Optimizing Features in Active Machine Learning for Complex Qualitative Content Analysis

Jasy Liew Suet Yan
School of Information Studies
Syracuse University, USA
jliewsue@syr.edu

Nancy McCracken
School of Information Studies
Syracuse University, USA
njmccrac@syr.edu

Shichun Zhou
College of Engineering and Computer Science
Syracuse University, USA
szhou02@syr.edu

Kevin Crowston
National Science Foundation, USA
crowston@syr.edu

Abstract

We propose a semi-automatic approach for content analysis that leverages machine learning (ML) being initially trained on a small set of hand-coded data to perform a first pass in coding, and then have human annotators correct machine annotations in order to produce more examples to retrain the existing model incrementally for better performance. In this "active learning" approach, it is equally important to optimize the creation of the initial ML model given less training data so that the model is able to capture most if not all positive examples, and filter out as many negative examples as possible for human annotators to correct. This paper reports our attempt to optimize the initial ML model through feature exploration in a complex content analysis project that uses a multidimensional coding scheme, and contains codes with sparse positive examples. While different codes respond optimally to different combinations of features, we show that it is possible to create an optimal initial ML model using only a single combination of features for codes with at least 100 positive examples in the gold standard corpus.

1 Introduction

Content analysis, a technique for finding evidence of concepts of theoretical interest through text, is an increasingly popular technique social scientists use in their research investigations. In the process commonly known as "coding", social scientists often have to painstakingly comb through large quantities of natural language corpora to annotate text segments (e.g., phrase, sentence, and paragraphs) with codes exhibiting the concepts of interest (Miles & Huberman, 1994). Analyzing textual data is very labor-intensive, time-consuming, and is often limited to the capabilities of individual researchers (W. Evans, 1996). The coding process becomes even more demanding as the complexity of the project increases especially in the case of attempting to apply a multidimensional coding scheme with a significant number of codes (Dönmez, Rosé, Stegmann, Weinberger, & Fischer, 2005).

With the proliferation and availability of digital texts, it is challenging, if not impossible, for human coders to manually analyze torrents of text to help advance social scientists' understanding of the practices of different populations of interest through textual data. Therefore, computational methods offer significant benefits to help augment human capabilities to explore massive amounts of text in more complex ways for theory generation and theory testing. Content analysis can be framed as a text classification problem, where each text segment is labeled based on a predetermined set of categories or codes.

Full automation of content analysis is still far from being perfect (Grimmer & Stewart, 2013). The accuracy of current automatic approaches on the best performing codes in social science research ranges from 60-90% (Broadwell et al., 2013; Crowston, Allen, & Heckman, 2012; M. Evans, McIntosh, Lin, & Cates, 2007; Ishita, Oard, Fleischmann, Cheng, & Templeton, 2010; Zhu, Kraut, Wang, & Kittur, 2011). While the potential of automatic content analysis is promising, computational methods should not be viewed as a replacement for the role of the primary researcher in the careful interpretation of text. Rather, the computers' pattern recognition capabilities can be leveraged to seek out the most likely examples for each code of interest, thus reducing the amount of texts researchers have to read and process.

We propose a semi-automatic method that promotes a close human-computer partnership for content analysis. Machine learning (ML) is used to perform the first pass of coding on the unlabeled texts. Human annotators then have to correct only what the ML model identifies as positive examples of each code. The initial ML model needs to learn only from a small set of hand-coded examples (i.e., gold standard data), and will evolve and improve as machine annotations that are verified by human annotators are used to incrementally retrain the model. In contrast to conventional machine learning, this “active learning” approach will significantly reduce the amount of training data needed upfront from the human annotators. However, it is still equally important to optimize the creation of the initial ML model given less training data so that the model is able to capture most if not all positive examples, and filter out as many negative examples as possible for human annotators to correct.

To effectively implement the active learning approach for coding qualitative data, we have to first understand the nature and complexity of content analysis projects in social science research. Our pilot case study, an investigation of leadership behaviors exhibited in emails from a FLOSS development project (Misiolek, Crowston, & Seymour, 2012), reveals that it is common for researchers to use a multidimensional coding scheme consisting of a significant number of codes in their research inquiry. Previous work has shown that not all dimensions in a multidimensional coding scheme could be applied fully automatically with acceptable level of accuracy (Dönmez et al., 2005) but little is known if it is possible at all to train an optimal model for all codes using the same combination of features. Also, the distribution of codes is often times uneven with some rarely occurring codes having only few positive examples in the gold standard corpus.

