International Journal of Social Research Methodology

ISSN: 1364-5579 (Print) 1464-5300 (Online) Journal homepage: https://www.tandfonline.com/loi/tsrm20

Using natural language processing technology for qualitative data analysis

Kevin Crowston , Eileen E. Allen & Robert Heckman

To cite this article: Kevin Crowston , Eileen E. Allen & Robert Heckman (2012) Using natural language processing technology for qualitative data analysis, International Journal of Social Research Methodology, 15:6, 523-543, DOI: 10.1080/13645579.2011.625764

To link to this article: https://doi.org/10.1080/13645579.2011.625764

Published online: 25 Oct 2011.

International Journal of Social Research Methodology
Vol. 15, No. 6, November 2012, 523–543

Using natural language processing technology for qualitative data analysis

Kevin Crowston*, Eileen E. Allen and Robert Heckman

School of Information Studies, Syracuse University, Hinds Hall, Syracuse, NY 13244, USA

(*Received 27 May 2010; final version received 19 September 2011*)

Social researchers often apply qualitative research methods to study groups and their communications artifacts. 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.

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

Introduction

In this paper, we discuss how computer systems that process text in human languages (i.e. natural language processing [NLP]) might partially automate a particular qualitative data analysis approach, namely content analysis, by semi-automatically extracting theoretical evidence from texts. Social researchers often employ qualitative methods to understand the social world by examining texts produced by individuals or groups. For example, researchers might examine transcripts of a group's discussions to understand how it solved some task and to understand the impact on performance of various problem-solving approaches adopted by different groups (e.g. Benbunan-Fich, Hiltz, & Turoff, 2003). However, because such data are textual, they require considerable manual effort to analyze. This work is tedious and difficult for humans to do reliably at scale. 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. While there is a wide diversity of techniques and

*Corresponding author. Email: crowston@gmail.com

ISSN 1364-5579 print/ISSN 1464-5300 online

http://dx.doi.org/10.1080/13645579.2011.625764
http://www.tandfonline.com

approaches, these technologies are often referred to collectively as NLP. In this paper, we introduce methodological techniques from 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 artifacts. 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 at a greater depth. The main contribution of this paper is to introduce 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. It is sometimes assumed that the goal of qualitative analysis is to uncover latent or hidden meanings in a text, e.g. to understand individuals' concepts of their social worlds from their communications, that is, that qualitative work is always interpretivist. But qualitative research can, in fact, adopt any research perspective: positivist, interpretivist or critical (Myers, 1997). In our studies, we assume that social processes of the groups studied are accurately reflected in the texts that they produce as part of their work together. Rather than hidden meanings, we look for explicit evidence of particular behavioral patterns of the participants. Our approach is thus essentially positivist, despite its reliance on qualitative data. We note though that the assumption that the behaviors are more-or-less explicit in the text is a limitation of the approach we propose, a topic we return to in the discussion.

In this paper, we explore how NLP techniques (and sublanguage theory in particular) can be applied to support a particular task in positivist qualitative research, namely coding for content analysis. 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 (or tagged) 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. For example, the coded text could be used to examine hypotheses such as the relationship between group member participation in decision-making and the overall effectiveness of a team. In this paper, we focus specifically on coding and do not address the use of the coded data, as critical as that is to the overall research process (Richards, 2002).

A key concern in coding is reliability, as measured by the degree of inter-rater agreement (also known in some circles as inter-coder reliability), that is, whether different human 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, which includes definitions of codes, and best examples of when to use them. However, a great deal of tacit knowledge is often used in coding, meaning that coders need to be trained extensively 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 labor 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. Furthermore, the tedium of the work makes it difficult for human coders to do reliably. Being able to analyze larger data sets is necessary to examine and compare multiple groups or to identify smaller effects within a group, but research teams often face limitations in the scope of analysis based on the available work force. It is this problem of reliable coding at 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 behaviors 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

NLP is a computational approach to text analysis (Jurafsky & Martin, 2009). 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). NLP is a large area of research and application, with numerous research problems and approaches. Example NLP tasks include automatic text summarization, machine translation, information search, and question answering. In the current paper, we discuss how NLP techniques 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) (Evans (1996) gives an overview of other computer applications). We focus our presentation on the particular NLP approaches we adopted and note that most of the techniques were pioneered by other researchers in other settings. Interested readers seeking further background will find numerous textbooks introducing NLP in more depth (e.g. Jurafsky & Martin, 2009; Manning & Schütze, 1999).

