Large parts of NLP are concerned with the understanding of natural language. To facilitate this understanding, algorithms can make use of grammatical rules, lexical databases and statistical linguistics. Such techniques aim to support understanding of written text and conversion into a more structured format.

Other NLP methods are more focused on the generation of such text from structured data as input. The majority of these techniques work both ways, but we are exclusively interested in parsing and understanding spoken (and transcribed) or written text for our autocoding algorithm.

In our scenario we can not only draw on the naturally spoken and transcribed text itself as well as common knowledge sources, but also include certain meta data embedded like the question-answer sequence and who is talking.

NLP is a supercategory for a variety of different methods and techniques. Here we will only present those relevant to our task of autocoding our interview data.

3.2.1 Employed NLP methods

In the following we will present all NLP technologies we have employed, or experimented with, for our autocoding algorithm. The categorizations in *Preprocessing*, *Conflation* and *Information Retrieval* are not intended as generalities, but rather reflect the order of their use in our processing pipeline.

In Figure 15 we present a graphical overview of the employed NLP methods and their position in our processing pipeline.

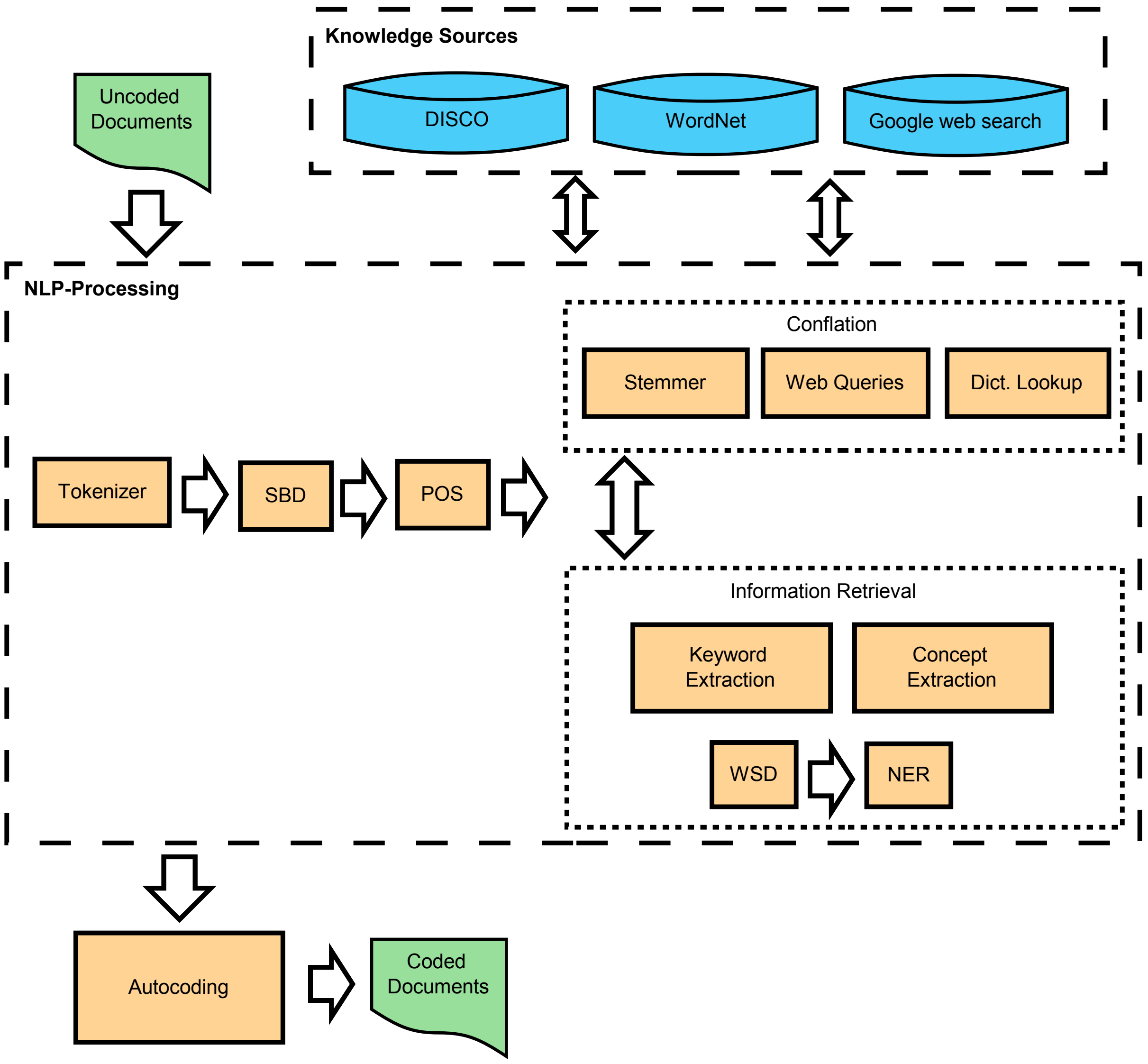

Figure 15: Employed NLP Methods

The exact integration of these techniques into our autocoding framework is described in chapter 2.4 Research Approach

3.2.1.1 Preprocessing

3.2.1.1.1 Tokenizing

Tokenizing is usually a necessary step early in the processing pipeline to enable further processing with more advanced analytical techniques. A tokenizer, sometimes referred to as lexical analyzer or as part of such, performs the process of splitting up a text into smaller atomic chunks (*tokens*). These tokens can be words, punctuation, formulae and other textual elements in the desired granularity.

3.2.1.1.2 Stoplist

Stoplists are lists of words, called stopwords, which are often removed in information retrieval systems. The length of these lists varies between less than ten to a few hundred words. Usually the stoplist has to be tailored for specific kinds of texts

Manning et al. (2008) described a trend in NLP that the use of stop lists degenerated from large stop lists containing several hundred words to small stoplists or complete abandonment of this practice.

General-purpose stoplists are publicly available and contain common words like conjunctions, articles, prepositions and may also contain qualifying words like *very* or *really.* These are usually not defining for a concept by themselves, but it remains important to realize that removal of such words can sometimes remove vital information from the text.

In our own experimentation we found general-purpose stop lists to be of little to no use, since our approach focuses mainly on already extracted keywords, and those seldomly contain any word that would usually be on such a stoplist. So at least for methods dealing with keywords instead of the original text the use of general-purpose stoplists had no impact on the results of our experiments.

Our second approach to utilize some form of stoplist was to populate it with frequent words relative to our test and training data, and with words which are uncharacteristic of a particular code or a smaller subset of codes in our coding-system. We have documented these results in chapter 2.6.3.1 Keyword Frequencies.

Our third approach to incorporate this practice is to populate the stoplist with keywords associated with a specific entity type (organization, location etc). This approach is is more context sensitive than a general-purpose stop list, so it has to be manually tuned to the specific context of each set of interviews. The results in our scenario however were discouraging. Our research results are documented in chapter 2.6.3.2 Named Entities .

3.2.1.1.3 Sentence Splitting

Sentence splitting, frequently also refereed to as *Sentence Boundary Disambiguation* (SBD), is performed as part of tokenization, and simply splits sequences of tokens at every sentence ending token (question mark, period etc.). In most languages the end of a sentence is marked with punctuation, however not every period mark in English texts marks the end of a sentence. Period marks, for example, are commonly used as a decimal point or as part of an abbreviation.

Palmer & Hearst (1997) tackle this problem using a machine learning algorithm called *Satz*. As training features they chose POS tags and capitalization. Mikheev (2000) measured the performance of Satz and several other state-of-the-art machine learning approaches on the Brown Corpus[7], and showed an error rate between 0.2% and 1.5%.

It is up to the tokenizer to decide if a character like a period belongs to another token or forms a separate token marking the end of a sentence.

Sentence splitting capabilities are integrated in the Annie pipeline, which we chose to use in our research.

3.2.1.1.4 Part-Of-Speech Tagging

Part-Of-Speech (POS) tagger exist in a variety of different forms, diverging radically in their approach to achieve the same goal. Some POS-taggers distinguish parts of speech by following handwritten rules. These may be rules specific for a particular kind of text, or general to the complete text universe of a specific language. Another popular approach are taggers based on hidden Markov models. Also automatically generated POS taggers based on the statistical analysis of manually tagged corpora and machine learning approaches have been explored.

POS tagging is, in principle, based on grammatical, not semantic, properties of the word (Schachter & Shopen, 1985). This distinction is vital to separate POS tagging from Word Sense Disambiguation. Although the grammatical properties may support a semantic disambiguation at a later stage in the processing pipeline.

According to Voutilainen (2003) the general architecture of many POS taggers contains the following three stages:

1. **Tokenization**
2. **Ambiguity look-up**

   Using a lexicon, the text will be annotated with all possible parts of speech for every word. Using state-of-the-art linguistic databases information on inflection and derivationally related forms will also be annotated.
3. **Ambiguity resolution**

   During this step one of the annotations of step 2 will be chosen as the most probable use of the word.

   The actual algorithm depends on the method applied to disambiguation. Information that is often used to this effect are probabilities derived from frequencies of how often each use of the word occurs in training corpora relative to each other. Also the sequence in which the word occurs may be used. For example a sentence with a sequence of subject, predicate, object is more likely to occur than a sentence

7 http://icame.uib.no/brown/bcm.html

without a subject. Syntactic relationships between a verb and an objects, a verb and a subject, or an adjective describing a noun can be identified and used in the disambiguation.

3.2.1.2 Conflation

Through conflation, sometimes also referred to as *token normalization*, we try to conflate certain terms that have a high probability of having an association with the same concept or code.

To achieve this, equivalence classes can be created and all occurrences of any word in the set may be replaced through a normalized denominator. We chose to preserve the original, not normalized, words and maintain the equivalence relationship separately. This allows us to draw on the original information if appropriate. Nonetheless, through the use of this method some more fine grain information within the text will inevitably be lost or disregarded, but through focusing on a more abstract level of conflated terms, new connections between the text to code and text of the training data may be revealed.

The effect of this technique on our metrics can, in most cases, be viewed as a tradeoff between *recall* and *precision,* since conflating terms before the training data is processed will increase the similarity of any two documents, thus the probability of a similar piece of text to a manually applied coding being identified will increase, which results in more automatically applied codes.

Some very common conflation criteria include:

- Capitalization
- Date and time format: 2014/01/02 is considered equal to 02.01.2014, and 2 pm should match occurrences of 14:00
- Different spelling, for example differences between British and American English
- Diacritics

The stage in the processing pipeline of any conflation method is especially important with regard to the use of a stoplist. If a word has many derivational forms, its frequency after stemming will be the sum of the frequencies of all derivational forms and may subsequently be added to the stoplist, while each individual derivation of the word would not have occurred frequently enough to be constitute a stropword. Since our approach to conflation is to maintain a separate list of equivalence classes instead of replacing the terms in the original text, the effect is the same as performing the stoplist removal before any conflation method, so only the exact derivational forms on the stoplist will be considered.

3.2.1.2.1 Stemming

Stemming describes the process of finding a base form (the *stem*) of every word in a text, and thus reduces the variety of different morphological variations in two documents. This stem form is usually described as the inflectional root of the word, but does not necessarily have to be the morphological root.

The terms (morphological) *root*, and *stem* are often used interchangeably, yet a subtle but clear distinction can be made. The root of a word is a morpheme that describes the core meaning of a word, while the stem includes the root and optionally derivational morphemes. The terms *reduce* and *deduce* for example are both stems containing the root 'duce' with a different derivational prefix (Payne, 2006). Roots and stems can be identical in many instances, but for the purpose of our research and most information retrieval tasks in general, abstracting any further than the stem removes too much vital information from the word. For example, we would like to stem the word *friendships* to *friendship*, not to *friend*.

