The Future of Coding:

A Comparison of Hand-Coding and

Three Types of Computer-Assisted Text Analysis Methods

Laura K. Nelson
Northeastern University
University of California, Berkeley

Derek Burk
Northwestern University

Marcel Knudsen
Northwestern University

Leslie McCall
CUNY Graduate Center

DRAFT: Comments welcome.

March 13, 2017

Acknowledgements: We are grateful for funding from the Russell Sage Foundation and comments from John Levi Martin on an earlier draft.

Abstract. Advances in computer science and computational linguistics have yielded new, and faster, computational approaches to structuring and analyzing textual data. These approaches perform well on tasks like information extraction, but their ability to identify complex, socially-constructed, and unsettled theoretical concepts – a central goal of sociological content analysis – has not been tested. To fill this gap, we compare the results produced by three common computer-assisted approaches – dictionary, supervised machine learning (SML), and unsupervised machine learning (UML) – to those produced through a rigorous hand-coding analysis of inequality in the news (N=1,253 articles). Although we find that SML methods perform best in replicating hand-coded results, we document and clarify the strengths and weaknesses of each approach, including how they can complement one another. We argue that content analysts in the social sciences would do well to keep all these approaches in their toolkit, deploying them purposefully according to the task at hand.

1.0. Introduction

Content analysis of text-based data is a well-established method in the social sciences, and advances in techniques for collecting and storing data, and in computational power and methods, are continually pushing it in new directions. These advances are typically aimed at making the process more scientific – more reliable, valid, and reproducible.[1] Previous advances include, for instance, intercoder reliability scores (e.g., Krippendorff 1970), designed to validate the coding of text across multiple people; qualitative data analysis software such as AtlasTI and NVivo, designed to enable both qualitative analysis and quantitative identification of patterns to support qualitative conclusions; and the application of algorithms and mathematical models to extract objective patterns in text (Bearman and Stovel 2000; Carley 1994; Franzosi, Fazio, and Vicari 2012; Martin 2000; Mische and Pattison 2000; Mohr and Duquenne 1997).[2]

This latter development, the application of algorithms and mathematical models to text-based data, is seeing renewed vigor from content analysts, as emerging methods in natural language processing and machine learning are enabling new, and faster, computational approaches to structuring and analyzing textual data, including "big" data (DiMaggio, Nag, and Blei 2013; Grimmer and Stewart 2011; Mohr et al. 2013). Advances in using computers to identify categories in text were initiated by computer scientists and computational linguists with the aim of classifying text into pre-specified categories, making text retrieval and information extraction more efficient and accurate (Andersen et al. 1992; Cowie and Lehnert 1996). To test the performance of these algorithms, computer scientists and computational linguists rely on a number of standard, labeled collections of text, such as the Reuters-21578 dataset ("Reuters-21578 Test Collection" n.d.) and the 20 Newsgroup dataset (Lang 1995). Categories in these benchmark datasets are determined by the collection curators and include topics such as "computers", "recreation", "science", and "economics", among others.

The general conclusion from this research is that, given an adequate supply of previously labeled data, researchers can find an algorithm, or an ensemble of algorithms, that will accurately classify unlabeled data into the chosen classification scheme. That is, supervised machine learning algorithms of this kind can promise greater efficiency, transparency, and replicability once a relatively small set of hand-coded documents has proven successful in "supervising" the computer to identify the desired content (Hanna 2013; King, Pan, and Roberts 2013). A number of software packages have therefore been developed to bundle algorithms and simplify their application in routine text analysis projects (e.g., RTextRools, scikit-learn, Stanford NLP, which we discuss below).

As accessibility expands, scholars outside of computer science are moving beyond the benchmark collections and applying them to their own, discipline- or domain-specific tasks. This raises three methodological questions: (1) Can algorithms benchmarked on the standard collections perform as well in other domains? (2) If so, can these algorithms, and other computational tools, be successfully incorporated into the workflow of domain-specific questions and analyses? (3) More ambitiously, can they replace hand-coded work altogether?

We address these questions from the perspective of the "domain" of sociology (and allied disciplines). Scholars are turning to machine learning and other computational methods to augment or replace one of the most common tasks in sociological content analysis: identifying and coding themes, frames, concepts, and/or categories within text. But, in contrast to computer scientists and computational linguists, social scientists are typically not as interested in classifying a massive amount of text into their dominant categories as they are in identifying complex, socially-constructed, and unsettled theoretical concepts, often with ill-defined boundaries, such as populism, rationality, ambiguity, and inequality (Bonikowski and Gidron 2016; Evans 2002; Griswold 1987). Most social scientists continue to rely on traditional human coding methods as the gold standard for the analysis of such phenomena (Benoit, Laver, and Mikhaylov 2009; Grimmer and Stewart 2011).

Our main objective in this paper is to empirically test the three most prominent computer-assisted content coding methods – the dictionary method, supervised machine learning (SML) methods, and unsupervised machine learning (UML) methods – against the gold standard of rigorous hand-coding for a complex topic of sociological interest. While there is considerable effort devoted to developing new algorithms and methods for specific domains and problems (see e.g., Bamman and Smith 2015, Nardulli, Althaus, and Hayes 2015), there is a dearth of empirical research to guide scholars in the selection and application of already established and packaged automated methods, especially with respect to the analysis of complex conceptual content. Can the leading fully-automated approaches to content analysis – dictionaries and unsupervised machine learning (UML) – circumvent the need for hand coding altogether? Indeed, are semi-automated methods like SML even up to the task of coding complex content?

Surprisingly, there has been no comprehensive comparison of how the various techniques perform relative to well established hand-coding methods when performing the kind of content coding of complex material that is of greatest interest to social scientists (including qualitative and quantitative researchers alike). Yet most social scientists do not have the resources to fully test these various approaches when embarking on their own content analysis project. We describe what, exactly, different automated techniques can and cannot do (in answer to the first question above), and show in the process that there can be significant complementarity among the various coding approaches (in answer to the second question above). In doing so we provide a guide to the implementation of these methods in the domain of the social sciences more generally.

Because our aim is not only to inform debates among specialists but also to reach a more general social science audience, we take a different benchmarking tack than is common in the technical literature. Rather than benchmarking *specific algorithms* using datasets coded to test *information retrieval* (as computer scientists and computational linguists have done extensively), we benchmark the three computer-assisted *approaches* (dictionary, SML, UML) on a dataset coded to *identify a complex and multi-faceted theoretical concept* (inequality), using rigorous hand-coding methods well-established among social scientists. We compare substantive findings across the methods by provisionally treating the hand-coded results as the yardstick of measurement. The hand-coding method's wider familiarity and acceptance among social scientists, along with its known weaknesses, enables us to root debates about content analysis methods firmly in realistic, social science data.

Though our focus is on substantive outcomes across the methods, we also offer practical guidance in the use of available software for computer-assisted text analysis. Supervised and unsupervised machine learning programs are at the leading edge of the field, yet even packaged programs require at least some knowledge of programming languages such as Python, Java, and R. We used the three most widely-used "off-the-shelf" packages for applying SML methods: RTextTools (Jurka et al. 2014), Python's scikit-learn (Pedregosa et al. 2011), and Stanford's Natural Language Processing (NLP) Classifier (Manning et al. 2014). Given that these three packages vary in the exact machine-learning algorithms included, the implementation of these algorithms, and default text-processing settings, we wanted to test whether they produced similar results, or whether they varied in their ability to replicate hand coding. We also sought to evaluate their ease of use, and we provide some practical advice and links to learning resources in an appendix.

The dataset, hand-coding methods, and general analytical strategy for testing the automated programs, given the features of our hand-coding project, are described in the next section (section 2.0). We then describe the metrics used to evaluate the accuracy of the automated methods in reproducing the hand-coded results in section 3.0. In section 4.0, we describe in greater detail the three automated approaches to textual analysis, and perform our empirical tests of these approaches, in three subsections on supervised machine learning methods, the dictionary method, and unsupervised machine learning methods. In the final section (5.0), we compare and contrast our results across the methods in order to highlight their strengths and weaknesses from a substantive perspective, and to summarize the ways in which research questions of a substantive and conceptual nature can be appropriately matched to the various content analysis strategies.

2.0. Data and Analytical Strategy

Data and Hand-coding Methods

In the hand-coding project, our substantive objective was to determine whether and when the new issue of rising economic inequality was covered by the media (McCall 2013). Following leading studies in political science on related topics such as welfare and race (Gilens 1999; Kellstedt 2000), we used the *Readers' Guide to Periodical Abstracts* to search for articles on economic inequality from 1980 to 2012 in the three major American newsweeklies of *Newsweek*, *Time*, and *US News & World Report*.[3] The *Readers' Guide* provides a pre-defined list of subject terms for each article, and we selected a set of terms that most closely described our subject matter ("income inequality," "wage differentials," "equality," and "meritocracy"). A surprisingly small number of articles had been assigned these inequality subject terms, however, so we expanded the search to include all articles that were assigned any of the 63 subject terms contained in this smaller set of articles. Because this population of articles numbered in the many thousands (approximately 8,500), we were forced to take a random sample stratified by year (10 to 15 percent of the population in each year)**.** The necessity of sampling when using hand-coding methods, caused by the labor-intensiveness of the process, is a well-known limitation of these methods (i.e., the inability to process large quantities of data). This sample (N=1,253) is the dataset of articles that we use in all subsequent analyses.

Crucial to the rationale for this paper is the fact that we encountered such a variety of subject terms and complexity of subject matter that we felt no choice but to code articles by hand. Unlike comparable studies of media coverage of welfare and race, we assumed neither that all articles (selected using the method described above) were relevant, nor that a preset list of phrases was exhaustive or definitive enough for use in a computer-coding program (at that time).[4] Rather, coding by hand enabled a more flexible approach to identifying and classifying subject matter that varies in form (i.e., the particular words or phrases used) but not necessarily in content (i.e., the concept of interest). This flexibility is perhaps especially necessary when the subject of analysis is a new multi-faceted social issue unfolding in real time, for which settled and durable cultural frames are unavailable. For instance, it was not feasible to deductively catalogue the complete set of metaphors for economic inequality that could be invoked over a three-decade span of news coverage (e.g., the metaphor of "Wall Street versus Main Street" spread wildly during the financial crisis in the late 2000s whereas stories about "union busting" were more germane in the early 1980s). Nor could we generate an exhaustive list of terms that are used to describe every potentially relevant social class group (i.e., the wealthy, the rich, executives, managers, professionals, the middle class, the unemployed, the poor, minimum wage workers, etc.).

Our coding scheme – developed in several stages and iteratively using deductive and inductive reasoning (Chong and Druckman 2009; Ferree et al. 2002; Griswold 1987) – attempted to encompass this wide range of coverage and, in addition, come to a better understanding of several gray areas of coverage (see Appendix A for our definition of inequality). In fact, the challenges we faced in reliably coding the *concept* of inequality – material that conveyed the reality of inequality without necessarily relying on stock *phrases* of inequality – meant that we had to abandon earlier efforts to also code the ways in which the issue was *framed*, particularly in terms of its causes and solutions.[5] As we discuss in subsequent sections, we anticipate using fully automated tools to perform these further analyses on the subset of articles identified by other methods (i.e., hand coding and SML) as mentioning inequality.

Our hand-coded results are presented in Figures 1 and 2. Over a third of articles were ultimately deemed Irrelevant[6] in the process of hand coding and the rest of the Relevant articles were divided into two groups: (1) those that reference the topic of Inequality, which are further broken down into articles with explicit or implicit mentions of inequality (respectively labeled Explicit and Implicit Inequality),[7] or (2) those that fell into a residual category focusing on related but more general trends in income, employment, and the economy. This group, which we term General Economic (or Economic for short) is also broken down into two categories. Figure 1 provides a visual representation of the five underlying categories along a continuum from Irrelevant to Explicit Inequality. The two aggregated relevant categories (Explicit/Implicit Inequality and General Economic) are also represented in the Figure. Two-coder reliability tests were high for the Irrelevant category (.78) and the combined Explicit and Implicit Inequality category (.92 in the first round of coding and .85 in the second round), and thus we focus on replicating them, and especially the central category of interest, Explicit/Implicit Inequality. The over-time trends for the two aggregated Relevant categories plus the Irrelevant category are charted in Figure 2.

[Insert Figures 1 and 2 here.]