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 (i.e. a piece of a word, such as a prefix or suffix) to word to sentence to paragraph to full document – the greater the variety 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 and the theories used to explain the data move more into the areas of cognitive psychology and artificial intelligence. Additionally, higher levels rely on the lower levels of language understanding.

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, qualitative researchers often use Computer-Assisted Qualitative Data Analysis Software (CAQDAS) tools to support

Table 1. Levels of linguistic analysis.

Level	Definition and examples
Phonological	Auditory features of language: sound, pitch, and inflection
Morphological	Smallest level of linguistic meaning, the morpheme. Prefixes and suffixes are the most familiar morphemes. 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
Lexical	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	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	Definitional meanings of words within context, whether the noun '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	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	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 alone can tell us

analysis (e.g. Atlas.ti, Hyper-research or Nudist, Barry, 1998; Lee & Esterhuizen, 2000). CAQDAS tools manage the traditional processes of manual coding and support retrieval of coded segments (Richards, 2002). The most advanced offer capabilities for some automatic coding of at the text; for example, it is common for CAQDAS tools to support various kinds of automated searches for keywords (lexical level) or regular expressions but not support for semantic or higher levels of language. While computer support does provide considerable benefits, analyzing significant volumes of text still requires considerable manual effort from researchers.

Other analysis programs provide greater automation. For example, the Linguistic Inquiry and Word Count program (http://www.liwc.net/) works at the lexical level, grouping words from a text into 68 semantic classes, e.g. positive or negative emotions, self-reference or religion (numerous other programs provide similar functions[1]). However, the automation comes at a cost, as the semantic classes may or may not fit the theory being investigated.

To analyze language at the higher levels of linguistic analysis, we draw in our work on sublanguage theory. A sublanguage is the particular language usage patterns that develop within the written or spoken communications of a language community, a community that uses this sublanguage to accomplish some common goal or to convey and discuss activities and events of common interest. Research in Sublanguage Theory (e.g. 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 that are used by creators of these genres. A sublanguage grammar (i.e. a set of rules at the syntax level) for a genre reflects the information structure of communication or texts in the domain, while the semantic classes of words used and the relationships between classes (i.e. a set of words in a specialized lexicon) reflect the domain's knowledge structure. The combination is a domain model that provides guidance for learning the particularized linguistic constructs used in the particular domain and thus for understanding the meaning of texts expressed in the sublanguage.

For example, take the problem of mapping the content of an educational learning standard – i.e. a statement of something a student should learn in a particular grade in school, such as being able to 'identify basic earth materials' by 4th grade – to another standard, e.g. from a different jurisdiction (Yilmazel et al., 2007). To process a learning standard requires understanding the genre (e.g. what kinds of sections a standard typically includes and what information is conveyed in each section) as well as the specialized vocabulary (e.g. what terms are synonyms or related terms). An NLP system 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. For example, Yilmazel et al. (2007) report on a system to process US national and state educational learning standards to show how the standards are related, with the goal of mapping learning materials (e.g. lesson plans) to multiple standards automatically (Devaul, Diekema, & Ostwald, 2011).

NLP systems use a variety of techniques to extract meaning based on features of language use. Two general approaches are in use: statistical and symbolic. The symbolic approach is knowledge-based, analyzing linguistic phenomena that occur within texts by reflecting syntactic, semantic and discourse information in human-developed rules and lexicons that can be used to extract meaning from text. The example of processing learning standards discussed above is an example of this approach (Yilmazel et al., 2007). On the other hand, corpus-based statistical methods apply mathematical techniques to build models of linguistic phenomena from actual examples (Manning & Schütze, 1999). For example, machine translation can be performed statistically by analyzing large numbers of bilingual documents to observe how particular words or phrases are typically translated in different contexts. Such an approach can be quite successful given a sufficiently large set of training examples and indeed has become the dominant mode of analysis for large corpora.