The loss of critical information through stemming is described as overstemming.

The stemmer we chose to use in our experimentations is the snowball stemmer which is integrated into the GATE framework. Snowball stemmers are based on the Porter stemmer (Porter, 1980), with rules written in Snowball (Porter, 2001).

Since the Porter algorithm became the de-facto standard for stemming English texts many variations of the algorithm have been developed. A traditional Porter stemmer follows the following steps.

1. Recode plurals
2. Remove -ed or -ing
3. Recode y to i, if stem contains another vowel
4. Handle double suffix
5. Remove suffixes, if the removal rule is not violated for the remaining stem

The only rule to avoid overstemming in the original implementation is the requirement, that any stem must at least contain one consonant-vowel-consonant sequence. If this is not the case, no more suffixes may be removed and the current sequence will be considered the final stem.

The drawback of such a rudimentary implementation is that it disregards any context of semantics of the word to be stemmed, which may lead to mis-stemming. For example the words *university* and *universe* would both be stemmed to the stem *univers.*

Another problem of stemmers following these rules is that they are not able to handle irregular inflection through their suffix rules. Porter (2001), however, argues that at least for irregular inflectional suffixes, attempts to handle these through the use of rewriting-rules or exception lists is of little use because they occur either in very infrequently used words like *ox/oxen,* or in very frequently used word like *see/saw* or *man/men* which often are problematic due to disambiguation issues. The terms *saw* and *man* for example, can both be used as nouns or as a verbs, which can not be solved by an exception list.

The quality of a stemming algorithm can be measured by the percentage of derivations that were correctly mapped to the same stem. However, for us the far more interesting measure is how much the stemming improves the results for information retrieval. As Flores et al. (2010)

have demonstrated for Portuguese stemmers, the two metrics do not necessarily have to be correlated.

The impact of stemming on information retrieval is slightly controversial. While the majority of studies investigating the impact of stemmers for various languages on information retrieval concludes a significant benefit of using stemming compared to an IR system without stemming (Fuller, 1998; Braschler, 2004; Kraaij & Pohlmann, 1996; Carlberger, Dalianis, Hassel & Knutsson, 2001), there are other studies finding insignificant variations in the results (Harman, 1991), or even found the use of a stemmer to be detrimental to IR in their environment (Figuerola, 2001).

One reason this kind of conflation became popular is because the computing resources required to process large amounts of texts is very low. Lexical lookups for example require a large manually maintained database, and the statistical methods we employed requires large corpora data as well as significant processing time.

To counter the main drawback of stemmers, overstemming and mis-stemming, more complex conflation algorithms, that include the semantics of a word may be considered. Such a method is often described as *lemmatization*.

In contrast to a stemmer, lemmatization requires more context for each word. A possible approach to include the context of the word for conflation is to consider POS-tagging. Yatsko et al. (2009) suggested, that overstemming could be effectively reduced by preliminary POS-tagging. They claimed a significant improvement from 88.87% to 98.7% accuracy through the use of this technique in combination with a Paice/Husk stemmer.

3.2.1.2.2 Lexical Lookup

A dictionary lookup may also be used to produce the correct base form of the word as it is found in the dictionary. Unlike a stem, this base form is always a word and is called the lemma of the word to lemmatize.

We did not incorporate a complete lemmatization package into our processing pipeline. We did, however, use a simple lexical lookup method for conflation.

For our experimentation we decided to use WordNet to conflate the words in our text with semantically similar words.

WordNet was developed at Princeton University since 1985, and is being maintained and updated by the Cognitive Science Laboratory. It is freely available and the version used in this research is 2.1. WordNet encompasses 177'000 distinct synsets (sets of cognitive synonyms), which are structured using different relationships. One of the relations we are interested in is the hypernom-hyponym relation. This semantic relation is a is-a relation, where the hypernym is the more general tern. Through this relation we create a hierarchy of abstractions.

For example the term *trust* would be contained in several synsets, one for each meaning of the word (trust-fund, corporate trust, relationship etc.). If we select the meaning “a trustful relationship” we see the synonym confidence, and the hypernyms {friendship, friendly relationship}, which should be interpreted as *"a trustful relationship is a friendly relationship, while trustful being more specific than friendly"*.

Besides the hypernym-hyponym relation WordNet provides useful information to us by connecting *derivationally related forms* to each synset. For our example {*trust, confidence* } in the context of a trustful relationship these are *confidential* (adj), *confide* (v), *trusty* (adj), *trust* (v), while the occurrence of the term trust among the derivationally related forms links to a different, but related, synset of *trust*.

Another semantic relationship we used for conflation which is supported by WordNet is the meronym/holonym relationship. This is a part-of relationship, where the holonym contains the meronym.

With this knowledge a map of these terms, and their relationships can be created. Such a map is also frequently refereed to as a lexical chain. An example of a short lexical chain is shown in Figure 16.

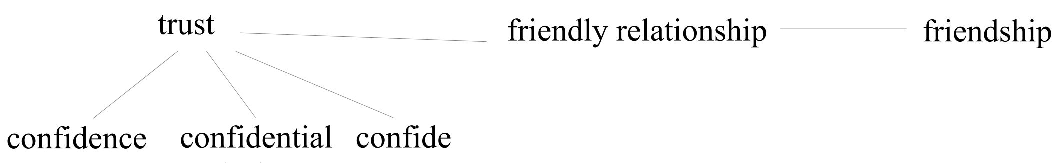

Figure 16: Lexical Chain

One possibility to create a useful lexical chain for a text is to exhaustively create every possible map of words, score each on its cohesion and select the map attributed with the highest score. Such a scoring algorithm could for example weigh the possible relationships in a chain, scoring synonyms higher than meronyms (Barzilay, 1997).

The creation of a lexical chain for an entire document using this method is associated with high computational costs, since due to ambiguities the number of possible lexical chains grows exponentially with the number of words in a text. Depending on the length of the document, the level of ambiguity and the used lexical database such a task might be unfeasible.

To reduce the computational costs the text may be segmented by a fixed number of words, sentences or if the text is structured by topics.

Silber & McCoy (2002) proposed a method which performs disambiguation of all words within the text, using the number of possible lexical relationships with other words in that text, and subsequently build the lexical chain, after this task has been completed.

Lexical chains have also been used by Hirst & St-Onge (1998) to detect and correct malapropisms and simply misspelled words by analyzing each word in its surrounding context in a lexical chain.

Typically a word map for a document contains not one single lexical chain, but a set of disjoint chains. The idea to capitalize on this for QDA, is to map a specific a subset to a code. The main challenge, besides generating a reliable disambiguated map is to identify a specific lexical chain as defining data for a significant semantic concept.

3.2.1.2.3 Statistical Similarities

Besides using morphological derivations and inflections or lexical superordinate relations between words, we also tried a statistical approach to inflate the set of keywords with statistically similar words, and thereby conflating their meaning.

We consider two words to be statistically similar if they have common co-locations. Statistical similarity can also be characterized by distributional similarity, but this property proved less effective in our scenario.

Through this we hoped to compensate for a relatively small set of training data, which usually impedes the performance of most machine learning algorithm, by enhancing the data we gathered with similar data from large corpora. We inflated the set of keywords for the documents to code, and the set of keywords for the training data. If applied to both, for some of our approaches the existing trends were simply amplified, and the end result stayed the same. We incorporated this techniques with mixed results. There are some instances, where our methods produced a correct replication of a manual coding, when this technique was applied, which were not replicated without it. Although such instances exist, they were mostly counterbalanced, if not outweighed, by codings that were no longer replicated when this technique was applied.

3.2.1.2.4 Web Queries

Another attempt to conflate terms that are associated with certain codes with terms that were previously unknown to the algorithm was to use web queries to generate keywords with a close semantic distance to the keywords directly extracted.

The use of web queries, is not commonly described in the literature on IR, however some related work to our approach was performed by Hovy, Kozareva & Riloff (2009). They used web queries in their algorithm to build a taxonomy for concept extraction as a counter proposal to using WordNet. They criticized some flaws of relying purely on WordNet as one of the most popular resources for building a taxonomy for concept extraction and classification. They identified several shortcomings of WordNet:

- All synsets for a term are treated as equally significant. Extracting the is-a relations for the term jaguar, for example would yield *animal* , *mammal* , *toy*, *sports-team*, *car-make* and *operating-system.* However they suggested, that the connection between *jaguar* and *animal* should be stronger, than between the other candidates.
- The is-a relationships in WordNet conflate conceptional (a physicist is a *human*) and instantiating (*Einstein* was a physicist) relationships.
- The taxonomy is incomplete, certain terms might be missed simply because they are not included in WordNet.

We also believe, that this approach might be particularly suited to be used for the less formal style of language in our interview transcripts, and is also capable to process compound terms more naturally.

Our first approach to using web queries for conflation was to use the keywords extracted in our training data, and finding news stories corresponding to them through Google News, for which an RSS feed is provided that can easily be automatically processed. The retrieved news articles were then again processed with AlchemyAPI to generate related keywords.

We experienced two main challenges when employing this technique:

The first challenge was to filter the search results for those news stories where the search term was actually central to the story. Such ranking is usually the task of the search engine, but if the queries were too general, the results were frequently only very loosely related. To increase the specificity of each query we used combinations of several keywords along with the name of the code.

The second challenge was to filter the list of keywords for those actually semantically related to the code. Here we were again in need for a measure of the semantic distance between words. Since we already employed DISCO for this purpose we also used it to filter the newly found keywords from our web query.

The results of these efforts were fairly disappointing. Although filtered, the extracted keywords were often either too general or within a different semantic domain.

We also experimented with the methods described by Hovy et al. (2009) in another attempt to integrate web queries, and the immense amount of data available through them, into our term-conflation process. To integrate these methods we made use of the *Doubly-Achored Pattern* (*DAP*) developed by Kozereva, Riloff & Hovy (2008) in our queries to increase specificity.

The DAP is a query using two concept-specific terms.

One is described as the root (`Seed Term 2`) and one is either a concept or an instance (`Seed Term 2`). The structure of the query is as follows:

```
DAP: [SeedTerm1] such as [SeedTerm2] and <X>
```

Figure 17: Doubly-Anchored Pattern

`Seed Term 1` is also called the *class name* and `Seed Term 2` is called the *class member*.

Since we want to apply this technique to our codes, and potentially our keywords, `Seed Term 2` is fixed to either the name of the code or a keyword associated with this code.

Depending on the choice of `Seed Term 1` the resulting terms can vary drastically. For our first attempt to use this technique `Seed Term 1` was set to *terms*, since we are interested in conflation of the term `Seed Term 2` with semantically similar terms.

We experimented with the code names as `Seed Term 2` as well as our keyword lists. When a new keyword was extracted using an existing keyword, we assigned its relevance value to the new keyword as well. If the keyword was extracted using a code title as `Seed Term 1`, we set this value to 1.1, scoring them higher than any keyword extracted from the manual codings. This is one factor why our Keyword Relevance method showed the most improvement through this conflation method.

Through a simple test with the query "*terms such as trust and* " we received similar results to the related terms extracted from WordNet. Within the first ten results, using the Google web search, the term *confidence* occurred twice, and the term *friendship* was extracted as well. These two already cover most of the semantic spectrum of the direct neighbors using WordNet. (Figure 16). In addition to this we also extracted the terms *responsibility* , *faith* , *fiduciary* , *settlor* , *trust fhnd* , *product.* These initial results demonstrate some strengths and weaknesses of using web queries to build a taxonomy.

