Using natural language processing technology for qualitative data analysis

Kevin Crowston[a*], Eileen E. Allen[a], Na "Lina" Li[b], Michael J. Scialdone[a] and Robert Heckman[a]

[a] School of Information Studies, Syracuse University, Syracuse, NY, United States
[b] Center for Graduate Studies, Baker College, Flint, MI, United States

30 May 2010

Social researchers often apply qualitative research methods to study groups and their communications artefacts. The use of computer-mediated communications has dramatically increased the volume of text available, but coding such text requires considerable manual effort. We discuss how systems that process text in human languages (i.e., natural language processing, NLP) might partially automate content analysis by extracting theoretical evidence. We present a case study of the use of NLP for qualitative analysis in which the NLP rules showed good performance on a number of codes. With the current level of performance, use of an NLP system could reduce the amount of text to be examined by a human coder by an order of magnitude or more, potentially increasing the speed of coding by a comparable degree. The paper is significant as it is one of the first to demonstrate the use of high-level NLP techniques for qualitative data analysis. (150 words)

Keywords: natural language processing, qualitative data analysis, coding, group maintenance

Word count: 5995 words

* Corresponding author. Tel: +1 315 464–0272. Fax: +1 866 265–7407. Email: crowston@syr.edu

Biographical notes

Kevin Crowston is a Professor in the School of Information Studies at Syracuse University. He received his PhD in Information Technologies from the Sloan School of Management, Massachusetts Institute of Technology. His research examines new ways of organizing made possible by the extensive use of information and communications technology. Specific research topics include the development practices of Free/Libre Open Source Software teams and work practices and technology support for citizen science research projects, both with United States National Science Foundation support.

Eileen E. Allen is an Analyst in the Center for Natural Language Processing in the School of Information Studies at Syracuse University. She has served as Project Leader and as a natural language processing analyst. In this role she has written pseudo-code that enables the TextTagger software to identify meaningful concepts and relationships within unstructured texts, like newspaper articles. Eileen holds New York State teaching certification for Elementary Education and for Music, and received her MLS from Syracuse University in 1990. She has been an academic librarian and has done free lance indexing and editing for faculty publications, doctoral dissertations and state and city government.

Na "Lina" Li is an Assistant Professor in the Center for Graduate Studies at Baker College. She received her PhD in Information Science & Technology from Syracuse University. Her research interests reside in human-computer interaction and group dynamics in virtual teams. Specific interests include group member behaviours in Free/Libre Open Source Software teams, information system evaluation and adoption, e-

commerce website evaluation and use, affect, cognition and behaviour in human-computer interaction.

Michael J. Scialdone is a doctoral candidate in Information Science and Technology at the School of Information Studies at Syracuse University. He earned a BA in Communication Arts from Utica College in 1999, and an MS in Information Design and Technology from State University of New York Institute of Technology (SUNYIT) in 2006. His research interests include social media integration within distance-based learning environments, and presentation of self in immersive 3D virtual worlds. He is currently serving as managing editor of Association for Information Systems *Transactions on Human-Computer Interaction*.

Robert Heckman is Senior Associate Dean and Associate Professor at the School of Information Studies at Syracuse University, where he teaches strategic management of information resources, information industry strategies, and information consulting. His research interests include Strategy and Planning for Information Resources, and Teaching and Learning Strategies for Information Professionals. He has more than 20 years experience in the information services industry as a senior manager of data processing operations, systems development, and information systems marketing. He received his PhD in information systems at the Katz Graduate School of Business, University of Pittsburgh.

Using natural language processing technology for qualitative data analysis

Introduction

Social researchers often employ qualitative methods to understand the work practices of groups. For example, researchers might examine transcripts of a group's discussions to understand how it solved some task and the impact of different approaches (e.g., Benbunan-Fich, Hiltz, & Turoff, 2003). Because such data are textual, they require considerable manual effort to analyze. To support such analyses, researchers often use Computer-Assisted Qualitative Data Analysis Software (CAQDAS) tools (e.g., Atlas.ti, Hyper-research or Nudist, Barry, 1998; Lee & Esterhuizen, 2000). The most advanced offer capabilities for automatically coding text, but most current CAQDAS tools simply manage the traditional processes of coding and retrieving coded segments (Richards, 2002). While computer support does provide considerable benefits, analyzing significant volumes of text still requires considerable effort from researchers, who must read and reread to make sense of the text and to locate evidence to support or refute their theories. As a result, qualitative research addressing important questions in social research often relies on small sample sizes because of the analysis effort required.

Fortunately, recent years have seen a great growth in the capability of computer systems to process text in human languages, paralleling the growth in the volume of computer-readable text. There is a wide diversity of techniques and approaches, but we refer to these technologies collectively as Natural Language Processing (NLP). In this methodological paper, we introduce methodological techniques from the discipline of NLP to social researchers to show how NLP might be applied to provide advanced analytic capabilities to support analysis of textual data such as communication artefacts.

If successful, NLP tools could advance the work of social researchers by extending the capabilities of current tools and enabling researchers to explore massive data sets in greater depth. The contribution of this paper is to introduce the NLP methodology to social researchers and to demonstrate the potential and limitations of NLP tools for supporting social research.