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 fits our situation. On the other hand, symbolic methods require considerable effort to develop rules. Furthermore, rules are often not easily transportable to other domains, thus limiting applicability to specific domains (Liddy, 2003). We will return to these limitations in the discussion section.

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 subsections, 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 behaviors in the effectiveness of FLOSS teams. Group maintenance behavior refers to the discretionary relationship-building behavior among members that binds the group, maintains trust, and promotes cooperation (Ridley, 1996). To understand and codify these behaviors, we drew on two theories describing pro-social, organizational behaviors: 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 et al. (2000, p. 89) 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".' Social presence has been shown to be a strong predictor of satisfaction with participation in CMC-supported groups (Gunawardena & Zittle, 1997). Strategies that people in CMC use to increase the degree of their social presence include the use of emoticons, humor, 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) and 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 behaviors to preserve self direction, freedom from imposed restrictions and a desire to be left alone, while positive face is exemplified by connectionist behaviors 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 (Brown & Levinson, 1987; 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 behaviors in the FLOSS data. This coding scheme described the 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 client: Gaim and Fire. The two projects were selected to be similar in terms of their project goals, nature of tasks, and potential users, and to allow for comparison of project effectiveness. Overall, Gaim emerged as a more effective project according to Crowston, Howison, and Annabi's (2006) multivariate measures. Evidence of Gaim's success can also be seen in that the project is still active (though now known as Pidgin), while Fire ceased active development in early 2007.

The data for the analysis was a subset of the available messages from the two project developer discussion lists. These lists were chosen because they are the primary channels through which developers interact and as such are the main venue for group maintenance. The sample was drawn by selecting the messages leading up to 60 decisions made in each of the two projects (a total of 120 decisions), with 20 decisions each from the beginning, middle and most recent period of the project lifespan. A subset of messages was analyzed because the available coder time was not sufficient to code the entire archive, an example of the problem we seek to address with NLP. The collection included 84,870 words in 797 messages, an average length of 106.5 words/messages.

Two PhD students using the Atlas.ti CAQDAS software package trained to code according to the constructs defined in Table 2. The overall process followed is shown in Figure 1. An iterative process of coding, inspection, discussion, and revision was carried out to inductively learn how the indicators evidenced themselves in the data. To measure inter-rater reliability, we used Holsti's coefficient of reliability, $2m/(n_1+n_2)$, where m represents the total number of coding decisions that the two coders agree on, n_1, the number of decisions made by coder 1, and n_2, the number made by coder 2. Rourke & Anderson, 2004 explain that this measure is 'the simplest and most common method of reporting inter-rater reliability.' This measure does not correct for chance agreement (unlike other measure such as kappa), but given the nature of the task (identifying small segments of text in a large corpus), the probability of chance agreement is low, and the simple measure seemed preferable.