- *Confidence, friendship, responsibility* and *faith* are intuitively very strongly related terms to *trust.* All of them may also be extracted through a lexical approach, but at least in WordNet they are associated to different synsets, and including all possible synsets, and all possible relationships of the first degree would yield a very large set of results. The retrieved weights for each relationship through their popularity rated by the Google web search improves on one of the identified shortcomings of a lexical approach.
- The term *trust fhnd* is clearly a spelling error for the term *trust fund* . These kind of errors usually do not occur in a carefully maintained lexical database. These may possibly be filtered through the number of search results for the misspelled term, however, for very frequent terms even misspellings will return a large number of matches on the indexed websites. Another possible approach would be to look the new found terms up in a dictionary, although this partly eliminates one of the advantages of this method, namely the possibility to extract terms that are not present in a lexical database such as WordNet.

  This method could, however, still uncover new relationships between terms that are not present, or at least of a higher degree, in the database. Except for the spelling error, all terms can be found in WordNet, although the semantic similarity is not defined. Also, for a simple lookup if a term is valid or just a spelling error, one could use multiple independent data sources, including such that do not offer any relationships between the terms.
- The terms *fiduciary & settlor* (and we may include *trust fund* as well) are terms specific to the financial domain, and were not extracted through WordNet, although a relationship exists within WordNet, but following the relationships beyond directly related terms expands the possible semantic distance between the two words significantly. We believe this is generally not desirable. However very specific terms like *fiduciary* and *settlor* would at least not introduce new ambiguities.

Through variation of the class name (`Seed Term 1`) a different set of class members can be extracted. For this purpose we also experimented with the class name *concept* , resulting in the search query "*concepts such as trust and* ".

Within the first 10 results using Google web search, we extracted the term *social capital* twice, as well as the terms *reciprocity* , *norms of reciprocity* , *distrust modeling* , *provenance* , *integration* , *trust network, dependability* and *risk.*

- Since the terms *reciprocity* and *norms of reciprocity* are closely related to *social capital*, four our of the 10 highest rated results were tied to the same very specific subdomain of trust. In this we see an example for another drawback of using web queries compared to a lexical approach.

  The universe of websites in of itself, as well as the use of one specific search engine introduces a new bias to the results, that is not or at least much less significantly, present in dictionaries that attempt to cover the whole spectrum of a language. In the sum of all websites indexed through Google however certain domains are over represented, and thus score higher in our very general search queries, while others are under represented.
- The semantic connection between trust and provenance is very weak. It occurs in the context of two examples of how concepts are represented on the internet. Such results might be filtered out by processing a larger set of results and reduce them to the most frequent occurrences.

Overall these results seem far less helpful for our coding task, than our previous results using the class name *"term"*. The level of abstraction was significantly lower, and unlike with our previous results there is also no overlap of these results with the related terms extracted through WordNet. A certain amount of overlap might be used as a gauge to determine the abstraction level of the extracted terms.

Hovy et al. (2009) also address the opposite challenge of avoiding to extract overly general terms through this technique. They propose the *Concept Positioning Test* (*CPT*).

```
(a) Concept such as Root Concept and <X>
(b) Root Concept such as Concept and <X>
```

Figure 18: Concept Positioning Test

The goal of the test is to determine if `Concept` is a subordinate to `Root Concept`. This is considered to be true, if query (b) yields more results in a web search than query (a). Otherwise the concept has failed the test.

The bootstrapping can be further extended beyond the original seed terms, by reversely applying the Doubly-Anchored Pattern, denoted *DAP* $^{-1}$. (Hovy et al., 2009)

```
DAP⁻¹: <X> such as [SeedTerm3] and [SeedTerm4]
```

Figure 19: Inverse Doubly-Anchored Pattern

To generate a new class name for specific terms selected in Figure 17, one can set `Seed Term 3` to *Seed Term 2* of the previous test , and `Seed Term 4` equal to <*X*> from Figure 17.

One possible query would be “such as trust and dependability”. This would yield new possible class names. Occurrences for <X> in this case would include *values*, *concepts* and *qualities*.

Although the examples presented in this chapter were manually retrieved from the Google web search, for our algorithm we used the Yahoo! web search.

3.2.1.3 Information Retrieval

The methods described in this subchapter are all coupled closely to the semantics of the words, rather than only relying on grammatical or syntactical features.

3.2.1.3.1 Keyword Extraction

The keyword extraction service we used for our experimentations was the keyword extraction service provided by AlchemyAPI. We would like to thank AlchemyAPI for their permission to use their service in our experimentations.

Interview elements were sent as chunks of text to AlchemyAPI via their REST interface, which returned a structured response, for which we chose XML. This response contained a list of all keywords, each attributed with a relevance score and a confidence score. Confidence measures the disambiguation, while relevance measures the semantic significance for the processed text.

Besides the use of AlchemyAPI for keyword extraction, we also experimented with several other keyword extraction services like OpenCalais and Zemanta. Through this experimentation we found that, at least in our scenario, keyword extraction should not focus too much on named entities. We chose AlchemyAPI for our final implementation due to its consistent performance, configurability, and the relevance score attributed to each keyword.

Although the technology we employ in our experiments is proprietary and from our perspective used as a black box service we would like to present some possible keyword extraction techniques that have been proposed in the literature.

Further, some of the approaches presented for keyword/keyphrase extraction showed promise during our experimentation to be transferable to concept extraction, or in our case autocoding, based on already extracted keywords.

Kupiec et al. (1995) identified several features of significance at the sentence level. Keyword extraction may be regarded as a more condensed form of text summarization as investigated by Kupiec et al. For this reason some of the features identified by them may also have value on the word level.

- The most frequent content words are considered to be of increased significance. Sentences would be scored on the number of occurrences of these words. The identification of “content words”, according to Paice (1990), is the task of keyword extraction, so this feature holds no value to us.

- Words contained in the title or section headers will also increase the score of a sentence. This is an important feature that can be capitalized on when the interviews are highly structured. In our second interview set, the subcodes of the code Interview Guideline Topics were frequently applied directly following a corresponding heading.
- Sentences in the beginning of a document are weighed to score higher than sentences at the end of a document. It is not clear if this is remains true for answers in interviews. There certainly is a threshold for short answers where this technique can not be applied. However for longer answers the interviewee might be inclined to describe an overview with the first few sentences of the answer and then go into more detail in the remainder of the answer.
- Certain pre-defined *indicator phrases* increase the probability of summarizing information to follow. Typical indicator phrases include single keywords like *results* or simple n-grams like "*This report...*" or *"In conclusion"*. This feature could be applied to interviews for QDA with a set of indicators specifically modified for interviews and spoken language. However a feature like this would fit better with a rule based approach instead of a machine learning approach, since the indicator lists would have to be tailored to the problem domain.
- A list of positive and negative *cue words* increases or lowers the score. Negative words are called *sigma* words, while positive words are called bonus words. A typical stigma words would be *hardly* or *impossible*, and a typical bonus word is *significant* and *greatest.*
- If a potential keyphrase occurs multiple times in a document, the first occurrence of the keyphrase is weighed as more important, than any subsequent occurrences of the term.

Similar word features have been used by Turney et al. (2000), who treated keyphrase extraction as a classification task, and applied a supervised machine learning algorithm to the problem. As such, a portion of the text has to be processed manually and is used as training data, attributing data about the used features to keywords in general. Then each word or n-gram in the remaining text is classified as either a keyphrase or not a keyphrase. Turney et al. used the general purpose classifier C4.5 which is based on decision trees, and a genetic algorithm specifically designed for keyphrase extraction. Frank et al. (1999) also investigated the same approach with an algorithm based on a naïve Bayes classifier.

Another statistical feature, that is often used for information retrieval is term frequency – inverse document frequency (*TF-IDF*). This measure uses the term frequency and the inverse term frequency to extract only such frequent words from a document, that are not frequent among all documents.

The basic formula to calculate the TF-IDF score is

$$tf(d,t) * \log \frac{N}{|d \in D : t \in d|}$$

Where *tf(d, t)* is the frequency of term *t* in document *d,* $|d \in D : t \in d|$ is the number of documents containing term *t,* and N is the total number of documents.

All of the features used by Kupiec et al. (1995), Turney (2000) and Frank et al. (1999) features as well as the TF-IDF exploit statistical and syntactical aspects of the text. In other words, they do not make use of any semantic properties of the words or even grammatical features of the sentences.

More recent approaches of information retrieval systems do incorporate such information for concept and keyword extraction.

Hulth (2003) for example used some of the statistical features, such as frequency, first occurrence etc. as a starting point and added information generated by POS-taggers, thus including grammatical features of the text. She concluded, that the five most common POS-sequences for keywords are:

1. Adjective Noun (singular or mass)
2. Noun-Noun (both singular or mass)
3. Adjective-Noun (plural)
4. Noun(singular or mass)-Noun (plural)
5. Noun (singular or mass)

In an attempt to also include semantic features of the words for keyword extraction Ercan & Cicekli (2007) presented a method that relied on lexical chains for keyword extraction. They compared an approach that relies on statistical and syntactical features as well as lexical chains with a baseline that did not include the lexical chains. In none of their measurements, the the use of lexical chains proved detrimental to the results, and in almost all they showed a significant improvement in precision of the keyword extraction algorithm.

Since we are working with interview transcripts we are also interested in the question of whether the characteristics of transcripts, for example the language that is used or perhaps more spontaneous changes of the topics have any effect on information retrieval. Liu, Pennel, Liu and Liu (2009) did some research on a related topic. They investigated the application of different keyword extraction methods specifically for meeting transcripts. They discovered some significant differences to the application of keyword extraction to written speech due to the characteristics of the transcripts.

They found, that an TFIDF approach outperforms a graph based approach which they attribute to the unstructured form of the data.

They also confirmed, that POS-tagging as well as the use of *sentence salience scores* improves the performance of both approaches for their meeting transcripts. A sentence salience score scores each sentence's similarity to the document as a whole. Such a score was also investigated for keyphrase extraction by Radev, Goldensohn & Zhang (2001), who used a linear combination of TFIDF values measuring the centrality of a term, a distance factor which is inversely correlated to the distance of a sentence to the beginning of the document and the overlap with the beginning sentence of a document.

Since unstructured interviews usually resemble an open discussion in which the used language is similar to a meeting, these findings may be transferable to interview transcripts.

3.2.1.3.2 Concept Extraction

After our initial success in the application of the keyword extraction service we tried to increase the abstraction level, and examined if the *concept extraction*, sometimes also called *concept mining*, service of the same service provider would yield information that could possibly be very valuable for an effort to create a new, or modify an existing, coding system based purely on evidence within the test data, and without any preconceptions from the data already gathered and coded.

We discovered that, while some abstractions were impressively accurate, others were very far from the actual context of the text. Some parts of the interviews were for example tagged with recording artists and generally music related concepts. We gathered, that the service we used is possibly biased towards a more widespread use of this technology in analyzing social media for market research or enriching web applications with additional content.

The revelation, that simply using a general-purpose tool, or at least one that was not mainly designed for formal qualitative research, nor tuned to our specific domain knowledge, would not yield consistently usable results was not unexpected. Still we believe that the intention of concept extraction and the idea of coding are congruent even though the background in their development is drastically different.

For this reason we incorporated several ideas of concept extraction to our autocoding algorithm, although we see true concept extraction primarily as a means of extracting evidence for codes which are not already present in the code system.