Before discussing NLP technology and its possible application to qualitative data analysis, we need to clarify the focus of our work. There is a widespread assumption that qualitative work is always interpretivist (i.e., focused on understanding individuals' concepts of their social worlds) but qualitative research can, in fact, adopt any perspective: positivist, interpretivist or critical (Myers, 1997). In our studies, we are interested in the behaviour of the groups studied and assume that their work and social processes are accurately reflected by the texts that they produce. Rather than seeking to uncover latent or hidden meanings in the text, we look for evidence of particular behavioural patterns of the participants. These features make our approach essentially positivist, despite its reliance on qualitative data.

In this paper, we explore how NLP techniques can be applied to support a particular task in positivist qualitative research, namely coding for content analysis. We focus in this paper specifically on coding and do not address the use of the coded data, as critical as that is to research overall (Richards, 2002). Content analysis is a qualitative research technique for finding evidence of concepts of interest using text as raw data (Myers, 1997). The result of the coding process is a text annotated with codes for the concepts exhibited (Miles & Huberman, 1994). In the approach we describe, codes are applied based on the features of specific segments of text, rather than with the goal of

understanding and interpreting the entire text as a whole. For example, if the focus of a study is group decision making, then transcripts of interactions are coded for evidence of theoretical constructs of interest, such as problem identification or introduction of an alternative. The goal of coding texts is to be able to study the relationship between concepts as expressed in the text (either inductively or deductively). For example, the coded text could be used to examine hypotheses such as the relationship between participation in decision-making and the overall effectiveness of a team.

A key concern in coding is reliability, as measured by the degree of inter-rater agreement, that is, whether different coders working on the same text identify the same set of codes. If coders do not agree, then they discuss the coding until they reach a better level of shared understanding of the code. Codes and coding decisions are documented in a codebook. However, it must be admitted that a great deal of tacit knowledge is used in coding, meaning that coders need to be trained to code reliably. Once the coders are coding reliably, they must read the texts to code them for the concepts of interest, which can be quite labour intensive for a large corpus. For example, to study decision making in an online group would require reading all (or a large number) of the emails exchanged among members looking for evidence of constructs related to decision making. As a result, research teams often face limitations in the scope of analysis feasible based on the available work force. It is this problem of scale that we seek to address by using NLP technology.

In the following sections, we first introduce NLP and discuss its capabilities and the underlying theoretical foundations of its use. We then present a short case study of its application to coding qualitative data to answer a social research question, namely an

examination of the role of group maintenance behaviours in online groups. We discuss limitations of our approach and present a cost-benefit analysis of the use of this novel approach to data analysis before concluding with a discussion of future work.

Natural language processing

Natural Language Processing is a computational approach to text analysis. It "is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications" (Liddy, 2003). In the current paper, we discuss how NLP can be used to automate (fully or partially) the process of qualitative data analysis by identifying segments of text that provide evidence for concepts of theoretical interest (i.e., coding).

NLP tools and approaches can be applied at different levels of analysis. The levels of linguistic analysis range from the lowest, *Phonological*, to the highest, *Pragmatic*, as shown in Table 1. Successively higher levels of linguistic processing reflect larger units of analysis, as well as increasing linguistic complexity and difficulty in processing. The larger the unit of analysis becomes (i.e., from morpheme to word to sentence to paragraph to full document), the greater the lexical variety, sentence structure and potential subtlety in meaning. Discerning meaningful regularities on which to build rules for processing text becomes a more difficult and elusive process as one moves from the lowest to the highest levels. Additionally, higher levels presume reliance on the lower levels of language understanding, and the theories used to explain the data move more into the areas of cognitive psychology and artificial intelligence.

[Insert Table 1 about here]

Because of these differences, lower levels of language processing have been more thoroughly investigated and incorporated into systems, as they are easier to encode with more reliable results. For example, it is common for CAQDAS tools to support various kinds of automated searches for keywords (lexical level), but support at semantic or higher levels is less common. Language processing for content analysis similarly depends on an understanding of the lower levels of language, but its promise is situated in the higher levels of semantic, discourse and pragmatic levels of understanding.

To analyze language at the higher levels of linguistic analysis, we draw on sublanguage theory. The early research in Sublanguage Theory (Liddy, Jorgensen, Sibert, & Yu, 1991; Liddy, Jorgensen, Sibert, & Yu, 1993; Liddy, McVearry, Paik, Yu, & McKenna, 1993; Sager, 1970; Sager, Friedman, & Lyman, 1987) has shown that there are differences in the linguistic phenomena amongst various genres (e.g., news reports, manuals, interviews, email). These genres exhibit characteristic lexical, syntactic, semantic, discourse and pragmatic features used by generators of these genres. Formally, a sublanguage is defined as the particular language usage patterns, which develop within the written or spoken communications of a language community that uses this sublanguage to accomplish some common goal or to convey and discuss activities and events of common interest.

Once developed for a genre, a sublanguage grammar (at the syntax level) reflects the *information structure* of communication or texts in the domain, while the semantic classes of words used and the relationships between classes reflect the domain's *knowledge structure*. The result of this type of theory-driven and data-focused approach is a domain model that provides guidance for learning the particularized linguistic

constructs to understand the meaning of texts expressed in the sublanguage. Technology can then be developed to simulate this understanding (Liddy, Jorgensen, et al., 1993) and instantiated within a system to extract meaning at multiple levels of understanding.

