Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

# Supporting Interview Analysis with Autocoding

Andreas Kaufmann
Computer Science Department
Friedrich-Alexander-University
Erlangen-Nürnberg, Germany
andreas.kaufmann@fau.de

Ann Barcomb
Computer Science Department
Friedrich-Alexander-University
Erlangen-Nürnberg, Germany
ann@barcomb.org

Dirk Riehle
Computer Science Department
Friedrich-Alexander-University
Erlangen-Nürnberg, Germany
dirk@riehle.org

## Abstract

*Interview analysis is a technique employed in qualitative research. Researchers annotate (code) interview transcriptions, often with the help of Computer-Assisted Qualitative Data Analysis Software (CAQDAS). The tools available today largely replicate the manual process of annotation. In this article, we demonstrate how to use natural language processing (NLP) to increase the reproducibility and traceability of the process of applying codes to text data. We integrated an existing commercial machine–learning (ML) based concept extraction service into an NLP pipeline independent of domain specific rules. We applied our prototype in three qualitative studies to evaluate its capabilities of supporting researchers by providing recommendations consistent with their initial work. Unlike rule based approaches, our process can be applied to interviews from any domain, without additional burden to the researcher for creating a new ruleset. Our work using three example data sets shows that this approach shows promise for a real–life application, but further research is needed.*

## 1. Introduction

Qualitative research methods form one of the main forms of inquiry in social sciences, psychology and market research. A typical goal of qualitative research is to understand behavior, reasoning and opinion. Often, this is accomplished through the analysis of unstructured textual data, which may be derived from expert interviews, discussions, or message logs. In the first stage of qualitative data analysis (QDA), the researcher develops a structured collection of categories of conceptual labels (codes) which are used to annotate the data. These codes represent the researcher's interpretation of the data and are grounded in knowledge of the domain and the research questions. The researcher applies codes to segments of text (coding) which address the concept for which the code was developed. As the researcher proceeds, codes are added, deleted, and modified, and the researcher reviews and revises earlier analysis. The researcher may be aided in this task by Computer–Assisted Qualitative Data Analysis Software (CAQDAS). Current CAQDAS applications provide assistance in identifying relationships between codes and overviews of the occurrence of codes within the text, but only replicate the manual method during the initial stage of coding.

One common concern about the process is *reproducibility* [1, 2, 3, 4]. Because coding depends on the researcher's knowledge and focus, two qualified researchers may emphasize different aspects of the data. This can lead to a lack of consistency, especially in large teams. The coding process may also be seen as lacking transparency and *traceability* because the context of a researcher's reasoning behind a specific coding—which is critical not only for reproduction but also for complete understanding—is not available to others. While these problems can be methodologically mitigated, for instance through investigator or theory triangulation, these measures for ensuring rigor of the research also incur a significant additional investment from the researcher.

We propose the use of natural language processing (NLP) technologies to improve the *reproducibility* and *traceability* of the coding process. In this paper, we present an approach which can be easily adapted for any domain. Our work does not introduce novel ML techniques, but applies established ML-based services like a concept extraction API, to a new domain by integrating them into a NLP pipeline using linguistic and statistical filtering of candidates, in order to improve the researcher's performance. Our algorithm uses previously coded interviews as training data and extracts the semantic context of each applied code in order to propose codes in new data. Inter–team reproducibility is improved by offering prompts which are consistent with previous applications of a code, ensuring that all members of the team have a similar view of a code's meaning. Traceability is improved

URI: https://hdl.handle.net/10125/63833
978-0-9981331-3-3

by automating identification of the underlying semantic context behind a code's application, offering insight into the researcher's thoughts.

While rule-based approaches to autocoding have shown promise in assisting researchers find new evidence for already existing codes, the success of such approaches comes at the cost of having to define rules that are mostly domain specific. This is almost unavoidable, since QDA is highly dependent on the researchers skill and the chosen research question under focus. This is not information that can be modeled with general purpose rules, or extracted from a general purpose corpus. We therefore try a training data based approach. However, the drawback of such in this context is the usually very small sets of data to train on.

The contribution of this work is to provide an initial exploration into the feasibility of using NLP and ML to assist the qualitative researcher in improving the reproducibility and traceability of the coding process through recommendations. Two key aspects of our approach are that it requires no extra effort on the part of the researcher, yet it can be applied to any domain where interview analysis is used. It is important to note that our work is an exploratory proof-of-concept, and is aimed at supporting, not supplanting, the qualitative researcher.

## 2. Related Work

Previous research has examined existing CAQDAS tools, both comparing them [5, 6] and instructing in their use [7].

