Crowd-sourced data coding for the social sciences:
massive non-expert coding of political texts[*]

Kenneth Benoit
London School of Economics
and Trinity College, Dublin

Drew Conway
New York University

Michael Laver
New York University

Slava Mikhaylov
University College London

Abstract

A large part of empirical social science relies heavily on data that are not observed in the field, but are generated by researchers sitting at their desks, raising obvious issues of both reliability and validity. This paper addresses these issues for a widely used type of coded data, derived from the content analysis of political text. Comparing estimates derived from multiple "expert" and crowd-sourced codings of the same texts, as well as other independent estimates of the same latent quantities, we investigate whether we can analyze political text in a reliable and valid way using the cheap and scalable method of crowd sourcing. Our results show that, contrary to naive preconceptions and reflecting concerns often swept under the carpet, a set of *expert* coders is also a crowd. We find that deploying a crowd of *non-expert* coders on the same texts, with careful specification and design to address issues of coder quality, offers the prospect of cheap, scalable and *replicable* human text coding. Even as computational text analysis becomes more effective, human coding will always be needed, both to validate and interpret computational results and to calibrate supervised methods. While our specific findings here concern text coding, they have implications for all expert coded data in the social sciences.

KEYWORDS: text coding; crowd sourcing; expert-coded data; reliability; validity

* Prepared for the third annual *New Directions in Analyzing Text as Data* conference at Harvard University, 5-6 October 2012. A very preliminary version of this paper was presented at the 70th annual *Conference of the Midwest Political Science Association*, Chicago, 12-15 April 2012. We thank Joseph Childress at CrowdFlower for assisting with the setup of the crowd-sourcing platform. This research was supported financially by the European Research Council grant ERC-2011-StG 283794-QUANTESS.

1. INTRODUCTION

A large part of empirical social science relies heavily on data that are not observed in the field, but instead are generated by researchers sitting comfortably at their desks. These researchers ingest and digest information from primary and secondary sources that for the most part contain qualitative information. Using rules or conventions that may or may not be explicit, researchers then use their "expert judgments" to turn the content of qualitative sources into the hard numbers that populate key variables.[1] What they are doing is generating coded data. Well-known and widely-used coded data in political science include, to name but a few prominent examples: [2]

- *Polity* scores, which rate countries on a 21 point scale "ranging from -10 (hereditary monarchy) to +10 (consolidated democracy)"[3];
- *Correlates of War* data that include indicators, for example, of conflict scope and intensity[4];
- *Comparative Parliamentary Democracy* data that include indicators, for example, of the "number of inconclusive bargaining rounds" in forming a government, or whether the fall of the previous government was "conflictual"[5];
- *Manifesto Project* data that comprise coded summaries of the policy content of party manifestos[6].

Fundamental questions about such coded data concern both validity and reliability. In relation to *validity*, we want to know whether direct observation of the latent quantity of interest would generate the same estimate. We typically address this question by using different indirect ways to estimate the same thing, feeling comforted when these generate the same substantive results. Below, we address the issue of *content validity* by comparing our estimates of party policy positions derived from expert and non-expert human codings of party manifestos with independent expert codings of the same manifestos and independent expert surveys of political scientists. Turning to *reliability*, we want to know whether different experts given the same coding task, or the same expert coding on different days, would generate the same set of codes. We can distinguish two different sources of inter-coder variation. First, assuming the latent quantity of interest has an objectively "true" but fundamentally unobservable value, with researchers observing only noisy realizations of this, it is likely that different coders, processing noise in different ways in search of the same underlying signal, will return different subjective scores. Second, and commonly in the social sciences, the latent quantity may itself be an

[1] They may also have had conversations with others – conversations that were, or could have been, transcribed into texts.
[2] There are many other examples of coded data, including: expert judgments on party policy positions of party positions (Benoit and Laver 2006; Hooghe et al. 2010; Laver and Hunt 1992); the democracy scores of Freedom House and corruption rankings by Transparency International.
[3] http://www.systemicpeace.org/polity/polity4.htm
[4] http://www.correlatesofwar.org/
[5] http://www.erdda.se/cpd/data_archive.html
[6] https://manifesto-project.wzb.eu/

essentially subjective construct (for example “level of democratization”, the “liberalness” of some policy position), subject to perfectly valid differences of interpretation by different coders. To uncertainty about measurements of a “certain” latent quantity in a noisy environment, we add uncertainly about the underlying quantity itself. For any coded latent quantity of interest, there is a vector of estimated scores, and there is a vector of estimates of the uncertainty of these scores – the latter decomposable into pure noise and fundamental uncertainty about the quantity being measured.

None of the canonical datasets we mention above estimate, or even discuss, uncertainty associated with their published scores. Coded data in the social sciences are thus typically reported as point measures with no associated estimate of uncertainty. Typically, this is for the prosaic reason that inter-coder (un)reliability can only be estimated in settings with multiple independent coders (or when we have a gold standard that contains the “truth”). The Manifesto Project (MP) is quite explicit that most manifestos are coded once, and only once, by a single coder. None of the other datasets we have mentioned reports, or explicitly takes account of, using multiple independent coders. The understandable if ignoble reason for this is cost. Generating the MP dataset, for example, involved massive effort and expense over decades by a team of dedicated researchers. The same can be said for the other datasets noted above. Despite the fact that distressingly low levels of inter-coder reliability have been found using the MP scheme in limited independent coding experiments (Mikhaylov et al. 2012), a proposal to recode each manifesto in the MP dataset ten times, using completely independent coders, would be a financial non-starter. While the MP data could *in theory* be recoded many times, *in practice* this is not going to happen. The constraint is research budget, not method. We tend not to have estimates of the uncertainty associated with key scores in canonical political science datasets, notwithstanding the fact we know such uncertainty surely exists, because we have not been able to afford multiple independent codings of the same latent quantity of interest – or, frankly, because we have preferred to spend our research budgets on more sexy things if funds might indeed have been available for multiple coding. As a result, while we do not know these canonical datasets fail the replication standard, we do not know they pass it.

Cheap and convenient platforms for crowd-sourced data coding now offer a radical new potential solution to this problem. Unlike classical data coding in the social sciences, which relies ultimately on the wisdom of a few experts, crowd sourcing relies on the wisdom of many non-experts. The core intuition is that, as anyone who has ever coded data will know, data coding is “grunt” work. It is boring, repetitive and dispiriting precisely because the ideal of the researchers employing the coders is – and certainly *should* be – that different coders will typically make the same coding decisions when presented with the same source information. Paradoxically, therefore, “expert” coders are not expected to unleash their unbridled expertise on a coding project and are positively discouraged from rendering idiosyncratic judgments or exercising any creativity whatsoever. The ideal is coder uniformity. One way to generate coder uniformity is to employ non-expert coders and give these a set of precise and simple instructions about how to complete the coding task. By analogy with the production of automobiles, skilled craftsmen who deeply understand what they are doing can build automobiles – but so can unskilled production line workers if the job is broken down into a set of clear and simple tasks.

Because crowd-sourcing requires jobs such as text coding to be broken down into a set of simple and very explicitly described tasks that can be understood in the same way by a wide variety of workers, the coding process is more precisely specified than for a typical expert coding project. It is therefore, at least in principle, more *replicable*. Because crowd-sourcing is so much cheaper than expert coding, many more codings can be generated for the same research budget. This compensates for the possibility that coders in the crowd are more prone to random error than expert coders. Provided coders in the crowd are not systematically biased in relation to the "true" value of the latent quantity we seek to estimate, the mean coding, even of erratic coders, will converge on this true value as the number of coders increases. Quickly and cheaply collecting multiple independent codings, we can generate distributions of scores for latent variables of interest, allowing us to *estimate the uncertainties* associated with our point measures. Because experts are axiomatically in short supply while members of the crowd are not, crowd-sourced solutions offer a straightforward and *scalable* method for the generation of *replicable* coded data in the social sciences. The reliability of a given set of estimates can be improved, or the range of cases to which they apply can be expanded, simply by buying more crowd-sourced coding.[7]

Here, we report the results of experiments in the crowd-sourced coding of the policy content of party manifestos. We proceed as follows. In section 2, we review the theory and practice of crowd-sourced coding using platforms such as Mechanical Turk and CrowdFlower. In section 3, we design a coding experiment that compares expert and crowd-sourced codings of the same documents using the same coding categories, as well as the effects of various modifications to procedures for classical expert coding that might improve coding quality more generally, but particularly the quality of crowd-sourced data coding. In section 4 we review issues for researchers that arise when running a crowd-sourced coding project distributed globally on the internet. In section 5 we report results of our coding experiments and in section 6 we conclude with recommendations for the future deployment of crowd-sourced data coding in the social sciences. While the specific application we discuss concerns crowd-sourced coding of party manifestos, our results apply to text coding more generally. Furthermore, we remind readers that almost all coded data at least implicitly involves coding text, since text provides most of the primary and secondary source material for all data coding.