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

Category	Indicator	Definition	Examples (verbatim from data)
Emotional expressions	Emoticons	Emphasis using emoticons	:) ;-)
	Capitalization	Emphasis using capitalization	'EVERYONE ON THE LIST' 'THINK'
	Punctuation	Emphasis using punctuation	'!!!' Underline'!'
Positive politeness	Colloquialisms/ slang	Use of colloquialisms or slang beyond group-specific jargon	'Saturdayish' 'BTW'
	Vocatives	Referring to or addressing a specific participant	'As sean said' 'Martin'
	Inclusive pronouns	Incorporating writer and recipient(s)	'we,' 'us,' 'let's,' 'our'
	Salutations/ closings	Personal greetings and closures	'Hi,' 'regards,'
	Complimenting	Complimenting others or message content	'The temporary message is a good idea' 'You guys have done an awesome job'
	Expressing agreement	Showing agreement	'Agreed' 'I suppose,' 'Correct'
	Apologies	Apologizing for one's mistakes	'Sorry again if I stepped on any toes' 'Sorry if I touched a nerve there Sean' 'Sorry about that …' 'I am sorry' 'Sorry for the inconveniences'
	Encouraging participation	Encouraging members of the group to participate	'Any comments welcome'
	Expressing appreciation	Showing appreciation for another person's actions	'Thanks for the help' 'Well thanks a lot for you hard work!'
Negative politeness	Disclaimers/ self-depreciation	Disclaiming prior to a FTA; self-depreciation to distance	'dumb fire question#1: which MSNService. nib "file" is the real one?' 'Sorry if I'm terribly ignorant somehow … I'm just getting into this stuff'
	Rational for FTA	Stating an FTA as a general rule to minimize impact	'In general we want to avoid forking the MSN library with our own changes so any changes there need to be sent onto Meredydd'
	Hedges/ hesitation	Tactics to diminish force of act; hesitation in disagreement	'um…' 'I'm not sure what the problem is …' 'it would be nice to at least …'
	Formal verbiage	Using formal wording choices	'please send the file to …'; 'please'

Training continued until the coders reached an inter-rater reliability over 0.80, a typical level of agreement expected for manual qualitative data analysis. The training process required two rounds of double coding on a total of approximately 400 messages. However, several indicators, such as *Humour*, had to be dropped from analysis, as they proved to be too difficult for the coders to reach consensus through subjective judgments (an unfortunate and not uncommon problem in qualitative data analysis). This code proved to be particularly difficult for our coders as they often did not understand jokes about programming. After the coders achieved reliability on the remaining codes, they independently coded the remaining messages. Table 2

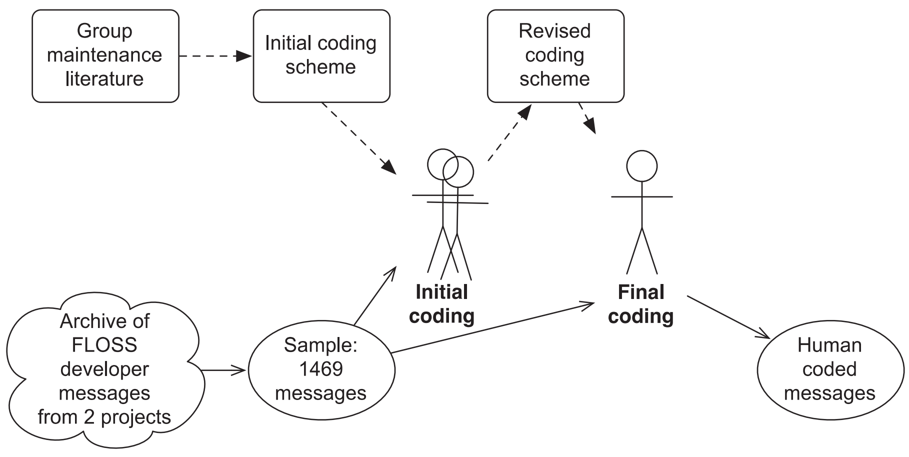

Figure 1. Manual data coding process.

presents the final coding scheme used to manually code the selected messages. In all, the coders coded 3011 segments of text (for the codes used in the study as explained below), not counting codes for metadata such as message sender or list.

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 overall process of content analysis using the system is shown in Figure 2. The automated content analysis coding