CAQDAS has been used to facilitate "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" [8], but existing software is largely limited to systematic data management and querying text and codes. Richards [9] argued that the lack of an active and critical academic debate on qualitative computing is due in part to the fact that new technologies are only assessed in terms of old methods, and software is seen as a mere tool to enhance speed and fluidity of an analysis process that would be applied regardless of the technological capabilities. A query of several research databases with the terms "*text mining AND qualitative*" showed that this area of research continues to languish [10].

Text mining and qualitative research have several common principles which make them epistemologically compatible [10]. A hypothetical overview of the use of four different NLP techniques—lexical management, statistical analysis, named entity recognition and pattern learning—hypothesized the effectiveness of NLP for social science analysis but the authors did not implement the tools for the identified use cases [11]. However, it is important for CAQDAS to avoid automating to the point that the reflection and creativity of the researcher are eliminated, and the interpretation is done by the software developers [12].

The only implementation of autocoding using NLP we found was performed by Crowston et al. [13]. They presented a case study in which they built a rule–system to perform autocoding for 12 predefined codes on a set of instant messaging logs and compared the results to a manual coding. In a related work [14], the preliminary results of a ML approach using the same data was published. They concluded that both approaches show promise, but only the rule-based approach was described in detail. Deriving the code system from the researcher, rather than from a general corpus, allows for the researcher's knowledge of the domain and the research questions to be taken into consideration. However, a rule–based approach necessitates explicitly expressing the concepts behind the codes in a specific format. By contrast, our work is able to derive the meaning from the application of codes. This makes our approach domain independent while still respecting the researcher's knowledge. This is demonstrated by making use of example data from two different domains.

## 3. Research Approach

Our goal was to determine how established NLP technologies and services may be applied to interview analysis. We do not propose that the use of NLP techniques should replace the interpretative ingenuity of the qualified researcher, but performing content analysis through the use of NLP and automating suggestions may aid the researcher, for instance by ensuring consistency. Naturally assistive tools should be developed with an awareness of the potential for tools to alter decision making processes, for instance by anchoring selections [15] or limiting the development of new codes. Our work is exploratory, with the purpose of determining if a general corpus-based NLP approach shows any promise in supporting QDA.

NLP comprises a range of methods and techniques applicable for different use cases, four categories of which appear to be well–suited for the enhancement of QDA. Table 1 provides an overview of the potential applications, with our area of research using corpus based tools in highlighted text. Complementary to Crowston et al.'s research on a rule–based approach, we explore a machine learning methodology and determine how accurately an autocoding algorithm incorporating existing technologies can reproduce the results of a trained researcher [13].

Table 1: NLP techinques for QDA

| Task | Method | Tools |
|---|---|---|
| Find new codings | Rule based | Manually written rules tailored to specific codes |
| | | Pattern analysis |
| | Corpus based (machine learning) | Keyword extraction |
| | | Named entity recognition |
| | | Lexical databases |
| | | Statistical linguistics |
| | | Sentiment analysis |
| Generate new codes | Machine learning | Concept extraction |
| | | Keyword extraction |
| | | Sentiment analysis |
| Enhance manual codes | Generation of structured metadata | Sentiment analysis |
| | | Named entity recognition |
| Facilitate translations | Dictation | Voice recognition software |

To assess the applicability of NLP to the problem of autocoding we implemented a prototype to process experimental results using three interview sets (see section 4). The manual coding was performed in MAXQDA[1] and exported to the XML file format.

Our algorithm is trained using a subset of the data, while the remainder is reserved for testing. Any ML method will require a certain minimum amount of data to perform well. Our method is therefore not aimed at projects consisting of two or three interviews. Because our available data sets were smaller than ideal, we used a high proportion of test data. We maximized the available training data by using N-1 of the N available interviews in Interview Sets 1 and 2 to train, and the remaining interview to test. We repeated this method for every interview in both data sets, resulting in N trials per data set. After training, we compared the results of the autocoding against the actual coding of the interviews by the original researcher. In the third trial, using Interview Set 3, we used coded interviews as training data and created suggested codings for uncoded interviews. The proposed codes were supplied to the original researcher, who manually evaluated the effectiveness of the autocoding.

We describe the method in more detail in Section 5, illustrating with examples from the data sets described in Section 4.

### 3.1. Metrics

To measure the success of our autocoding algorithm on Interview Sets 1 and 2, we used *recall* and *precision*. Recall describes the percentage of manual codes that could be replicated by the algorithm, while precision measures the percentage of autocoded codes that are correctly applied.

Both metrics are, to some extent, inversely correlated. Coding each interview with all possible codes would yield 100% recall, but this approach would be punished with a low precision value. Applying only correct codes sparingly would give high precision but low recall. We consider the ideal algorithm to be one which applies all the same codes as a human coder, with no additional codes, thus displaying 100% for both recall and precision.

The algorithm only applies one most likely code for each paragraph, meaning that if that code was also coded by the researcher this paragraph is considered coded correctly. A paragraph with many codes applied to it is thus easier to correctly autocode than one with only one manually applied code, leading to higher recall and precision values for densely coded data.