Natural Language Processing uses a variety of techniques to extract meaning from context based on features of language use. Two general approaches are in use: statistical and symbolic approaches. The symbolic approach is knowledge-based, analyzing linguistic phenomena that occur within texts and reflecting syntactic, semantic and discourse information in human-developed rules and lexicons to extract meaning from text. On the other hand, corpus-based statistical methods apply mathematical techniques to develop approximate generalized models of linguistic phenomena based on actual examples, usually treating documents as independent "bags of words". We will present our experiences applying the symbolic approach. Compared to statistical techniques, symbolic approaches have an advantage of not requiring large data sets for training, which fit our situation. On the other hand, symbolic methods require considerable effort to develop rules and the rules often are not easily transportable to other domains, thus limiting applicability to specific domains (Liddy, 2003), limitations that we will return to in the discussion.

Example study

In this section, we present a case study of NLP for qualitative data analysis. The authors are collaborating on a study of the work practices of teams of free/libre open source software (FLOSS) developers. These teams are geographically and temporally distributed, rarely interact on a face-to-face basis, and coordinate their activities primarily through electronic channels (Raymond, 1998; Wayner, 2000). Large archives of these

interactions are available for analysis. In the following sections, we describe the stages of the study, starting with conceptual development and coding system development through manual coding before discussing the use of NLP for this coding task. In keeping with our focus on research methods, here we present only enough detail about the study for a reader to understand the method applied and the role of NLP, omitting specific discussion of the study results.

Conceptual development

In this study, we examined the role of Group Maintenance behaviours in the effectiveness of FLOSS teams. Group maintenance behaviour refers to the discretionary relationship-building behaviour among members that binds the group, maintains trust and promotes cooperation (Ridley, 1996). To understand and codify these behaviours, we drew on two theories describing pro-social, organizational behaviours: social presence (Garrison, Anderson, & Archer, 2000; Rourke, Anderson, Garrison, & Archer, 1999) and face work in computer-mediated communications (CMC) (Morand & Ocker, 2003). We discuss each in turn.

Social presence. Garrison, Anderson & Archer (2000) defined social presence to be "the ability of participants… to project their personal characteristics into the community, thereby presenting themselves to the other participants as 'real people'" (p. 89). Strategies that people in CMC use to increase the degree of their social presence include the use of emoticons, humour, vocatives (a direct reference to another person), phatics (speech used to share feelings rather than information), inclusive pronouns, complimenting, expressions of appreciation and agreement, and non-standard or

expressive punctuation and conspicuous capitalization as means to express emotion (Rourke, et al., 1999).

Face work. Referring to Goffman (1959), Morand (1996) explains that face is "the positive value individuals claim for the public self they present" (p. 545). Face is the result of two desires: independence of action (also known as negative face) and the need for approval and regard (also known as positive face) (Duthler, 2006; Meier, 1995). Negative face is exemplified by distancing behaviours to preserve self direction, freedom from imposed restrictions and a desire to be left alone, while positive face is exemplified by connectionist behaviours that seek respect, approval and a sense of belonging to the community (Duthler, 2006). However, whatever the public image one claims, face can be easily threatened or lost during interactions through face-threatening acts (FTAs). Thus, maintaining one's own face, as well as that of others, permeates social interaction (Holtgraves, 2005; Morand, 1996).

Politeness is a mitigation strategy that individuals use to moderate face threats in communicating with others (Morand, 1996). Politeness in CMC takes the form of linguistic acts that can be either positive tactics to invoke positive face or negative tactics to invoke negative face (Morand & Ocker, 2003). Examples of positive politeness tactics include use of colloquialisms or slang, inclusive pronouns, vocatives, agreement and sympathy. Examples of negative politeness include use of apologies, formal verbiage, hedges, indirect inquiries, subjunctives, honorifics, passive voice and rationales for FTAs (Morand, 1996; Morand & Ocker, 2003).

Based on these theories and their discussion in the literature, an initial coding scheme was created deductively to investigate Group Maintenance behaviours in the

FLOSS data. This coding scheme described the Group Maintenance indicators of interest, described their characteristics and included definitions and examples to guide coders.

Manual data analysis

For the study, two FLOSS projects were selected, both of which had a goal of developing an Instant Messaging (IM) client: Gaim and Fire. The two projects were selected to be similar in terms of their project goals, nature of tasks, and potential users, as to allow for comparison of project effectiveness. Overall, Gaim emerged as a more effective project, according to Crowston et al.'s multivariate measures (Crowston, Howison, & Annabi, 2006). Evidence of Gaim's success can also be seen in that the project is still active (now known as Pidgin), while Fire ceased active development in early 2007.

The data for the analysis was a sample of 1469 messages, a subset chosen randomly from the available data from the two developer discussion lists. These lists were chosen because they are the primary channel through which developers interact and as such are the main venue for group maintenance. A random sample was selected because the available coder time was not sufficient to code the entire archive, an example of the problem we seek to address with NLP.