[7] While there is some theoretical upper bound to this, the CrowdFlower platform we deploy in this research offers access to millions of potential coders (www.crowdflower.com).

2. HARVESTING THE WISDOM OF CROWDS

The term crowdsourcing was first coined by Jeff Howe in a *Wired* magazine article (Howe 2006) and further developed in Howe (2008). The original idea can be traced back at least to Sir Francis Galton (1907) who noticed that an average of a large number of individual judgments by fair-goers of the weight of an ox is close to the true answer and, importantly, closer than any of the individual judgments (for a general introduction see Surowiecki 2004). Crowdsourcing has emerged as a paradigm for applying human intelligence to problem-solving on a massive scale. Crowdsourcing systems are now widely used for various data-processing tasks such as image classification, video annotation, form data entry, optical character recognition, translation, recommendation, and proofreading. Crowdsourcing is becoming a popular tool in social sciences (Bohannon 2011), particularly as a cheap alternative to traditional experimental studies (e.g. Lawson et al. 2010; Horton et al. 2011; Paolacci et al. 2010; Mason and Suri 2012).

As with any other empirical research, the internal and external validity of crowdsourced data are of major importance. Recent studies in political science (Berinsky et al. 2012a), economics (Horton et al. 2011) and general decision theory (Paolacci et al. 2010; Goodman et al. forthcoming), report some differences in the demographics of crowdsourced and traditional subjects in behavioral research, but find high levels of external validity for crowdsourcing methods.[8] The *external* validity of the “subject pool” is less relevant to our use of crowdsourcing for data coding. We need not concern ourselves with whether our coders represent some wider human population. We are perfectly happy if they are completely unrepresentative of the population at large, as long as they code data in a reliable and valid way. In this sense data coding, as opposed to running online experiments, represents a canonical use of crowdsourcing as described by Sir Francis Galton. This does, however, raise the importance of minimizing threats to *internal* validity of crowdsourcing.

The studies mentioned above also focused on internal validity, particularly the comparison between the quality of data produced by crowdsourcing and more traditional data generation techniques (Horton et al. 2011; Paolacci et al. 2010; Berinsky et al. 2012a). While a general consistency of crowdsourcing data with those derived from previous results has been shown, these studies emphasize the necessity to ensure coder quality (see also Kazai 2011). In general, all human generation of data requires some level of expertise. At one extreme, a single human coder (expert) with infinite knowledge and consistency would be capable of generating a dataset that is both perfectly valid and completely reliable. A much more realistic scenario, however, is that each human coder combines reliable expertise on some aspects of the problem at hand, with ignorance or unreliability on other aspects of the problem. Our hope in harvesting the wisdom of crowds in data coding is that the procedure will exploit the knowledge and expertise of our human coders, while remaining relatively unaffected by their relative lack of expertise on areas in which they are less knowledgeable or reliable. Several empirical studies have explored this

[8] These use Amazon’s Mechanical Turk as a crowdsourcing platform

issue and found that, while a single expert typically produces more reliable data, this performance can be matched and sometimes even improved, at much lower cost, by aggregating the judgments of several non-experts (Alonso and Mizzaro 2009; Hsueh et al. 2009; Snow et al. 2008; Alonso and Baeza-Yates 2011; Carpenter 2008).[9] Recent studies also show that the "wisdom-of-the-crowd" effect extends beyond simple human judgment tasks to more complex multidimensional problem solving tasks like ordering problems (Steyvers et al. 2009), the minimum spanning tree problem and traveling salesperson problem (Yi et al. forthcoming).

Specific procedures for aggregating non-expert judgments may influence the quality of data and also convergence on the "truth" (or trusted expert judgment). Snow and colleagues analyzed how many non-experts it would take to match the accuracy of an expert in natural language task (Snow et al. 2008). They found that, using a simple majority rule, they required from two to nine non-experts, and a smaller number with more complicated aggregating algorithms. Similar results have been found when applying the crowdsourcing to machine learning tasks (Sheng et al. 2008) and information retrieval tasks (Nowak and Rger 2010). This follows general results in mathematical and behavioral studies on aggregations of individual judgments where simpler methods perform just as well and often more robustly than more complicated methods (e.g. Ariely et al. 2000; Clemen and Winkler 1999).

All human decision-making is prone to error, and coding decisions are no exception. Coders will differ because they bring to the exercise different bundles of knowledge, different viewpoints, and because they may interpret questions, scales, or instructions in slightly different ways. Snow et al. (2008) identify three general approaches to improve data coding quality given imperfect coders. One is to build in redundancy by employing more coders. Another provides financial incentives to high quality coders by comparing coders to some gold standard (accuracy of coding) or assessing inter-coder agreement on coding items (inter-coder reliability). Finally, we can model the reliability and biases of individual coders and apply corrections to these. Statistical issues of redundancy are discussed in Sheng et al. (2008) and Ipeirotis et al. (2010). They show that repeated coding can improve the quality of data as a function of the individual qualities of the coders and their number, particularly when the coders are imperfect and coding categories (labels) are "noisy". Using the expectation maximization (EM) algorithm proposed in Dawid and Skene (1979) researchers can identify both the "most likely" answer to each task and a "confusion matrix" for each coder. The confusion matrix contains estimates of coder biases that can be used to correct the coding decisions. The confusion matrix contains misclassification probabilities that can be aggregated into a single measure of coder quality. Snow et al. (2008) extend the EM algorithm to recalibrate coders' responses to the gold standard (a small amount of expert coding results). The coding decisions of different coders are assumed to be conditionally

[9] An assumption in the wisdom of crowds results is that coders are independent in making their judgments (Surowiecki 2004). The importance of this assumption has been experimentally confirmed in some instances in Lorenz et al. (2011), while also experimentally shown not to matter in Miller and Steyvers (2011).

independent of each other and follow a multinomial distribution. Their bias recalibration delivers small but consistent improvements in coding results.

When a gold standard is not available Carpenter (2008) and Raykar et al. (2010) develop Bayesian models of data classifiers that simultaneously assess coder quality. Welinder and Perona (2010) develop a classifier model that integrates data difficulty and several coder characteristics. Welinder et al. (2010) take this approach further and develop a unifying model of different characteristics of coders and data. Extending the work of Whitehill et al. (2009), they assume that each text unit (image in their image classification setup) can be characterized using different factors that locate it on an abstract Euclidian space. Each coder, in turn, is represented as a multidimensional entity characterized by competence, expertise and bias. Welinder et al. can thus identify characteristics of data difficulty and different groups of coders based on their underlying characteristics.

An alternative use of the coder quality scores is to use them as a tool for automated acceptance or rejection of coding results (e.g. Ribeiro et al. 2011). Thus coders achieving a very low quality score (often called cheaters or spammers) will be simply rejected from the task together with their submitted coding results. Wang et al. (2011) suggest that simply discarding low quality coders can be losing valuable information, and due to implementation of redundancy this can be costly even given the generally low cost of non-experts in crowdsourcing. They suggest that a better approach is to differentiate between true error rates and biases. While the former are unrecoverable, Wang et al. extend the EM algorithm and show that useful information can be extracted from the latter. Their results show that estimated quality of coders who provide consistently and predictably poor answers can be increased.

3. DESIGNING AN EXPERIMENT IN CROWD-SOURCED DATA CODING

Our primary objective here is to test the proposition that *well-designed crowd-sourced data coding can generate estimates of latent quantities of interest that are statistically indistinguishable from those generated by expert coders*.

The particular project we report here concerns the estimation of party policy positions by coding the content of party manifestos. The core of our experiment involves multiple codings, under different experimental conditions, of about five thousand sentences populating six British party manifestos – the Conservative, Labour and Liberal Democrat manifestos for 1987 and 1997. These manifestos were chosen for two main reasons. First, we have access to extensive and diverse cross-validation of placements of these parties for these periods, using contemporary expert survey estimates of these manifestos, as well as independent MP expert codings of the same texts. Second, there are very well documented shifts in party positions, and especially those of the Labour Party, between these dates. The ability of coders in the crowd to pick up these shifts is a useful test of external *substantive* validity.

To allow more variation in the key quantities being estimated and more extensive external validation, a second phase of the design involves coding manifestos of the three main British parties at all elections between 1987 and 2010, making 18 manifestos in all. Table A1 in the Appendix gives sentence counts for each of the manifestos coded, and shows that the text database under investigation includes a total of about 18 thousand natural sentences.