This paper presents our attempt in optimizing the initial ML model through feature exploration using gold standard data created from a multidimensional coding scheme, including codes that suffer from sparseness of positive examples. Specifically, our study is guided by two research questions:

a) *How can features for an initial machine learning model be optimized for all codes in a text classification problem based on multidimensional coding schemes? Is it possible to train a one-size-fits-all model for all codes using a single combination of features?*
b) *Are certain features better suited for codes with sparse positive examples?*

2 Machine Learning Experiments

To optimize the initial machine learning model, we systematically ran multiple experiments using a gold standard corpus of emails from a free/libre/open-source software (FLOSS) development project coded for leadership behaviors (Misiolek et al., 2012). The coding scheme contained six dimensions: 1) social/relationship, 2) task process, 3) task substance, 4) dual process and substance, 5) change behaviors, and 6) networking. The number of codes for each dimension ranged from 1 to 14. There were a total of 35 codes in the coding scheme. Each sentence could be assigned more than one code. Framing the problem as a multi-label classification task, we trained a binary classification model for each code using support vector machine (SVM) with ten-fold cross-validation. This gold standard corpus consisted of 3,728 hand-coded sentences from 408 email messages.

For the active learning setup, we tune the initial ML model for high recall since having the annotators pick out positive examples that have been incorrectly classified by the model is preferable to missing machine-annotated positive examples to be presented to human annotators for verification (Liew, McCracken, & Crowston, 2014). Therefore, the initial ML model with low precision is acceptable.

Category	**Features**
Content	Unigram, bigram, pruning, tagging, lowercase, stopwords, stemming, part-of-speech (POS) tags
Syntactic	Token count
Orthographic	Capitalization of first letter of a word, capitalization of entire word
Word list	Subjectivity words
Semantic	Role of sender (software developer or not)

Table 1. Features for ML model.

As shown in Table 1, we have selected general candidate features that have proven to work well across various text classification tasks, as well as one semantic feature specific to the context of FLOSS development projects. For content features, techniques that we have incorporated to reduce the feature space include pruning, substituting certain tokens with more generic tags, converting all tokens to lowercase, excluding stopwords, and stemming. Using the wrapper approach (Kohavi & John, 1997), the same classifier is used to test the prediction performance of various feature combinations listed in Table 1.

Model	SINGLE		MULTIPLE	
Measure	**Mean Recall**	**Mean Precision**	**Mean Recall**	**Mean Precision**
Overall				
All (35)	0.690	0.065	0.877	0.068
Dimension				
Change (1)	0.917	0.011	1.000	0.016
Dual Process and Substance (13)	0.675	0.069	0.852	0.067
Networking (1)	0.546	0.010	0.843	0.020
Process (3)	0.445	0.006	0.944	0.024
Relationship (14)	0.742	0.083	0.872	0.089
Substance (3)	0.735	0.061	0.919	0.051

Table 2. Comparison of mean recall and mean precision between SINGLE and MULTIPLE models.