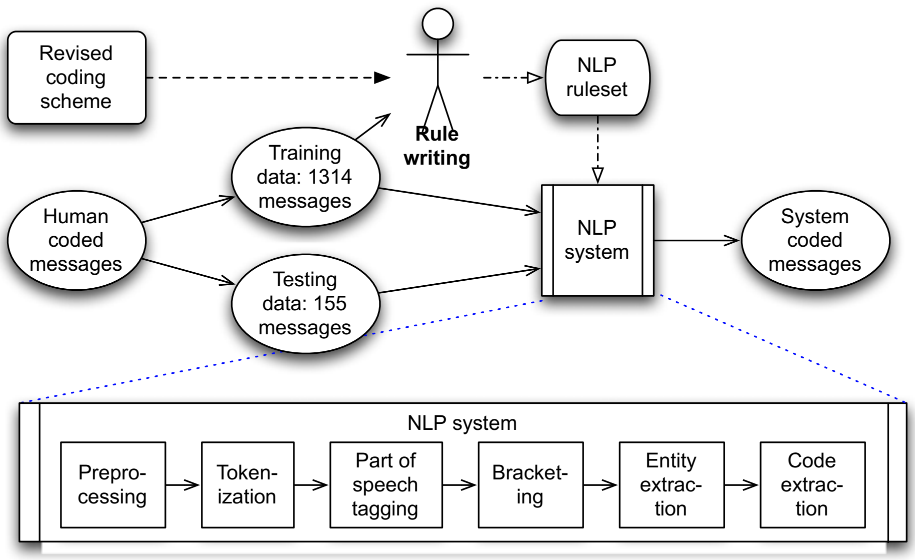

Figure 2. NLP data coding process, showing stages of NLP processing.

was approached as an information extraction (IE) task, meaning that segments showing evidence of group maintenance behaviors were 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 TextTagger, a text-processing system developed at the Syracuse University Center for NLP (Ingersoll, Yilmazel, & Liddy, 2007). This system provides a number of standard text processing functions that we could use for our analysis, including preprocessing of text, tokenization, part-of-speech tagging, and entity extraction (these functions are explained below). As well, the system provides a platform for building customized processing, specifically, a custom-developed set of rules for extracting the theoretical constructs of interest (i.e. those shown in Table 2). Our system is quite similar in function to other NLP systems, such as the open-source package General Architecture for Text Engineering (GATE) (http://gate.ac.uk/). We used TextTagger for this project because we had experience using it in previous studies and understood its limitations, but the same results could have been achieved using tools such as GATE, making the process replicable by other researchers. Indeed, Robinson and Vlas (2011) used GATE to perform an analysis comparable to the one described here (though on different texts, looking for different constructs).

We now describe the processing steps in more detail. The first stage is *preprocessing*, in which the raw text in all its variability is reduced to a uniform format for further processing. Specifically, we converted the raw messages to a format that would preserve the metadata elements (e.g. sender, receiver, subject, date) and identified sections of the messages that should not be processed, such as signature lines, source code, or quoted messages (i.e. sections of an email message included in a reply to that message). The preprocessing step was tailored for the particular kind of text we were analyzing, namely email messages among software developers.

The second stage is *tokenization*, which identifies the smallest complete units within a text, usually words, as well as sentence detection. Sentence detection was hampered by the informal nature of email messages, which often did not have correct punctuation or completely grammatical sentence structures, something that had to be taken into account in writing rules at later stages.

In the third stage, each identified token was tagged with a *part of speech* representing morphological and lexical levels of language understanding. These conversions are performed with a combination of a lexicon (i.e. a list of words and their possible parts of speech) and rules about where in sentences different parts of speech can be used. Tokenization and part-of-speech tagging are built-in functions of the NLP tool we used and of similar tools, such as GATE, though a specialized lexicon may be needed for unique vocabulary. 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 (i.e. eliminating suffixes or changes for tense or person).