Simplifying the coding scheme

While the MP has been coding party manifestos for decades, we decided not to use its 56-category policy coding scheme for three main reasons. The first is methodological: the complexity of the MP scheme and the uncertain boundaries between many of its coding categories have been found in coding experiments where multiple expert coders use the MP scheme to code the same documents, to be a major source of inter-coder unreliability (Mikhaylov et al. 2012). The second is practical in a crowd sourcing context: it is effectively impossible to write clear and precise instructions, that could reliably be understood by a globally distributed and diverse set of non-expert coders, for using this detailed and complex scheme. The MP scheme is quintessentially designed for highly trained expert coders. Third, the complexity of the MP scheme is largely redundant. Third-party users of data on party policy positions, including most end users of the MP dataset, almost never want a 56-dimensional policy space. They typically want a reliable and valid *low-dimensional* map of party policy positions. For all of these reasons, we focus here on estimating positions on the two dimensions most commonly used by empirical spatial modelers seeking to represent party policy positions. The first describes policy on matters we think of as “economic policy: left vs right”. The second describes policy on matters we think of as “social policy: liberal vs conservative”. Not only is there evidence that these two dimensions offer an efficient representation of party positions in many countries[10], but these same dimensions were used in expert surveys conducted by Laver and Hunt, Benoit and Laver, and Hooghe et al. (Benoit and Laver 2006; Hooghe et al. 2010; Laver and Hunt 1992). This allows cross-validation of all estimates we derive against widely used expert survey data.[11]

It is not our intention in this paper to engage in a debate on what an ideal text coding scheme might look like. Our prime interest here is the *method*, not the policy dimensions *per se.* This means that any coding scheme must be easy to communicate to non-expert coders distributed worldwide on the Internet, and easy for these coders to deploy using an intuitive coding interface. If the method we propose works effectively, then other scholars may use it to estimate scores on other latent variables of interest to them. Indeed, if a cheap and scalable crowd-sourced data coding system can be developed, the ideal will be for researchers to design and run data coding projects designed specifically to meet their own precise needs, rather than relying on some giant canonical dataset designed long ago by other people for other purposes. A

[10] See Chapter 5 of Benoit and Laver (2006) for an extensive empirical review of this matter in relation of a range of contemporary democracies using expert survey data.
[11] As of October 2012, there were over 1500 Google Scholar citations of either Laver and Hunt or Benoit and Laver.

shift to reliable and valid crowd sourcing would mean that the scientific object of interest would no longer be the *dataset* per se, but the *method* for collecting the required data on demand.

While expert surveys tend to locate political parties on 20-point policy scales, “taking everything into consideration”, we argue that having a coder in the crowd locate a manifesto sentence on a policy dimension is analogous to having a voter answer a question in public opinion survey. We therefore asked coders to code sentences using the five point scales[12] that are tried and tested in mass surveys.[13]

Specifying natural sentences as the unit of text analysis

Before coding can start, we must specify the basic unit of analysis. Automated “bag of words” methods treat single words as the text unit. A method such as Wordscores, as its name implies, scores every word in a set of training (“reference”) documents in terms of the information it provides about the position of the author on some latent dimension of interest (Laver et al. 2003). The MP method for expert text coding specifies the basic text unit as a “quasi-sentence” (QS), defined as a portion of a natural sentence deemed by the coder to express a single policy idea or issue. Text unitization by MP coders is thus *endogenous to the process of text coding*. An obvious alternative is to use “natural” sentences, defined exogenously and causally prior to the process of text coding using a set of pre-specified punctuation marks. The rationale for endogenous text unitization using QS is to allow for the possibility that authors, as a matter of literary style, load different policy ideas into a single natural sentence. A major disadvantage of doing this is that endogenous text unitization by humans is, axiomatically, a matter of subjective judgment, raising the issue of unreliability in specifying the fundamental unit of analysis. This is a real as well as a potential problem. Däubler et al. show that MP coders cannot agree on how to unitize a political text into QS, even when carefully trained to follow clear instructions (Däubler et al. 2011). This introduces substantial unreliability into text unitization and, consequently, into the entire coding process. Däubler et al. also found that using exogenously defined natural sentences as the basic unit of text analysis makes no statistically significant difference to CMP point estimates, while reducing the unreliability of these. We therefore specify the basic unit of text analysis as a natural sentence. This has the great advantage that text unitization can be performed mechanically, with perfect reliability, in a way that is completely exogenous to the substantive scoring of the text.

Coding text units in sequence or random order?

Having specified the text unit as a natural sentence, the next issue that arises concerns the sequence in which sentences are coded. It might seem obvious to start at the beginning of a

[12] Ranging from “very left” to “very right” on economic policy, from “very liberal” to “very conservative” on social policy.
[13] Figure A1 in the Appendix shows our coding instructions, identical for both expert and non-expert coders, specifying definitions of the economic left-right and social liberal-conservative policy dimensions we estimate.

document and work through, sentence by sentence, until the end – perhaps leaving open the sequence in which *documents* are coded, perhaps making more explicit decisions about this, such as coding according to the date of authorship. In what we term “classical” expert coding, experts draw on a fixed coding scheme to read and classify the (natural) sentences in a text in their original sequence, by assigning each sentence a single code. From a practical point of view, however, many workers in a crowd-sourced coding project may not complete the coding of an entire long policy document, and discarding partially coded documents would be horrifically inefficient. From a theoretical point of view, moreover, coding sentences in sequence creates a situation in which coding one sentence may affect priors for subsequent sentence codings, with the result that summary scores such as means calculated for particular documents are not aggregations of i.i.d. scores. In particular, coded sentences tend to occur in “runs” of similar topics, and hence codes, because of the perfectly natural tendency to section a text into clusters of similar topics – just as paragraphs tend to contain multiple sentences on the same topics. Bearing in mind these practical and theoretical matters, we specify the “atomic” text coding task as scoring a single natural sentence on a latent variable of interest. Considering the sequence in which text units are coded, one possibility is to code these in their naturally occurring sequence, as with “classical” expert coding. Another is to sample text units for coding in random order, so that each coding can be seen as an i.i.d. estimate of the latent variable of interest. This has the added big advantage in a crowdsourcing context of *scalability*. Jobs for individual coders can range from being very small to very large; coders can pick up and put down coding tasks at will; every little piece of coding in the crowd contributes to the overall database of text codings.

Qualitative feedback from expert coders stressed the huge difference between coding each manifesto from beginning to end in sequence, and coding the same set of sentences from the same manifestos, served up in random order. This suggests that the difference between classical coding and coding sentences in random order might be an important treatment in our coding experiments. This informed a preliminary coding experiment, in which expert coders coded the same sets of manifesto sentences both in sequence and in random order. We report results from this experiment in the Appendix, but the headline news is that we did not find any systematic effect on our estimates that depended on whether manifesto sentences were coded in their naturally occurring sequence or in random order. This informs our decision, discussed below, to use the more general and tractable random sentence sequencing in the crowd-sourcing method we specify.

Coding text units with or without knowing the name of the author?

Another largely unremarked but obvious feature of classical expert coding is that expert coders typically know the author of the document they are coding. Especially in relation to a party manifesto, it is not necessary to read very far into the document, even if the cover and title page have been torn off, to figure out which party wrote it. (Indeed, we might well think that an expert coder who cannot figure this out is not an expert.) In effect, it is extremely likely that coders will bring non-zero priors to coding manifesto sentences and that the precisely same sentence ("we must do everything we can to make the public sector more efficient") will be coded in different ways if the coder knows the text comes from a right-wing rather than a left-wing party. Yet codings are typically aggregated into estimated party positions as if the coders had zero priors. Crudely speaking, we don't really know how much of the score given to any given sentence is the expert coder's judgment about the actual content of the sentence, and how much is a judgment about its author.

This raises the issue, if we serve sentences at random from the text corpus, of whether or not to reveal the name of the author, especially since we found during the early stages of our project that it is often surprisingly difficult to guess the author of a single random manifesto sentence. Accordingly, in our preliminary coding experiment, expert coders coded the same sets of manifesto sentences both knowing and not knowing the name of the author. We report details of this analysis in the Appendix. The headline news is that we did find systematic coding biases arising from knowing the identity of the document's author. Specifically, we found that coders tended to code the same sentences from the Conservative manifestos as more right wing, if they knew that these sentences came from a Conservative manifesto. This informs our decision, discussed below, to withhold the name of the author in the crowd-sourcing method we specify.

Providing context for the target sentence

"Classical expert coding", which begins at the beginning of a document with a known author and ends at the end, creates a "natural" context in which every individual sentence is coded in light of the text surrounding it. Often, it is this surrounding text that gives a sentence substantive meaning. Given the results we note in the previous two sections, our crowd-sourcing method will specify the atomic crowd-sourced text coding task as coding a "target" sentence selected at random from a text, with the name of the author not revealed. This leaves open the issue of how much context either side of the target sentence we provide to assist the coder. The final objective of our preliminary coding experiment was to assess the effects of providing no context at all, or a +/- two sentence context. We report results in the Appendix, but the headline news here is that we found significantly better correspondence between coder judgments and "golden" codings when we provided a context of two sentences before and after the sentence to be coded. This informed out decision to settle on a two-sentence context for our crowd-sourcing method.