General Analytical Strategy

In addition to the complexity of the coding scheme noted above, we highlight several other aspects of our data and coding process that have implications for how we perform our tests of the automated methods and for our expectations. First, the coding and development of the coding instructions took place prior to the spread of the new automated approaches to textual analysis; thus, the coding was not performed *in order to* test the automated programs. Second, and relatedly, we sought to determine only whether the fact of economic inequality, as we defined it, was *ever* mentioned in an article. Notably, this means that many articles were coded as inequality even if the *primary topic* was another issue. As a consequence of these two aspects of the hand-coding process, the automated programs will have to tune out a considerable amount of noise in order to correctly classify the articles (i.e., to agree with the classification of the hand coders). At the same time, the distinctions among the categories could be challenging to detect because most of the articles contain economic material to some degree.

Thus, we have set a high bar for the computer-assisted methods to meet, even those that are trained by previously hand-coded data. With respect to the automated methods that do not have this built-in advantage, the bar may be unreachably high. Our tests are nevertheless instructive, as they clarify exactly what will result, substantively, from the application of each method alone to the data (something we believe is fairly common practice). Specifically, we examine whether, starting from scratch, fully automated methods (UML) isolate the topic of theoretical interest (i.e., inequality) from the potentially numerous other ways in which our data can be categorized. Analogously, we examine whether sophisticated dictionary lists are exhaustive enough to detect the scope and variation of coverage of inequality over time. In short, we use the hand-coding results as a yardstick against which to empirically identify the relative strengths and weaknesses of each of the three broad approaches to computer-assisted textual analysis.

3.0. Measures of Fit

In performing our tests, we utilize three widely used measures of fit: *precision*, *recall*, and *F1* scores (Van Rijsbergen 1979).

Precision refers to the proportion of positive results that are "true" positives according to the hand-coding. For instance, if half of the articles that an automated program classified as mentioning inequality were similarly classified by the hand coders, then the precision score would be .50. *Recall* refers to the proportion of true positives that are also coded by the automated methods as positives. Thus, an analysis with high precision and low recall will be correct in most of its positive classifications, but will miss a large proportion of articles that should have been classified as positive. *F1* scores are the harmonic mean of precision and recall and provide a measure of overall accuracy for each category. While in most situations, F1 scores are taken as the best indicator of fit, we found that precision and recall offered a better sense of where a model is succeeding and erring. Accordingly, we pay as much, if not more, attention to these indicators as to the F1 score. Because these scores are calculated for each category (i.e., inequality, irrelevant, etc.), we also use a weighted average of precision, recall, and F1 score across coding categories as an overall measure of method accuracy.[8]

We add to these standard measures a comparison of the time trends estimated by each of the computer-assisted methods. Not only is the identification of time trends one of the most common objectives of a textual analysis project, but one concern about automated approaches is their potential insensitivity to changes in language over time (Hopkins and King 2010:242). We therefore test for the ability of computer-assisted approaches to reproduce the time trend from the original data, which is based on the proportion of articles in each year coded as falling into our predetermined categories, such as articles that contain explicit and/or implicit mentions of inequality. After using two-year moving averages to smooth the data, we use the correlation between these proportions for the automated programs and for the hand-coded method as a measure of accuracy. These analyses provide an answer to the question of whether computer-assisted coding will yield substantive conclusions similar to those derived from traditional methods.

4.0. Results

We begin with the method that is most similar to hand coding in that it requires hand-coded input (SML). We then evaluate the more fully automated methods in the following sections. Sections on each method are in turn broken down into three subsections: (1) a brief overview of the method, including references to the technical literature in both the text and corresponding appendix for readers interested in greater detail; (2) a description of the analytical strategy, which differs slightly for each method as we calibrate our data and analysis to the particularities of the methods; and (3) the results.

4.1. Supervised Machine Learning (SML) Methods

Brief Description

SML methods leverage both computers' ability to detect patterns across large numbers of documents and human coders' ability to interpret textual meaning. Based on a "training set" of documents hand coded into categories of interest, a SML analysis consists of three steps. First, documents are converted into "vector[s] of quantifiable textual elements," which are called "features" (e.g., words). Second, a machine learning algorithm is applied to find a relationship between these numeric feature-vectors and the hand-coded categories assigned to them, producing a model called a "classifier." Finally, the analyst uses the classifier to code documents not in the training set (Burscher, Vliegenthart, and De Vreese 2015: 124).

In SML methods, then, a document is represented as a vector of word counts, or "bag of words." On its face, treating documents as bags of words seems wrongheaded, given how context can drastically change a word's meaning. Because of the complexity of our hand-coding scheme, changes over time, and the concept of inequality itself, our analysis poses a difficult test for the bag of words approach. However, in practice this strategy has been shown to perform well for many classification schemes of interest to researchers (Hopkins and King 2010). By experimenting with different combinations of our five underlying content codes (see Figure 1), we further examined which types of hand-coded distinctions, if any, the SML methods could replicate.

Analytical Strategy

If we were performing a SML analysis from scratch, we would first hand code a subset of documents from our population of interest. This subset of hand-coded documents is the training set. Second, we would test our SML setup by selecting random subsets of the hand-coded documents to train SML classifiers and try to replicate the classification of the remaining hand-coded documents (called the "test set"). Low levels of agreement would suggest the need to refine the hand-coding scheme or change the specifications for training the SML classifier. Finally, once an acceptable level of agreement was reached (based on precision, recall, and F1 scores), we would train a classifier using *all* the hand-coded documents as input (i.e., the training set), and then use it to classify the larger population of uncoded documents (called the "unseen set").

Because our focus was on testing the ability of SML to replicate hand coding, we only applied our classifiers to hand-coded documents. We constructed 25 artificial training and test sets by randomly selecting roughly 500 articles to be the training set and using the rest (roughly 750) as the test set. We present the range of metrics across the 25 sets for "averaged total" precision, recall, and F1 scores across all categories (see columns 7 to 9 in the first panel of Table 1) but focus our presentation and discussion on the metrics for the individual categories of the median performing set. The metrics for this set are the main entries in all columns of the first panel of Table 1, and the accompanying figures chart the proportion of articles classified by SML into the specified categories over time, again for this median performing set.[9]

[Insert Table 1 here.]

In addition to varying the training and test sets, we also tested three combinations of our five underlying categories. In the first coding scheme, Relevant vs. Irrelevant, we distinguish between all substantively relevant articles and irrelevant articles. In the second coding scheme, Inequality vs. Not Inequality, we distinguish between articles mentioning inequality – whether explicitly or implicitly – and all other articles. In the third coding scheme, Inequality vs. Economic vs. Irrelevant, we distinguish between articles mentioning inequality, those discussing general economic issues but not mentioning inequality, and irrelevant articles. By comparing SML's performance among these three coding schemes, we evaluate the method's ability to replicate distinctions of different types and different levels of aggregation.

We performed our SML analysis using the three most widely adopted SML software packages: RTextTools, Stanford's Natural Language Processing routines, and Python's scikit-learn.[10] We ran each program with comparable settings, within the limits of the options provided by each package, because we wanted to compare the "off-the-shelf" products and minimize the need for users to employ additional scripting (see Appendix Table D1 for the settings for each program). Figures 3-5 show the time trends for all three programs, to demonstrate their commensurability. Because our results were similar across software packages, and because Python's scikit-learn is the most actively developed program of the three, we present results only from that package in Table 1, but include results from the other programs in Appendix Table D2. We also include in Appendix D brief descriptions of each program, along with links to helpful tutorials and learning resources.

Outcomes

Our analyses reveal that SML methods perform well in terms of both precision and recall. Looking first at the metrics averaged across the categories for each of the three classification schemes (column 9 in Table 1), we find average F1 scores for the median test set close to or well above the .70 rule of thumb for good fit often followed in the literature (Caruana et al. 2006). F1 scores are generally quite high for both the Relevant vs. Irrelevant (.83) and Inequality vs. Not Inequality (.78) schemes, indicating that the inequality articles (combining explicit and implicit articles) and the irrelevant articles represent well-defined groupings. F1 scores are lower for the Inequality vs. Economic vs. Irrelevant scheme (.69), suggesting that lower levels of aggregation lead to fuzzier distinctions among categories that are more challenging for the algorithms to recognize, at least in our data. The low F1 score for this scheme stems from lower metrics for the Economic category, which are not shown in Table 1. The F1 score for this category was .52, compared to .69 and .80 for the Inequality and Irrelevant categories, respectively. The recall for this Economic category was especially poor. Of the 200 test-set articles hand-coded into this category, only 93 (47%) were correctly classified by the SML algorithm. The algorithm struggled most in distinguishing between the Economic and Inequality categories, classifying 67 (34%) of these 200 Economic articles as Inequality articles. We return to the significance of this point below.

[Insert Figures 3 to 5 here.]

These methods also reproduce to a remarkable degree the time trend in media coverage of inequality found in the hand-coded data, as shown in Figures 3 to 5 (with two-year moving averages depicted along with correlations). Here, we measure coverage of inequality as the proportion (as opposed to number) of articles coded into the inequality or relevant category, and we include the whole sample (training and test sets) of articles to get the best estimate of actual coverage of inequality. Just as in the hand-coded analysis, the SML results show peaks in inequality coverage in the early 1990s and around the period of the Great Recession. The over-time correlation between the hand-coded and SML trends ranges from .69 to .75 (shown in column 10 of Table 1).

An important takeaway from our analyses is that the selection of classification schemes may depend more on the precision and recall metrics for individual categories of interest than on the average total F1 score across categories, which is a more common practice in the literature. For example, in testing different three-category coding schemes, we obtained a higher overall F1 score with an Explicit Inequality vs. Implicit Inequality/Economic vs. Irrelevant scheme than with the Inequality vs. Economic vs. Irrelevant scheme presented in Table 1. This higher F1 value was due to better precision and recall for the combined Implicit Inequality/Economic category compared to the Economic category, suggesting that the poor recall for the Economic category, as described above, was likely due to similarities between articles with implicit mentions of inequality and those discussing related economic issues. However, the trade-off to obtain this higher overall F1 value was markedly worse performance in identifying Explicit

Inequality articles. Given our substantive interest in inequality, we opted for a coding scheme that better identified articles mentioning inequality over one with better performance overall.

In sum, supervised machine learning models were not only successful at replicating the hand-coded results overall and over time, thus, importantly, boosting confidence in the reliability of those results, but they also prompted a deeper analysis and understanding of the subject matter. This is particularly the case for distinctions between articles with explicit versus implicit mentions of inequality and between the “middle” General Economic category and the categories that bookend it. Ultimately, given the results of this exercise, a researcher could then gather the full population of articles (recall that we had only a 10 to 15 percent sample), or another sample/population of articles (e.g., from the *New York Times*) and code them using these semi-automated methods, assuming coverage features are roughly equivalent across different kinds of publications. Indeed, an object file containing the relevant information for classifying articles into categories based on our full set of hand-coded articles (as the training set) can be made available to other researchers, which not only eliminates the need for hand coding within specific content domains but facilitates the usage of SML methods and the analysis of larger corpuses of text.

4.2. Dictionary and Unsupervised Learning Methods

Because SML algorithms require a non-trivial number of hand-coded texts, a time-intensive and often expensive task, social scientists are exploring more fully automated text analysis methods to circumvent the need for hand-coding text. However, fully automated methods (Grimmer 2010) and dictionary methods (Loughran and McDonald 2011) cannot be mechanistically applied; their output is typically tested by hand *post facto*. That is, the methods are implemented on a corpus and then hand coders go back through on a sample of the corpus to test the validity of the computer-assisted codes. In this respect, dictionary and fully automated methods rely to a non-trivial degree on the judgment of the analyst to interpret and verify the results, at best using the most rigorous tests of reliability adopted by hand coders. Given that we have a large set of hand-coded results already in hand, our analysis is intended to make these judgment points explicit, along with the consequences for drawing substantive conclusions from the application of each method, had it been chosen originally as the only method of analysis of our data.

4.2.1 Dictionary Method

Brief Description

The dictionary method is the most straightforward and widely used of the automated textual analysis tools available. This is especially the case when a media content analysis is not the central objective of a scholarly piece of research, but instead is employed to quickly chart issue salience in the media or to supplement findings from a survey-based analysis with more contextual data. On the subject of inequality, for instance, the Occupy Wall Street movement prompted what appeared to be an increase in media coverage of inequality. The impulse to quantify this shift led researchers to use keyword searches of "inequality" in the news to draw conclusions about the extent to which the public was being exposed to new information, as this is considered a key determinant not only of issue salience but of issue-specific political and policy preferences (McCall 2013; Milkman et al. 2013).