Therefore the result of our NLP pipeline is a relevance value for each code to each paragraph of uncoded data. The relevance value determines if a code should be automatically applied, or reccomended, for a specific paragraph in the new data.

Precision and recall are not the only metrics that could be employed to measure replication of human behavior; a detailed discussion of intercoder agreement in QDA can be found in Krippendorff [16]. We opted to use two standard metrics for information retrieval (IR) for the sake of simplicity.

For Interview Set 3, we relied on the human coder's report and focused on the insightful suggestions, as our previous experiments provided a clear picture of how the techniques performed at replication.

## 4. Data Sources

For our research we focused on three different sets of interviews, which were chosen as examples in part due to their availability to us. All three were in English.Table 2 summarizes the data by the number of codes used in our analysis (top– or second–level), the total number of codes, and the number of codings applied by the researchers. The depth of the code system was 4 levels in data sets 1 and 2, and 3 levels in data set 3. The Each set is described in more detail below.

*Interview Set 1* comes from the domain of software engineering and consists of six unstructured interviews about inter–cultural challenges that ensue when collaborating with Chinese software development teams, from the perspective of German and US developers [17]. An example code in this data set would

[1] http://www.maxqda.de

Table 2: Overview of Data Sources

| Set | Domain | Used codes | Total codes | Codings |
|---|---|---|---|---|
| 1 | Soft. Eng. | 13 | 68 | 441 |
| 2 | Sociology | 9 | 40 | 315 |
| 3 | Soft. Eng. | 46 | 104 | 506 |

be *Trust* > *Transparency* > *IP Rights*.

*Interview Set 2* contains 11 structured interviews about general life satisfaction [18].We used the second data set to determine if our approach was suitable not only for software engineering, but other domains. An example code in this data set would be *People* > *Parents*.

The training data for *Interview Set 3* was six semi–structured interviews with Free/Libre and Open Source Software community managers on the topic of episodic volunteering in their communities [19]. With this data set we used second-level codes rather than top-level codes. We proposed codes for two new interviews. An example code in this data set would be *Contributors* > *motivation* > *altruism*.

We used this data set to evaluate our method manually. The researcher was asked to evaluate the suggested codes during the coding process to determine if they aligned with her own choice, were inappropriately applied, or provided new insight resulting in the application of a code she would not have organically applied.

## 5. NLP Pipeline and Results

As the purpose of our study was to examine the feasibility of using NLP to assist in QDA, we present our results as a more detailed description of our approach, interwoven with examples from each data set in turn.

### 5.1. Data Preparation

Pre-processing of interview data is necessary to preserve useful meta information before parsing the document. For this study the process was only partially automated, as the data was not consistently formatted and needed to be manually reformatted into a common format. When the text is parsed we can identify section headers and differentiate between questions and answers.

While parsing the document some metadata is also removed which is not useful and may skew the results. These include time markers which may be included in audio transcriptions. Such imperfections in the data represent one of the challenges to be faced when applying text processing software to real–world situations.

We also generate statistics of word frequencies relative to the current document and all documents.

### 5.2. Processing

An overview of the entire process, from data preparation through autocoding, is shown in Figure 1.

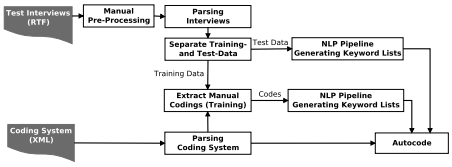


Figure 1: Process overview

The NLP pipeline component of Figure 1 is shown in more detail in Figure 2. Each component of the NLP pipeline is described in greater detail below.

**Keyword Extraction.** For each code we look up the coded text within the training data and use the AlchemyAPI[2] keyword extraction service on each coding. The keywords extracted will be assigned as semantic content for the corresponding code and can later be used to identify semantically similar text in the test data. The same process is used to determine the semantic content of each interview element. An interview element is either a question, an answer, or metadata.

Keyword extraction is used to identify semantically significant words in a text. In Figure 3 we present an example result, taken from Interview Set 1, where the extracted keywords are highlighted.