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 > men-
tion|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, *entity extraction*, 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>
```

Again, these processes, bracketing and entity extraction, are standard features of text processing tools, though they can be guided by domain-specific lexicons.

Rule-writing effort

So far, all of the functionality we have described was provided by the NLP tool and used with little customization. In the final stage of analysis, hand-written *IE* rules are applied to the text to identify embedded text segments that reflect constructs of theoretical interest (i.e. the constructs in Table 2). While applying the rules to the text is a built-in feature of the text-processing tool, the rules themselves must be custom-written for the particular application. These rules were the main product of our development effort, which is described in this section.

An NLP analyst developed IE rules for the group maintenance codes in Table 1. Because of time limitations, we developed NLP rules for only 12 of the 15 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 12, though *Vocatives* pose particular problems that we discuss below.) 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 list 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 simple example rule in Figure 1, a rule for finding *Agreement*. While this rule is a simple example of the possible power of NLP rules to

Premise	`<S> ($it\|$anypos) (do|VBZ) (seem|$anypos) ($anywd|$anypos)* (more\|$anypos) ($anywd\|$anypos) (th[ae]n|$anypos) ($i|$anypos) ($anywd|$anypos)* </S>`
Action	`==> generic ($&, 'entity', 'gm', 'agreement', sf($1,$2,$3,$4,$5,$6,$7,$8,$9));`

Figure 3. Example NLP rule showing premise (text to be sought) and action (output created when text is found).

handle syntax, in using higher-levels of language, the processing goes well beyond the typical capabilities of CAQDAS tools.

As shown in Figure 3, the rules used for this project have a two-part structure: a premise and an action. The premise (the top part of the example rule) defines the matching criteria for the rule, that is, the pattern of words sought in the text. The action (the bottom line of the rule) defines the resulting output when a string in the text being processed 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, and `</S>`, at the end. 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 `$it`, which can be the word 'it,' 'this,' 'that' or 'these,' the actual word represented by `do|VBZ` can be 'do' (i.e. these do) or 'does' (i.e. it does), a simple example of the use of syntax. The element `$anywd|$anypos` represents any token in a candidate text string (any word tagged with any part of speech); the * latter in the rule means that the previous token can be repeated. 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.`

as shown below:

Text	Matching segment of premise
	`<S>`
`It`	`($it\|$anypos)`
`does`	`(do|VBZ)`
`seem`	`(seem|$anypos)`
`to be`	`($anywd\|$anypos)*`
`more`	`(more|$anypos)`
`trouble`	`($anywd\|$anypos)`
`then`	`(th[ae]n|$anypos)`
`i`	`($i\|$anypos)`
`thout at first`	`($anywd\|$anypos)*`
	`</S>`

It should be noted that the example sentence includes a word with a non-standard spelling, namely 'thout.' Such a word would pass through the lower levels of processing largely untouched, though the part of speech might be inferred from the word's location in the sentence. The frequency of use of such non-standard words in texts such as email messages complicates the application of NLP techniques. In this case though, it matches a wild-card section of the rule.

The rule set 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 797 messages from Fire and Gaim were used for this effort as the so-called 'Gold Standard' (GS) data, 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 (81 messages, or about 10%, stratified by source) was set aside for final testing of the completed rule set. The majority of the messages were used to assess the performance of the rule set as it was being built.

Results

To test the performance of the automated process, the developed rule set was run on the GS data reserved for testing. Each message in the test set was inspected to see which instances of group maintenance behaviors 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 IE 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). For example, for the code *Hedges*, the human coders coded a total of 156 segments of text in the set-aside test messages as representing hedges, but the system found only 116 of these, making the Recall 116/156 = 74% (see Table 3). Precision measures the proportion of the automatically extracted data that was coded correctly as compared to the GS data (i.e. accuracy). For example, for the code *Hedges*, the system coded a total of 155 segments of text in the set-aside test messages as representing hedges, but only 116 of these were correct (i.e. matched the human coders), making the Precision 116/155 = 75%. It is usually difficult to have high performance on both measures: in general, the more accurate the results (i.e. the higher the Precision), the smaller the coverage of the target data (i.e. the lower the Recall), 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 raw message data 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 selected for automation. 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*

Table 3. System performance.

	Recall		Precision		GS instances		NLP instances	
Code	Training (%)	Testing (%)	Training (%)	Testing (%)	Training	Testing	Training	Testing