Dictionary methods consist, then, of a search through a corpus of documents for a list of words or phrases predetermined by the researcher, offering a quick and relatively easy way to code large volumes of data. Dictionary methods can be considerably more sophisticated, however, requiring a carefully curated list that indicate the category of interest. Standard dictionaries such as the Linguistic Inquiry and Word Count (Tausczik and Pennebaker 2010) have been shown to be reliable, but only in limited domains. Creating specialized dictionaries has the benefit of being domain-specific, but it is still unclear whether dictionaries can be reliably used to code complex text and unsettled concepts, the focus of our analysis.

Analytical Strategy

We use the text mining tools available in the statistical package R to search for articles with keywords from a combined list of two comprehensive dictionaries on inequality (Enns et al. 2015; Levay 2013). Because these lists are comprised of variations on the term inequality and its synonyms (i.e., divide, gap, etc.), we compare the results of this method to the results from the hand-coded Explicit Inequality category only. In a subsequent analysis, we also attempt to translate our own hand-coding instructions in Appendix A as closely as possible into a list of terms and search instructions to identify explicit *and* implicit mentions of inequality. Appendices B and C provide these lists and instructions, respectively. If any term or phrase in the dictionary instructions is present in an article, the article is placed in the Inequality category; otherwise, the article is placed in the Irrelevant category.[11] This is consistent with our hand-coding procedure, in which a single mention of relevant content is sufficient to place an article in the Inequality category, and it is a lenient test of the dictionary method.

[Insert Figure 6 here.]

Outcomes

The results of our analysis of the hand-coded articles using these two dictionaries are presented in the second panel of Table 1 and in Figure 6. What we find is that the carefully constructed lists of terms provided by Enns et al. (2015) and Levay (2013) – which are combined in our analysis – are remarkably successful at identifying articles hand coded as containing explicit coverage of inequality. With a precision of .91 (see column 1 of second panel of Table 1), this method was highly unlikely to misidentify non-inequality articles as inequality articles; that is, it resulted in few false positives. Yet, as is often the case, precision came at a cost: with a recall score of just .25 (see column 2 of Table 1), many of the articles hand coded as explicit were overlooked, not to mention articles that were coded as implicitly covering inequality (which we excluded from the Inequality category for these tests). This substantial degree of underestimation is visually apparent in Figure 6, which compares the time trends revealed by the hand-coding and dictionary methods.[12] By contrast, the instructions intended to mirror the complexity of our own hand-coding process, including both implicit and explicit mentions of inequality, erred in the opposite direction: with high recall (.84) and low precision (.48), coverage of inequality was over-identified, as also illustrated in Figure 6.

We draw two conclusions from this exercise. First, dictionary lists can accurately identify the most explicit instances of coverage, and, somewhat to our surprise, even approximate a time trend of coverage (the correlation with the trend of articles hand-coded as explicit was .42 when we use a 2-year moving average, as shown in column 10 of Table 1), but they are likely to miss more nuanced portrayals of a topic and thus significantly underestimate overall occurrence. If frequency of occurrence matters, then this is a serious shortcoming.[13] Second, a more complex set of instructions can effectively net a larger share of relevant articles, and even better approximate the time trend (*r*=.62), but they will in the process erroneously categorize a large share of irrelevant articles as relevant. Although it may be possible to fine-tune the dictionary instructions to arrive at a happy medium between the two extremes represented by our two versions of the dictionary method,[14] we underscore again that researchers beginning from scratch will not know, as we would, when they have arrived at this happy medium.

4.2.2 Unsupervised Machine Learning (UML) Methods

Brief Description

Finally, there is hope that fully automated methods – or unsupervised machine learning (UML) tools – can inductively identify categories and topics in text, thus replacing human input altogether, at least on the front end (Carley 1994; Bearman and Stovel 2000; Franzosi 2004; Grimmer and Stewart 2011; Lee and Martin 2015; Mohr 1998). Rather than classifying text into pre-determined categories, as is the case with the dictionary and SML methods, fully automated text analysis techniques simultaneously generate categories and classify text into those categories. In theory, these techniques will inductively categorize text into the objectively "best" categories. In practice, there are multiple ways to classify text, with no clear metrics to determine which classification is better than others (Blei 2012; Grimmer and Stewart 2011). When a fully-automated method offers multiple ways to group texts, researchers may qualitatively consider the topics covered as well as statistical fit. The complexity of these algorithms, the "black box" nature of their implementation and interpretation, and the sometimes cryptic output they generate has meant that social science researchers, in particular sociologists who are attuned to the complexity of language and concepts, are hesitant to fully embrace their use (e.g., Lee and Martin 2015).

Because the coding scheme in our hand-coded data was done in part inductively as well, as is common in qualitative analysis, and the categories are detailed enough to represent bounded, though complex, topics, we have the opportunity to compare computationally inductive techniques to the hand-coding technique. Our findings thus build on debates about the potential to substitute (allegedly) faster and more replicable UML methods for traditional content analysis (e.g., Bail 2014; DiMaggio, Nag, and Blei 2013; Lee and Martin 2015).

Analytical Strategy

We used three fully automated methods in an attempt to identify inequality themes in these data.[15] The first two are from the probabilistic topic modeling family (Blei 2012). Using the co-occurrence of words within documents, probabilistic topic models use repeated sampling methods to simultaneously estimate topics and assign topic weights to each document. In other words, topic models assume that each document is made up of multiple topics with varying levels of prevalence, rather than assigning each document to one topic or category. We estimate two topic models using two different algorithms, Latent Dirichlet Distribution (LDA), the most basic topic model (Blei 2012), and Structural Topic Models (STM) (Roberts, Stewart, Tingley, and Airoldi 2013), a topic modeling algorithm that provides a way to incorporate document-level covariates into the model. Because the language used to discuss inequality changed over time, we include the document year as a covariate in our STM. As with many fully automated methods, the researcher must choose the number of topics to be estimated by the algorithm, and we ranged the number of topics from 5 to 100 at various intervals for both algorithms, looking at the highest weighted words per topic to determine the content of the topic.

As noted, topic models do not assign articles to topics as hand coders do; rather, each document is a weighted distribution over all topics. In order to compare these results to those obtained using hand-coded methods, then, we classified an article as being about inequality if the associated topic weight was in the 95th percentile of the topic score among articles hand-coded as irrelevant. This is intended to avoid classifying articles as about inequality if they simply contained routine mentions of the words (common in everyday language) associated with the inequality topic. In addition, because of the infrequency of our topic using these methods, we measure their performance against the category of articles hand-coded as explicit only (and not implicit), much like in the evaluation of the first dictionary method.

While increasingly popular in the social sciences, topic modeling has been criticized outside of the social sciences for its poor predictive performance and its lack of reproducibility (e.g., Lancichinetti et al. 2014). Simpler clustering techniques often perform just as well, and sometimes better, than more complicated hierarchical and topic modeling techniques (Steinbach, Karypis, and Kumar 2000; Schmidt 2013). Our third fully automated technique is thus the relatively simple k-means clustering algorithm, an established and ubiquitous algorithm that uses Euclidean distance measures to cluster articles into *mutually exclusive* groups (Lloyd 1957, Jain 2010). Like topic modeling, the number of clusters is determined by the researcher, using visual methods, mathematical methods (e.g., Bayesian Information Criterion), or qualitatively by examining the coherence of the clusters (Rousseeuw 1987). We ranged the number of clusters from 2 to 70, looking at the most frequent words per cluster to determine the content of the cluster.

Outcomes

The metrics are provided in the third panel of Table 1 and the time trend for the STM results are shown in Figure 7 (there were too few relevant articles from the k-means analysis to construct a time trend, and the LDA results were similar to the STM results). We begin with the k-means analysis before examining the more complex methods. Using the silhouette method (Rousseeuw 1987) combined with the Bayesian Information Criterion (BIC) (Pelleg and Moore 2000), the 18-cluster model produced the most distinctive clusters, but none of these clusters were clearly about inequality. Beginning with the 20-cluster model, there was one cluster that seemed to center on inequality, and there were two such clusters in the 60-cluster model. The word “inequality” never appeared as a frequent word, however, in any of these clustering solutions (despite the fact that the SML methods were capable of identifying inequality content within the broader corpus of articles). The silhouette method isolated the 30-cluster model as having the most distinctive clusters in the second set of models (20-70 clusters, in which the BIC steadily declines after 20 clusters), and it included a cluster that appeared consistent with our theme (see the first column of Table 2 for the most frequent words in this inequality cluster). Yet with only 42 articles in this cluster, these methods appeared to quite dramatically undercount the number of articles about inequality in our data.

[Insert Table 2 here.]

The results from this k-means analysis suggest two important conclusions. First, there is no guarantee that the clusters produced by the k-means algorithm will line up with the topics of interest to the researcher. Furthermore, the mathematically "best" clustering solution may not necessarily be the best solution from a substantive perspective, as was the case with our data (i.e., the mathematical methods were no better than visual inspection at identifying models containing clusters with an inequality theme). Second, these results confirm the intuition that, in our data, discussions of inequality are woven throughout articles whose main focus is a separate topic; that is, inequality as a dominant topic is relatively infrequent. K-means is thus better suited to the analysis of thematically focused articles, such as tweets or press releases, and does not perform well in picking up themes that may be buried within discussions of different topics.

[Insert Figure 7 here.]

The other fully automated method we use, topic modeling, is designed to address this shortcoming by picking up more minor themes running across many articles. After failing to find a computer-generated topic on the subject of inequality when the number of topics for the STM was set to 5, 10, and 20, one did emerge in the output of a 30-topic model. The 20-topic and the 60-topic models, however, produced the most coherent topics as measured by the distribution of the top weighted topic over all documents, a mathematical solution that indicates distinctive topics. Because the 60-topic model also produced an inequality topic, we analyze the results from this model (see the second column of Table 2 for a list of the top weighted words associated with this topic from the 60-topic model).[16]

Generally speaking, the results mirror those for the first dictionary method, in which precision is high but recall is low. While the recall is extremely low using the k-means method (.14), there is somewhat more balance in the results from the STM method, in which a larger share of explicit articles are identified as compared to the first dictionary method (compare the recall score of .45 for STM with the recall score of .25 for the first dictionary method, as shown in column 2 of Table 1); consequently, the F1 score is also higher (.53 versus .40, as shown in column 3). The correlation of the two-year moving averages also improves (compare .58 for STM versus .42 for the first dictionary method). Given that our approach to hand coding was not "topical," in the sense that we were searching for *any* coverage of inequality in articles on *any* subject matter (broadly on economic matters), it is perhaps impressive how well the topic modeling algorithms actually correspond to the hand-coded articles. On the other hand, like the first dictionary method, the fully automated methods are under-counting the number of “true” inequality articles. If we had used only these methods for the original analysis, as we suspect many content analysts are now doing, we would have missed almost all of the implicit discussions of inequality, and many of the explicit ones as well (as demonstrated by the low recall of .45).

Overall, given that some of the clustering or topic modeling solutions did not pick up an inequality topic, and given the low recall for STM, we suggest that it may be best to deploy this method *after* categories have been defined and articles classified, in order to focus the topic modeling exercise on only the primary category of interest. For instance, once articles mentioning inequality have been selected with some degree of confidence (i.e., using either conventional hand-coding metrics of reliability, the dictionary method, supervised learning methods, or some combination of these), one could use unsupervised machine learning methods to examine the content of these articles in greater detail. They could be used, for example, to identify the range of frames and topics with which inequality often co-occurs – such as the discussion of taxes, immigration, education, and so on. As topic modeling assumes each document is structured from multiple topics, this could be an appropriate method for doing so. However, we would ideally want a larger corpus of articles than we currently have at our disposal to perform this deeper content analysis.

5.0. Discussion and Conclusion

Our main conclusion is that these new computer-assisted methods can effectively *complement* traditional human approaches to coding complex and multifaceted concepts in the specialized domain of sociology (and related disciplines), but the evidence is mixed as to whether they can fully *replace* traditional approaches. SML methods successfully approximated and thus may partially substitute for hand-coding, whereas the other methods are best implemented in conjunction with hand coding (or SML). In this section, we highlight the strengths and weaknesses of the various approaches in evaluating our hand-coded data, focusing in particular on the substantive conclusions that would have been drawn from the results produced by each method. The larger objective of this discussion, however, is to shed light on the pros and cons of each method for a broader array of text analysis projects. Taken together, our results confirm the effectiveness of each of these methods for specific roles in the workflow of a content analysis project.