In this example, the text was manually coded with the code *Turnover*. It demonstrates the significance of attributing actual semantic context to a code. Some autocoding features in existing 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 our example, this technique would have led to the false

[2]http://www.alchemyapi.com

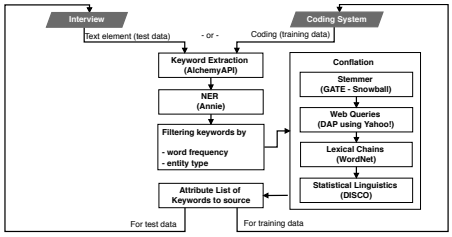


Figure 2: NLP pipeline overview

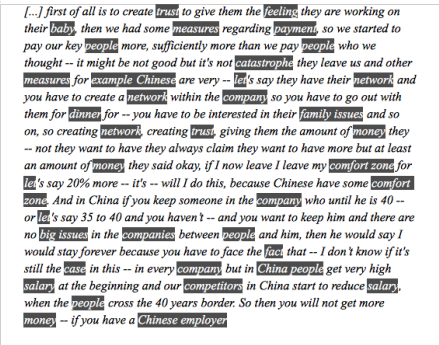


Figure 3: Example keyword extraction

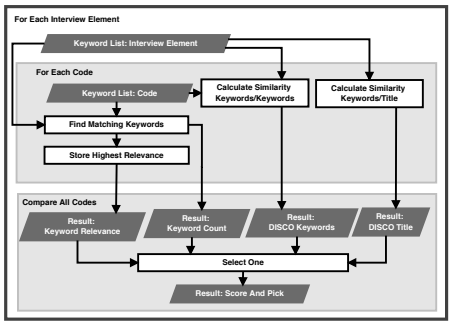


Figure 4: Autocoding overview

application of the code *Trust*, because this word occurs twice. We assume that the researcher did not apply the code *Trust* in this example because uses of this code have the semantic context of (mis)trust between collaborating software developers, whereas in this paragraph the term occurs in the context of motivation. Our algorithm correctly detects that most of the text concerns money, payment and salary, which are closely identified with the code *Turnover*.

**Conflation and Elimination.** An important part of our autocoding algorithm is different conflation methods, creating equivalence classes for token normalization. This allows us to match semantically similar codes to a text even though the vocabulary within the training data may differ from that used in the test data.

Conflation can also be performed on derivations of the word, as well as semantic relationships from a lexical database. The more coarse–grained our conflation methods are, the more information will usually be lost, but the more connections between codes and text may be uncovered. The drawback of lost information is that these new connections may be based on false assumptions. Conflating "China" with "china" draws a connection between a country and porcelain which is undesired, because although the terms are etymologically connected, they are semantically distant.

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

For lexical chains we used WordNet[3]. 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 term. Through this relation we create a hierarchy of abstractions. We then configured our prototype to conflate a term with a fixed degree of parent and child nodes.

For stemming we chose the snowball stemmer integrated in the GATE framework [4], which is based on the Porter stemmer. The stemmer performs the following 5 steps: (1) recode pluras, (2) remove -ed or -ing, (3) recode y to i, if the stem contains another vowel, (4) handle double suffix and (5) remove suffixes, if the removal rule is not violated for the remaining stem.

For conflation through web queries we relied on Doubly-Anchored Pattern (DAP) [20] with the Yahoo! web search engine. The DAP pattern is *[SeedTerm1] such as [SeedTerm2] and [X]*. For SeedTerm1 we used the word *concept* and for SeedTerm2 we used the term we were looking to expand. An example search term would be *concepts such as trust and*.

## 5.3. Autocoding

During the autocoding process we match each interview element with a code that is considered to be semantically closest to the content of the text.

We used complete paragraphs as the unit of coding. We chose this because the theme of an answer in the interview became much stronger in the sentences were not addressed individually. We unsuccessfully experimented with linguistic inter-sentence dependencies based on rules using part-of-speech tagging.

An overview of the autocoding process is shown in Figure 4. The autocoding process is performed on all interview element types. This is helpful because a question may sometimes give a more precise hint about the context of an answer.

The coding algorithm has two main input parameters. First, we have the interview element, containing the actual text, metadata on the interview

[3]http://wordnet.princeton.edu

[4]https://gate.ac.uk

element type, and a list of keywords associated with the text. Second, we have a set of codes (the coding system), each consisting of a set of codings and keywords associated with this code. In order to simplify the problem, we flattened the coding hierarchy and worked with individual codes.

The goal of our algorithm is to select the correct code from the coding system or to select none, if the algorithm determines that the semantic similarity of the text to any codes is below a certain threshold. We initially set a low threshold, so that even weak semantic connections appear when comparing our different methods.

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 eliminate the least likely candidates among all assigned codings. Criteria for this selection are the length of the text to be coded and the score of the assigned code, which measures the strength of the semantic connection between code and text. Codes assigned to questions or metadata are also deleted in this step, to align with a coding guideline employed by the human coders of data set 1 and 3 where questions were only coded together with the answer when the answer was not comprehensible on its own.

**Coding Methods.** To calculate the semantic similarity an interview element to a set of codings, we experimented with different technologies.

Our first approach was **Keyword Count**, which compares a list of keywords generated for an interview element to each list of keywords assigned to each code, and counts the number of matching keywords. We then normalized the count, to account for the fact that codes which have been assigned more frequently are more likely to contain matching keywords.