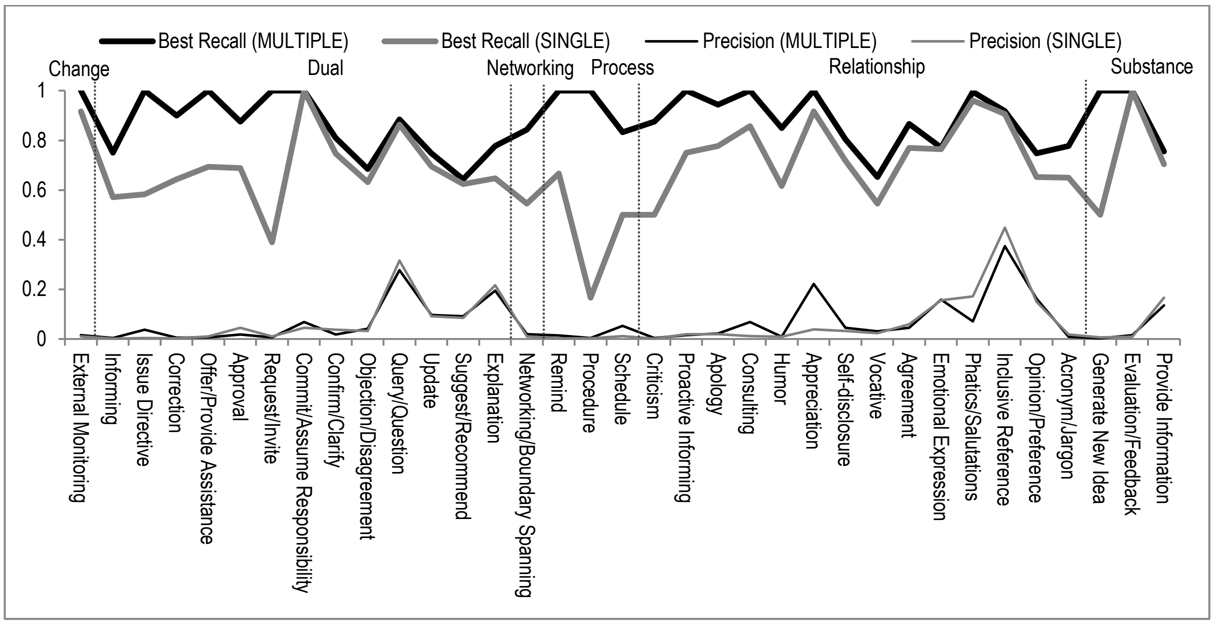

Figure 1. Recall and precision for each code (grouped by dimension).

3 Results and Discussion

We ran 343 experiments with different combinations of the 13 features in Table 1. We first compare the performance of the best one-size-fits-all initial machine learning model that produces the highest recall using a single combination of features for all codes (SINGLE) with an "ensemble" model that uses different combinations of features to produce the highest recall for each code (MULTIPLE). The SINGLE model combines content (unigram + bigram + POS tags + lowercase + stopwords) with syntactic, orthographic, and semantic features. None of the best feature combination for each code in the MULTIPLE model coincides with the feature combination in the SINGLE model. For example, the best feature combination for code "Phatics/Salutations" consists of only 2 out of the 13 features (unigram + bigram).

The best feature combination for each code in the MULTIPLE model varies with only some regularity noted in a few codes within the Dual and Substance dimensions. However, these patterns are not consistent across all codes in a single dimension indicating that the pertinent linguistic features for codes belonging to the same dimension may differ despite their conceptual similarities, and even fitting an optimal model for all codes within a single dimension may prove to be difficult especially when the distribution of codes is uneven, and positive examples for certain codes are sparse. There are also no consistent feature patterns observed from the codes with sparse positive examples in the MULTIPLE model.

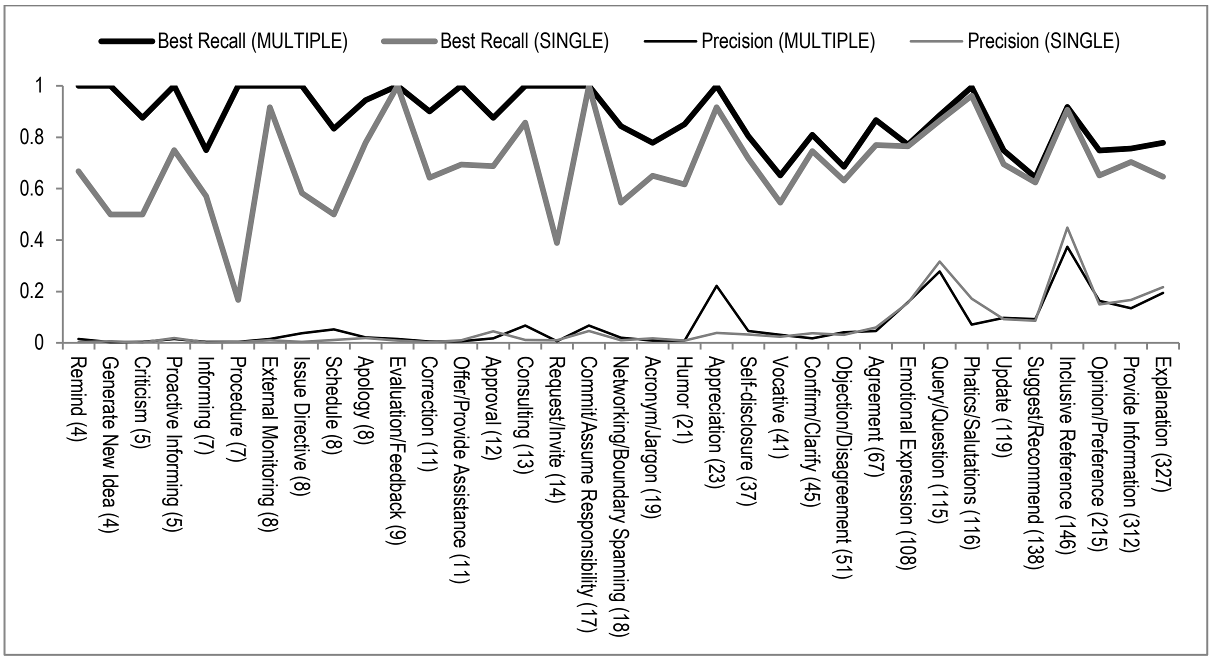

Figure 2. Recall and precision for each code (sorted by gold frequency)