Our approach of using hypernyms and hyponyms to generate hierarchies of word maps, with the goal of semantic conflation of terms, is very similar to the concept extraction method described by Gelfand, Wulfekuler and Punch (1998).

To apply this technique to our autocoding algorithm we build on the keywords extracted by AlchemyAPI. These sets of keywords are processed analogous to the bags of words used by Gelfand et al., without consideration of word order. However, while every word in a paragraph was considered for his word maps, our keywords are the result of extensive preliminary analysis of the text.

Although our keywords do have a relevance score and a confidence level, which may have been influenced by word order, part of speech etc., and which we ultimately considered in our coding algorithm, for building the word map each word is equally considered.

Like Gelfand et al. we also chose WordNet as our lexical database to build the word-map, as described in section 3.2.1.2 Conflation.

The hypernym-hyponym relation, the holonym-meronym relation, as well as the derivationally different forms expand the vocabulary specific to the semantic theme of the text we wish to code as well as the training data. This potentially increases the evidence for our autocoding choice where we might have missed an obvious thematic connection simply due to derivations of the same word.

To perform autocoding for a fixed coding-system one could use very simple methods like counting the matching keywords with the expanded vocabulary, but it may also be applied to cluster the expanded bag of words by their relations to identify the strongest themes and eliminating matches between text and code that are mostly coincidental because they did not belong to a significant cluster. This way this technique could also be used to contribute positively to the precision metric.

Although we did not fully apply Gelfand's concept extraction approach we consider this to be a promising approach to generate new theories from interviews.

3.2.1.3.3 Named Entity Recognition

Named Entity Recognition is concerned with extracting words identifying an entity belonging to a specific entity category. Such categories include names, locations, organizations, time, money-values etc.

State-of-the-art NER systems are capable of performing their task with an F-measure of over 90%, the F measure being the harmonic mean of recall and precision (Lin & Wu, 2009).

We chose to perform NER with the Annie plugin, since it is one of the core plugin of the GATE framework, which we used in our implementation.

The GATE NER processing resource relies on a gazetteer. A gazetteer contains lists of entities as plain text files. One such file would, for example, contain European currency units. These files are then grouped by the entity category they belong to, and listed in an index file. The files can be compiled into finite state machines, which will annotate the text according to the features specified in the index file (Cunningham et al., 2009).

However, the GATE NER resource does not only perform a gazetteer lookup. Grammar rules are defined in JAPE format, which are prioritized and processed in a specific order.

Besides using the gazetteer list the rule for organizations will for example also consider certain keywords and other contextual features. The following are a few examples of how these features are used.

- Organizations can be identified by certain keywords like *Ltd.* for corporations or *St.* for churches
- Identification of many entities can make use of capitalization and POS tagging.
- Names can be identified by titles like *Dr.* or *Prof* or trigger words like *Mr.* or *Mrs.*
- In phrases like 'X joined Y' X is likely to be a Person and Y is likely to be an organization or another person.

- Names can be recognized in the context of other entities. <NAME>Anne</Name> **and** <UNKNOWN>Kenton</UNKNOWN> can be resolved to <NAME>Anne</Name> **and** <NAME>Kenton</NAME> due to the context of another NAME entity and the conjunction 'and'.

Ambiguities are resolved at several stages in the pipeline based on statistics. For example will a christian name followed by a location, like 'Ken London' be annotated as a person rather than a location (Cunningham et al., 2009).

In recent development in NER there is a trend towards machine learning algorithms instead of rule based algorithms, although the availability of training data for certain entity types is still limited. In such cases semi-supervised machine learning techniques have been proposed (Nadeau & Sekine, 2007). These techniques use a bootstrapping stage to mitigate the lack of large training corpora. During the bootstrapping a small set of *seed rules* is expanded to a larger set of training data (Collins & Singer, 1999).

Although the general-purpose processing resource we employed for NER tagging performed well for our test data we are interested in the question whether the type of data we use, the hand-transcribed interviews, has any influence on NER. Poibeau & Kossheim (2001) studied the impact of the type of corpus being processed for NER. In their research they used emails and hand-transcribed telephone conversations, and especially the latter share many characteristics with interview transcripts. They found, that the lack of strict writing constraints leads to a significant drop of 20%-40% performance when rules written for journalistic texts are applied to the new type of data. They concluded, that a significant portion of the negative effect induced by the informal corpora data can be mitigated by in increase of contextual rules and they further suggest, that using a machine learning approach would mitigate the impact of different kinds of corpora, since the same kind of data is used in the training data as well as the test data.

We have used NER for considering or disregarding certain keywords in our autocoding algorithm, which proved not very useful. It may hold more value by identifying characteristic keyword categories to a code.

Codes about *Relationships*, *Partners*, *Family* and *Parents,* like they were present in our second interview set about life satisfaction, will contain *Person* entities with near certainty. Codes like *Turnover,* on the other hand, are more likely to contain an organization entity.

3.2.1.3.4 Word Sense Disambiguation

Word Sense Disambiguation is the task of distinguishing words with the exact same spelling but a different semantic. One disambiguation task we had to deal with in our research is choosing a synset for a word if multiple synsets are available within the WordNet database.

An important variable relating to Word Sense Disambiguation in our research is the *confidence* score AlchemyAPI returns for each keyword in its keyword extraction service along with the *relevance* score.

While the *relevance* metrics attempts to quantify the semantic significance of the word within its context, the *confidence* metric is an attempt to quantify how likely the word has been correctly disambiguated. In other words the confidence can be seen as a precondition to the relevance A high relevance score means the word is significant to the meaning of the processed text under the assumption, that the word was correctly categorized (disambiguated).

Disambiguation can make use of other NLP techniques, such as Part-Of-Speech tagging, to distinguish between verbs, nouns etc. On different abstraction levels disambiguation is both a part of as well as a more general form of Part-Of-Speech tagging.

Often the use of lexical databases, such as WordNet, is also helpful to search for semantically similar words that may occur in the context of the word to disambiguate.

Other statistical information that may be used for word sense disambiguation includes a collocation matrix for large corpora. Yarowsky (1995) studied how often a word was used in the same context of the same collocations but with a different sense of the word. He hypothesized that for each collocation of a word there is usually exactly one sense. Through utilization of this hypothesis the disambiguation algorithm he tested performed significantly better exceeding 96% accuracy.

These findings also encouraged us to experiment with statistical analysis tools like DISCO, as described in chapter 3.2.1.2 Conflation, to extend our autocoding algorithm and finding evidence for a concept by searching for frequent collocations of significant keywords.

3.3 Further Results

3.3.1 Structured vs. Unstructured

Both of our test sets were inherently different in semantic content and style, besides the fact that one was structured and one unstructured. The length of the interviews was a significant difference between the two sets of interviews, the structured one being significantly shorter.

The variety of differences results in a large set of confounding factors. This makes it difficult to isolate a specific feature that might be considered responsible for different behaviour of our autocoding algorithm.

From our experience a structured interview offers more possibilities for rule-based coding. But also our ML approach benefits from the fact that all interview are more similar to each other. In the unstructured set we also had two interviews that were more similar to each other than to the others, and experienced a significant drop in performance if in the autocoding of one of them, the other was not in the training data. The unstructured interviews however were longer, meaning more text for our training data. Although we spent significantly more time experimenting with the unstructured set the coding agreement for the structured set was still higher, which conforms to our initial subjective expectation.

3.3.2 Detailed Data on Conflation

The diagrams showing the impact of the conflation methods stemming and web queries we presented in section 2.6.3 of our research chapter, show the aggregated results for each method, disregarding the differences in each specific interview. In the following subchapters we present more detailed results on our experimentation with conflation methods.

3.3.2.1 Stemming

Figure 20: Stemming (Detailed) shows the results of our autocoding algorithm with and without stemming for each of our coding methods

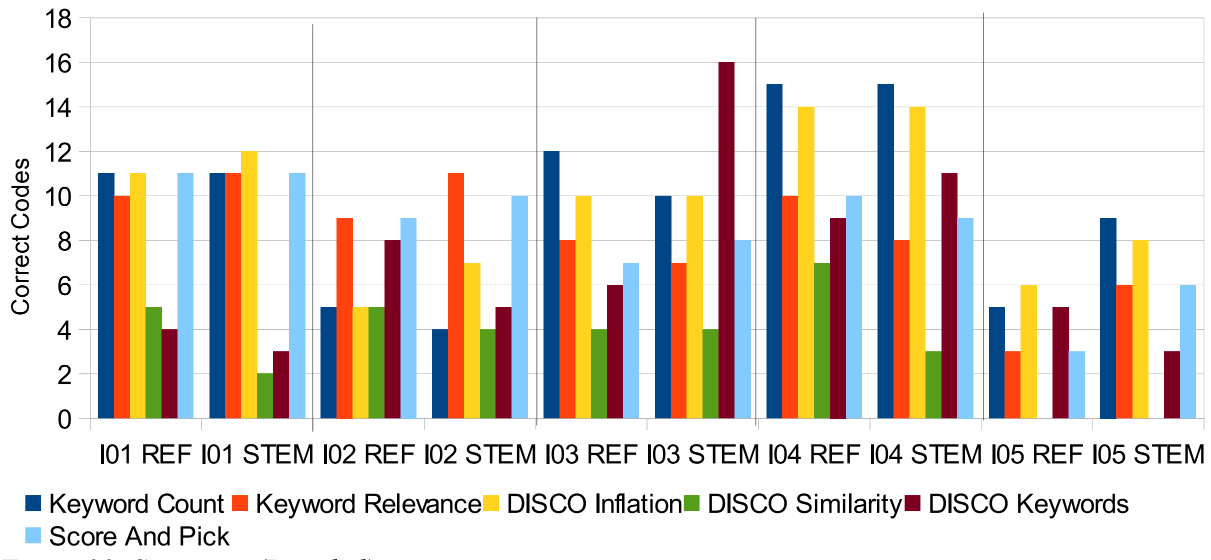

Figure 20: Stemming (Detailed)

3.3.2.2 Web Queries

Figure 21 shows the results of adding keywords to each code based on web queries using the DAP method described in 3.2.1.2.4 Web Queries. We used the word “Terms” as Seed Term 1 and the title of the code as Seed Term 2.

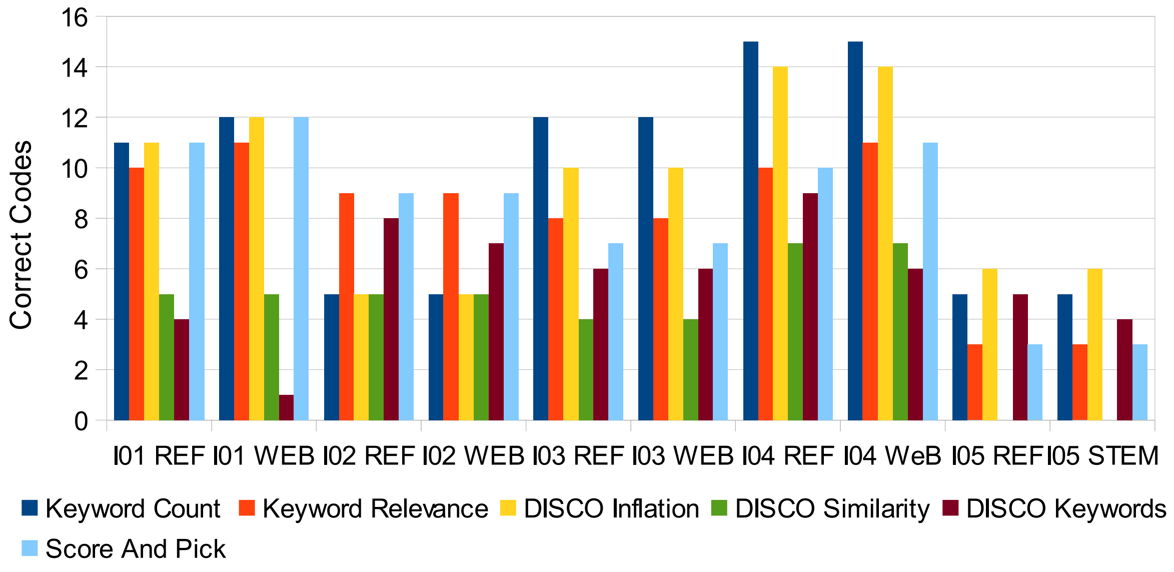

Figure 21: Web Queries - Title (Detailed)

To extend the impact of this method we repeated the experiment while using each of the keywords associated with a specific code as Seed Term 2 to gather more keywords. The results show a higher variance compared to using only the code title as Seed Term 2. The detailed results by coding method and interview is presented in Figure 22.