Experimental design

We take as our baseline the coding method we call "classical expert coding" (ES), in which experts (E) code sentences in sequence (S), beginning within the first sentence in a document and finishing with the last. With the exception that we have adhered to Daubler et al's (2012) prescription to take strict natural sentences as the coding unit, ES coding closely conforms to the practice of expert coding used by the bulk of political science content analysis approaches (e.g. MP and the Policy Agendas Project). Given the methodological decisions we discussed in the three previous sub-sections, we specify a single crowd-sourced "treatment" in this experiment (CR). This uses the crowd (C) to code precisely the same set of sentences, randomly (R) selected from the text corpus, set in the context of two sentences before and after and without revealing the name of the author. Since moving from classical expert coding (ES) to the crowd-sourced treatment (CR) involves changing two things at the same time, we specified a second treatment in which experts code the same manifestos using exactly the same method as the crowd coders – coding randomly-served sentences with +/- two-sentence contexts, withholding the name of the author. This treatment (ER) allows us to compare ES with ER coding, and the ER with CR coding, and thereby to factor out the independent effects of moving to random sequence coding and moving into the crowd.

For the expert coding, we designed a web interface to serve up individual sentences for coding according to the relevant treatment. Coding was accomplished with simple mouse clicks. A total of six expert coders[14] coded the six core manifestos with the sentences appearing in sequence (ES). These expert coders then coded the same manifestos but with the sentences appearing in random order, according to the ER treatment. To extend the coding tasks to the "crowd", we set up a nearly identical coding interface on the CrowdFlower platform, as discussed below, in such a way that crowd coders were given batches of sentences to code that were randomly selected (without replacement) from the text corpus (CR), until the entire text corpus has been coded.

4. DISTRIBUTING CROWD-SOURCED CODING TASKS VIA THE INTERNET

There are many online platforms available for distributing crowd-sourced Human Intelligence Tasks (HITs) via the Internet. The best-known is Amazon's Mechanical Turk (MT), which provides an interface for creating jobs, combined with a massive global pool of workers who use the system. The system allows employer-researchers to distribute a large number of tasks that require human intelligence to solve, but are nonetheless extremely simple to specify. Workers receive a small amount of compensation, often a few pennies, for completing each task. The system is well suited for tasks that are easy for humans to complete, but difficult for computers. This includes jobs such as: labeling images – "is this a picture of a cat or a dog?"; translating short pieces of text; or classification – "is this website about food or sports?" Given its ability to generate large amounts of data very quickly and cheaply, there is tremendous potential for using this type of service in the social sciences. While MT, and crowd-sourcing more generally, are potentially very valuable in a number of different settings, there are problems with "out-of-the-box" implementations of MT. The most obvious is quality assurance. Given the natural economic motivation of workers to finish as many jobs in as short a time as possible, it is easy for workers to submit badly or falsely coded data. Such workers are often referred to as

[14] Three of the authors of this paper, plus three senior PhD students in Politics from New York University.

"spammers". The data they generate is useless but, given the open nature of the platform, it is difficult to prevent them from participating in a job (Nowak and Rger 2010; Eickhoff and de Vries 2012; e.g. Kapelner and Chandler 2010; Berinsky et al. 2012b). Spamming would be a serious problem for our text coding project, because it is crucial that workers thoroughly read both the coding instructions and sentences being coded. For this reason, we implemented our design using the CrowdFlower (CF) platform, which also deploys jobs on MT, but provides a robust layer of screening that helps prevent spammers from participating in the job.[15] This is provided by CF's system of "gold" questions (see below) and related filters for "trusted" and "untrusted" judgments. In the remainder of this section we explain how these filters work, how to prepare manifesto text for coding, and the interface workers use to code manifesto sentences.

Preparing the data for coding

The CF platform implements a two-step process to improve the quality of results submitted through crowd-sourcing. This uses "gold" questions, both to screen out potential spammers from the job and to provide insight into the quality of each coded unit. For any task, gold questions are pre-labeled versions of questions with very clear answers, analogous to those that will be completed during the task. For example, if the task is to label unknown pictures as either dogs or cats, the system would use several pictures of cats and dogs for which the labels were already known. Coders would be asked to label these, and their answers to these "gold" questions used to determine the quality, or potential "spamminess" of each coder.[16] If a coder gets too many gold questions wrong then the coder's trust level goes down and s/he may be ejected completely from of the job. We supplemented this with a training mode offered by CF, according which workers are only allowed onto a job after they have correctly answered four "gold" questions.[17] This resulted in a huge improvement in the quality of our responses. Perfecting this training mode turned out to make the crucial difference in getting quality coder results from the crowd, versus unreliable results from crowd coders unable or uninterested in providing good results. As we discuss below, the crowd results finally reported here resulted from a single overnight job. Perfecting the web system on Crowdflower, tweaking the gold questions, and testing and refining the training mode took us over six months.

We specified our gold questions as manifesto sentences that all six of our experts coded as having the same policy content (economic, social or neither), and the same direction (liberal, conservative, or neither).[18] An example of a gold sentence we used was: "Our aim is to ensure Britain keeps the lowest tax burden of any major European economy". This came from the actual 1997 Conservative manifesto. All six expert coders coded it as either "somewhat" or "very" to

[15] For more information on CrowdFlower is http://www.crowdflower.com.
[16] For CrowdFlower's formal definition of gold questions see: http://crowdflower.com/docs/gold.
[17] Workers giving wrong codes to gold questions are given a short explanation of why they are wrong.
[18] Thus, for example, a sentence was eligible as gold if it was labeled by all expert coders as being about Economic policy, and by all of them as either "Somewhat" or "Very Conservative".

the right on economic policy, so "Economic policy: somewhat or very right wing" is specified as the golden coding.

In addition to using the CrowdFlower "gold" system for rating the coders in the crowd, we used a number of "screener" sentences to ensure that coders were paying attention throughout the coding process (Berinsky et al. 2012b). These sentences are set in a natural +/- two sentence context, but replace the target sentence with one the morphs into a simple instruction, for example " We will restore honesty in sentencing by abolishing automatic early release; however, code this sentence as having economic policy content with a score of Very Right.". Observing the codes entered for screener sentences enables us to check the extent to which the coder was paying attention and potentially to weight coder input accordingly.

Developing the interface

Once gold data have been identified, CF has a flexible system for working with many different types of crowd-sourcing task. In our case, preparing the manifesto texts for CF coders requires converting the text into a matrix-organized dataset with one natural sentence per row. CF uses its own proprietary markup language, CrowdFlower Markup Language (CML), to build jobs on the platform.[19] The language is based entirely on HTML, and contains only a small set of special features that are needed to link the data being used for the job to the interface itself. To create the coding tasks themselves, some additional markup is needed. Here we use two primary components: a text chunk to be coded, and the coding interface. To provide context for the text chunk, we include two sentences of preceding and proceeding manifesto text, in-line with the sentence being coded. The line to be coded is colored red to highlight it. The data are then linked to the job using CML, and the CF platform will then serve up the coding tasks as they appear in the dataset. To design the interface itself we use CML to design the form menus and buttons, but must also link the form itself to the appropriate data. Unlike the sentence chunk, however, for the interface we need to tell the form which columns in our data will be used to store the workers' coding; rather than where to pull data from. In addition, we need to alert the CF platform as to which components in the interface are used in gold questions. Figure A2 in the Appendix shows the precise CML used to design our CF interface. With all aspects of the interface designed, the CF platform uses each row in our data set to populate tasks, and links back the necessary data. Each coding task is served up randomly by CF to its pool of workers, and the job runs on the platform until the desired number of trusted judgments has been collected.

5. RESULTS

Independent benchmark

[19] For complete details on the CrowdFlower Markup Language, see http://crowdflower.com/docs/cml

Table 1 shows independent estimates of policy positions of the British Labour, Liberal Democrat and Conservative parties, at or around the time of the general elections of 1987 and 1997. The top panel shows estimates based on expert surveys conducted by Laver and Hunt in 1989, and by Laver immediately after the 1997 election (Laver and Hunt 1992; Laver 1998). The bottom panel shows estimates based on the widely used right-left dimension "Rile" developed by the MP, using procedures proposed by Benoit et al for calculating bootstrapped standard errors, and by Lowe et al for rescaling the same data using a logit scale (Lowe et al. 2011; Benoit et al. 2009).