Two PhD students using the Atlas.ti software package trained to code according to the initial codebook. An iterative process of coding, inspection, discussion and revision was carried out to inductively learn how the indicators evidenced themselves in the data. Training continued until the coders reached an inter-rater reliability over 0.80, a typical level of agreement expected for qualitative data analysis. However, several indicators, such as humour, were dropped from analysis, as they proved to be too difficult for the coders to reach consensus through subjective judgments. Table 2 outlines the final coding

scheme used to manually code the selected messages. After the coders achieved reliability, they independently coded the remaining messages.

[Insert Table 2 about here]

Automated group maintenance coding

In this section, we discuss how NLP was applied to perform the qualitative data analysis described in the previous sections. The automated content analysis coding was approached as an Information Extraction (IE) task, meaning that text showing evidence of group maintenance behaviour was to be identified and extracted from the text for further analysis, using symbolic rules developed by an analyst. However, in applying NLP, our goal was to develop a system that could support, rather than replace, a human coder. Therefore, we assumed that the output of the system would be reviewed and corrected by a human coder, rather than being used as is.

For this project, we used a text-processing engine developed at our University (anonymized for review). The system processes the input text through a series of stages, beginning with preprocessing, which takes the raw text and stores it in a uniform format for processing. We converted the raw messages to a format that would preserve the metadata elements, identify significant features of the data, such as signature lines or quoted messages and prepare the data for processing with our text processing engine, thus encoding the discourse structure for further use. We also extracted structured information, such as date, sender and subject.

The second stage is tokenization, which identifies the smallest complete units within a text, usually words, as well as sentence detection. Then each token is tagged

with a part of speech (morphological and lexical understanding). For example, for the sentence,

```
Alan Helfer mentioned these security updates back
in July
```

applying tokenization and part-of-speech tagging would result in the following string:

```
<sentence> Alan|NP Helfer|NP mention|VBD these|DT
security|NN update|NNS back|RB in|IN July|NP .|.
</sentence>
```

Note that the tense of 'mentioned' and the plurality of 'updates' is embedded in the part-of-speech tag (VBD-past tense verb; NNS-plural noun) and the word itself is converted to its lemmatized form. This string is next fed into bracketing stages, which identify numeric and temporal phrases, common noun phrases and proper noun phrases, reflecting lexical, syntactic, semantic and pragmatic understanding of the sentence, as in the following example:

```
<sentence> <proper noun> Alan|NP Helfer|NP
</proper noun> mention|VBD these|DT <noun phrase>
security|NN update|NNS </noun phrase> back|RB
in|IN <temporal> July|NP </temporal> .|.
</sentence>
```

The next stage of automatic processing interprets the phrases and assigns each a category (person, organization, date, etc.), resulting in a sentence marked up as follows:

```
<sentence> <proper noun, "person"> Alan|NP
Helfer|NP </proper noun> mention|VBD these|DT
<noun phrase, "unknown"> security|NN update|NNS
</noun phrase> back|RB in|IN <temporal, "month">
July|NP </temporal> .|. </sentence>
```

After the text has been marked up for these entities, in the final stage of analysis, hand-written information extraction rules are applied to extract a variety of kinds of information embedded in the text, such as metadata elements, relationships among entities, descriptors, or, in our application, evidence of a theoretical construct of interest.

Rule-writing effort

An NLP analyst developed information extraction rules for the group maintenance codes in Table 1. Because of time limitations, we developed NLP rules for only twelve of the fifteen manual codes: all except *Vocatives*, *Disclaimers/Self-depreciation* and *Stating Rationale for FTA*. (Preliminary work for these three codes suggests that the issues surrounding the automation of the coding would be similar to the issues for the other twelve.) The rule-writing process was iterative: rules were written to code the most abundant and obvious examples of the coded text and then progressively refined for coverage and accuracy.

Some rules, as for *Capitalization*, were primarily based on regular expressions to detect upper case. Other rules, as for *Apology*, focused on specific lexical items—'sorry', 'apologies'—or a lexicon of lexical items. But others required the use of the full range of NLP features such as part of speech, actual word, semantic class and syntax, as seen in the example rule in Figure 1, a rule for finding *Agreement*.

[Insert Figure 1 about here]

As shown in Figure 1, the rules used for this project have a two-part structure: a premise and an action. The premise defines the matching criteria for the rule. The action defines the resulting output when a text string matches the premise of the rule. In the example rule, the `<S>` in the premise indicates that the matching text must be situated at the beginning of a sentence. The elements `seem`, `more`, and either `than` or `then` are specific lexical items (words) that must appear in the sentence. The element `do|VBZ` combines a lemmatized lexical item, `do`, with tense information from the part of speech `VBZ` (present tense). In combination with the semantic class represented by `$it`, which

can be the word ‘it’, ‘this’, ‘that’, or ‘these’, the verb which actually is represented by `do|VBZ` can be ‘do’ (“these do”), or ‘does’ (“it does”), an example of the use of syntax. The element `$anywd|$anypos` represents any token in a candidate text string (any word tagged with any part of speech). When a candidate text string matches this rule, the resulting action tags the text with the code `agreement`. For example, the rule would match and code as *Agreement* the sentence:

```
It does seem to be more trouble then i thout at
first.
```

The ruleset included both positive rules, to code sections of text, and negative rules, to cancel out the coding of text. For example, if a rule finds “Sorry that I caused a problem here,“ it would be coded as *Apology*. However the presence of “not” in “I’m not sorry that I caused a problem here,” indicates quite the opposite, and thus another rule is added that is intended to rectify the coding when “not” appears.