To begin with the most widely used of the automated methods, the dictionary method successfully identified a subset of the most explicit discussions of inequality in our data, as evidenced by the dictionary-identified articles “The Inequality Dodge,” “Rich America, Poor America,” “To the Rich, From America,” and “The Quagmire of Inequality.” However, this method missed more nuanced but nonetheless obvious (to a knowledgeable coder) discussions of inequality. Specifically, this method failed to detect a large share of articles in the early 1990s that were hand coded as about inequality (see Figure 6). Media coverage at this time dealt primarily with the problem of rising wage and earnings inequality in the labor market, as opposed to Wall Street or the top one percent, in articles such as “Bridging the Costly Skills Gap” and “Bob Reich's Job

Market." These articles discussed the simultaneous rise in productivity and stagnation of male wages, the gap in wages between college and non-college-educated workers, and excessive executive pay. Concerns of fairness in the labor market were paramount as transformations in the economy appeared to threaten the financial security of many workers.

The SML algorithms, alternatively, confirmed the rise in coverage of inequality in the early 1990s that was identified by the hand coders (see Figure 4). The features (words) that most distinguish the Inequality from Not Inequality categories include "class," "middle," "pay," and "wage" – words indicative of the inequality discussion in the 1990s. However, they also include words one would not immediately associate with inequality, such as "benefit" or "families," suggesting that the SML approach represents more than just a glorified dictionary method. One article in particular highlights the difference between the dictionary and SML methods. An article printed in 1994 titled "Reining in the Rich" was correctly identified by the SML programs as being about inequality, but was not identified by the dictionary method. The story never uses words like income gap or income inequality. Instead, the discussion is about how social security subsidizes the lifestyles of the affluent:

> The costliest welfare load isn't for the poor, it's for the well-to-do...[A rich retiree] knows he is being subsidized by the 12.4% payroll tax being paid by employers and their younger and lower-paid workers, like his granddaughter Amanda Fargo, 21, who earns $5 an hour as a receptionist in a beauty salon. Savage approves of taxpayer subsidies for the elderly

poor, but adds, “It's unconscionable . . . to take money away from these kids and give it to well-off people.”

While this article reflects on a well-known aspect of inequality, it does not contain any of the words or phrases in the carefully curated dictionary developed by previous researchers.

Our research thus suggests that dictionary methods will struggle with the identification of broader concepts, but can play a role when specific phrases are of interest (e.g., the “one percent”) or accuracy and prevalence are not at a premium. For example, tracking the use of the word “inequality” could be useful in revealing shifts in the way that the underlying concept of inequality is being represented, especially if it could be shown that the deployment of the inequality term itself has substantively meaningful consequences (e.g., for understanding how public discourse reflects or shapes public perceptions and views about inequality). By contrast, we show that dictionaries are not an appropriate method if the purpose is to identify complex concepts or themes with myriad meanings and definitions, particularly over long periods of time when the terms chosen to represent them are likely to vary.

Supervised machine learning algorithms, on the other hand, are well equipped to recognize these more complex concepts, even as the specific content related to the concepts changes over time; we were therefore able to almost completely replicate our hand-coding scheme using SML algorithms. The success of this method in discerning significant shifts in discussions of inequality gives us confidence that it can be used on most concepts or themes of interest to sociologists, provided they are reasonably bounded (recall the difficulties SML methods encountered distinguishing Implicit Inequality from General Economic articles). This method does, however, require much more investment at the front end of the project to correctly hand code a non-trivial number of articles. With this caveat in mind, SML approaches can replace hand-coding approaches if the objective is to code large quantities of text and capture nuanced discussions of complex concepts.

Finally, structural topic modeling (STM) is also well equipped to identify salient clusters of words, and like the SML algorithms, it correctly identified the above article on inequalities in the social security tax and transfer system. Likewise, it correctly picked up the rise in discussion of inequality in the early 1990s. But, as the presentation of results using this method illustrated, UML approaches will not necessarily identify in every model the specific concepts or themes of interest to a researcher. And, if it does, the qualitative decision points involved, such as choosing the number of topics and types of words to include, should give deductive researchers pause. Additionally, to tag an article as having mentioned inequality, we used a cutoff determined by the distribution of topic probabilities across articles *formerly hand coded* as irrelevant. If we were doing a content analysis project from scratch (i.e., without any prior hand coding), we would not be able to perform this sort of benchmarking, creating another choice-point for the researcher. More likely, researchers using this method would examine the proportion of words structured from a topic – charting, for example, this proportion over time – rather than tagging entire documents into categories (and charting the proportion of articles falling into these categories over time, as we did).

If the goal is not to categorize documents into known categories but to inductively explore textual data and the themes that emerge from them, or to explore how topics co-occur in texts, topic models are a good solution. In particular, once relevant content has already been identified using other, more reliable methods, such as SML, fully automated methods can then be used to examine the content in greater detail and in a more inductive fashion (e.g., in our case, we would investigate exactly *how* inequality is covered or framed in the relevant articles or which other topics inequality is most commonly associated with). Our results demonstrate the ability of topic models to recognize patterns of theoretical interest in textual data, indicating that they can be used to complement other forms of analysis. We do not think our data are unusual in this respect, as many researchers use the media as their source of data, though it is certainly possible (and may be preferable) to apply these methods to more thematically focused material (e.g., Hanna 2013).

In closing, we wish to underscore that even though our conclusion regarding the significant complementarities among the methods we discussed is based on the current state-of-the art, we believe it will continue to apply in the foreseeable future as new computer-assisted text analysis methods and techniques are being developed. For example, on the one hand, new work in word embeddings, which incorporate the context in which a word is used more effectively than in previous methods, can further improve the performance of natural language processing algorithms (Goth 2016; Mikolov et al. 2013). Sociologists would therefore benefit from an ongoing engagement with this literature to elevate their own application of computer-assisted techniques. Yet, on the other hand, we as a discipline should think carefully about exactly how these new methods correspond to the types of research questions and data at the core of our scholarly enterprise, including those that privilege humanistic interpretation. Comparing and contrasting automated methods to nuanced hand-coding methods provides an empirical foundation that has been lacking in debates over the relationship between our methodological traditions and the new computer-assisted techniques, and that we hope advances these debates to better understand the future of textual analysis in sociological research.

Endnotes

[1] How exactly to make content analysis "scientific", and if that is even possible, is of course contested (see, e.g., Biernacki 2012; Reed 2015; Spillman 2015).

[2] Past research has used semi-automated methods to quantify the structural narrative of texts (Bearman and Stovel 2000; Franzosi, Fazio, and Vicari 2012), clustering and block modeling methods to measure latent cultural structures embedded in text (Martin 2000; Mische and Pattison 2000; Mohr and Duquenne 1997), and map and network analyses to measure relationships between concepts within texts (Carley 1994).

[3] Due to a less straightforward subject matter, however, our method ended up differing substantially from prior studies that were able to obtain their target population of articles with only a few keywords, such as race and welfare.

[4] See Gilens (1999), Dyck and Hussey (2008), and Kellstedt (2000) for approaches that retain all articles from the search as relevant and then either code pictures only or use computerized methods to identify frames.

[5] Sociologists "code" text in a variety of ways that vary in complexity, including classifying whole or parts of text into different categories, identifying different themes or frames in text, and identifying rhetorical techniques such as persuasion, satire, or ambiguity, to name a few. We see our hand-coded data as a form of complex text classification, complex enough to entail a challenge for these automated methods. Further research could investigate different types of coding tasks in a similar way as we do here.

[6] Irrelevant articles were on the following topics: racial or gender inequality, gay rights, inequality in other countries, individuals whose names are part of a subject term

(e.g., Marc "Rich"), popular culture items that include part of a subject term (e.g., a movie named "Big Business"), clearly personal affairs about a single individual, non-economic elites (e.g., in art or religion), and social class as a predictor of non-economic phenomenon (e.g., health, drug use).

[7] Appendix A describes the distinction between explicit and implicit mentions of inequality (see in particular panel 4).

[8] Specifically we take the macro average across categories: average total precision = average of precision scores for each category; average total recall = average of recall scores for each category; average total F1 = (2*average_precision*average_recall)/(average_precision + average_recall).

[9] Whereas the table reports metrics for the test set, the graphs provide trends for the entire sample of articles, including both test and training sets, as the substantive results for the entire sample (and by inference, the population) are of interest to the researcher.

[10] We also performed extensive tests of the ReadMe program, which is available as a package for R or as a stand-alone program (Hopkins, King, Knowles, and Melendez 2013). We include information about and results from that analysis in Appendix D. However, because ReadMe directly estimates the proportion of documents falling in each category rather than classifying documents individually, it was not possible to create precision, recall, and F1 statistics.

[11] To account for the fact that keywords may occur by chance in articles not related to inequality, we also considered a threshold-based approach to classification, whereby, for example, an article would be placed in the inequality category only if the incidence of

keywords exceeded the 95th percentile of keyword-incidence among articles hand-coded as irrelevant. However, because there is no established procedure for setting such a threshold in the literature, we opted to present results for the simpler "one-occurrence" dictionary-coding scheme.

[12] An alternative method for constructing a time trend from a keyword dictionary is to chart the incidence of keywords as a proportion of total words in each year, as opposed to charting the proportion of articles containing at least one keyword. We tested this alternative method, but found that the trend in keyword incidence was prone to wild swings from year to year and did not closely follow the trend constructed through hand coding. The correlation between the trend in keyword incidence and the proportion of articles hand-coded as explicitly covering inequality was 0.46, compared to 0.59 between the proportion of articles containing at least one keyword and the hand-coded trend.

[13] On the other hand, if explicit and implicit coverage are correlated, then inferences about overall coverage and trends in coverage may not be overly biased (though a comparison of these trends in Figure 3 reveals that the trend for explicit articles differs from the trend for combined explicit and implicit articles).

[14] For example, to improve the recall of the two-word, modifier-noun keyword approach, we could expand the list of keywords in order to capture more of the ways in which inequality is discussed. On the other hand, to improve the precision of our more complex scheme, we could require that two keywords occur in the same sentence, or the same paragraph, rather than anywhere in the article.

[15] For the topic models and the k-means model below, we performed common pre-processing steps: we converted all letters to lower case, removed punctuation, and stemmed words using the Porter Stemmer algorithm.

[16] We also ran a 60-topic LDA model, and the results were similar to STM. With the LDA model, we identified 150 articles as having content on inequality, whereas we identified 278 articles with STM. The F1 score was similar for the two (.52 for LDA and .53 for STM), with recall higher for STM (.45 compared to .41 for the LDA model) and precision lower for STM (.63 compared to .71 for the LDA model). Given the similar F1 scores and the fact that STM flagged more articles, we focus on the results from the STM analysis only.

References

Andersen, Peggy M., Philip J. Hayes, Alison K. Huettner, Linda M. Schmandt, Irene B. Nirenburg, and Steven P. Weinstein. 1992. "Automatic Extraction of Facts from Press Releases to Generate News Stories." Pp. 170-177 in *Proceedings of the Third Conference on Applied Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics.

Bamman, David, and Noah A. Smith. 2015. "Open Extraction of Fine-Grained Political Statements." Pp. 76-85 in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics.

Biernacki, Richard. 2012. *Reinventing Evidence in Social Inquiry: Decoding Facts and Variables*. New York: Palgrave Macmillan.

Blei, David M. 2012. "Probabilistic Topic Models." *Communications of the ACM* 55(4): 77-84.

Bonikowski, Bart, and Noam Gidron. 2016. "The Populist Style in American Politics: Presidential Campaign Rhetoric, 1952-1996." *Social Forces* 94 (4): 1593-1621.

Bearman, Peter S., and Katherine Stovel. 2000. "Becoming a Nazi: A Model for Narrative Networks." *Poetics* 27 (2–3): 69–90.

Benoit, Kenneth, Michael Laver, and Slava Mikhaylov. 2009. "Treating Words as Data with Error: Uncertainty in Text Statements of Policy Positions." *American Journal of Political Science* 53(2): 495-513.

Burscher, Bjorn, Rens Vliegenthart, and Claes H. De Vreese. 2015. "Using Supervised Machine Learning to Code Policy Issues: Can Classifiers Generalize across Contexts?" *The ANNALS of the American Academy of Political and Social Science* 659 (1): 122–31.