As a baseline to compare the performance of our Keyword Count method, and to evaluate the suitability of the chosen keyword extraction service is **Word Count**. Word Count works exactly like Keyword Count, except that every word is treated as a keyword.

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 in the **Keyword Relevance** method. Through experimentation, we found that this method, like Keyword Count, benefits from factoring in the size of the text to code and the existing evidence of a code.

We wanted to determine how similar a text is to the training data even when the vocabulary is completely disjunct. For this we made use of the DISCO[5] tool. To determine the semantic distance of two words, DISCO uses statistical analysis on large corpa. 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 two words in a three word window within the training corpus or by measuring the distributional similarity [21]. DISCO creates word position triplets containing two words and their distance in a window of +/- 3 words. As a first metric, the number of co-occurrences of both words normalized by their overall frequency is calculated. In a further step, the two words are then compared based on their sets of distributionally similar words to identify words that are not necessarily co-located directly but have similar sets of co-occurring words.

In our **DISCO Title** approach, 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. A significant advantage of this approach is that no training data is required to perform the autocoding.

We extended our previous DISCO approach with the **DISCO Keyword** method, which included the training data. The similarity between each keyword of each code and each keyword in the text are also calculated and normalized.

In another approach, **DISCO Inflation**, we extended the vocabulary of the keyword list by adding frequently co–occurring words within the BNC extracted through DISCO, and subsequently applied the Keyword Count method.

We ultimately implemented a weighting scheme to leverage the benefits of the techniques under different circumstances. The weighting scheme, **Score and Pick**, chooses one of the methods described previously and assigns the recommended code to a paragraph. For instance, Score and Pick would choose the code suggested by Keyword Relevance if the assigned relevance variable was above a given threshold, or if the similarity to the codes name was significantly above average it would choose DISCO Title. We found that identifying good values for parameters is highly sensitive to the type of data, even within one study. Therefore these should ideally be learned on a training set of the actual data.

Finally for our third data set and the manual evaluation, we added suggestions based on a simple pattern matching algorithm similar to what current CAQDAS packages offer, but different in that it only tries to match code names with identified keywords and any of their conflated and related terms or their word stems. We refer to this method as **Direct Match**.

[5]http://www.linguatools.de/disco/disco_en.html

[6]http://www.natcorp.ox.ac.uk

Table 3: Recall and Precision for data set 2

| Codes | Recall | Precision | Manual Codings |
|---|---|---|---|
| Day-to-Day issues | 79.31% | 67.64% | 29 |
| Interview Guideline Topics | 70.00% | 43.75% | 40 |
| Key Quotes | 0.00% | 0.00% | 6 |
| People | 14.29% | 33.33% | 7 |
| Challenges | 0.00% | 0.00% | 1 |
| **Total** | **62.65%** | **50.98%** | |

The input to all these methods was already processed by the rest of the NLP pipeline.

## 5.4. Initial Results

Table 3 shows the results of our algorithm applied to two interviews from data set 2. The table shows recall and precision for each of the codes as well as the number of instances this code was applied by the researcher in the test data.

Our method only applied one top-level code to each paragraph. This proved to perform more reliably, because for each of the top level codes there was more aggregated training data compared to each individual lower level code. When autocoding all individual codes on any abstraction level, the precision fell to 29.7% for all codes aggregated. We attribute this effect mostly to the different amount of training data per code, but it remains open how much other factors, such as the abstraction level itself, contribute to this relationship between number of codes and precision. The statistical effect of a higher chance to choose the wrong code out of a larger set by chance does not appear to be the main factor in our case, since recall was not only not affected, but even slightly improved to 69% when autocoding on all abstraction levels of the code system.

The code *Challenges* shows an inherent drawback of our approach requiring training data. This code appeared exclusively in one interview. Therefore, when our data was divided into training and test data, there was no possibility for instances to be included in both.

## 5.5. Methods for Semantic Matching

**Keyword Frequencies.** We experimented with the frequency threshold under which a keyword is considered during autocoding. The apparent trend of our Keyword Count method is that it performs better if we allow keywords to be associated with a higher number of codes. On the other hand, we observed that methods that draw on each keyword's relevance, or semantic meaning, show the opposite reaction to