Rule writing was interspersed with testing to assess performance on the training data during the development process. The results of the manual coding of the 1469 messages from Fire and Gaim were used for this effort as the so-called “Gold Standard” data (GS), meaning that these data are assumed to be correct and so can be used to check the performance of the NLP system. A portion of the coded data (155 messages, or about 10%) was set aside for final testing of the completed ruleset. The remainder was used to assess the performance of the ruleset as it was being built.

Results

To test the performance of the automated process, the developed ruleset was run on the 10% of the GS data reserved for testing. Each message in the test set was inspected to see

which instances of Group Maintenance were correctly coded, which were missed, how many additional instances were erroneously coded by the automated process and to understand the nature of the errors.

Two standard information extraction metrics were used to evaluate the automated system, Recall and Precision. Recall measures the proportion of the codes in the GS data that was identified and extracted by the system (i.e., coverage). Precision measures the proportion of the automatically extracted data that was coded correctly, as compared to the GS data (i.e., accuracy). It is usually difficult to have high performance on both measures: in general, the more accurate the results, the smaller the coverage of the target data, and vice versa. To completely automate coding, it would be necessary to achieve good performance on both measures. However, given our goal of developing a support system, in building the rules, we optimized the automated system for Recall, with a goal of 80%. We took this approach because we felt that it would be easier for someone reviewing the system output to remove incorrectly coded data (included due to low Precision) than to search the message logs to find evidence that had not been coded at all (the result of low Recall).

Table 3 shows the system performance for the 12 Group Maintenance codes. The training and testing columns compare the performance of the system on the training and testing data. The training performance is generally higher because the rules were developed in reference to these data. Examining the codes in more detail, we see that Recall is highest for *Emoticon*, *Inclusive Pronouns*, and *Formality*, reflecting the regularity of the realization of these constructs in the text. It is lower for codes such as *Slang* or *Appreciation* that show higher variability. The Precision of the results is lower,

reflecting our deliberate decision to favour Recall over Precision. Nevertheless, Precision is quite good for a number of codes, such as *Emoticon* or *Salutations*, and with the exception of *Capitalization* and *Punctuation*, all are at usable levels. We discuss below the problems that lead to the unexpectedly low level of Precision for these codes.

[Insert Table 3 & 4 about here]

Another way to show the performance of the system is with a table comparing the GS and system decisions, as shown in Table 4 for the test data for one construct, *Hedges*. The first row of the table shows that the GS test data (the reserved 10% of messages) included 244 instances of *Hedges*, of which the system correctly coded 181 and missed 63, giving a Recall of 74%. The first column of the table shows that the system coded a total of 262 segments of text as being *Hedges*, of which 181 matched the GS and 81 did not, giving a Precision of 69%.

Not shown is the final cell, i.e., the number of segments of text in the corpus that neither the human coders nor the system coded as being a hedge. Because of the use of thematic units as the unit of coding (a limitation discussed below), it is not possible to give a precise figure for this cell. However, the test data included 155 messages, suggesting the number of units was in the thousands. As a result, even with the current level of performance, the system could reduce the amount of text to be examined by a human coder by an order of magnitude or more (in this case, from thousands of units to 262), potentially increasing the speed of coding by a comparable amount. The performance impact would be greater for codes occurring less frequently (for which the narrowing would be greater), but lower for codes for which the system exhibits lower Precision.

Discussion and limitations

Overall, the use of NLP to code qualitative data seemed quite promising, as the rules that were developed showed good performance on a number of codes. In analyzing the results of our work, we identified several issues that impacted performance. In this section we discuss these issues, before concluding with a discussion of the cost and benefits of this approach.

Insufficient preprocessing. Preparing the data for processing is an important and often time-consuming part of NLP. For this effort, messages were preprocessed in various ways, e.g., to section off headers, forwarded messages and signature blocks, because human analysts generally excluded these sections from manual coding. However, messages also include lines of programming language, error logs, source file comparisons (diffs) and messages copied in from other sources that are difficult to reliably identify and exclude from processing. Unfortunately, including this content particularly affected Precision for *Punctuation*, *Capitalization* and *Emoticon*, as it frequently included strings of punctuation, capitalized words or characters that resemble emoticons.

Unit of coding. In manually coding the data, the researchers chose the thematic unit as the unit of coding, a common choice in qualitative data analysis. A word, a phrase, a sentence or an entire paragraph might be marked as capturing the group maintenance evidence. Unfortunately, with this variability in scope it is difficult to exactly match the boundaries of text to be captured using NLP rules. For the results reported above, any overlap between the text coded manually and that coded automatically was considered to be a match, as is often done when comparing human coders. To facilitate future

comparisons of human and automatic coding though it would be better to pick a more easily defined unit of coding, such as the sentence or even an entire message.