Carley, Kathleen. 1994. "Extracting Culture through Textual Analysis." *Poetics* 22 (4): 291–312.

Caruana, Rich, and Alexandru Niculescu-Mizil. 2006. "An Empirical Comparison of Supervised Learning Algorithms." Pp. 161-168 n *Proceedings of the 23rd International Conference on Machine Learning*. New York: ACM.

Chong, Dennis and James N. Druckman. 2009. "Identifying Frames in Political News." Pp. 238-287 in *Sourcebook for Political Communication Research: Methods, Measures, and Analytical Techniques* edited by E. P. Bucy and R. L. Holbert. New York: Routledge.

Cowie, Jim, and Wendy Lehnert. 1996. "Information Extraction." *Communications of the ACM* 39 (1): 80–91.

DiMaggio, Paul, Manish Nag, and David Blei. 2013. "Exploiting Affinities between Topic Modeling and the Sociological Perspective on Culture: Application to Newspaper Coverage of U.S. Government Arts Funding." *Poetics* 41 (6): 570–606.

Enns, Peter, Nathan Kelly, Jana Morgan, and Christopher Witko. 2015. "Money and the Supply of Political Rhetoric: Understanding the Congressional (Non-)Response to

Economic Inequality." Paper presented at the APSA Annual Meetings, San Francisco.

Evans, John H. 2002. *Playing God?: Human Genetic Engineering and the Rationalization of Public Bioethical Debate*. Chicago: University Of Chicago Press.

Ferree, Myra Marx, William Anthony Gamson, Jurgen Gerhards, and Dieter Rucht. 2002. *Shaping Abortion Discourse: Democracy and the Public Sphere in Germany and the United States*. New York: Cambridge University Press.

Franzosi, Roberto. 2004. *From Words to Numbers: Narrative, Data, and Social Science*. Cambridge, UK: Cambridge University Press.

Franzosi, Roberto, Gianluca De Fazio, and Stefania Vicari. 2012. "Ways of Measuring Agency: An Application of Quantitative Narrative Analysis to Lynchings in Georgia (1875–1930)." *Sociological Methodology* 42 (1): 1–42.

Gilens, Martin. 1999. *Why Americans Hate Welfare: Race, Media, and the Politics of Antipoverty Policy*. Chicago: University of Chicago Press.

Goth, Gregory. 2016. "Deep or Shallow, NLP is Breaking Out." *Communications of the ACM* 59(3): 13-16.

Grimmer, Justin. 2010. "A Bayesian Hierarchical Topic Model for Political Texts: Measuring Expressed Agendas in Senate Press Releases." *Political Analysis* 18 (1): 1–35.

Grimmer, Justin, and B. M. Stewart. 2011. “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.” *Political Analysis* 21 (3): 267–97.

Griswold, Wendy. 1987. “A Methodological Framework for the Sociology of Culture.” *Sociological Methodology* 17:1-35.

----. 1987. “The Fabrication of Meaning: Literary Interpretation in the United States, Great Britain, and the West Indies.” *American Journal of Sociology* 92 (5): 1077–1117.

Hanna, Alex. 2013. “Computer-Aided Content Analysis of Digitally Enabled Movements.” *Mobilization: An International Quarterly* 18(4): 367-88.

Hopkins, Daniel, and Gary King. 2010. “A Method of Automated Nonparametric Content Analysis for Social Science.” *American Journal of Political Science* 54 (1): 229–47.

Hopkins, Daniel, Gary King, Matthew Knowles, and Steven Melendez. 2013. *ReadMe: Software for Automated Content Analysis*. Version 0.99836. http://gking.harvard.edu/readme.

Jain, Anil K. 2010. “Data clustering: 50 Years Beyond K-Means.” *Pattern Recognition Letters*, *31*(8), 651–666.

Jurka, Timothy P., Loren Collingwood, Amber E. Boydstun, Emiliano Grossman and Wouter van Atteveldt. 2014. *RTextTools: Automatic Text Classification via Supervised Learning*. R package version 1.4.2. https://cran.rproject.org/web/packages/RTextTools/index.html

Kellstedt, Paul M. 2000. “Media Framing and the Dynamics of Racial Policy Preferences.” *American Journal of Political Science* 44(2): 239-255.

King, Gary, Jennifer Pan, and Margaret Roberts. 2013. “How Censorship in China Allows Government Criticism but Silences Collective Expression.” *American Political Science Review* 107 (2): 1–18.

Krippendorff, Klaus. 1970. “Bivariate Agreement Coefficients for Reliability of Data.” *Sociological Methodology* 2: 139–50.

Lancichinetti, Andrea, M. Irmak Sirer, Jane X. Wang, Daniel Acuna, Konrad Körding, and Luís A. Nunes Amaral. 2015. “High-Reproducibility and High-Accuracy Method for Automated Topic Classification.” *Physical Review X* 5 (1): 011007.

Lang, Ken. 1995. “NewsWeeder: Learning to Filter Netnews.” Pp. 331-339 in *Proceedings of the 12th International Machine Learning Conference*. Morgan Kaufmann Publishers Inc.

Lee, Monica and John Levi Martin. 2015. “Coding, Culture, and Cultural Cartography.” *American Journal of Cultural Sociology* 3: 1-33.

Levay, Kevin. 2013. “A Malignant Kinship: The Media and Americans’ Perceptions of Economic and Racial Inequality.” Unpublished paper, Northwestern University Department of Political Science.

Loughran, Tim, and Bill McDonald. 2011. “When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks.” *The Journal of Finance* 66 (1): 35–65.

Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. “The Stanford CoreNLP natural language processing toolkit." Pp. 55-60 in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics.

Martin, John Levi. 2000. "What Do Animals Do All Day?: The Division of Labor, Class Bodies, and Totemic Thinking in the Popular Imagination." *Poetics* 27 (2–3): 195–231.

McCall, Leslie. 2013. *The Undeserving Rich: American Beliefs about Inequality, Opportunity, and Redistribution.* Cambridge: Cambridge University Press.

Mikhaylov, Slava, Michael Laver, and Kenneth R. Benoit. 2012. "Coder Reliability and Misclassification in the Human Coding of Party Manifestos." *Political Analysis* 20: 78-91.

Mikolov, Thomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space." In *Proceedings of Workshop at International Conference on Learning Representations.*

Milkman, Ruth, Stephanie Luce, and Penny Luce. 2013. *Changing the Subject: A Bottom-Up Account of the Occupy Wall Street Movement in New York City*. New York: The Murphy Institute, City University of New York.

Mische, Ann, and Philippa Pattison. 2000. "Composing a Civic Arena: Publics, Projects, and Social Settings." *Poetics* 27 (2): 163–194.

Mohr, John W. 1998. "Measuring Meaning Structures." Annual Review of Sociology 24 (1): 345–70.

Mohr, John W., and Vincent Duquenne. 1997. “The Duality of Culture and Practice: Poverty Relief in New York City, 1888-1917.” *Theory and Society* 26 (2/3): 305–56.

Mohr, John W., Robin Wagner-Pacifici, Ronald L. Breiger, and Petko Bogdanov. 2013. “Graphing the Grammar of Motives in National Security Strategies: Cultural Interpretation, Automated Text Analysis and the Drama of Global Politics.” *Poetics* 41 (6): 670–700.

Nardulli, Peter F., Scott L. Althaus, and Mathew Hayes. 2015. “A Progressive Supervised-learning Approach to Generating Rich Civil Strife Data.” *Sociological Methodology* 45(1):145-83.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and É Duchesnay. 2011. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825−2830.

Pelleg, Dan, and Andrew W. Moore. 2000. “X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters.” In *Proceedings of the Seventeenth International Conference on Machine Learning*, 727–734. San Francisco: Morgan Kaufmann Publishers Inc.

R Core Team. 2014. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Reed, Isaac Ariail. 2015. “Counting, Interpreting and Their Potential Interrelation in the Human Sciences.” *American Journal of Cultural Sociology* 3 (3): 353–64.

"Reuters-21578 Test Collection." n.d. http://www.daviddlewis.com/resources/testcollections/reuters21578/. Accessed 09 March 2017.

Roberts, Margaret, Brandon Stewart, Dustin Tingley, and Edorardo M. Airoldi. 2013. "The Structural Topic Model and Applied Social Science." *Neural Information and Processing Society*.

Rousseeuw, Peter J. 1987. "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Computational and Applied Mathematics* 20: 53-65.

Schmidt, Benjamin M. 2012. "Words Alone: Dismantling Topic Models in the Humanities." *Journal of Digital Humanities* 2(1). http://journalofdigitalhumanities.org/2-1/words-alone-by-benjamin-m-schmidt/.

Spillman, Lyn. 2015. "Ghosts of Straw Men: A Reply to Lee and Martin." *American Journal of Cultural Sociology* 3 (3): 365–79.

Steinbach, Michael, George Karypis, and Vipin Kumar. 2000. "A Comparison of Document Clustering Techniques." In *KDD Workshop on Text Mining*.

Tausczik, Yla R., and James W. Pennebaker. 2010. "The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods." *Journal of Language and Social Psychology* 29 (1): 24–54.

Van Rijsbergen, C. J. 1979. *Information Retrieval*. London ; Boston: Butterworth-Heinemann.

Table 1. Evaluation metrics for each automated method.

Method (Coding Scheme)	Inequality/Relevant			Not Inequality/Irrelevant			Averaged Total			Time Trends		Support
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision[2]	Recall[2]	F1 Score[2]	Corr (2yr MA)	Correlation	N
(1) SML[1]	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Relevant vs. Irrelevant (A)	0.85	0.90	0.87	0.81	0.74	0.77	0.83 (.81-.86)	.84 (.81-.86)	.83 (.81-.86)	0.75	0.74	745
Inequality vs. Not Inequality (B)	0.73	0.60	0.84	0.80	0.88	0.84	0.78 (.74-.80)	.78 (.75-.80)	.78 (.74-.80)	0.69	0.63	745
Inequality vs. Economic vs. Irrelevant (C)	0.67	0.70	0.69	0.76	0.84	0.80	0.68 (.64-.71)	.69 (.65-.71)	.69 (.64-.71)	0.72	0.69	745
(2) Dictionary												
Levay-Enns (D)	0.91	0.25	0.40	0.83	0.99	0.91	0.85	0.84	0.80	0.42	0.59	1253
McCall (B)	0.48	0.84	0.61	0.86	0.52	0.65	0.73	0.63	0.64	0.66	0.44	1253
(3) Unsupervised ML (UML)												
Topic Model vs. Explicit (D)	0.63	0.45	0.53	0.87	0.86	0.87	0.75	0.67	0.71	0.58	0.68	1253
K-Means vs Explicit (D)	0.88	0.14	0.24	0.99	0.81	0.89	0.94	0.48	0.63	N/A	N/A	1253

[1]SML values are for test set only [2]Parentheses contain range across the 25 test/training set pairs

Coding Scheme A: Relevant (Explicit, Implicit, General Economic) Irrelevant (Irrelevant)

Coding Scheme B: Inequality (Explicit, Implicit) Not Inequality (General Economic, Irrelevant)

Coding Scheme C: Inequality (Explicit, Implicit) Economic (General Economic) Irrelevant (Irrelevant)

Coding Scheme D: Inequality (Explicit) Irrelevant (Implicit, General Economic, Irrelevant)

Table 2. Top words for inequality topics in k-means and STM models.

Clustering: Inequality Group[1]	Topic Model: Inequality Group[2]
incom	percent
percent	incom
class	american
rich	famili
middl	inequ
poor	class
famili	wage
america	poor
gap	top
top	rich
colleg	econom
live	middl
averag	earn
increas	increas
gain	household
wage	gap
school	year
educ	one
today	colleg
social	averag

[1]K-means model with 30 clusters; most frequent words.

[2]Structural Topic Model with 60 topics; highest weighted words.