Table 1: Expert survey estimates of positions of British political parties in 1989 and 1997

	Conservative		*Liberal Democrat*		*Labour*	
	1987	*1997*	*1987*	*1997*	*1987*	*1997*
Expert surveys[20]						
Economic policy						
Mean	17.2	15.1	8.2	5.8	5.4	10.3
SE	*0.4*	*0.2*	*0.4*	*0.2*	*0.4*	*0.2*
Social policy						
Mean	15.3	13.3	6.9	6.8	6.5	8.3
SE	*0.5*	*0.3*	*0.4*	*0.2*	*0.4*	*0.2*
Manifesto project						
Rile						
Mean	30.4	25.8	-4.2	-5.8	-13.7	8.2
BLM SE	*2.2*	*2.4*	*2.3*	*2.4*	*3.1*	*2.6*
Logit Rile						
Mean	1.13	0.85	-0.16	-0.22	-0.48	0.29
Lowe et al. SE	*0.10*	*0.09*	*0.09*	*0.09*	*0.11*	*0.10*

Source: Expert surveys: Laver (1998). *Rile:* Replication dataset for Benoit et al (2009). *Logit Rile:* Replication dataset for Lowe et al (2011)

These independent estimates comport with received wisdoms about core substantive features of British party politics during the period under investigation. In particular, we take the following to represent to core substantive features of party positioning and movement that any method we propose should capture. The Labour party is shown in all independent estimates as moving sharply and significantly to the right on economic policy, or Rile, during the period of transition to "New Labour" between 1987 and 1997. Over the same period, the Liberal Democrats are shown as moving to the left – the shift being statistically significant for expert survey estimates though not for the MP Rile estimates. As a result, Labour and the Liberal Democrats switch positions on the economic policy scale according to expert survey estimates, and on the Rile scale according to MP data. The Conservatives are always estimated as being significantly to the right of the other parties, though are estimated to have moved somewhat towards the center between 1987 and 1997.

[20] Expert survey results refer to 1989 not 1987

Classical expert coding (ES)

Table 2 presents the first substantive results from our six independent "classical" expert codings of the six core manifestos on two policy dimensions, each sentence coded in sequence, starting from the beginning. For each sentence in each coded manifesto, we calculated a mean of the six independent expert codings on each dimension, and the numbers reported in Table 2 are the means of the mean sentence codings for all sentences in each manifesto.

Table 2: ES estimates of positions of British political parties in 1987 and 1997

	Conservative		*Liberal Democrat*		*Labour*	
	1987	*1997*	*1987*	*1997*	*1987*	*1997*
Economic policy						
Mean	0.41	0.40	-0.39	-0.48	-0.89	-0.28
SE	*0.03*	*0.03*	*0.03*	*0.03*	*0.03*	*0.03*
Social policy						
Mean	0.68	0.51	-0.61	-0.96	-0.79	-0.05
SE	*0.05*	*0.05*	*0.06*	*0.06*	*0.08*	*0.07*

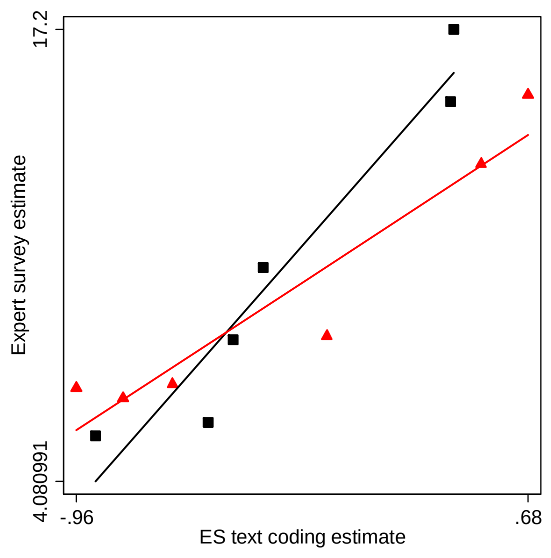


Figure 1: British party positions 1987 and 1997 on economic (black squares) and social (red triangles) policy estimated by "classical" expert text coding (ES) and independent expert survey

The ES text coding estimates capture all of the key substantive patterns noted in the previous section. Labour moves sharply to the right on economic policy, and in a conservative direction on social policy. The Liberal Democrats move in the opposite direction on both dimensions, resulting in reversed positions on both dimensions for Labour and LibDems. The Conservatives are always significantly to the right of the other parties on both dimensions. Figure 1 plots the ES estimates of party positions on economic and social policy against the same positions estimated using independent expert surveys. The linear regression lines summarizing these plots show that ES text coding estimates predict the expert survey estimates on each dimension very well.

The pretty results in Table 2 and Figure 1, however, sit on top of messy inter-coder variation in individual sentence codings, variation that shows that *the expert coders are themselves a crowd.* Figure 2 categorizes each of the coded sentences in the manifestos under investigation according to how many of the six expert coders coded the sentence as having economic policy content (left panel) or social policy content (right panel) – setting aside for now the directional coding of each sentence over and above this.[21] If expert coders were in perfect agreement on the policy content of each manifesto sentence, there would be only two bars in each panel. Either all six coders would code each sentence as dealing with economic policy, or none would. This is clearly not true. For economic policy there was unanimous expert agreement only for 46% of coded manifesto sentences. This involved agreement on no economic policy content for about 37% of sentences, with unanimous agreement that about 10% of sentences concerned economic policy. For the remaining 54% of manifesto sentences, at least one expert coded the sentence as dealing with economic policy but at least one expert disagreed.

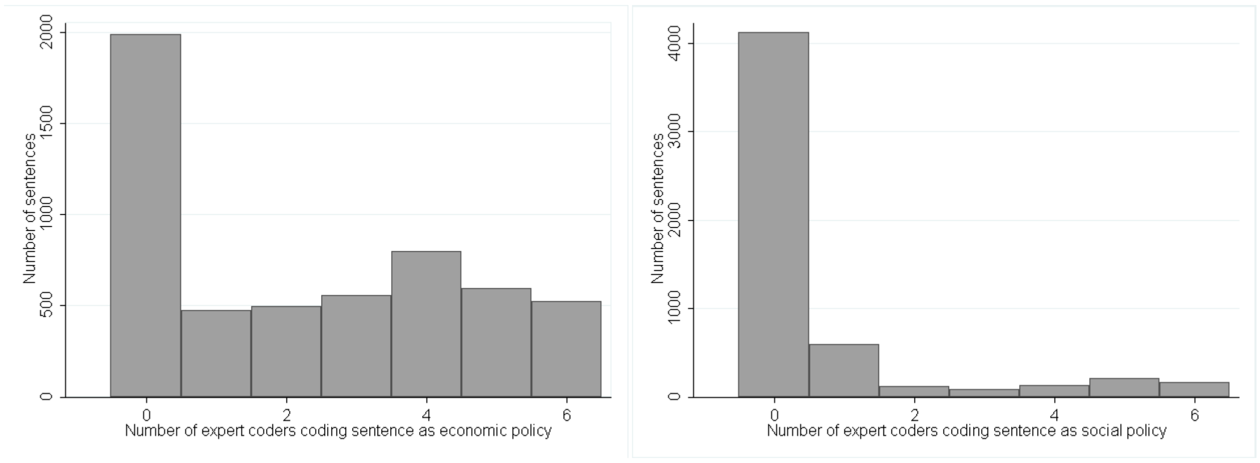


Figure 2: Agreement between six expert coders on classification of manifesto sentences

Turning to substantive scoring of sentences for directional policy content, Figure 3 summarizes, for sentences in which more than one expert found economic or social policy content, standard deviations of expert scores for each sentence. Complete scoring agreement on a sentence among experts who agree the sentence deals with economic policy, for example, implies a zero standard deviation of expert scores. Clearly this was not true for most scored sentences.

In a nutshell, while Figure 1 shows that the point estimates we derive from classical expert coding have excellent external validity, Figures 2 and 3 show considerable variation in the data underlying these point estimates, even using trained expert coders of the type usually used

[21] More detailed information can be found in Table A2 in the Appendix.

to generate coded data in the social sciences. As noted above, the experts are themselves a crowd and should be treated as such.