Adequate training examples. Some codes were so sparse in the data as to provide unreliable training data. In general, to apply NLP requires hundreds of examples of correctly coded text or even more to apply statistical techniques. This shortcoming in our data is evident in performance differences; in general, the difference between training and testing is greater for the codes that had fewer than 100 training examples, with the exception of *Formal Verbiage*, which performed surprisingly well.

Manual coding error. In order to assess the benefit of automatic coding, performance is compared against the human results. However, any errors in manual coding are propagated in the automatic processing, as rules are built based on possibly erroneous data. We focused on gaining complete agreement between human coders, but on further review found that the GS data they created still contained coding errors, for several reasons. First, it took some time for inter-coder reliability to stabilize. The GS data used in our evaluation represents coding prior to and including the stabilization period, meaning that not all of the coding is of the same quality. Second, coding is a tedious, fatiguing process, so errors both of commission and omission are likely to occur in coding – perfect reliability is simply not achievable with human coders in reasonable time. As well, the human coders were somewhat disadvantaged in their assessment of some codes, for example *Slang*, because they were not from the community of developers, and therefore not adequately familiar with some of the terms or community-specific meanings.

We attempted to quantify the effect of manual coding error on our results by re-judging NLP false hits for correctness according to the codebook (vs. against the GS data). Our expert NLP analyst judged that with this correction, Precision for all codes would have risen. The most dramatic increases would be seen for codes that have the fewest examples in the GS data, for which a few errors makes a noticeable difference in the result. However, for *Inclusive Pronouns*, there was a difference of 31% between the achieved Precision and the analyst's estimate correcting for manual coding errors, reflecting the ease with which automated techniques can find such regular forms and the difficulties human coders have. This result shows that the automatic process may in some case be even more reliable for finding instances than human coders.

Language and Meaning. The thorniest problems for automation of content analysis deal with the incredible richness of language. This richness varies by code of interest. For example, very few rules were needed for good performance for the codes *Formal Verbiage*, *Apology* and *Agreement*, which exhibit regularities in their expression. On the other hand, *Hedges* and *Vocatives* (which was explored but not formally evaluated) proved more difficult for a variety of syntactic and semantic reasons:

- *Context.* Content analysis can be highly dependent upon context. Unfortunately, the processing engine we used currently does not have a way to consider text outside of a sentence boundary, except for co-reference purposes. Thus, context outside of a single sentence is not available for consideration. This limitation prevents full exploitation of discourse structure and context of an entire message.
- *Syntactic variety and synonymy*. Language holds an infinite variety of ways to structure and convey meaning using differing syntactic structures, synonymous terms,

and embellishments (adverbial and adjectival clauses). While the sublanguage of software engineering does not reflect the full variety of language, good automation can require much more training data than was available to capture this richness.

- *Multiple aspects of meaning*. Various clue words were helpful in identifying *Hedges*, for example, 'probably'. Others, such as 'seem', 'would', 'of course', were more problematic, as sometimes they were indicators and at other times, not. Context, both within the sentence and beyond the sentence, can subtly affect meaning, which under many circumstances can be difficult for an automated system to capture. A solution for a particularly difficult problem in correctly identifying *Vocatives* has not yet been explored, that is, identifying the differentiating features that indicate when 'you' refers to a specific individual and when 'you' refers to 'a person', as in the sentence, "When you open up the file, you will see two items". Without the ability to interpret context surrounding this sentence, or an associated response to the message, it is difficult to code *Vocatives* with high recall and reasonable precision.
- *Implicit meaning*. NLP systems are only now just beginning to explore the extraction of meaning that is implicit in text (Snyder, D'Eredita, Yilmazel, & Liddy, 2009). This is a very challenging area of research, since even humans have difficulty in this space, as evident in the difficulty our analysts encountered with the *Humour* code.

Cost/benefit

Finally, an important component to consider in the evaluation of NLP-enabled content analysis is the potential cost-benefit to a research project. While NLP can potentially automate parts of the coding process, additional effort is required to develop and validate a ruleset. In the approach we took, NLP coding was built upon a manually crafted

codebook (as is required for any such qualitative study). However, the NLP ruleset required additional development for the rules and testing time to determine performance, both requiring the time of a trained NLP analyst. For a large-scale analysis system handling very large volumes of textual data, particularly discourses spanning long periods, some development time might also be needed to adjust for changes in data format, new discoveries and evolution in both the data content and the analytic thinking.

For the case reported here, a software engineer and a linguistic analyst each committed approximately 5 weeks of effort (FTE) over the course of a year for data preprocessing, rule-writing, development and testing. In comparison, two human coders worked half time on the project for the same period and were able to code only two projects, though some of this effort went to refining the codebook that was used as a basis for both the manual and NLP coding. Once the coders were trained and codebook stabilized, manually coding 700 messages on all 15 codes took approximately 100 hours of effort for one coder.

In light of the additional work needed, an NLP-supported approach would not make sense for small (e.g., a thousand or so messages) unique data sets that can be handled by training content coders within a relatively short time span. Furthermore, we note that the NLP approach is only appropriate for theoretical concepts that find a regular expression in text. NLP would be unlikely to work for coding that draws heavily on subjective interpretation and context. However, for suitable codes, and after development resources have been invested, benefits can be realized for large-scale studies by processing and analyzing large volumes of data with reduced human coder effort. Specifically, the investment of time in writing rules should enable order of magnitude

reductions in the effort needed to code additional text, potentially allowing the analysis of hundreds of groups with hundreds of thousands of messages. Indeed, such automation is arguably the only way to reliably handle such large amount of text that would otherwise require hundreds of person-years of coder effort.