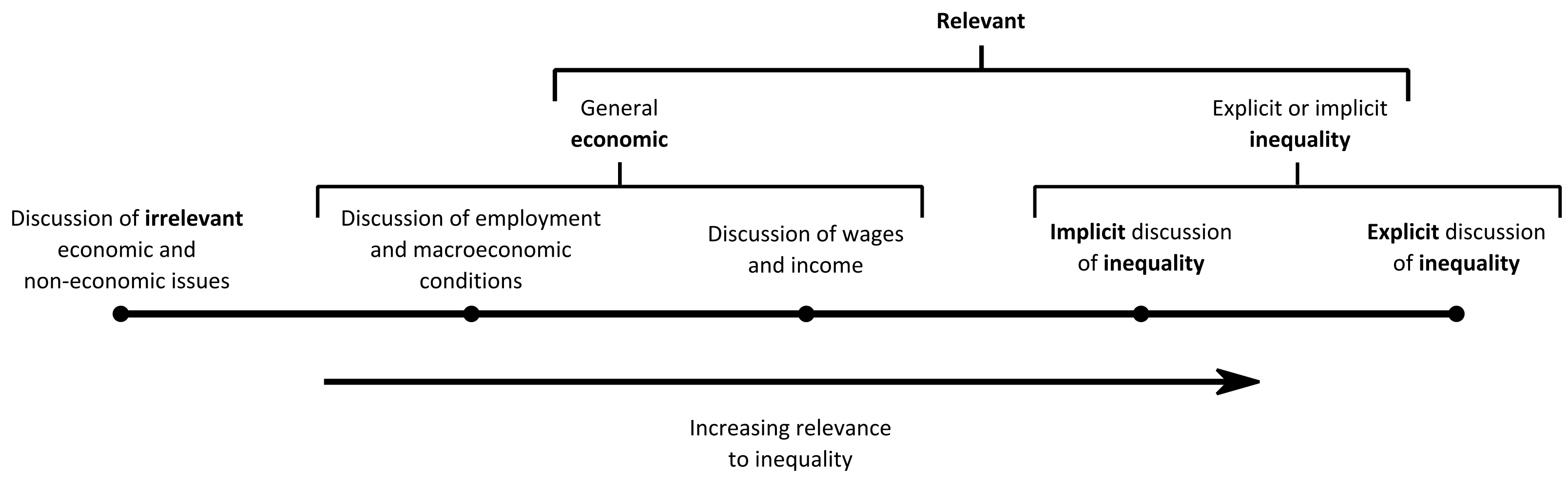
Relevant
General economic
Explicit or implicit inequality
Discussion of irrelevant economic and non-economic issues
Discussion of employment and macroeconomic conditions
Discussion of wages and income
Implicit discussion of inequality
Explicit discussion of inequality
Increasing relevance to inequality

Figure 1. Categorization of hand-coded articles.

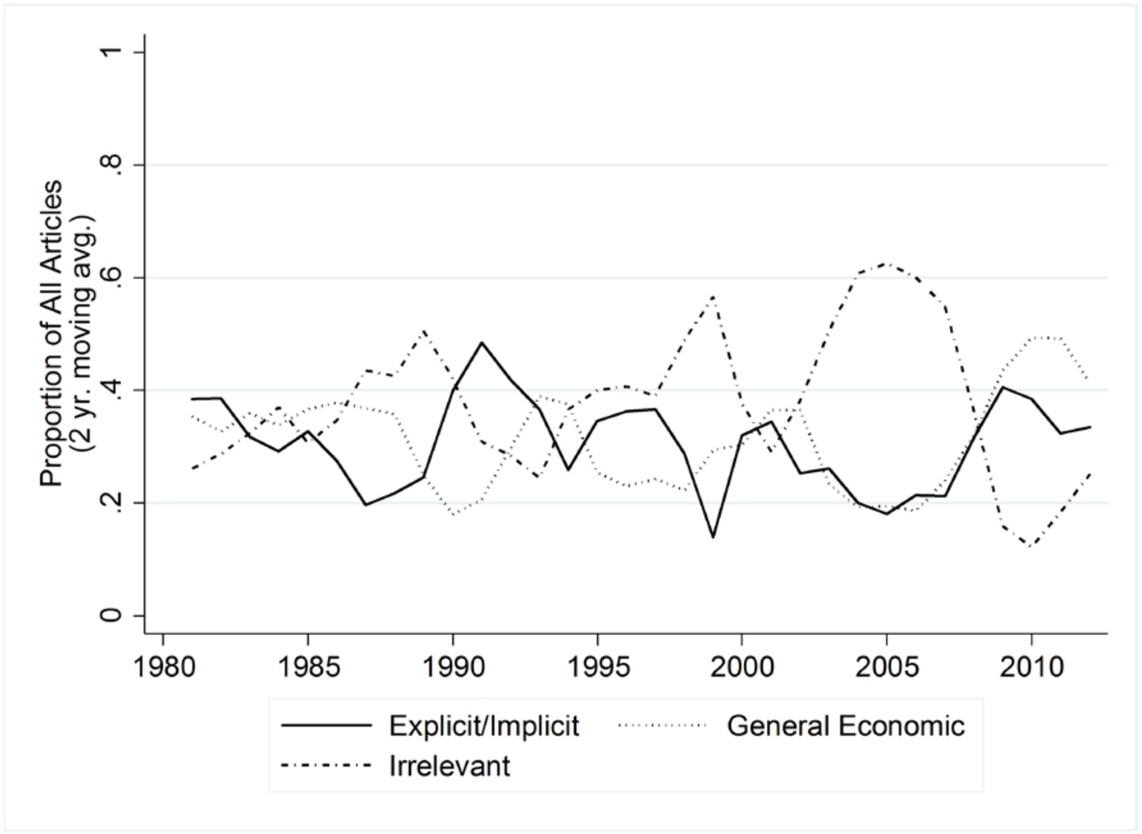

Figure 2. Trends in preferred three code scheme of hand-coded articles
(Explicit/Implicit Inequality vs. General Economic vs. Irrelevant categories).

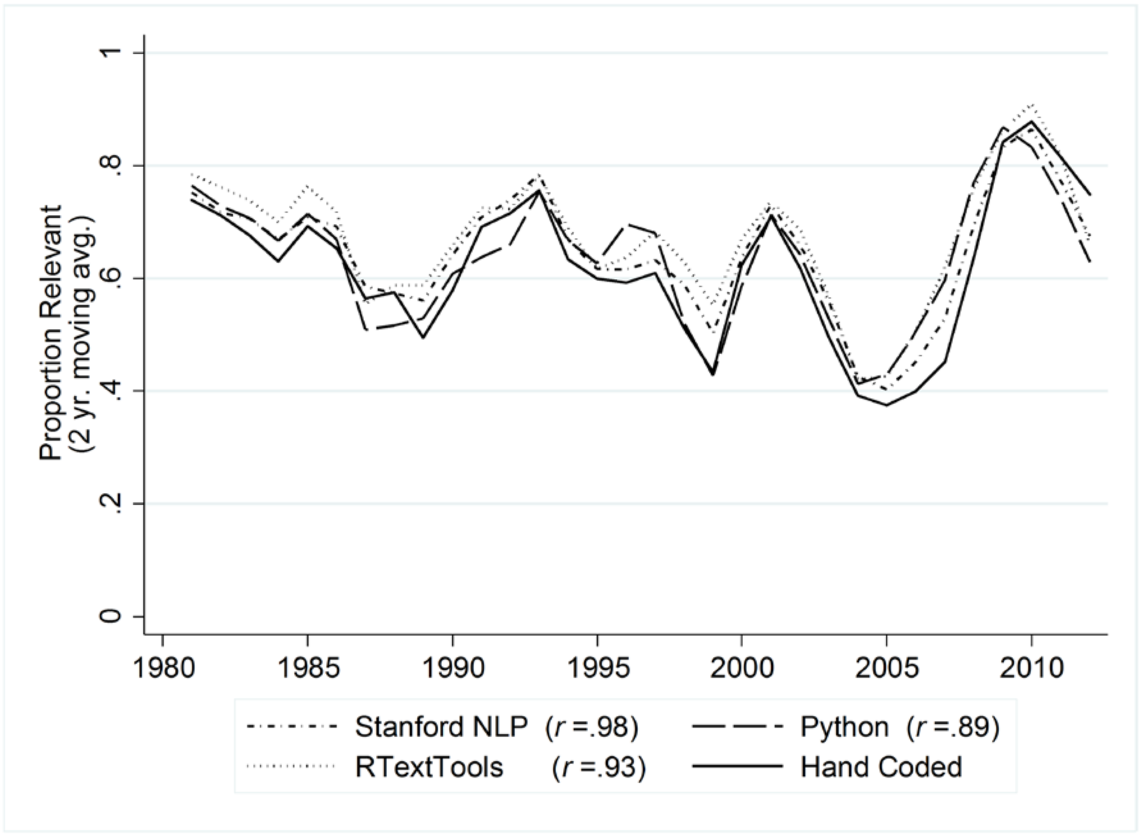

Figure 3. Trends in SML analysis of hand-coded articles for Relevant vs. Irrelevant binary scheme (combined Relevant substantive categories vs. Irrelevant category; combined Relevant substantive categories shown).

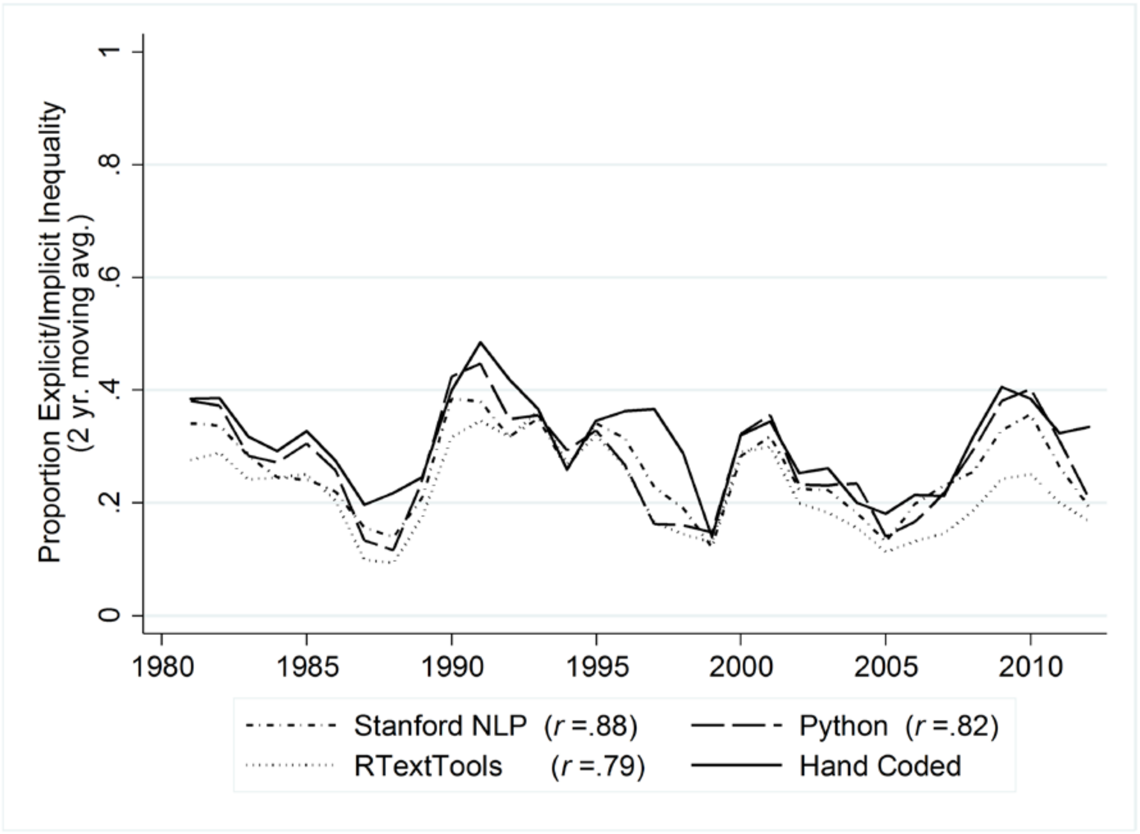

Figure 4. Trends in SML analysis of hand-coded articles for Inequality vs. Not Inequality binary scheme (Explicit/Implicit Inequality vs. all other categories; Explicit/Implicit Inequality category shown).