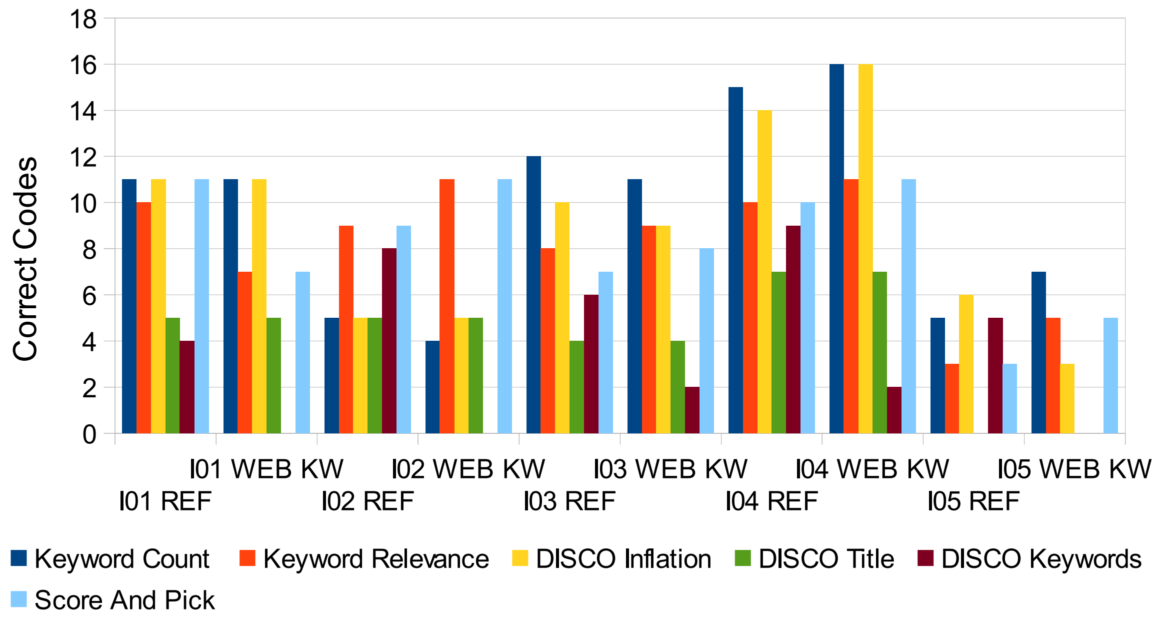

Figure 22: Web Queries - Keywords (Detailed)

3.3.2.3 WordNet

The following results show the results for word conflation using WordNet separately for hypernyms, hyponyms, holonyms and meronyms. The left figure will always show the results without conflation through stemming and web queries, and the right figure shows combining all three methods compared to only stemming and web queries as reference.