Conclusions

In this paper, we explored the possibilities and limitations of applying NLP techniques to the task of qualitative data analysis, specifically content analysis of communication artefacts from online groups. Our future work for this project has three aspects.

First, we are building a system around the NLP text processor that will allow a user to quickly check the system-applied codes. Second, we will use the system to support our study of Group Maintenance behaviour in FLOSS teams. We have developed some initial hypotheses based on patterns we saw in the human coding, but the current volume of manually coded data allows for only a comparative case study of the two teams. By applying NLP, we hope to be able to analyze hundreds of teams, thus providing an evidentiary basis for stronger findings.

Finally, a key bottleneck in the current study is the reliance on a trained NLP analyst to develop and tune the rulesets. To avoid this bottleneck, we plan to explore the use of machine learning (ML) techniques to build rules. The most significant limitation of an ML approach though is that it requires even more GS data as input: a rule of thumb is 500 examples of each code, which we have for only two of the codes in our example case.

In summary, our small case study demonstrates the promise of NLP support for this particular style of qualitative data analysis. The performance of the rulesets we

developed suggests that this approach has considerable promise for coding at least some kinds of concepts. This approach seems most promising for projects with content analysis codes that are readily evident in large data sets, projects that analyze multiple data sets over time and projects where manual coding is simply not feasible due to the volume of the data, an increasingly common challenge as social researchers study online groups.

Tables and Figures

Table 1. Levels of linguistic analysis.

Level	Definition and examples
Phonological	Phonological analysis pertains to the auditory features of language: sound, pitch, and inflection.
Morphological	Morphological analysis occurs at the smallest level of linguistic meaning, the morpheme. For example, '-ed' added to the end of a word can signify an action that occurred in the past, and 'un-' added before a word, such as 'tested', radically alters a word's meaning. Prefixes and suffixes are the most familiar morphemes.
Lexical	Lexical analysis occurs at the word level. Part of speech is a feature of lexical analysis affecting meaning. Consider, for example, the difference in meaning between 'book' as a noun (read a book), and 'book' as a verb (to book a flight).
Syntactic	Syntactic analysis pertains to the meaning that derives from the sequence of words in a phrase or sentence. For example, consider the different meanings of 'the man hit the ball' and 'the ball hit the man'.
Semantic	Semantic analysis deals with the definitional meanings of words within context, whether 'bank', for example, refers to a river bank, or to a financial institution. Semantic analysis can deal with fine gradations of meaning depending on context.
Discourse	Discourse analysis reveals meaning based on a larger unit than a sentence, where the meaning of a particular sentence is affected by the text that precedes it, or its placement within a document. Discourse analysis has led to the identification of genres of documents, where information can be predictably found through document structure (introduction, byline, research findings, etc.).
Pragmatic	Pragmatic analysis involves the incorporation of world knowledge to determine meaning, that is, connotations based on experience, and shared understandings. For example, we understand much more about "Third World Countries" than the component words can tell us.

Table 2. Group maintenance coding scheme showing conceptual categories, indicators and definitions.

Category	***Indicator***	***Definition***
Emotional Expressions	*Emoticons*	Emphasis using emoticons
	Capitalization	Emphasis using capitalization
	Punctuation	Emphasis using punctuation
Positive Politeness	*Colloquialisms/Slang*	Use of colloquialisms or slang beyond group-specific jargon
	Vocatives	Referring to or addressing a specific participant
	Inclusive pronouns	Incorporating writer and recipient(s)
	Salutations/Closings	Personal greetings and closures
	Complimenting	Complimenting others or message content
	Expressing agreement	Showing agreement
	Apologies	Apologizing for one's mistakes
	Encouraging participation	Encouraging members of the group to participate
	Expressing appreciation	Showing appreciation for another person's actions
Negative Politeness	*Disclaimers/Self-depreciation*	Disclaiming prior to a face-threatening act (FTA); self-depreciation to distance
	Rational for FTA	Stating an FTA as a general rule to minimize impact
	Hedges/Hesitation	Tactics to diminish force of act; hesitation in disagreement
	Formal verbiage	Using formal wording choices

Figure 1. Example NLP rule showing premise and action.

Premise	`<S> ($it\|$anypos) (do|VBZ) (seem|$anypos) ($anywd|$anypos)* (more\|$anypos) (th[ae]n|$anypos) ($i|$anypos) ($anywd|$anypos)* </S>`
Action	`==> generic ($&, 'entity', 'gm', 'agreement', sf($1,$2,$3,$4,$5,$6,$7,$8));`

Table 3. System performance.