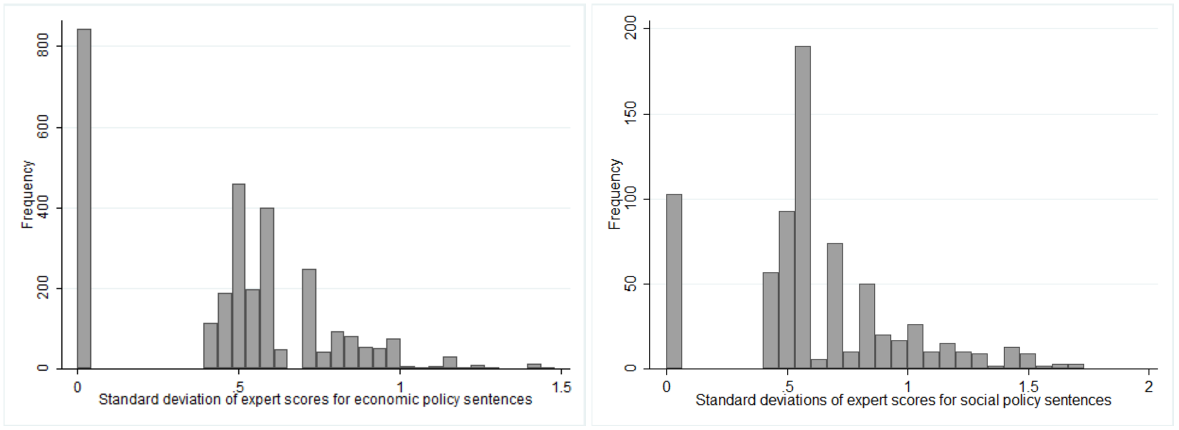


Figure 3: Agreement between six expert coders on scoring of manifesto sentences

There is no paradox in the facts that, while a crowd of experts disagree substantially on the policy content of individual manifesto sentences, we can nonetheless derive externally valid estimates of the policy positions of manifesto authors if we aggregate the judgments of all experts in the crowd on all sentences in the manifesto corpus. This happens because, while each expert judgment on each manifesto sentence can be seen as a noisy realization of some underlying signal about policy content, the expert judgments taken as a whole scale nicely – in the sense that in aggregate they are all pulling in the same direction and seem to be capturing information, albeit noisy, about the same underlying quantity.

This is shown in Table 3, which reports a reliability analysis for the economic policy scale derived by treating the economic policy scores for each sentence allocated by each of the six expert coders as six sets of independent estimates of economic policy positions. Two of the coders (Experts 3 and 6) were clearly much more cautious than the others in classifying sentences as having economic policy content. Notwithstanding this, overall scale reliability, measured by a Cronbach's alpha of 0.96, would be rated "excellent" by any conventional standard.[22] All coders contribute information to the scale – the final column shows that the scale alpha would decrease if any coder were excluded. All sets of codings correlate highly with the resulting synthetic scale, as well as with other codings that comprise the scale. Notwithstanding the variance in the individual sentence codings, therefore, the resulting aggregate economic policy scale is extraordinarily reliable by these measures.[23] Our expert coders are a crowd in the sense they often make different judgments about the same thing. But they are a "good" crowd in the sense that their aggregated judgments all appear to relate to the same underlying constructs, as well as having good external validity. Clearly, what we hope is that our non-expert crowd-sourced coders are also a "good" crowd in this sense.

[22] Conventionally, an alpha of 0.70 is considered "acceptable".
[23] Results for social policy are almost identical and are available from the authors.

Table 3: Inter-coder reliability analysis for an economic policy scale generated by aggregating all expert scores for sentences judged to have economic policy content

Item	*N*	*Sign*	*Item-scale correlation*	*Item-rest correlation*	*Cronbach's alpha*
Expert 1	2298	+	0.88	0.79	0.95
Expert 2	2715	+	0.88	0.79	0.95
Expert 3	1045	+	0.86	0.75	0.95
Expert 4	2739	+	0.89	0.77	0.95
Expert 5	2897	+	0.89	0.79	0.95
Expert 6	791	+	0.89	0.84	0.94
Economic policy scale					0.96

Expert coding of random sentences (ER)

Taking classical expert coding (ES) as the baseline, the first treatment we report is ER, whereby experts coded exactly the same set of sentences, but with these sentences served in random order, with the name of the author withheld, but with +/- two sentences of context. This is identical to the CR treatment we report below, except that we use experts rather than coders in the crowd. Since we have now completed all expert coding phases of data collection, we report ES and ER results for the 18 manifestos issued by the three main British parties at all elections between 1987 and 2010. Figure 4 summarizes a lot of information about the positions of the 18 party manifestos on the two policy dimensions of interest – at total of 36 estimates derived from both ES and ER coding methods. Each point with error bars plots the ES and ER estimates for a single manifesto on a single dimension.

The headline news is that the ES and ER point estimates are very highly correlated. In a nutshell, moving from classical expert coding to having experts code sentences in random order, withholding the name of the author, does NOT make much of a substantive difference to the point estimates that are generated. ES and ER estimates seem to be measuring the same thing. This is important to know moving forward to the CR coding treatment, since the shift to coding random sentences does not seem to be making a big difference to substantive results.

The other news is that we do find some evidence of the "centrist" bias we found in our preliminary coding trials. Conservative manifestos tend systematically to be below the 45 degree line and Labour manifestos to be above this. Our preliminary trials suggest strongly that this arises from withholding the name of the author in the ER treatment. Knowing a manifesto comes from the Conservative party, for example, as they do with the classical expert coding (ES), expert coders are somewhat more likely to code any given sentence in it as being on the right. The converse is true for the Labour party. Substantively, this effect is not huge, but it is systematic. This raises the interesting, and to our knowledge hitherto unexplored, question of what is the "correct" thing to do. Do we want expert coders to code sentences in the context of some prior about the position of the author on the sentence to be coded, so that the same sentence may be coded in different ways by the same coder, if it is known to come from different authors? Or do we want coders to code sentences purely on their local content, without bring any prior about the author into this judgment? This is of course a generic problem for data coding in the social sciences. Expert coders who generate codes for the Polity dataset, for example, presumably do this in light of substantive priors they have about the country in question.

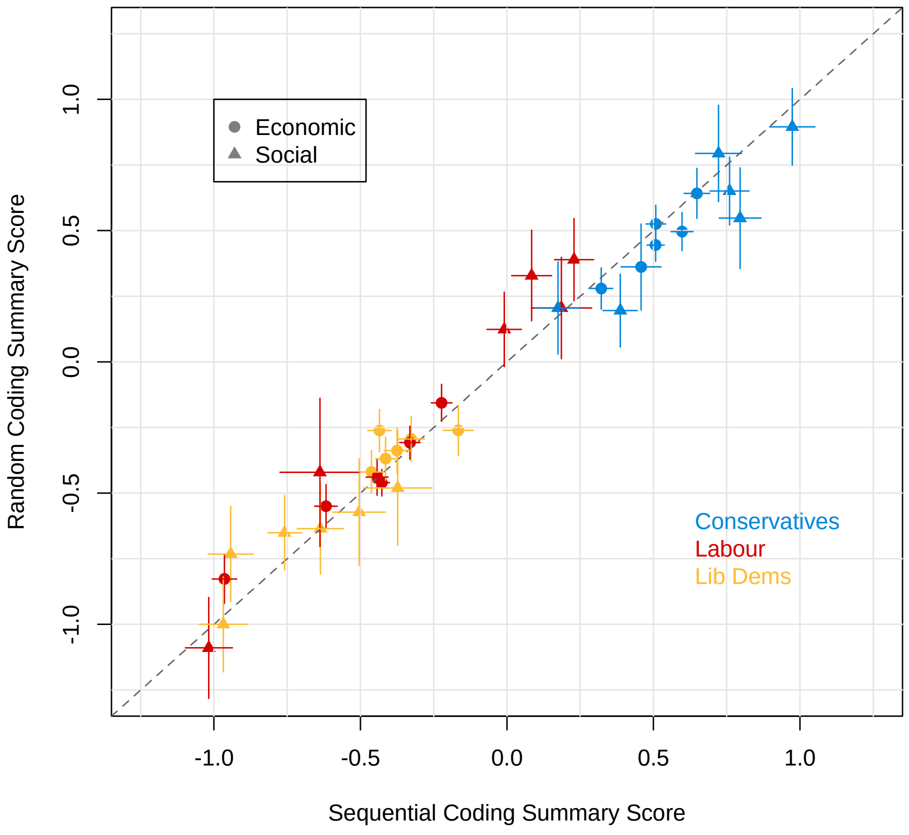


Figure 4: Comparison of summary left-right scores from expert coding, comparing sequential versus random coding

Crowd-sourced results of random sentence (CR) coding

We divided the sentences in the six party manifestos for 1987 and 1997 into a series of Crowdflower jobs that completed after achieving a minimum of five codings of each non-gold sentence by trusted coders.[24] The vast bulk of these jobs were deployed one afternoon, EST, and fully completed overnight. A total of 263 unique coders, based in 21 different countries, participated in the CR exercise. Some coders coded no more than ten sentences, while the most

[24] Given their role in assessing coder quality, gold sentences tend to be served much more frequently, so there are many more than five codings of each gold sentence.

prolific coded 2196 sentences. Country origins of crowd codings are shown in Table 4; about 57 percent of codings come from the US, 23 percent from India, and six percent from Britain.