Hyponyms

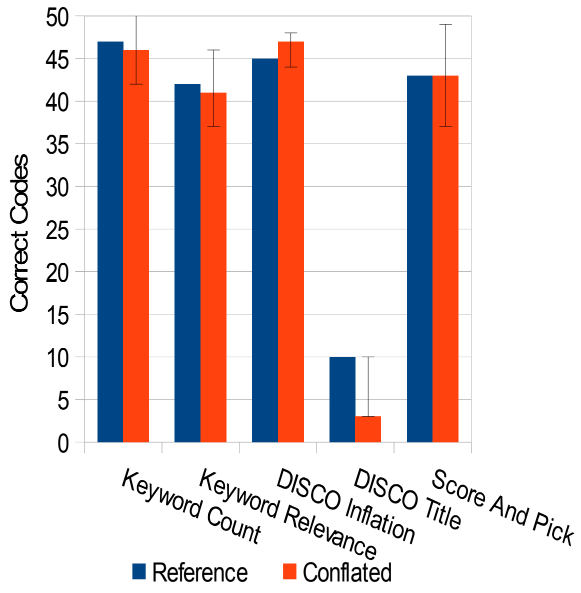

Figure 23: Hyponyms

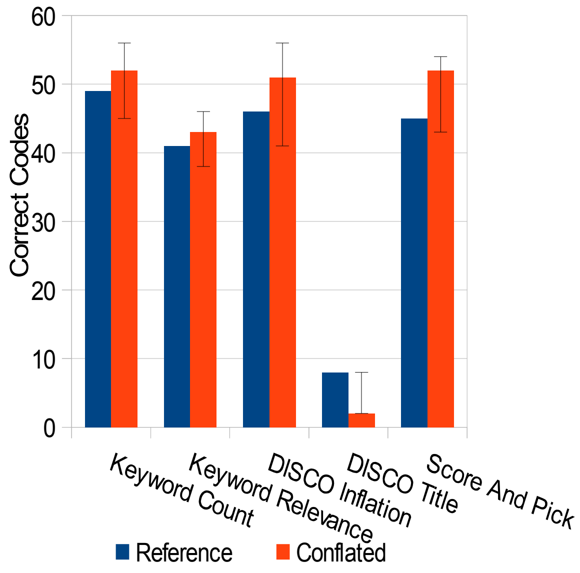

Figure 24: Hyponyms with Stemming & DAP

Hypernyms

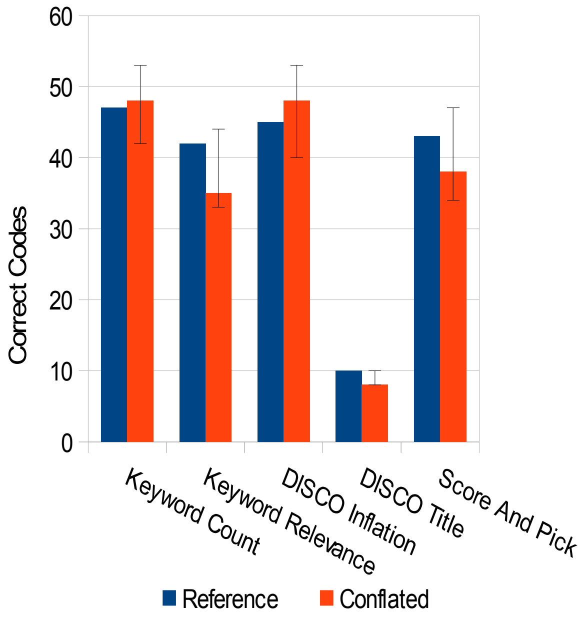

Figure 25: Hypernyms

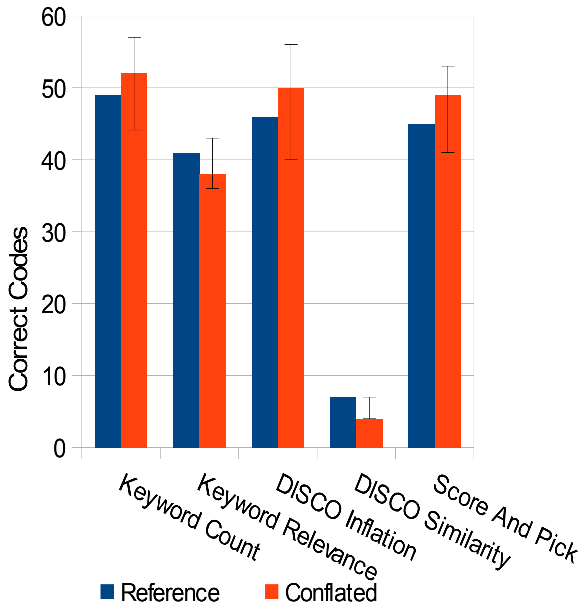

Figure 26: Hypernyms with Stemming & DAP

Meronyms

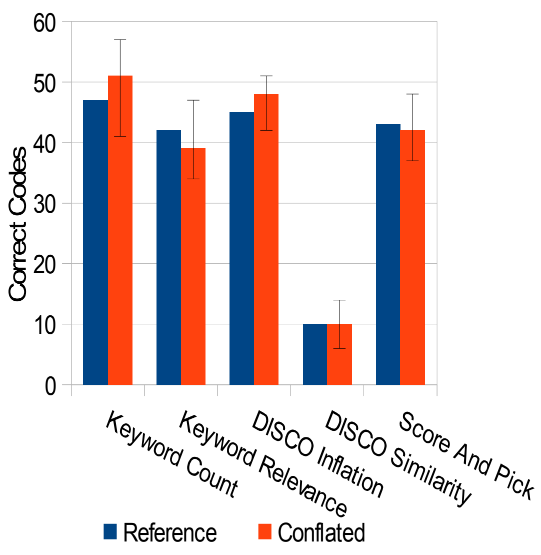

Figure 27: Meronyms

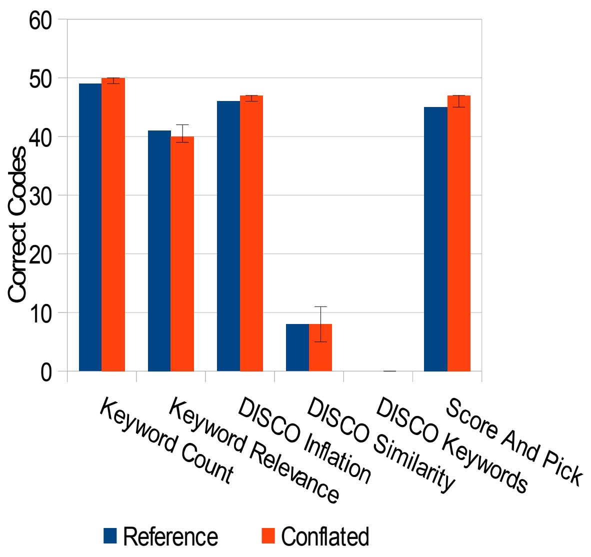

Figure 28: Meronyms with Stemming & DAP

Holonyms

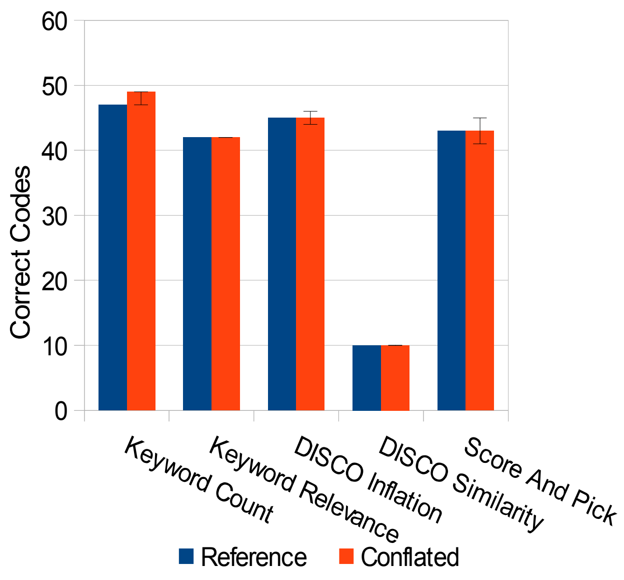

Figure 29: Holonyms

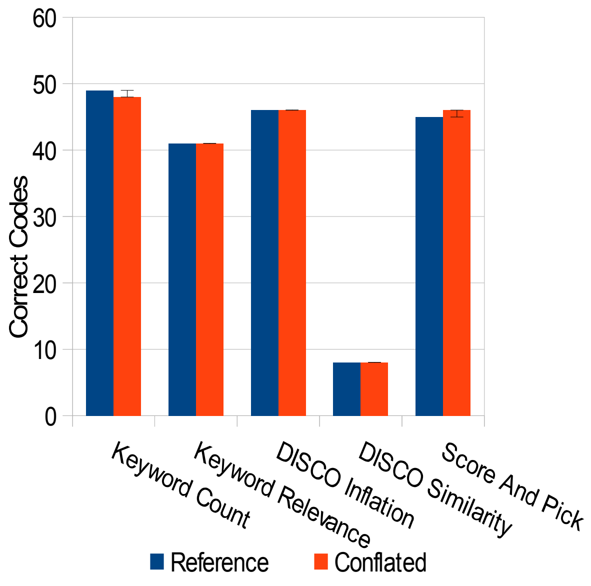

Figure 30: Holonyms with Stemming & DAP

3.3.2.3.1 Interpretation

Although the data shows moderate variance two trends emerged from our experiments. Overall we observed, that adding more specific keywords, hyponyms and meronyms, showed better performance than adding more general terms with hypernyms and holonyms. Also at least for our test data, and especially true for hypernyms and hyponyms, this method of conflation seems to work better in conjunction with other conflation methods. Our Score And Pick method performed better in every instance with stemming, DAP and WordNet compared to only DAP and stemming. Interestingly we see the variance for conflation using meronyms and holonyms drop significantly when used in conjunction with stemming and the DAP.

3.3.3 Statistical Machine Learning For Autocoding

Besides using machine learning for an algorithm, that trains on semantic features of the training data and applying this knowledge directly to the test data, we also experimented with a machine learning approach that builds on the semantic data gathered as described in 2.4 Research Approach and uses statistical analysis of this data for a classification task, and either apply one of the codes generated by one of our methods or discard them. This is similar to our method Score And Pick, but without manual configuration.

To accomplish this we calculated statistics on the score of each individual matching method (Keyword Count, Keyword Relevance, DISCO Title etc.), and whether the score correlated with a match to the manually performed coding or not. We stored this information in an .arff file to be processed by Weka. Weka is a machine learning software developed and maintained by the University of Waikato. It provides a powerful toolbox, that supports a wide variety of learning schemes including naïve Bayes, several decision tree algorithms, the Gaussian process and others. More detailed information on Weka was published by Hall et al. (2009)

It became obvious that codings based on a below average relevance score can be discarded with a high probability. However, it was still the case, that the instances with higher relevance scores still contain a large amount of false codings.

Our goal was to derive a weighing algorithm that decides, which of the possible coding results, produced by each coding technique, should be used as the final coding. The weights for each technique should be learned through statistical evaluation of the training data.

However, the results of this experimentation were disappointing. Both recall and precision dropped significantly if the weighing scheme was constructed in this way, which lead us to discard this approach for the final implementation and stick with a rudimentary, but hand crafted, weighing scheme. From this we do not conclude, that this methodology may not potentially be useful, but rather that more extensive study of this approach is necessary to determine its value for autocoding.

3.4 Beyond Autocoding

While our experimentation mainly focused on the aspect of autocoding, our initial research on possible applications of NLP to QDA produced some other applications that we did not have the resources to extensively test in our software implementation. These may be subject to further research.

3.4.1.1 Sentiment Analysis

Sentiment analysis is a powerful tool for market researchers to gauge the public opinion towards a product, a company, or a brand by monitoring and automatically processing large amounts of news reports and social media activity.

Another area of application for sentiment analysis has been developed in finances. The application of sentiment analysis to process finance related news stories has been discussed in multiple publications in recent years (Larkin & Ryan, 2008; Bollen, Mao & Zeng, 2011; Junqué de Fortuny, De Smedt, Martens & Daelemans, 2014). By 2010 about 35% of all investment firms on Wall Street were exploring automatic processing of news stories for indications for new trends, compared to only 2% in 2008 (Bowley, 2010).

Early approaches to sentiment analysis simply calculated the polarity of a sentence, by using dictionaries which assigned words either a negative or positive sentiment value. For example the words *bad*, *deterioration* & *poor* would impact a sentence sentiment negatively while *good*, *excellent* & *superb* would contribute a positive sentiment value. Slightly more sophisticated algorithms would also consider modal words such as *always*, *must*, *might* & *possibly.*

Recently Socher et al. (2013) presented an algorithm for sentiment that is based on a treebank of training data, that assigns all parts of a sentence an individual score, and through this is able to recognize patterns of negation, restriction, generalization etc. The algorithm suggested along with the use of a treebank utilizes Recursive Neural Tensor Networks to analyze this data. They suggested, that this combination yields a 5.4% increase of polarity based sentiment classification compared to other state-of-the-art algorithms.

The algorithm supports 5 sentiment classes (- -, - , 0, +, + +) which were condensed from a more fine grain 25 different possible values which had to be manually assigned in the training data on word level as well as on parts of the sentence. Each sentence can be judged by multiple persons. Originally each sentence was categorized by 3 judges. When parts of the tree are categorized by many participants, one may consider the statistical distribution on the sentiment classes. For example in the live demo of the algorithm, the word cleverness was categorized by 7% of participants as "+ +", by 81% as "+", by 4% as "0", by 6% as "-", and by 2% as "- -".

To test the integration into a QDA environment, we integrated the algorithm presented by Socher et al. (2013) into our test framework.

The integration proved straight forward, using the original training data used by Socher et al., the *Stanford Sentiment Treebank*. This treebank consists of 215,154 manually sentiment labeled phrases in 11,855 sentences.

The domain of the treebank used in the original publication was movie reviews, however in our initial experimentation the training domain of the treebank did not impair the performance severely. Some of the characteristics of movie reviews may be shared by interview data. Mainly that its vocabulary is less formal.

The results the algorithm produced in our environment seemed promising, although we did not measure accuracy by any means, since our focus was on autocoding.

3.4.1.2 Voice Recognition

NLP is not limited to understanding written texts, but many of the methods can also be applied to creating written text. The field of speech recognition (SR) is concerned with the translation of spoken words to writing. Texts created in this way are also frequently called speech-to-text (STT) transcriptions.

Concerning the application of such methods to the transcription process for qualitative research Franzosi (2010) wrote in a hopeful way “... by the time you read this book, new technological solutions to the problem of digitizing text may have become available (e.g. voice recognition software with 100% reliability)”

Such hopes have been around for many years. Brown (2002) made an even significantly earlier claim, that “voice recognition software is now sufficiently well developed to allow researchers to "talk" in data to a text or audio file”. But, it is our opinion, that reality never caught up with the expectations. In reliability engineering as a discipline of software engineering, we have the understanding, that 100% reliability, in terms of correct behaviour, is a praiseworthy goal, but can usually not be guaranteed. Therefore reliability is usually measured with probabilistic values (O'Connor & Kleyner, 2011). In these terms the hope formulated by Franzosi (2010) is unrealistic.

Still, in recent years, the possibility of using *voice recognition software* (*VRS*) to transcribe audio data gathered in qualitative research has been raised in a few publications.

There are two distinct ways of applying VRS to interview transcriptions. One possibility is training software with the researchers voice and dictate while listening to the recording, and one is to use the audio recording of the interview itself as input to create a STT transcript.

Matheson (2007) investigated both methods of using VRS to transcribe digital interview data in qualitative research. He found, that using the original audio recording as input produced considerably inaccurate transcriptions. He claimed, that the transcription using voice recognition as a way of dictating the text while listening was still faster, as well as mentally and physically less tedious. Due to the latter property the use of this technique may be recommendable for people with disabilities, or simply less typing experience, who may consider the transcription process as a burden (Matheson, 2007).

Earlier research by Park and Zeanah (2005), also on interview data, support this notion that transcription using VRS is not necessarily faster, but showed significant advantages for researchers with disabilities and slow typists. The cost savings, compared to hiring a secretary are also mentioned as advantage, however as we have elaborated in chapter 3.1.4.2 Transcription such a delegation of the transcription process has implications far beyond the monetary issue. Further, it has been noted, that not having to focus on spelling while typing improved the awareness for detail in the interview. This claim, however, is very subjective and no measures for verification are provided by Park and Zeanah (2005).

Johnson (2011) performed a comparative study challenging the notion of improved speed of the transcription process when using voice recognition versus a transcription performed by a professional typist. He concluded that the listen-and-type approach took 14.2% less time than a computer assisted approach using voice recognition. While the manually transcribed document had an accuracy of 98% in its unedited version, the unedited version of the auto transcribed document had an accuracy of 96.4%. Both required proof reading and editing, but the proof reading of the auto transcribed document took significantly longer mitigating any time advantages during the initial transcription. Further Johnson found that the errors made by the voice recognition software made certain passages difficult to comprehend because the erroneous word was only phonetically similar. Typing errors however were orthographically similar, which allowed readers to easily identify and correct the error.

Frequent issues with STT transcripts, especially when used directly with an audio recording, also include the lack of, or incorrect, punctuation and capitalization, which negatively impacts the readability. Gibson et al. (2004) researched such factors on readability besides accuracy of correctly spelled words. They concluded, that transcript enhancements such as adding punctuation and capitalization increased the readability by 5% - 10%.

3.4.1.3 Integration of NLP for Querying

As discussed in 2.1.4.1 Criticism of Automation in QDA, the automation of interpretative processes is controversial in qualitative research. Although autocoding may play an increasingly significant role, we see other applications of NLP, that may be described more as enabling, than as automatizing.

Analysis such as NER and sentiment analysis may be performed, and stored as meta data separate from the coding system. The researcher may then use this data in new querying applications. Correlations of sentiments with different codes could be considered, or investigating which persons frequently occur in the context of a code.

3.5 Implementation Details

The implementation of our prototypical autocoding framework consists of about 6000 LoC, written in Java, spread over 41 classes. The size of the project is therefore easily manageable for a single software developer. We will limit our elaboration on key design choices, that proved to be reliable solutions for our task, and which we would reuse in a similar application.

3.5.1 User Settings and Parameters

We feature five kinds of user specific settings, which can be configured through a .config file:

- Identification of interviewer and interviewee
- Selection of the keyword extraction method to be used
- Selection of the autocoding methods used
- Filtering: It may be defined, at which frequency a keyword should be dismissed.
- Conflation: It may also be configured which conflation methods should be used (stemming, lexical lookup or web queries)

3.5.2 Structure

Part of our core data structure shown in Figure 31: Implementation: Core Data Structure.