CODE	RECALL			PRECISION			GS # of INSTANCES	
	Training	Testing		Training	Testing		Training	Testing
Apologies	89%	67%		81%	67%		19	3
Formality	90%	89%		55%	53%		29	9
Complimenting	88%	67%		70%	40%		40	6
Agreement	87%	80%		61%	60%		71	15
Capitalization	96%	60%		27%	19%		73	10
Appreciation	90%	64%		91%	45%		90	14
Emoticon	91%	91%		30%	81%		122	32
Salutations	77%	86%		79%	86%		159	28
Punctuation	79%	71%		16%	22%		257	34
Slang	89%	67%		71%	69%		274	81
Inclusive Pronouns	98%	98%		90%	58%		478	55
Hedges	80%	74%		63%	69%		1136	244

Note: Recall is the percentage of human applied codes found the system; precision is the percentage of codes found by the system that match the human codes. Testing results are on the 10% of data held back for final testing.

Table 4. System decisions compared to Gold Standard decisions for test data for *Hedges.*

		System		
		Coded	Not coded	Total
Gold standard	Coded	181	63	244
	Not coded	81		
	Total	262		

References

Barry, C. A. (1998). Choosing qualitative data analysis software: Atlas/ti and Nudist compared. *Sociological Research Online, 3*(3).

Benbunan-Fich, R., Hiltz, S. R., & Turoff, M. (2003). A comparative content analysis of face-to-face vs. asynchronous group decision making. *Decision Support Systems, 34*(4), 457–469.

Crowston, K., Howison, J., & Annabi, H. (2006). Information systems success in Free and Open Source Software development: Theory and measures. *Software Process—Improvement and Practice, 11*(2), 123–148.

Duthler, K. W. (2006). The politeness of requests made via email and voicemail: Support for the hyperpersonal model. *Journal of Computer-Mediated Communication, 11*(2), 500–521.

Garrison, R., Anderson, T., & Archer, W. (2000). Critical thinking in a text-based environment: Computer conferencing in higher education. *The Internet and Higher Education, 2*(2-3), 87–105.

Goffman, E. (1959). *Presentation of Self in Everyday Life*. Garden City, NY: Doubleday.

Holtgraves, T. (2005). Social psychology, cognitive psychology and linguistic politeness. *Journal of Politeness Research. Language, Behaviour, Culture, 1*(1), 73–93.

Lee, R. M., & Esterhuizen, L. (2000). Computer software and qualitative analysis: Trends, issues and resources. *International Journal of Social Research Methodology, 3*(3), 231–243.

Liddy, E. D. (2003). Natural Language Processing *Encyclopedia of Library and Information Science* (2nd ed.). New York: Marcel Decker.

Liddy, E. D., Jorgensen, C. L., Sibert, E. E., & Yu, E. S. (1991). *Sublanguage grammar in natural language processing.* Paper presented at the RIAO '91 Conference, Barcelona.

Liddy, E. D., Jorgensen, C. L., Sibert, E. E., & Yu, E. S. (1993). A sublanguage approach to Natural Language Processing for an expert system. *Information Processing & Management, 29*(5), 633–645.

Liddy, E. D., McVearry, K., Paik, W., Yu, E. S., & McKenna, M. (1993). *Development, implementation and testing of a discourse model for newspaper texts.* Paper presented at the ARPA Workshop on Human Language Technology, Princeton, NJ.

Meier, A. J. (1995). Passages of politeness. *Journal of Pragmatics, 24*(4), 381–392.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook* (2nd ed.). Thousand Oaks: Sage Publications.

Morand, D. A. (1996). Dominance, deference, and egalitarianism in organizational interaction: A sociolinguistic analysis of power and politeness. *Organization Science, 7*(5), 544–556.

Morand, D. A., & Ocker, R. J. (2003). *Politeness theory and computer-mediated communication: A sociolinguistic approach to analyzing relational messages.* Paper presented at the 36th Hawai'i International Conference on System Sciences (HICSS-36).

Myers, M. D. (1997). Qualitative research in information systems. *MIS Quarterly, 21*(2), 241–242. MISQ Discovery, archival version, June 1997, http://www.misq.org/discovery/MISQD_isworld/. MISQ Discovery, updated version, last modified: November 2008, accessed May 2010.

Raymond, E. S. (1998). Homesteading the noosphere. *First Monday, 3*(10).

Richards, L. (2002). Qualitative computing: A methods revolution? *International Journal of Social Research Methodology, 5*(3), 263–276.

Ridley, M. (1996). *The Origins of Virtue: Human Instincts and the Evolution of Cooperation*. New York: Viking.

Rourke, L., Anderson, T., Garrison, D. R., & Archer, W. (1999). Assessing social presence in asynchronous, text-based computer conferencing. *Journal of Distance Education, 14*(2), 51–70.

Sager, N. (1970). The sublanguage method in string grammars. In R. W. Ewton & J. Ornstein (Eds.), *Studies in Language and Linguistics*. El Paso, TX: University of Texas at El Paso.

Sager, N., Friedman, C., & Lyman, M. S. (1987). *Medical language processing: Computer management of narrative data*. Reading, Mass: Addison-Wesley.

Snyder, J., D'Eredita, M. A., Yilmazel, O., & Liddy, E. D. (2009). *Towards a cognitive approach for the automated detection of connotative meaning*. Paper presented at the International Conference on Computational Semantics.

Wayner, P. (2000). *Free For All*. New York: HarperCollins.