Table 4: Country origins of CR codings

Country	N codes	Mean trust score	% codes
ARE	67	0.87	0.21
ARG	545	0.90	1.74
BEL	15	1.00	0.05
BRA	35	0.78	0.11
CAN	245	0.83	0.78
COL	160	0.79	0.51
GBR	1,828	0.82	5.85
HUN	30	1.00	0.10
IND	7,100	0.76	22.73
ITA	110	0.91	0.35
LVA	25	0.90	0.08
MKD	781	0.75	2.50
MYS	15	0.86	0.05
NLD	115	0.88	0.37
OMN	10	1.00	0.03
PAK	275	0.93	0.88
PRT	25	0.88	0.08
ROU	2,006	0.87	6.42
URY	20	0.83	0.06
USA	17,793	0.84	56.96
ZAF	40	0.83	0.13
Total	27,092	0.82	100.00

As we noted above, the CrowdFlower system uses answers to a set of “Gold” questions (sentences in our case) with known answers (consensus expert codings in our case) to assess the quality of each coder. In addition, the training mode only allows coders into the system in the first place if they correctly coded four of our gold sentences. The rate of correct answers to gold questions generates a trust score for each coder, which can be taken into account when generating aggregate results. Table 4 also reports mean coder trust scores by country, and shows that, while the party manifestos were British, it was not the case that British coders had higher than average trust scores. Indeed, of countries generating more than one percent of the codings, the highest average trust score came from Pakistan. Figure 5 reports the distribution of coders’ trusts scores, and shows that the bulk of coders had trust scores in the 0.8 – 0.9 range.

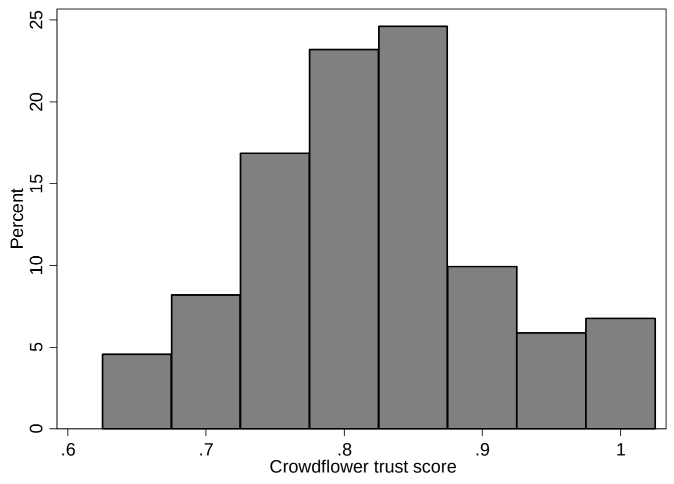


Figure 5: Distribution of trust scores, by coder

Table 5 shows CR-generated estimates of party manifesto positions on the two policy scales, both for all crowd coders, and using only "trusted" coders.[25] As before, these are the means of the mean CR codings for each manifesto sentence. Figure 6 plots these CR-generated estimates using all qualified crowd coders (left panel) and only trusted coders (right panel) against ES-generated estimates of the six party manifesto positions on the two policy dimensions.

Table 5: CR estimates of positions of British political parties in 1987 and 1997

	Conservative		*Liberal Democrat*		*Labour*	
	1987	*1997*	*1987*	*1997*	*1987*	*1997*
Economic policy						
CR Mean (all)	0.27	0.28	-0.33	-0.13	-0.69	-0.38
CR SE (all)	*0.03*	*0.02*	*0.03*	*0.03*	*0.03*	*0.02*
CR Mean (trusted)	0.37	0.41	-0.13	-0.03	-0.83	-0.28
*CR SE (trusted)**	*0.08*	*0.04*	*0.08*	*0.05*	*0.08*	*0.04*
Social policy						
CR Mean (all)	-0.15	0.02	-0.40	-0.48	-0.74	-0.50
CR SE (all)	*0.04*	*0.03*	*0.04*	*0.03*	*0.04*	*0.04*
CR Mean (trusted)*	0.15	0.67	-0.64	-0.24	-0.75	0.01
*CR SE (trusted)**	*0.12*	*0.05*	*0.32*	*0.11*	*0.11*	*0.08*

The results are very encouraging. Starting with Figure 6, both panels show that the benchmark expert codings are well-predicted by crowd-sourced codings, though there is noticeably more attenuation in crowd-sourced estimates[26] when all coders are used, rather than only trusted coders. Our three key tests of substantive validity, documented in Table 1, were that: the

[25] Coders with a with a trust score > 0.70.
[26] The range of CR point estimates is smaller

Conservatives were substantially to the right of the other parties on both dimensions in both years; Labour moved substantially to the right on both economic and social policy between 1987 and 1997; Labour and the Liberal Democrats consequently switch positions between 1987 and 1997 on the economic policy dimension. Table 2 shows that all of these patterns are observed in the expert (ES) codings. Table 5 shows that they are also observed, bar the switch of Labour and LibDem positions on economic policy, in the crowd-sourced (CR) codings. This is the case whether we use the full set of codings or only those from trusted coders, though estimated changes in Labour Party positions are much more pronounced when results are restricted to trusted coders. The Conservatives are always on the right in the CR codings. Labour shifts sharply to the right on both economic and social policy. Labour and the LibDems converge, but switch positions only on social policy, as estimated by trusted coders. Particularly if we consider the results generated by trusted crowd coders in the right panel of Figure 6, however, it does look very much as if crowd-sourced and expert codings of the same documents are measuring the same latent quantities. Non-expert coders scattered around the world are generating essentially the same estimates of party policy positions as "expert" political science professors and graduate students. And the crowd-sourced results were generated overnight, at much lower cost, rather than over three months of a summer vacation

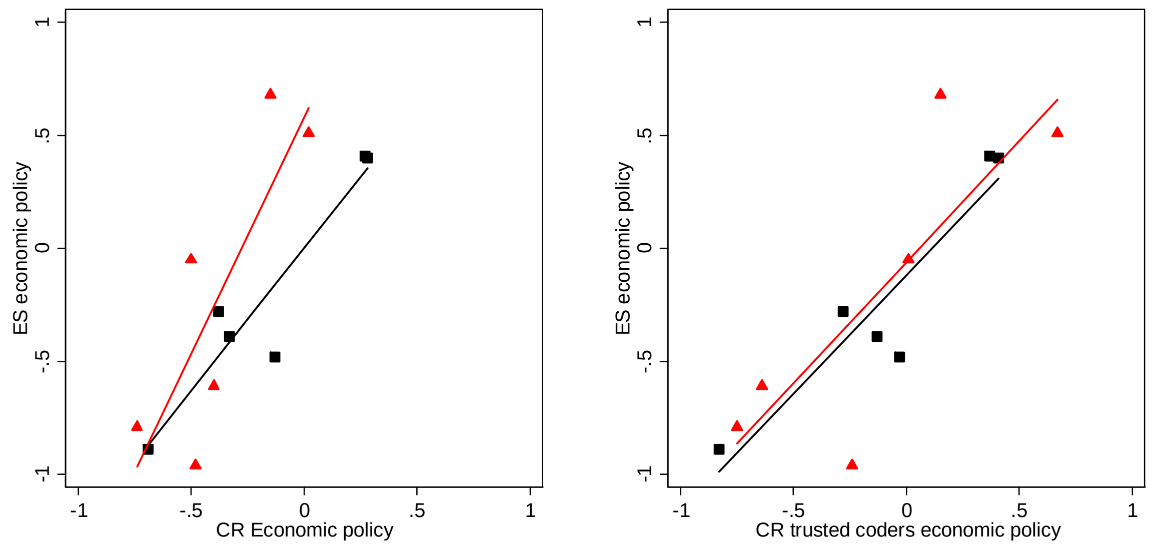


Figure 6: CR and ES estimates of manifestos' economic (black) and social (red) policy positions

Once again, the nice pictures in Figure 6 show "well-behaved" aggregate estimates of party policy position that represent signal extracted from a fairly noisy underlying dataset. Instead of plotting estimated positions of entire party manifestos, Figure 7 plots CR against ES estimated positions of individual manifesto sentences, and shows considerable scatter. It is by no means the case that crowd sourced and expert sentence codings map directly into each other. Table 6 reports linear regressions predicting the benchmark ES expert codings of individual sentences from our CR crowd sourced codings. Notwithstanding the scatter in Figure 6, these results show, in the aggregate, that there is a strong and significant statistical relationship between the CR and ES coding of individual manifesto sentences. Despite a lot of noise at individual sentence level, the CR codings in the aggregate contain a lot of signal. This is the essence of crowd-sourcing.