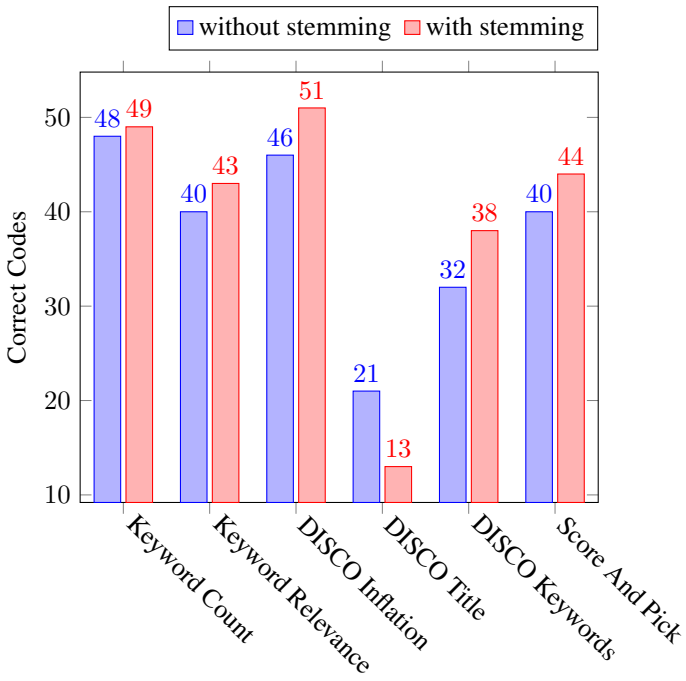


Figure 5: Impact of conflation (stemming)

changing the threshold variable. The DISCO Title method is unaffected by this variable, as it does not rely on training data.

When comparing the Keyword Count method to the baseline Word Count method, We observed that using a sophisticated keyword extraction method can double the number of correctly identified codings.

We also measured the impact of limiting the frequency of a word in regards to its overall occurrence. 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 the test runs, the only method which slightly benefited from a lower threshold was DISCO Keywords.

**Named Entities.** In an attempt to tailor the parameters of the algorithm more closely to our semantic context, we experimented with removing specific entitles that are possibly too common in our context. However, the results of measuring the impact of named entity keywords showed, overall, the opposite trend. Only the methods Keyword Count and DISCO Keywords were affected.

**Stemming.** In Figure 5 we present the impact of one form of conflation, stemming the keywords on our autocoding algorithm. The results show a significant variance, but aggregated we see a slight advantage

in including stemming. For instance, with DISCO Inflation, the number of correctly applied codes rose from 46 to 51 when conflated through stemming. This method is clearly not suited for DISCO Title, but it increased the number of correctly applied codes for all other methods by an average of 9.23%.

**Web Queries.** Our conflation approach using web queries was trialed on code titles exclusively, and on all keywords. Overall, Keyword Count and Keyword Relevance showed a slight improvement with this method, while DISCO Keywords performed more poorly. DISCO Title was unaffected as it does not rely on training data, while DISCO Inflation benefited from using web queries on code titles and suffered from its use on all keywords.

**WordNet.** We conflated all keywords with hyponyms, hypernyms, meronyms, holonyms and derivationally different forms. While this conflation alone did not provide a significant improvement, it showed promising results for interviews which had already been processed using stemming and web queries, further improving the number of correctly applied codes by up to 17%.

### 5.6. Manual Evaluation

With data set 3, the researcher worked through the two test interviews by first coding a paragraph and then comparing her codings to the codings proposed by each method. She indicated if the proposed code had already applied (accepted), was not appropriate for the context (rejected), or was appropriate and had not already been applied (inspirational). Table 4 shows the aggregate of all three types of results across the two interviews.

In terms of replicating the researcher's work, DISCO Keywords has the highest fidelity, with only two rejected codings. If we consider only these codings, the success rate of our recommendations is excellent. However, these recommendations are limited to a fixed unit of coding. A researcher could apply multiple codings to a paragraph, whereas our method is designed to promote a single code for a paragraph or choose none. This limitation reduces the complexity of the recommendation system significantly and makes sensible suggestions possible in the first place while limiting the scope of the recommendations.

Several methods were able to identify appropriate keywords which the researcher overlooked. Of the 14 successes, there were 12 unique codings proposed for 11 different paragraphs.

One example of an inspirational coding was the following, which the researcher coded as *practices* $>$ *guiding*, and *practices* $>$ *communication*. DISCO Inflation suggested *contribution types* $>$ *translation*.

> Yes. The contact is synchronize first, to manage the status, to notify about the new events, to introduce newly funded project and to tell them how the projects work, where the resources if they have questions to answer their questions. There is a specific case of the translation project because in the translation it's important that all the applications have the same English word and the same expression always translate the same way. So there is a need to big—to match communication between the translators.

The former example could have been detected by a naïve approach, as the code appears in the text. It could be said that the researcher merely overlooked the applicable coding. However, there were also instances of more revealing codings, which required inference, such as the suggestion by DISCO Inflation of *market share* for the following paragraph.

> So I mean we don't really know how many users we have, I mean that's probably millions that would be a really—really large community but of course people are not interacting usually, so I would say the core community that's probably few thousand people once in a while do something related to KDE.

To suggest the code *market share*, the algorithm needed to link millions with non–active people in the community, and find the similarity between this and other applications of the code, which involved counts of users.