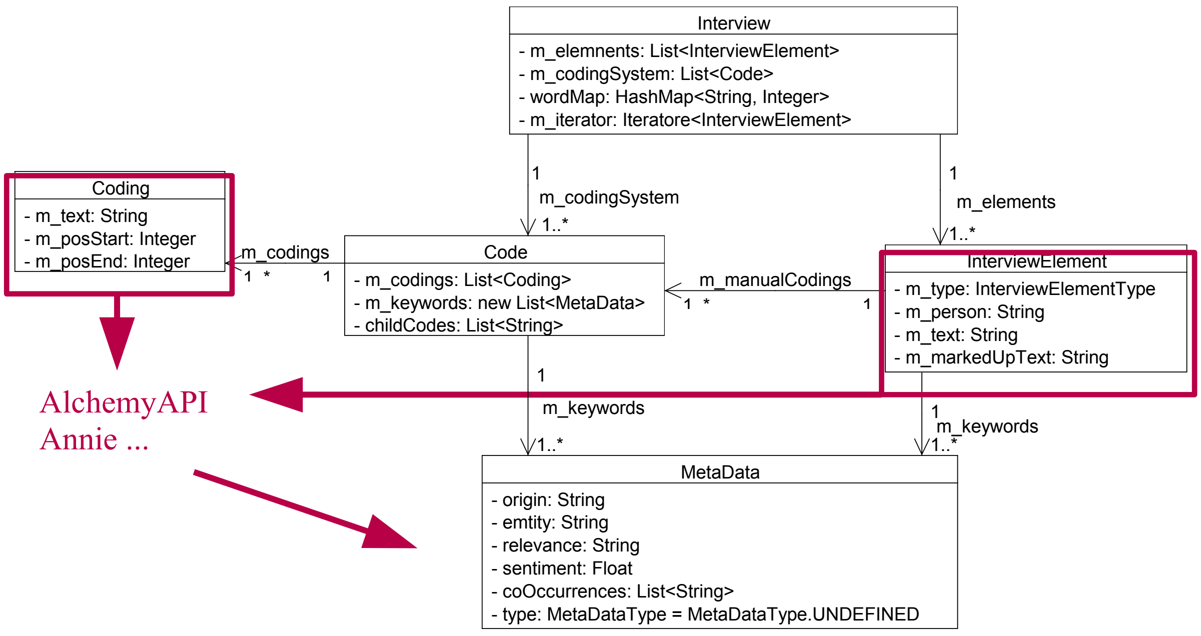

Figure 31: Implementation: Core Data Structure

Each `Interview` has set of `InterviewElements`, and set of `Codes`. Each `Code` may have multiple `Codings` associated with it. The association *`m_manualCodings`* from `InterviewElement` to `Code` is used for validation.

The text of each `Coding` and the text of each `InterviewElement` will be processed by IR methods generating `MetaData` instances. Depending on the IR method, new `MetaData` instances are parametrized instanced as keywords, entities etc.

3.5.3 GATE

The use of the GATE framework proved to be a reliable foundation for NLP assisted interview analysis. GATE is open source software, and built for flexibility. The Annie plugin was successfully used for its capabilities of fine grain NER. Although different available plugins can be added to the pipeline, none of them are capable of performing IR at the semantic abstraction level we needed for our keyword extraction.

For each additional processing resource, like AlchemyAPI, Zemanta, OpenCalais etc. we built a class capable of

1. reading data from the GATE document, which is passed as a parameter.
2. processing the text, for example by querying the REST interface of AlchemyAPI.
3. integrating the result into the GATE document.

A simplified representation of this integration of GATE is presented in Figure 32. GATE classes are highlighted in gray.

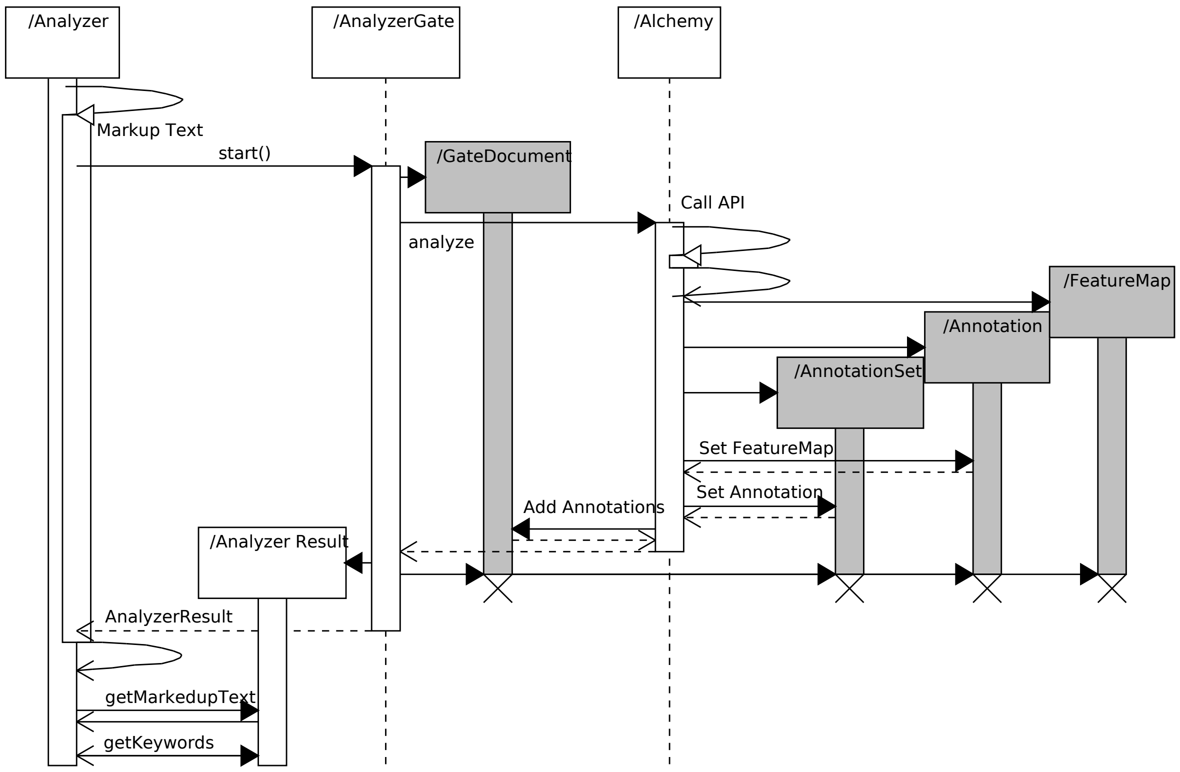

Figure 32: GATE Integration Sequence

Before the GATE document would be destroyed, and our own `AnalyzerResult` instance created, further processing such as NER would be performed on the same *GateDocument*.

Also, one concrete *AnnotationSet* would usually contain many `Annotation` instances with the same type of features in their specific *FeatureMap*.

Each GATE document can be assigned multiple named *AnnotationSets*. Each annotation set contains a *FeatureMap*, and an offset to identify the part of the text which shall be annotated with the feature. Our adapter to the AlchemyAPI interface would for example extract the *AnnotationSet* "Sentence" to extract all sentences from the document, then request a keyword extraction on groups of sentences, and create an *AnnotationSet* called "Keywords_Alchemy" with a *FeatureMap* containing the type, relevance and the extracted data.

3.5.4 Caching

The time needed to compute the statistical similarities between all possible combinations of keywords increases exponentially with the length of the training and test data. To make large amounts of test runs possible we had to cache similarities between all keywords occurring in our texts in a large XML file, which we parse to a hashmap on startup.

Caching was also performed for the keyword extraction service as well as for the results of our web queries for term conflation.

3.6 Further Research

At the time of this work, there was very little research on the application of machine learning using NLP for autocoding available. Therefore we devoted our limited time and resources to constructing an experimental framework to test a multitude of techniques and services, both successful and unsuccessful. The end result in terms of coding agreement, however, remained underwhelming. We firmly believe that building on this research a focused approach with more manpower would yield a better performing autocoding algorithm measured by recall and precision. An assessment based on different intercoder agreement measures may also be performed.

We focused our autocoding approach by design purely on machine learning and statistical methodologies. At least one successful, academic, application of rule-based autocoding is known to us (Crowston, 2010). We suspect that there is significant potential to combine both approaches.

Further investigation of a bootstrapping mechanism to tune the variables of the machine learning algorithm to a new interview project may succeed, where our experiments using Weka did not. This would further lessen the burden of configuring the algorithm to specific projects, and thus extend on one of the strengths of ML algorithms. If the amount of training data is large enough, a genetic algorithm for this purpose may even be considered.

This research is academic and retrospective in nature, and autocoding was not considered in the original qualitative research process generating the data we experimented with. We suggest, that a prospective study that includes NLP techniques at an earlier stage, would give new insights into its application. The scientific documentation of a hybrid approach combining technologies and manual expertise would also be interesting.

We also suggest the a comparative investigation of the impact of the size of the data being processed. Typically machine learning performs better, if the amount of available training resources is high. Although the interviews available to us were quite lengthy, the set contained only 6 interviews. This is not uncommon for QDA, however there are instances of larger studies for which a ML approach might be particularly suited. Alternatively further investigation of bootstrapping mechanisms could be an area of future research, to compensate the lack of large collections of coded texts.

A deeper investigation, whether and how clustering of word maps built on lexical chains may be incorporated into a coding algorithm. Particularly its suitability for creating new codes could be assessed.

The focus of our research has been the autocoding for an existing coding system. As we have observed during our experiments there may also be an application for NLP techniques to develop, or refine, the coding system itself. Through the application of concept extraction methods, it might be possible for an coding algorithm to adapt to the new data even after the algorithm has been fully trained with dedicated training data.

We also suggest further research into enhancing manual codings through NLP, to empower the researcher with new search tools. Possible research areas include the analysis of the relationships between codes and the involvement of different actors in each code which could reveal new relationships between the actors. This could be implemented relatively easy though the use of NER

Another area for future research is to investigate how the hierarchical structure of a coding system may be utilized in an autocoding system using ML.

References

Auerbach, C. F. & Silverstein, L. B. (2003). *Qualitative data*. New York: New York University Press.

Aston, G., & Burnard, L. (1998). *The BNC handbook*. Edinburgh: Edinburgh University Press.

Barzilay, R. (1997). *Lexical chains for summarization* (Doctoral dissertation, Ben-Gurion University of the Negev).

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1-8.

Bong, S. A. (2002, May). Debunking Myths in Qualitative Data Analysis. In *Forum: Qualitative Social Research* (Vol. 3, No. 2).

Bowley, G. (2010). „Computers That Trade on the News". *New York Times*, *22*, 12.

Braschler, M., & Ripplinger, B. (2004). How effective is stemming and decompounding for German text retrieval?. *Information Retrieval*, *7*(3-4), 291-316.

Britten, N. (1995). Qualitative interviews in medical research. *BMJ: British Medical Journal*, *311*(6999), 251.

Brown, D. (2002, May). Going Digital and Staying Qualitative: Some Alternative Strategies for Digitizing the Qualitative Research Process. In *Forum: Qualitative Social Research* (Vol. 3, No. 2).

Bucholtz, M. (2000). The politics of transcription. *Journal of Pragmatics*, *32*(10), 1439-1465.

Carlberger, J., Dalianis, H., Hassel, M., & Knutsson, O. (2001, May). Improving precision in information retrieval for Swedish using stemming. In *the Proceedings of NODALIDA* (pp. 21-22).

Charmaz, K. (2006). *Constructing grounded theory*. London: Sage Publications.

Collins, M., & Singer, Y. (1999, June). Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora* (pp. 100-110).

Crowston, K., Allen, E. E. & Heckman, R. (2012). Using natural language processing technology for qualitative data analysis. *International Journal Of Social Research Methodology*, 15 (6), pp. 523--543.