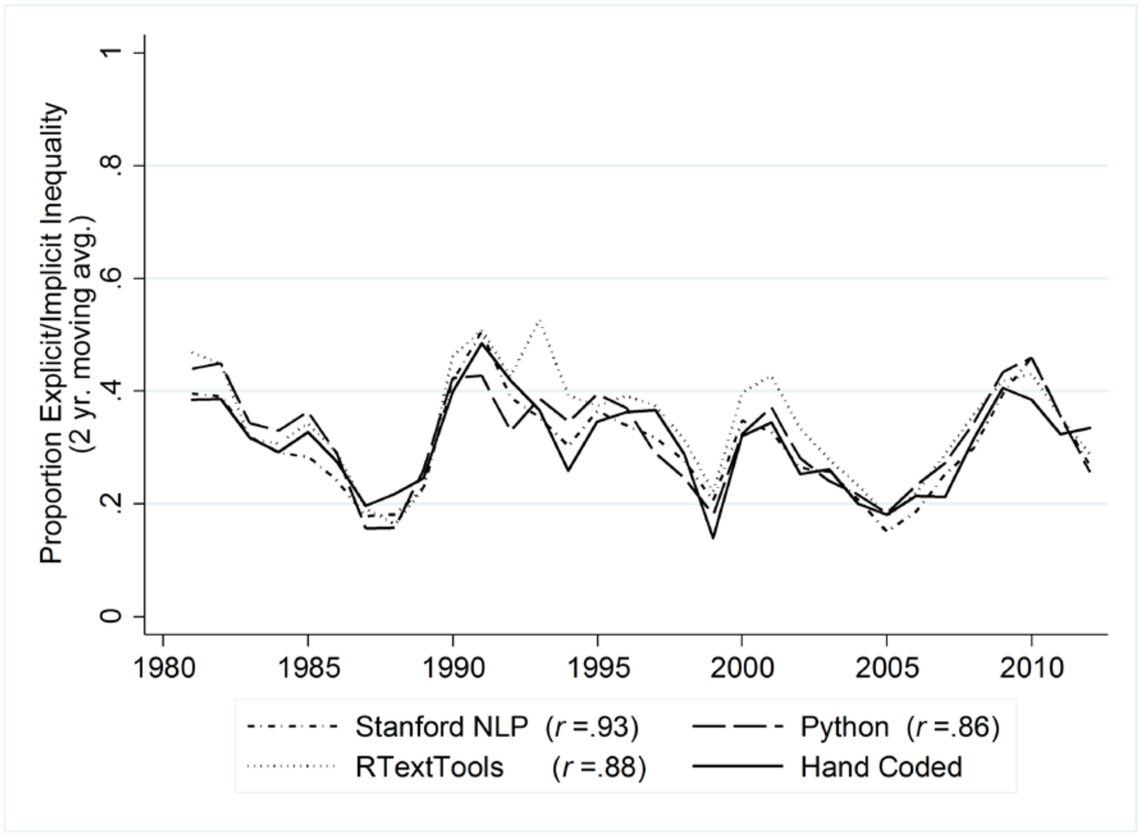

Figure 5. Trends in SML analysis of hand-coded articles for preferred three code scheme (Explicit/Implicit Inequality vs. General Economic vs. Irrelevant categories; Explicit/Implicit Inequality category shown).

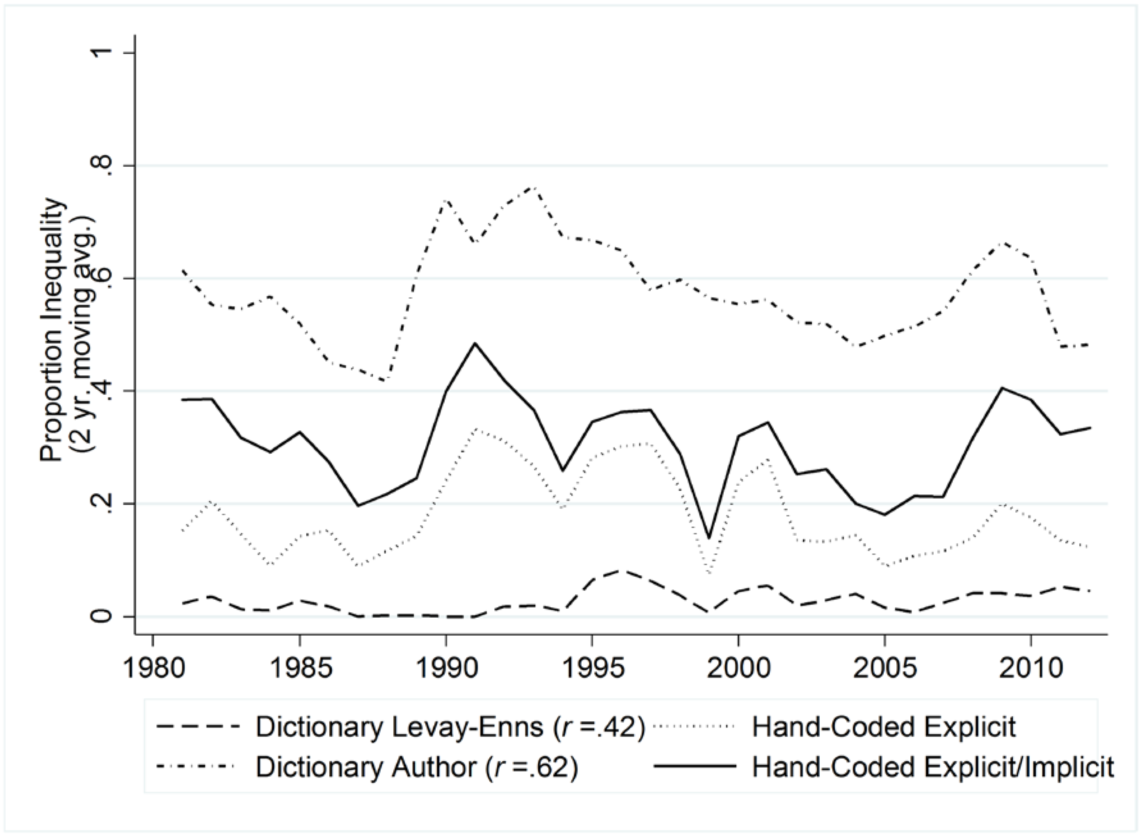

Figure 6. Trends in dictionary analysis of hand-coded articles (compare Levay-Enns to hand-coded Explicit Inequality trend; compare McCall to hand-coded Explicit/Implicit Inequality trend)**.**

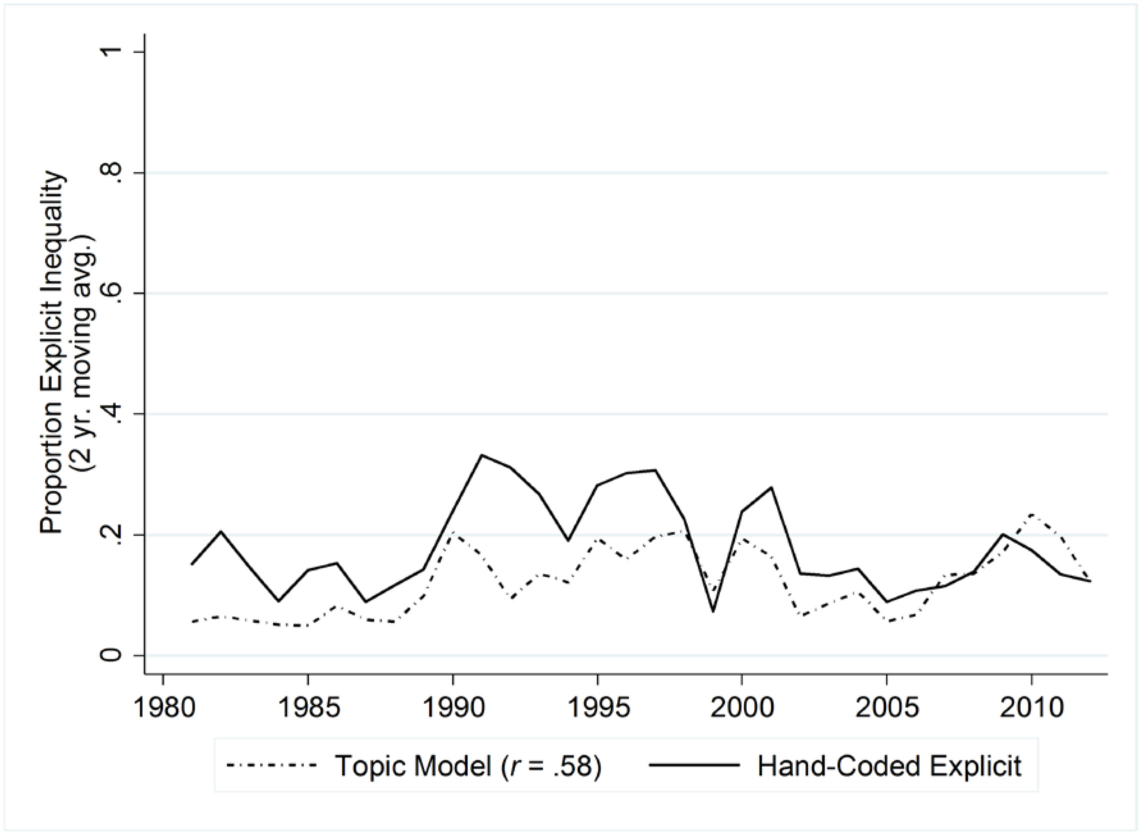

Figure 7. Trends in UML analysis of hand-coded articles (Explicit Inequality category shown).

Appendix A. Definition of media coverage of economic inequality.

(1) Type of inequality

(a) Labor market inequality, unaffected directly by government taxes and transfer programs.

(b) Total income inequality, including the contribution of family formation patterns and government taxes and transfer programs.

(2) Causes and policy solutions associated with inequality

(a) Multiple causes of labor market inequality include skill-biased technological change, globalization, immigration, and decline in wage setting institutions and social norms, and associated policy solutions.

(b) Multiple causes of total income inequality include declines in social spending on low income individuals and families (e.g., unemployment benefits and means-tested programs) and declines in progressive taxation, and associated policy solutions.

(3) Social class groups

(a) Categories of workers (typically large groupings) identified in economic literature on rising earnings and income inequality, including skill groups, union workers, minimum wage workers, immigrants, executives, the rich and wealthy (with incomes from both labor and capital).

(b) Categories of individuals (typically large groupings) identified in broader literature on inequality, in popular culture, and by social program receipt, including the poor, unemployed, middle class, blue collar workers and white collar workers.

(4) Relational or comparative language connecting social class groups

(a) *Explicit* comparisons or relations among two or more hierarchically ordered social class groups, including relational language such as inequalities, gaps, differences, disparities, and divides or comparative language such as winners, losers, primary beneficiaries, etc.

(b) Presence of two or more hierarchically ordered social class groups without explicit comparative or relational language among them but with *implicit* reference to different and unequal economic circumstances or redistributive processes.

Appendix B. List of keywords in the Enns et al. (2015) and Levay (2013) dictionaries.

concentration of income
concentration of wealth
distribution of income
distribution of incomes
distribution of wealth
distributional
economic disparities
economic disparity
economic distribution
economic divide
economic equality
economic inequalities
economic inequality
economic insecurity
egalitarian income
egalitarian wealth
employment insecurity
equal economic
outcomes
equal income
equal pay
equal wage
equal wealth
equality of economic
outcomes
equality of income
equality of incomes
equality of wealth
equalize income
equalize incomes
equalize wealth
equalizing income
equalizing incomes
equalizing wealth
equitable distribution
equitable income
equitable wealth
equity in income
equity in incomes
equity in wealth
equity of income
equity of incomes
equity of wealth
gini
income concentration
income decile
income difference
income differential
income disparities
income disparity
income distribution
income divide
income equality
income equalization
income gap
income inequalities
income inequality
income inequity
income polarization
income quintile
income redistribution
income stratification
inegalitarian income
inegalitarian wealth
inequality of economic
outcomes
inequality of income
inequality of incomes
inequality of wealth
inequitable distribution
inequitable income
inequitable wealth
inequity of incomes
inequity of wealth
job insecurity
maldistribution
pay difference
pay differential
pay divide
pay equality
pay gap
pay inequality
redistribution of income
redistribution of
incomes
redistribution of wealth
top incomes
unequal economic
outcomes
unequal economy
unequal income
unequal incomes,
unequal wealth
unequal pay
unequal wage
unequal wealth
uneven distribution
wage difference
wage differential
wage disparities
wage disparity
wage divide
wage equality
wage gap
wage inequalities
wage inequality
wealth concentration
wealth difference
wealth differential
wealth disparities
wealth disparity
wealth distribution
wealth divide
wealth equality
wealth equalization
wealth gap
wealth inequalities
wealth inequality
wealth inequity
wealth polarization
wealth redistribution
wealth stratification

Appendix C. List of keywords and instructions for dictionary based on McCall (2013).