The comparison between the two models in Table 2 further demonstrates that the MULTIPLE model outperforms the SINGLE model both in the overall mean recall of all 35 codes, as well as the mean recall for each dimension. Figure 1 (codes grouped by dimensions) illustrates that the feature combination on the SINGLE model is ill-suited for the Process codes, and half the Dual Process and Substance codes. Recall for each code for the SINGLE model are mostly below or at par with the recall for each code in the MULTIPLE model. Thus, creating a one-size-fits-all initial model may not be optimal when training data is limited. Figure 2 (codes sorted based on gold frequency as shown beside the code names in the x-axis) exhibits that the SINGLE model is able to achieve similar recall to the MULTIPLE model for codes with over 100 positive examples in the training data. Precision for these codes are also higher compared to codes with sparse positive examples. This finding is promising because it implies that creating a one-size-fits-all initial ML model may be possible even for a multidimensional coding scheme if there are more than 100 positive examples for each code.

4 Conclusion and Future Work

We conclude that creating an optimal initial one-size-fits-all ML model for all codes in a multi-dimensional coding scheme using only a single feature combination is not possible when codes with sparse positive examples are present, and training data is limited, which may be common in real world content analysis projects in social science research. However, our findings also show that the potential of using a one-size-fits-all model increases when the size of positive examples for each code in the gold standard corpus are above 100. For social scientists who may not possess the technical skills needed for feature selection to optimize the initial ML model, this discovery confirms that we can create a "canned" model using a single combination of features that would work well in text classification for a wide range of codes with the condition that researchers must be able to provide sufficient positive examples above a certain threshold to train the initial model. This would make the application of machine learning for qualitative content analysis more accessible to social scientists.

The initial ML model with low precision means that the model is over-predicting. As a result, human annotators will have to correct more false positives in the machine annotations. For future work, we plan to experiment with different sampling strategies to pick the most "profitable" machine annotations to be corrected by human annotators. We will also work on designing an interactive and adaptive user interface to promote greater understanding of machine learning outputs for our target users.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1111107. Kevin Crowston is supported by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors wish to thank Janet Marsden for assisting with the feature testing experiments, and gratefully acknowledge helpful suggestions by the reviewers.

References

Broadwell, G. A., Stromer-Galley, J., Strzalkowski, T., Shaikh, S., Taylor, S., Liu, T., Boz, U., Elia, A., Jiao, L., Webb, N. (2013). Modeling Sociocultural phenomena in discourse. *Natural Language Engineering*, *19*(02), 213–257.

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

Dönmez, P., Rosé, C., Stegmann, K., Weinberger, A., & Fischer, F. (2005). Supporting CSCL with automatic corpus analysis technology. In *Proceedings of 2005 Conference on Computer Support for Collaborative Learning* (pp. 125–134).

Evans, M., McIntosh, W., Lin, J., & Cates, C. (2007). Recounting the courts? Applying automated content analysis to enhance empirical legal research. *Journal of Empirical Legal Studies*, *4*(4), 1007–1039.

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

Grimmer, J., & Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297.

Ishita, E., Oard, D. W., Fleischmann, K. R., Cheng, A.-S., & Templeton, T. C. (2010). Investigating multi-label classification for human values. *Proceedings of the American Society for Information Science and Technology*, *47*(1), 1–4.

Liew, J. S. Y., McCracken, N., & Crowston, K. (2014). Semi-automatic content analysis of qualitative data. In *iConference 2014 Proceedings* (pp. 1128–1132).

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Sage.

Misiolek, N., Crowston, K., & Seymour, J. (2012). Team dynamics in long-standing technology-supported virtual teams. Presented at the Academy of Management Annual Meeting, Organizational Behavior Division, Boston, MA.

Zhu, H., Kraut, R. E., Wang, Y.-C., & Kittur, A. (2011). Identifying shared leadership in Wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 3431–3434). New York, NY, USA.