Table 4: Results of manual evaluation (data set 3)

| Method | Accept | Reject | Inspire |
|---|---|---|---|
| Direct Match | 185 | 43 | 4 |
| Word Count | 178 | 51 | 3 |
| Keyword Count | 188 | 44 | 0 |
| Keyword Relevance | 179 | 50 | 3 |
| DISCO Title | 194 | 38 | 0 |
| DISCO Keywords | 230 | 2 | 0 |
| DISCO Inflation | 186 | 44 | 2 |
| Score And Pick | 183 | 47 | 2 |

## 6. Discussion

Since inter-rater agreement scores are highly dependent on the context of the study it is hard to define

generally applicable values for ”good” agreement. Overall, we found that our autocoding approach, not surprisingly, fell short of replicating human effort. However, DISCO Keywords showed high fidelity when it was manually compared to Interview Set 3, which suggests that the success or failure of the method may be related to the researcher or domain. Especially if precision can be improved in the future, we believe an autocoding recommendation system can significantly improve the QDA process.

DISCO Title had the benefit of being independent of any training data, however it relies heavily on good naming of the codes. As it is evidenced in Figure 5 the method doesn't work when the codes are too broad like in our experimentation with data set 1 and 2. However, it doesn't under-perform anymore when the codes are more precise and specific like in our experimentation with data set 3.

When we attempted to provide new information to a researcher rather than to compare our results to existing codings, the method showed that the algorithm can find evidence of new, appropriate codes that the human coder overlooked. Of course the difference between inspirational and accepted codes as we defined them was whether the researcher had also applied them. However, the fact that inspirational codes existed demonstrates that the measurement used in the first two data sets, comparing the recommendations to previously coded text segments, does not fully measure success of the method, as the data may be missing codings.

In most of the instances the researcher reported that the suggestions were not so much inspirational as helpful in ensuring consistency given a large number of codes to keep in mind. In a situation such as the one presented by Interview Set 3, where there were 104 different codes, the reproducibility of the work is improved when the CAQDAS is able to recommend codings based on keywords retrieved from data coded by the original researcher.

Within Interview Set 3, the inspirational codings were never identified with multiple methods. In most cases, the other methods all proposed one coding, while the dissenting method revealed an insightful, less obvious coding. No method clearly stood out as consistently providing inspirational codings, indicating that each method may have distinct benefits which are not present in the other methods.

This research demonstrates that an approach using ML and NLP has potential in an assistive capacity for qualitative interview analysis. Our results show more support for assisting researchers in maintaining consistency given a large number of codes, but there were also instances where the researcher may be directed toward additional meaning through NLP. There is significant room for further research to improve the quality of the results and to present the information to the researcher in a helpful but unobtrusive manner. Nonetheless, we feel that CAQDAS could potentially include this type of support, particularly if discipline–specific corpa could be employed.

## 7. Limitations

During our research process we identified some limitations to autocoding in general and our approach specifically.

While keyword extraction for natural language is largely independent of the analysis being done, the semantic generalization needed for autocoding is not. For general purpose NLP, large corpora of texts can be used as training data, but such a large set of training data is not available for autocoding. Instead data explicitly tailored to the domain of analysis is required. The amount of training data available is therefore small by comparison, constituting only a few interviews in each case within our examplary cases.

One technical problem, which precluded us from using multiple data sources coded with different tooling, was the lack of a standardized representation of coded data. One development which will aid in the exploration of using NLP to assist QDA is the REFI-QDA XML[7] exchange format, which will simplify the task of writing software by eliminating the data preparation phase.

A general limitation of any ML algorithm arises due to the requirement of a certain amount of training data. Any code which has few occurrences in the training data might not be detected. These types of codes are common although they should only constitute a small fraction of the code system. A general restriction in this respect is the frequently small sample size of many qualitative research studies. For smaller sample sizes a rule–based approach may be more appropriate.

We addressed potential overfitting of our algorithm by using three different qualitative studies. However, some variables in our NLP pipeline were configured manually. It remains open how well a fully automated algorithm would adjust to a new data set.

Another limitation is, that only the top-1 result is either chosen or discarded for any paragraph. Further investigation into an interactive recommendation system based on a ranking like ours is needed.

[7] https://www.qdasoftware.org

## 8. Conclusions and Future Work

We assessed the applicability of a wide range of NLP methods in an experimental autocoding system for unstructured, semi–structured and structured interviews. We achieved 45.83% recall and 45.83% precision for Interview Set 1 and 62%–69% recall and 29.71%–50.98% precision on Interview Set 2, depending on the abstraction level of the codes. For Interview Set 3, our methods were able to identify some suggested codings which the researcher chose to adopt.

We identified beneficial effects for word conflation through stemming, DAP and lexical chains. Considering the question–answer sequence of interviews was also helpful. No positive effect was observed for using word frequencies related to the documents, or filtering keywords by entity type.