Crowston, K., Liu, X. & Allen, E. E. (2010). Machine learning and rule-based automated coding of qualitative data. *Proceedings Of The American Society For Information Science And Technology*, 47 (1), pp. 1—2.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., ... & Funk, A. (2009). *Developing Language Processing Components with GATE Version 5:(a User Guide)*. University of Sheffield.

Davidson, C. (2009). Transcription: Imperatives for Qualitative Research. *International Journal of Qualitative Methods*, *8*(2).

De Ruyter, K., & Scholl, N. (1998). Positioning qualitative market research: reflections from theory and practice. *Qualitative market research: An international journal*, *1*(1), 7-14.

Denzin, N. K. & Lincoln, Y. S. (2003). *Strategies of qualitative inquiry.* Thousand Oaks, CA: Sage.

Denzin, N. K. & Lincoln, Y. S. (2005). *The sage handbook of qualitative research*. Thousand Oaks: Sage Publications.

Ercan, G., & Cicekli, I. (2007). Using lexical chains for keyword extraction. *Information Processing & Management*, *43*(6), 1705-1714.

Fasick, F. A. (1977). Some uses of untranscribed tape recordings in survey research. *Public Opinion Quarterly*, 549-552.

Figuerola, C., Gómez-Díaz, R., Zazo, Á. F., & Alonso-Berrocal, J. L. (2001). Stemming in Spanish: A first approach to its impact on information retrieval. In *Results of the CLEF 2001 Cross-Language System Evaluation Campaign. Working Notes for the CLEF 2001 Workshop. 3 September, Darmstadt, Germany* (pp. 197-202).

Flores, F. N., Moreira, V. P., & Heuser, C. A. (2010). Assessing the impact of stemming accuracy on information retrieval. In *Computational Processing of the Portuguese Language* (pp. 11-20). Springer Berlin Heidelberg.

Francis, J. J., Johnston, M., Robertson, C., Glidewell, L., Entwistle, V., Eccles, M. P., & Grimshaw, J. M. (2010). What is an adequate sample size? Operationalising data saturation for theory-based interview studies. *Psychology and Health*, *25*(10), 1229-1245.

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., & Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction.

Franzosi, R. (Ed.). (2010). *Quantitative narrative analysis* (No. 162). Sage.

Fuller, M., & Zobel, J. (1998, August). Conflation-based comparison of stemming algorithms. In *Proceedings ofADCS'98 Third Australian Document Computing Symposium* (p. 7).

Gelfand, B., Wulfekuler, M. & Punch, W. (1998). Automated concept extraction from plain text.

Gibson, E., Wolf, F., Fedorenko, E., Jones, D., Chuang, C., & Patel, R. (2004, October). Two new experimental protocols for measuring speech transcript readability for timed questionanswering tasks. In *DARPA EARS RT-04F Workshop*.

Glaser, B. G. (2002). Constructivist grounded theory?. *Forum: Qualitative Social Research*, 3 (3).

Glaser, B. G. & Strauss, A. L. (1967). *The discovery of grounded theory.* Chicago: Aldine Pub. Co.

Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough? An experiment with data saturation and variability. *Field methods*, *18*(1), 59-82.

Halcomb, E. J., & Davidson, P. M. (2006). Is verbatim transcription of interview data always necessary?. *Applied Nursing Research*, *19*(1), 38-42.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter, 11*(1), 10-18.

Harman, D. (1991). How effective is suffixing?. *JASIS*, *42*(1), 7-15.

Hirst, G., & St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, *305*, 305-332.

Holloway, I., & Wheeler, S. (2013). *Qualitative research in nursing and healthcare*. John Wiley & Sons. (p. 282)

Hovy, E., Kozareva, Z., & Riloff, E. (2009, August). Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2* (pp. 948-957). Association for Computational Linguistics.

Hulth, A. (2003, July). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing* (pp. 216-223). Association for Computational Linguistics.

Hutchby, I., & Wooffitt, R. (2008). *Conversation analysis*. Polity.

Jefferson, G. (1983). *An exercise in the transcription and analysis of laughter.* Tilburg Univ., Department of Language and Literature.

Johnson, B. E. (2011). The speed and accuracy of voice recognition software-assisted transcription versus the listen-and-type method: A research note. *Qualitative Research*, *11*(1), 91-97.

Junqué de Fortuny, E., De Smedt, T., Martens, D., & Daelemans, W. (2014). Evaluating and understanding text-based stock price prediction models. *Information Processing & Management*.

Knoblauch, H. (2013, September). Qualitative Methods at the Crossroads: Recent Developments in Interpretive Social Research. In *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research* (Vol. 14, No. 3).

Kolb, P. (2008). Disco: A multilingual database of distributionally similar words. *Proceedings of KONVENS-2008, Berlin*.

Kolb, P. (2009). Experiments on the difference between semantic similarity and relatedness. *Forum: Qualitative Social Research*.

Kozareva, Z., Riloff, E., & Hovy, E. H. (2008, June). Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *ACL* (Vol. 8, pp. 1048-1056).

Kraaij, W., & Pohlmann, R. (1996, August). Viewing stemming as recall enhancement. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 40-48). ACM.

Krippendorff, K. (2011). Agreement and information in the reliability of coding. *Communication Methods and Measures*, *5*(2), 93-112.

Kupiec, J., Pedersen, J., & Chen, F. (1995, July). A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 68-73). ACM.

Larkin, F., & Ryan, C. (2008). Good news: using news feeds with genetic programming to predict stock prices. In *Genetic Programming* (pp. 49-60). Springer Berlin Heidelberg.

Lewins, A., & Silver, C. (2009). Choosing a CAQDAS package.

Lin, D., & Wu, X. (2009, August). Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (pp. 1030-1038). Association for Computational Linguistics.

Liu, F., Pennell, D., Liu, F., & Liu, Y. (2009, May). Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics* (pp. 620-628). Association for Computational Linguistics.

MacLean, L. M., Meyer, M., & Estable, A. (2004). Improving accuracy of transcripts in qualitative research. *Qualitative Health Research*, *14*(1), 113-123.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1, p. 6). Cambridge: Cambridge university press.

Martin, A., & Stenner, P. (2004). Talking about drug use: what are we (and our participants) doing in qualitative research?. *International Journal of Drug Policy*, *15*(5), 395-405.

Matheson, J. L. (2007). The Voice Transcription Technique: Use of Voice Recognition Software to Transcribe Digital Interview Data in Qualitative Research. *Qualitative Report*, *12*(4), 547-560.

Maykut, P. S. & Morehouse, R. (1994). *Beginning qualitative research*. London: Falmer Press.

Mays, N., & Pope, C. (1995). Rigour and qualitative research. *BMJ: British Medical Journal*, *311*(6997), 109.

Mehrabian, A., & Ferris, S. R. (1967). Inference of attitudes from nonverbal communication in two channels. *Journal of consulting psychology*, *31*(3), 248.

Mehrabian, A., & Wiener, M. (1967). Decoding of inconsistent communications. *Journal of personality and social psychology*, *6*(1), 109.

McCracken, G. (Ed.). (1988). *The long interview* (Vol. 13) (pp. 41 - 48). Sage.

McLellan, E., MacQueen, K. M., & Neidig, J. L. (2003). Beyond the qualitative interview: Data preparation and transcription. *Field methods*, *15*(1), 63-84.

Mikheev, A. (2000, April). Tagging sentence boundaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference* (pp. 264-271). Association for Computational Linguistics.

Muhr, T. (2000). Increasing the reusability of qualitative data with xml. *Forum: Qualitative Social Research*, 1 (3).

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3-26.

Neale, J., Allen, D., & Coombes, L. (2005). Qualitative research methods within the addictions. *Addiction*, *100*(11), 1584-1593.

Ochs, E. (1979). Transcription as theory. *Developmental pragmatics*, 43-72.

O'Connor, P., & Kleyner, A. (2011). *Practical reliability engineering*. John Wiley & Sons.

Palmer, D. D., & Hearst, M. A. (1997). Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, *23*(2), 241-267.

Park, J., & Zeanah, A. E. (2005). An evaluation of voice recognition software for use in interview-based research: A research note. *Qualitative Research*, *5*(2), 245-251.

Paice, C. D. (1990). Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management*, *26*(1), 171-186.

Payne, T. E. (2006). *Exploring language structure: a student's guide*. Cambridge University Press.

Poibeau, T., & Kosseim, L. (2001). Proper name extraction from non-journalistic texts. *Language and computers*, *37*(1), 144-157.

Poland, B. D. (1995). Transcription quality as an aspect of rigor in qualitative research. *Qualitative inquiry*, *1*(3), 290-310.

Poland, B., & Pederson, A. (1998). Reading between the lines: Interpreting silences in qualitative research. *Qualitative Inquiry*, *4*(2), 293-312.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, *14*(3), 130-137.

Porter, M. F. (2001). Snowball: A language for stemming algorithms.

Radev, D. R., Blair-Goldensohn, S., & Zhang, Z. (2001). Experiments in single and multi-document summarization using MEAD. *Ann Arbor*, *1001*, 48109.

Richards, L. (2002). Qualitative computing--a methods revolution?. *International Journal Of Social Research Methodology*, 5 (3), pp. 263—276.

Saillard, E. K. (2011, January). Systematic Versus Interpretive Analysis with Two CAQDAS Packages: NVivo and MAXQDA. In *Forum: Qualitative social research* (Vol. 12, No. 1).

Saldaña, J. (2012). *The coding manual for qualitative researchers* (No. 14). Sage.

Schachter, P. & Shopen, T. (1985). 1 parts-of-speech systems.

Seale, C., & Silverman, D. (1997). Ensuring rigour in qualitative research. *The European Journal of Public Health*, *7*(4), 379-384.

Silber, H. G., & McCoy, K. F. (2002). Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, *28*(4), 487-496.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631-1642).

Strauss, A. L. (1987). *Qualitative analysis for social scientists*. Cambridge [Cambridgeshire]: Cambridge University Press.

Strauss, A. L. & Corbin, J. M. (1990). *Basics of qualitative research*. Newbury Park, Calif.: Sage Publications.

Strauss, A. L. & Corbin, J. M. (1998). *Basics of qualitative research*. Thousand Oaks: Sage Publications.

Ten Have, P. (2007). *Doing conversation analysis*. Sage.

Turney, P. D. (2000). Learning algorithms for keyphrase extraction. *Information Retrieval*, *2*(4), 303-336.

Verspoor, K., Sanfilippo, A., Elmore, M. & Mackerrow, E. (2006). Deploying natural language processing for social science analysis.

Voutilainen, A. (2003). Part-of-speech tagging. *The Oxford handbook of computational linguistics*, 219-232.

Welsh, E. (2002). Dealing with data: using nvivo in the qualitative data analysis process. *Forum: Qualitative Social Research*, 3 (2).

Yarowsky, D. (1995, June). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* (pp. 189-196). Association for Computational Linguistics.

Yatsko, V. A., Starikov, M. S., Larchenko, E. V., & Vishnyakov, T. N. (2009). The algorithms for preliminary text processing: Decomposition, annotation, morphological analysis. *Automatic documentation and mathematical linguistics*, *43*(6), 336-343.

Yu, C. H., Jannasch-Pennell, A., & DiGangi, S. (2011). Compatibility between Text Mining and Qualitative Research in the Perspectives of Grounded Theory, Content Analysis, and Reliability. *Qualitative Report*, *16*(3), 730-744.

Zaghloul, B. (2014). A Theory of Problems and Solutions in German/Chinese and American/Chinese Software Engineering Collaborations.