Apologies	89	N/A	81	0	19	0	21	1
Formality	90	100	55	50	29	3	47	6
Complimenting	88	75	70	43	40	4	50	7
Capitalization	96	50	51	40	67	8	126	10
Agreement	87	79	61	65	71	14	102	17
Appreciation	90	63	91	100	90	8	89	5
Emoticon	93	87	46	80	97	23	197	25
Salutations	77	85	79	92	159	13	155	12
Punctuation	77	64	17	23	194	25	899	70
Slang	89	67	70	89	274	58	348	44
Inclusive Pronouns	98	98	90	94	478	45	521	47
Hedges	80	74	63	75	1136	156	1444	155

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 messages held back for final testing.

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

	System			
	Coded	Not coded	Total	Recall
GS				
Coded	116	40	156	74%
Not coded	39			
Total	155			
Precision	75%			

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 favor 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.

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 test messages) included 156 instances of *Hedges*, of which the system correctly coded 116 and missed 40, while the first column of the table shows that the system coded a total of 155 segments of text as being *Hedges*, of which 116 matched the GS and 39 did not. 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 81 messages and 8037 words, 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 in the raw data to the 155 identified by the system), 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 pre-processing

Preparing the data for processing is an important and often time-consuming part of NLP. For this effort, messages were pre-processed 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, extracts from error logs, source file comparisons (diffs), and messages copied in from other sources that were 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. Furthermore, email messages are often non-grammatical, further complicating the development and application of structured rules.

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 thematic unit may consist of a word, phrase, sentence, or even an entire paragraph that is felt by the coders to provide evidence of a given concept. 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 less amorphous unit of analysis for coding, such as a 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, applying NLP requires many examples of correctly coded text for an analyst to consider (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 was compared against the human results, the GS data that are assumed to be correct. In the manual coding, we attempted to reach complete agreement between human coders, but further review revealed 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* or *Humour*, because they were not from the community of developers, and therefore not adequately familiar with some of the terms or community-specific meanings.

Unfortunately, errors in manual coding are propagated in the automatic processing, as rules are built based on possibly erroneous data. Furthermore, the performance of the NLP coding may be judged incorrectly when compared to erroneous data. 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 (i.e. the human coders missed some instances of codes that the NLP rules found). 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 and that having human coders review the NLP output may thus provide superior coding results.

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, TextTagger, 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 technical limitation prevents full exploitation of discourse structure and context of an entire message.
- *Syntactic variety and synonymy*. Natural language provides a nearly 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 currently active but 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 rule set. In the approach we took, NLP coding was built drawing upon a manually crafted qualitative coding codebook (as is required for any such qualitative study). However, the NLP rule set required additional development for the rules and testing time to assess 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 five weeks of effort 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 and for coding additional messages beyond those analyzed for this paper. Once the coders were trained and codebook stabilized, manually coding 700 messages on all 15 codes took approximately 100 h 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. With the current approach, 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 artifacts 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 behavior 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 rule sets. 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 is that it requires even more GS data as input. The number of cases needed depends on several factors, such as the learning algorithm, the number of labels (i.e. codes), and the complexity of the phenomenon, but would typically be in the hundreds, rather than the tens of examples that we have for most codes.

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

Acknowledgement

This research was partially supported by United States National Science Foundation Grant HSD 05-27457. The authors acknowledge the assistance of Na “Lina” Li and Michael Scialdone with the data analysis.

Note

1. http://academic.csuohio.edu/kneuendorf/content/cpuca/ccap.htm provides a helpful list of computer content analysis programs.

Notes on contributors