Further research is needed to improve upon the recall and precision, and to address some of the complexities of QDA we simplified for our investigation. In particular, future work could incorporate the hierarchy of the code system, include memos, and consider overlapping and variable–length codings.

We had relatively small data sets, which showed some limited value of NLP. Future work could examine how much data is necessary for our approach to be beneficial to the researcher. This might be considered in relation to saturation, which is a common standard by which the qualitative researcher determines if the existing data is sufficient.

Further research may also focus on some of the other fields of application, like generating recommendations for new codes.

Our research demonstrates the exceptional adaptability of a domain independent NLP and ML based approach for autocoding using parts of the study to be aided as training data. Although our final results show significantly lower coding agreement compared to the rule–based approach, the ML approach does not require the researcher to adapt the algorithm for a specific domain or data set. Our work demonstrates the potential of using NLP approach to autocoding to increase reproducibility and traceability in QDA.

## References


[1] N. Mays and C. Pope, "Qualitative research: rigour and qualitative research," *Bmj*, vol. 311, no. 6997, pp. 109–112, 1995.

[2] A. Martin and P. Stenner, "Talking about drug use: what are we (and our participants) doing in qualitative research?," *International Journal of Drug Policy*, vol. 15, no. 5, pp. 395–405, 2004.

[3] J. Neale, D. Allen, and L. Coombes, "Qualitative research methods within the addictions," *Addiction*, vol. 100, no. 11, pp. 1584–1593, 2005.

[4] K. De Ruyter and N. Scholl, "Positioning qualitative market research: reflections from theory and practice," *Qualitative market research*, vol. 1, no. 1, pp. 7–14, 1998.

[5] A. Lewins and C. Silver, "Choosing a CAQDAS package," tech. rep., University of Surrey, 2009.

[6] E. K. Saillard, "Systematic versus interpretive analysis with two CAQDAS packages: NVivo and MAXQDA," *Forum: Qualitative Social Research*, vol. 12, no. 1, 2011.

[7] E. Welsh, "Dealing with data: Using NVivo in the qualitative data analysis process," *Forum: Qualitative Social Research*, vol. 3, no. 2, 2002.

[8] S. A. Bong, "Debunking myths in qualitative data analysis," *Forum: Qualitative Social Research*, vol. 3, no. 2, 2002.

[9] L. Richards, "Qualitative computing–a methods revolution?," *International Journal of Social Research Methodology*, vol. 5, no. 3, pp. 263–276, 2002.

[10] C. H. Yu, A. Jannasch-Pennell, and S. DiGangi, "Compatibility between text mining and qualitative research in the perspectives of grounded theory, content analysis, and reliability," *The Qualitative Report*, vol. 16, no. 3, p. 730, 2011.

[11] K. Verspoor, A. Sanfilippo, M. Elmore, and E. MacKerrow, "Deploying natural language processing for social science analysis," in *Chicago Colloquium on Digital Humanities and Computer Science*, 2006.

[12] H. Knoblauch, "Qualitative methods at the crossroads: Recent developments in interpretive social research," *Forum: Qualitative Social Research*, vol. 14, no. 3, 2013.

[13] K. Crowston, E. E. Allen, and R. Heckman, "Using natural language processing technology for qualitative data analysis," *International Journal of Social Research Methodology*, vol. 15, no. 6, pp. 523–543, 2012.

[14] K. Crowston, X. Liu, and E. E. Allen, "Machine learning and rule-based automated coding of qualitative data," *Proceedings of the American Society for Information Science and Technology*, vol. 47, no. 1, pp. 1–2, 2010.

[15] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl, "Is seeing believing?: how recommender system interfaces affect users' opinions," in *Conference on Human factors in computing systems*, pp. 585–592, ACM, 2003.

[16] K. Krippendorff, "Agreement and information in the reliability of coding," *Communication Methods and Measures*, vol. 5, no. 2, pp. 93–112, 2011.

[17] B. Zaghloul, "A theory of problems and solutions in German/Chinese and American/Chinese software engineering collaborations," Master's thesis, Peking University, 2014.

[18] C. Silver and A. Lewins, *Using software in qualitative research: A step-by-step guide*. Sage, 2014.

[19] A. Barcomb, A. Kaufmann, D. Riehle, K.-J. Stol, and B. Fitzgerald, "Uncovering the periphery: A qualitative survey of episodic volunteering in free/libre and open source software communities," *IEEE TOSEM*, 2018.

[20] Z. Kozareva, E. Riloff, and E. H. Hovy, "Semantic class learning from the web with hyponym pattern linkage graphs.," in *ACL*, vol. 8, pp. 1048–1056, 2008.

[21] P. Kolb, "Experiments on the difference between semantic similarity and relatedness," in *ODALIDA*, pp. 81–88, 2009.