GROUP I TERMS: EXPLICIT DISTRIBUTIVE LANGUAGE – 1 AND (2 OR 3)

(1) Distribution

inequality	differential	equity
equality	difference	inequity
unequal	disparity	inequitable
distribution	polarization	egalitarian
gap	dualism	inegalitarian
divide	dual society	concentration

AND **((**

(2) Income/wealth (private):

economic	compensation	bonus
wage	benefit	investment
income	wealth	tax
earning	asset	stock **)**
pay	stock return	

OR **(**

(3) Income/wealth (govt):

cash transfer	Medicaid	social spending
non cash transfer	Medicare	social program
welfare	housing assistance	redistribution
food stamp	public housing	redistributive **))**
unemployment	earned income tax	
insurance	credit	
social security	EITC	

GROUP II: IMPLICIT DISTRIBUTIVE LANGUAGE – 1 AND (2 OR 3)

(1) Social class groups (terms from at least two groups):

top	professional	manager
rich	white collar	1%
executive	high income	middle class
CEO	high wage	blue collar
affluent	high skill	middle income
wealthy	investor	median wage
wealthier	upper class	median earner
wealthiest	employer	average wage

average earner	union	99%
poor	low income	unemployed
worker	lower class	
minimum wage	bottom	
worker	low wage	

AND **((**

(2) Income/wealth (private):

economic	compensation	bonus
wage	benefit	investment
income	wealth	tax
earning	asset	stock **)**
pay	stock return	

OR **(**

(3) Income/wealth (govt):

cash transfer	Medicaid	social spending
non cash transfer	Medicare	social program
welfare	housing assistance	redistribution
food stamp	public housing	redistributive **))**
unemployment	earned income tax	
insurance	credit	
social security	EITC	

Notes:
In all cases, to allow for plurals, we allow for optional trailing "-ies" for terms ending in "y" preceded by a consonant; trailing "-es" for terms ending in "ch", "s", or "x"; and trailing "-s" for all other words. An article must contain at least one listed term/boolean expression to satisfy a given category.

Appendix D. Description of supervised machine learning programs.

We used three general-use programs to conduct our supervised machine learning (SML) analyses described in the main text: Stanford Classifier, RTextTools, and scikit-learn. (Although not included in our discussion and results, we also conducted extensive analyses using the ReadMe program, which we discuss at the end of this appendix). All three programs are freely downloadable. None of the three requires knowledge of the mathematics behind machine learning algorithms, and we found them all to be easy to use for an analyst with previous exposure to computer programming languages. Moreover, all three produced comparable results in our analysis. In sum, all three are good options for researchers interested in using SML for text analysis.

All three programs are actively supported and updated, but scikit-learn has the largest community of developers and most frequent update schedule. Researchers with experience in one of the languages underlying these programs will find that program most accessible: Stanford Classifier is written in Java, RTextTools in R, and scikit-learn in Python. For those without experience in any of these languages, we have provided some comparisons that may help you decide which is right for you. Also, in the "Resources" section below, we have provided links to some of the most helpful documentation and tutorials on each of these programs, and in Appendix Table D1, we provide the settings we used to run our analyses.

Installation

Stanford Classifier requires installation of Java, which can be installed from https://java.com/en/download/help/index_installing.xml. Stanford Classifier can then be downloaded at http://nlp.stanford.edu/software/classifier.html. Users need only unzip the downloaded directory, and the program is ready to use. More advanced uses may require familiarity with a command line interface.

RTextTools requires the installation of the R statistical computing program. R installers for Mac, Linux, and Windows can be downloaded from https://cran.r-project.org/. Users may also want to install the popular R IDE RStudio, which is free from https://www.rstudio.com/products/rstudio/download/. RStudio includes features such as code completion, syntax highlighting, and integrated function documentation. After installing R, users can install the RTextTools package from within the R (or RStudio) graphical user interface, using either drop-down menus or the "install.packages" command.

Like RTextTools, scikit-learn is an add-on package, and requires installation of a Python programming language interpreter, unless your operating system comes packaged with such an interpreter. Although there are many ways to install a Python interpreter and Python packages, one good option is the Anaconda distribution, which includes the most popular packages used by scientists and researchers, with installers downloadable from https://www.continuum.io/downloads. This option helps avoid the extra step of installing scikit-learn after installing Python.

Data Input

When used as a standalone program, Stanford Classifier does not provide tools to reshape data into the required format of a tab- or comma-separated text file. It also requires that training set and test set data be contained in separate files, so preprocessing in some other program will likely be necessary. A properties file can be used to provide parameters for the analysis. Once your data is properly formatted, however, Stanford Classifier reads, processes, and analyzes your data all in one command.

R can read in data in a wide variety of formats, including the proprietary formats of other statistical software such as Stata. Once your data is read into R, RTextTools requires the user to point to the data columns containing the text to be analyzed and the content codes, and to specify which texts are in the training and test sets. These tasks can be performed with a basic understanding of column- and row-subsetting in the R language.

Python can also read data in many formats. We recommend using the package “pandas”, included in the Anaconda distribution, which provides utilities for reading tabular data and for column- and row-subsetting. Using pandas, users can easily point scikit-learn to the columns and rows that contain text and label information for the training and test sets.

Computing Requirements

Our analysis with Stanford Classifier was run in a single command, with options specified in a simple properties file. Each analysis run only took a few seconds.

Some of RTextTools’s built-in algorithms have excessive memory and computing requirements. In some cases, training a classifier in RTextTools consumed more than 16 GB of memory, forcing us to use a batch computing server. These models could also take several minutes of computing time, which slows the pace of analysis when testing different model options.

Scikit-learn rivals Stanford Classifier in speed, with the training process taking only a few seconds.

Customizability and Extensibility

Stanford Classifier only offers built-in support for a couple SML algorithms, and has minimal built-in options for preprocessing tasks such as word stemming, term frequency-inverse document frequency (TFIDF) weighting, and exclusion of common and rare words. RTextTools offers built-in access to nine algorithms, ensemble classification options that automatically combine results from multiple algorithms, and a fair range of preprocessing options, including TFIDF weighting. Scikit-learn offers built-in access to many algorithms, and includes the widest range of preprocessing options.

RTextTools and scikit-learn are both part of robust environments for all kinds of data analysis, and thus offer more options for data manipulation and additional analyses than Stanford Classifier.

Bugs and Configuration Issues

While using RTextTools and Stanford Classifier, we ran into small problems that required some debugging of application code (RTextTools) or editing of a helper file to point to the actual location of application files (Stanford Classifier). We had no such problems with scikit-learn. However, problems like this can occur in using any software, and will depend on the configuration of the user's system and the details of their dataset.

Accessibility of the Language

For basic use, Stanford Classifier requires almost no knowledge of any programming language. Using RTextTools and scikit-learn involves learning a few commands, but this is made easier by the availability of tutorials (see links below under "Resources"). For those with experience in object-oriented programming languages, Python and scikit-learn will likely be easier to pick up than RTextTools. For those whose programming experience is only in other statistical programs, RTextTools and R will likely allow you to get up and running more quickly.

Resources

Stanford Classifier
- Basic tutorial: http://nlp.stanford.edu/wiki/Software/Classifier
- List of analysis options: http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/classify/ColumnDataClassifier.html

RTextTools
- Journal article about the package, which walks through a typical analysis: https://journal.r-project.org/archive/2013-1/collingwood-jurka-boydstun-etal.pdf

scikit-learn
- Tutorial on text analysis: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

ReadMe

ReadMe is a supervised machine learning program, but it differs from the other computer-assisted methods used in our analysis in that it does not classify individual texts (Hopkins, King, Knowles, and Melendez 2013). Instead, ReadMe directly estimates the incidence of hand-coded categories in the corpus as a whole. For this reason, we could not assess agreement between the ReadMe results and hand coding, using metrics such as recall and precision, as we did with the other computer-assisted methods. However, ReadMe's estimates of the overall incidence of hand-coded categories in the test set showed a relatively high level of agreement with the actual hand coding of test-set articles. For two binary coding schemes, the average error across 25 random training-set / test-set configurations of ReadMe's estimate of the percentage of articles in each category relative to the hand-coded percentage was just 0.9% and 1.9% respectively, although the difference was as large as 14.9% for one configuration. Results for the three-code scheme were similar, with average differences of 0.4%, 1.7%, and 1.2%, and a maximum difference of 11.6%. In sum, ReadMe did an accurate job of replicating our hand-coding results, especially given that our hand-coding scheme does not strictly follow the recommendations of ReadMe's authors (specifically, our categories were not designed as mutually-exclusive, internally coherent topics, but rather as indicators of the presence or absence of one particular topic, so that residual categories such as "irrelevant" encompass a wide variety of topics).

Appendix Table D1. Settings used for supervised machine learning analyses.

	Stanford Classifier	RTextTools	Scikit-learn
Algorithm(s)	Maximum entropy	Ensemble (boosting, bootstrap aggregating, decision tree, generalized linear model, maximum entropy, neural networks, random forests, supervised latent Dirichlet allocation, support vector machine)	Linear support vector machine
Min. number of times word must appear in document to be counted	1	1	1
Max. number of times word can appear in document before being ignored	No maximum	No maximum	No maximum
Min. word length	1	1	1
Max. word length	No maximum	No maximum	No maximum
N-gram length	1	1	1
Remove numbers?	No	No	No
Remove punctuation?	Yes	Yes	Yes
Stem words?	No	No	No
Remove words appearing in *X*% or more documents?	No	No	Yes, 95%
Remove words appearing in *X*% or fewer documents?	No	Yes, 1%	No
Remove common English words (stop words)?	Yes	Yes	No
Convert all text to lowercase?	Yes	Yes	Yes
Strip whitespace?	Yes	Yes	No
Weighting	Term frequency	TFIDF	TFIDF

Appendix Table D2. Evaluation metrics for all three SML programs.

Method (Coding Scheme)	**Inequality/Relevant**			**Not Inequality/Irrelevant**			**Averaged Total**			**Time Trends**		**Support**
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision[2]	Recall[2]	F1 Score[2]	Corr (2yr MA)	Correlation	N
(1) Python scikit-learn[1]	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Relevant vs. Irrelevant (A)	0.85	0.90	0.87	0.81	0.74	0.77	0.83 (.81-.86)	.84 (.81-.86)	.83 (.81-.86)	0.75	0.74	745
Inequality vs. Not Inequality (B)	0.73	0.60	0.84	0.80	0.88	0.84	0.78 (.74-.80)	.78 (.75-.80)	.78 (.74-.80)	0.69	0.63	745
Inequality vs. Economic vs. Irrelevant (C)	0.67	0.70	0.69	0.76	0.84	0.80	0.68 (.64-.71)	.69 (.65-.71)	.69 (.64-.71)	0.72	0.69	745
(2) RTextTools[1]												
Relevant vs. Irrelevant (A)	0.83	0.95	0.88	0.89	0.68	0.77	0.86 (.83-.88)	.82 (.77-.83)	.84 (.80-.85)	0.79	0.77	745
Inequality vs. Not Inequality (B)	0.82	0.46	0.59	0.76	0.94	0.84	0.79 (.74-.80)	.70 (.66-.74)	.74 (.69-.77)	0.65	0.64	745
Inequality vs. Economic vs. Irrelevant (C)	0.67	0.76	0.70	0.75	0.84	0.80	0.68 (.65-.71)	.67 (.69-.75)	.67 (.65-.70)	0.78	0.75	745
(3) Stanford Classifier[1]												
Relevant vs. Irrelevant (A)	0.83	0.73	0.78	0.85	0.91	0.88	0.84 (.83-.87)	.82 (.79-.84)	.83 (.80-.85)	0.92	0.90	745
Inequality vs. Not Inequality (B)	0.80	0.91	0.85	0.77	0.57	0.66	0.78 (.75-.80)	.74 (.68-.76)	.75 (.69-.78)	0.79	0.75	745
Inequality vs. Economic vs. Irrelevant (C)	0.72	0.69	0.70	0.76	0.83	0.80	0.68 (.65-.70)	.68 (.65-.70)	.68 (.65-.70)	0.86	0.82	745

[1]SML values are for test set only [2]Parentheses contain range across the 25 test/training set pairs

Coding Scheme A: Relevant (Explicit, Implicit, General Economic) Irrelevant (Irrelevant)

Coding Scheme B: Inequality (Explicit, Implicit) Not Inequality (General Economic, Irrelevant)

Coding Scheme C: Inequality (Explicit, Implicit) Economic (General Economic) Irrelevant (Irrelevant)