Kevin Crowston is a Professor in the School of Information Studies at Syracuse University. He received his Ph.D. (1991) in Information Technologies from the Sloan School of Management, Massachusetts Institute of Technology (MIT). 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 NSF support. He is also conducting research on the application of natural language processing technology to social science research. He is currently secretary of the International Federation for Information Processing (IFIP) Working Group 8.2 on Information Systems and Organizations and Program Chair for Academy of Management Organizational Communications and Information Systems Division.

Eileen E. Allen is a Research Administrator in the School of Information Studies at Syracuse University. She is also an Analyst for the School’s Center for Natural Language Processing (CNLP). She received her M.L.S. in 1990 from Syracuse University and has been an analyst for a wide range of projects and applications for natural language processing since 1996.

Dr. Robert Heckman is Senior Associate Dean at the School of Information Studies, Syracuse University. He leads the Global Enterprise Technology Curriculum project, developing practice-based learning opportunities in collaboration with industry and university partners. Dr. Heckman's research interests include teaching and learning strategies for information professionals, design of work-based and practice-based learning experiences, and collaboration in virtual communities and teams.

References

Barry, C.A. (1998). Choosing qualitative data analysis software: Atlas/ti and Nudist compared. *Sociological Research Online, 3*(3). Retrieved from http://www.socresonline.org.uk/socresonline/3/3/4.html

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. doi: 10.1016/S0167-9236(02)00072-6.

Brown, P., & Levinson, S.C. (1987). *Politeness: Some universals in language usage*. Cambridge: Cambridge University Press.

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.

Devaul, H., Diekema, A.R., & Ostwald, J. (2011). Computer-assisted assignment of educational standards using natural language processing. *Journal of the American Society for Information Science and Technology, 62*(2), 395–405. doi: 10.1002/asi.21437

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. Retrieved from http://jcmc.indiana.edu/vol11/issue2/duthler.html

Evans, W. (1996). Computer-supported content analysis: Trends, tools and techniques. *Social Science Computer Review, 14*(3), 269–279. doi: 10.1177/089443939601400302

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.

Gunawardena, C.N., & Zittle, F.J. (1997). Social presence as a predictor of satisfaction within a computer-mediated conferencing environment. *American Journal of Distance Education, 11*(3), 8–26.

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

Ingersoll, G., Yilmazel, O., & Liddy, E.D. (2007, 17–20 April). Finding questions to your answers. In *Proceedings of IEEE 23rd International Conference on Data Engineering Workshop*, Istanbul, Turkey, pp. 755–759. doi:10.1109/ICDEW.2007.4401064

Jurafsky, D., & Martin, J.H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics and speech recognition*. Englewood Cliffs, NJ: Prentice-Hall.

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. In M.A. Drake (Ed.), *Encyclopedia of library and information science* (pp. 2126–2136). New York, NY: Marcel Decker.

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.A., Paik, W., Yu, E., & McKenna, M. (1993, 21–24 March). Development, implementation and testing of a discourse model for newspaper texts. In *Proceedings of Workshop on Human Language Technology*, Princeton, NJ. Association for Computational Linguistics. doi: 10.3115/1075671.1075707

Manning, C.D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA: The MIT Press.

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*. Thousand Oaks, CA: Sage.

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. In *Proceedings of Hawai'i International Conference on System Sciences* (*HICSS-36*). doi: 10.1109/HICSS.2003.1173660.

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

Raymond, E.S. (1998). Homesteading the noosphere. *First Monday, 3*(10). Retrieved from http://tuxedo.org/~esr/writings/homesteading/homesteading/

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, NY: Viking.

Robinson, W., & Vlas, R. (2011). A rule-based natural language technique for requirements discovery and classification in open-source software development projects. In *Proceedings of Hawai'i International Conference on System Science*, Poipu, HI. IEEE. doi: 10.1109/HICSS.2011.28.

Rourke, L., & Anderson, T. (2004). Validity in quantitative content analysis. *Educational Technology Research and Development, 52*(1), 5–18.

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. Retrieved from http://cade.athabascau.ca/vol14.2/rourke_et_al.html

Sager, N. (1970). The sublanguage method in string grammars. In R.W. Ewton & J. Ornstein (Eds.), *Studies in language and linguistics* (pp. 89–98). 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, MA: Addison-Wesley.

Snyder, J., D'Eredita, M.A., Yilmazel, O., & Liddy, E.D. (2009, 7–9 January). Towards a cognitive approach for the automated detection of connotative meaning. In *Proceedings of International Conference on Computational Semantics*, Tilburg, The Netherlands. Retrieved from http://portal.acm.org/citation.cfm?id=1693797

Wayner, P. (2000). *Free for all*. New York, NY: HarperCollins.

Yilmazel, O., Balasubramanian, N., Harwell, S.C., Bailey, J., Diekema, A.R., & Liddy, E.D. (2007, 3–6 January). Text categorization for aligning educational standards. In *Proceedings of Hawai'i International Conference on System Sciences* (*HICSS-40*), Waikoloa, HI. doi:10.1109/HICSS.2007.517.