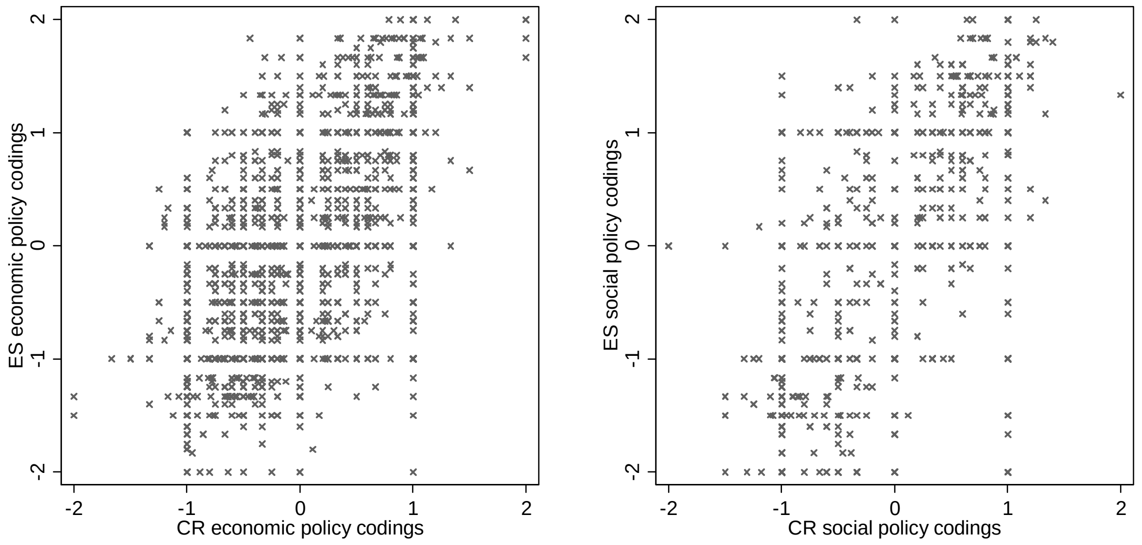


Figure 7: CR and ES estimates of policy content of individual British party manifesto sentences on economic (left panel) and social (right panel) policy.

Table 6: Predicting benchmark ES text codings of individual manifesto sentences using crowd-sourced codings

	ES economic policy		*ES social policy*	
CR economic policy	0.72** (0.02)			
Trusted CR economic policy		0.90** (0.04)		
CR social policy			0.88** (0.04)	
Trusted CR social policy				1.11** (0.07)
Constant	-0.02 (0.01)	0.15 (0.03)	0.04 (0.03)	0.15 (0.05)
Adjusted R^2	0.34	0.43	0.36	0.54
N	2799	593	976	235

*** = significant at better than 0.001 level*

CONCLUSIONS

Crowd-sourcing offers huge potential advantages for the creation of data traditionally generated using expert judgments. Drawing on a vast pool of untrained coders to carry out human judgment tasks is a scalable, cost-effective, and efficient means to generate data formerly the preserve of a single trained expert. We presented results of experiments that use multiple raters to code the policy content of sentences in party manifestos. We used our method with both an expert panel of raters, and with a panel recruited from the crowd, using the CrowdFlower system for recruiting and assessing workers sourced worldwide via Mechanical Turk.

Our method involves a simplified coding scheme which partitions all sentences in any text under investigation according to whether they deal economic policy, social policy, or neither of these. For sentences dealing with economic or social policy, we ask coders the substance of policy on a five-point scale. Using this simplified scheme, we achieved good results from our expert panel of coders, cross-validated against completely independent sources. Applying our method using crowd-sourced coders, we found that it is indeed possible to replicate estimates of party policy positions that are generated using expert text coders. So far so good.

APPENDIX

Table A1: Sentence counts for British party manifestos, 1987-2010

	Expert Coded Sentences			Total Sentences in
Manifesto	Sequential	Random	Total	Manifesto
Con 1987	6,090	1,830	7,920	1,015
LD 1987	5,268	1,527	6,795	878
Lab 1987	2,730	770	3,500	455
Con 1997	7,026	2,081	9,107	1,171
LD 1997	5,238	1,609	6,847	873
Lab 1997	6,312	1,889	8,201	1,052
Con 1992	8,654	3,061	11,715	1,731
LD 1992	4,420	1,593	6,013	884
Lab 1992	3,305	1,144	4,449	661
Con 2001	3,740	1,289	5,029	748
LD 2001	5,890	2,106	7,996	1,178
Lab 2001	8,758	3,103	11,861	1,752
Con 2005	2,070	723	2,793	414
LD 2005	3,379	1,462	4,841	821
Lab 2005	4,744	2,137	6,881	1,186
Con 2010	4,960	2,182	7,142	1,240
LD 2010	3,420	1,514	4,934	855
Lab 2010	5,396	2,372	7,768	1,349
Total	91,400	32,392	123,792	18,263

Table A2: Agreement between six expert coders on classification of policy content: Frequency of scales assigned per sentence for 1987 and 1997 manifestos, sequential codings

	Social policy							
Economic policy	0	1	2	3	4	5	6	Total
0	1,193	196	67	59	114	190	170	1,989
1	326	93	19	11	9	19	0	477
2	371	92	15	15	5	0	0	498
3	421	117	12	7	0	0	0	557
4	723	68	10	0	0	0	0	801
5	564	31	0	0	0	0	0	595
6	527	0	0	0	0	0	0	527
Total	4,125	597	123	92	128	209	170	5,444

APPENDIX

An experiment to assess the effect of context and sequence of sentence coding

The aim of this methodological experiment was to assess the effects of: coding manifestos in their natural sequence or in random order (Treatment 1); providing a +/- two-sentence context for the target sentence (Treatment 2); revealing the title of the manifesto and hence the name of its author (Treatment 3). The text corpus to be coded was a limited but carefully-curated set of 120 sentences. We removed some surrounding sentences that had proper party names in them, to maintain some degree of manifesto anonymity. These were chosen on the basis of the classical expert coding (ES) phase of our work to include a balance of sentences between expert-coded economic and social policy content, and only a few sentences with no economic or social policy content. The coder pool comprised three expert coders, all co-authors of this paper, and 30 “semi-expert” coders who were Masters students in Methods courses at either LSE or UCL. The detailed design for the administration of treatments to coder is available from the authors. The analysis depends in part on the extent to which the “semi-expert” coders agreed with a master or “gold” coding for each sentence, which we specified as the majority scale and code from the three "expert” coders.

For each sentence that was master-coded as referring to none, economic, or social policy, Table A2 reports exponentiated coefficients from a multinomial logit predicting how a coder would classify a sentence, using the sentence variables as covariates. This allows direct computation of misclassification, given a set of controls. Since all variables are binary, we report odds ratios. Thus the highlighted coefficient of 3.272 in Model 1 means that, when the master coding says the sentence concerns neither economic nor social policy, the odds of a coder misclassifying the sentence as economic policy were about 3.3 times higher if the sentence displayed a title, all other things held constant. More generally, we see from Table A2 that providing a +/- two-sentence context does tend to reduce misclassifications (with odds ratios less that 1.0) while showing the coder the manifesto title does tend to increase misclassification (with odds ratios greater than 1.0).

Confining the data to sentence codings for which the coder agreed with the master coding on the policy area covered by the sentence, Table A3 reports an ordinal logit of the positional codes assigned by non-expert coders, controlling for fixed effects of the manifesto. The base category is the relatively centrist Liberal Democrat manifesto of 1987. The main quantities of interest estimate the interactions of the assigned positional codes with title and context treatments. If there is no effect of title or context, then these interactions should add nothing. If revealing the title of the manifesto makes a difference, this should for example move economic policy codings to the left for a party like Labour, and to the right for the Conservatives. The highlighted coefficients show that this is a significant effect, though only for Conservative manifestos.

Table A2: Scale misclassification

		(1)	(2)	(3)
		Master Scale		
Equation	Independent Variable	Neither	Economic	Social
Economic	Context	0.492*		2.672
		(0.214 - 1.132)		(0.702 - 10.18)
	Sequential	1.069		0.896
		(0.578 - 1.978)		(0.396 - 2.030)
	Title	3.272***		1.053
		(2.010 - 5.328)		(0.532 - 2.085)
Social	Context	0.957	0.822	
		(0.495 - 1.850)	(0.583 - 1.160)	
	Sequential	0.867	1.05	
		(0.527 - 1.428)	(0.800 - 1.378)	
	Title	1.540**	1.064	
		(1.047 - 2.263)	(0.877 - 1.291)	
None	Context		0.478***	0.643
			(0.280 - 0.818)	(0.246 - 1.681)
	Sequential		1.214	2.598**
			(0.758 - 1.943)	(1.170 - 5.766)
	Title		0.854	0.807
			(0.629 - 1.159)	(0.505 - 1.292)
	N	750	3,060	1,590

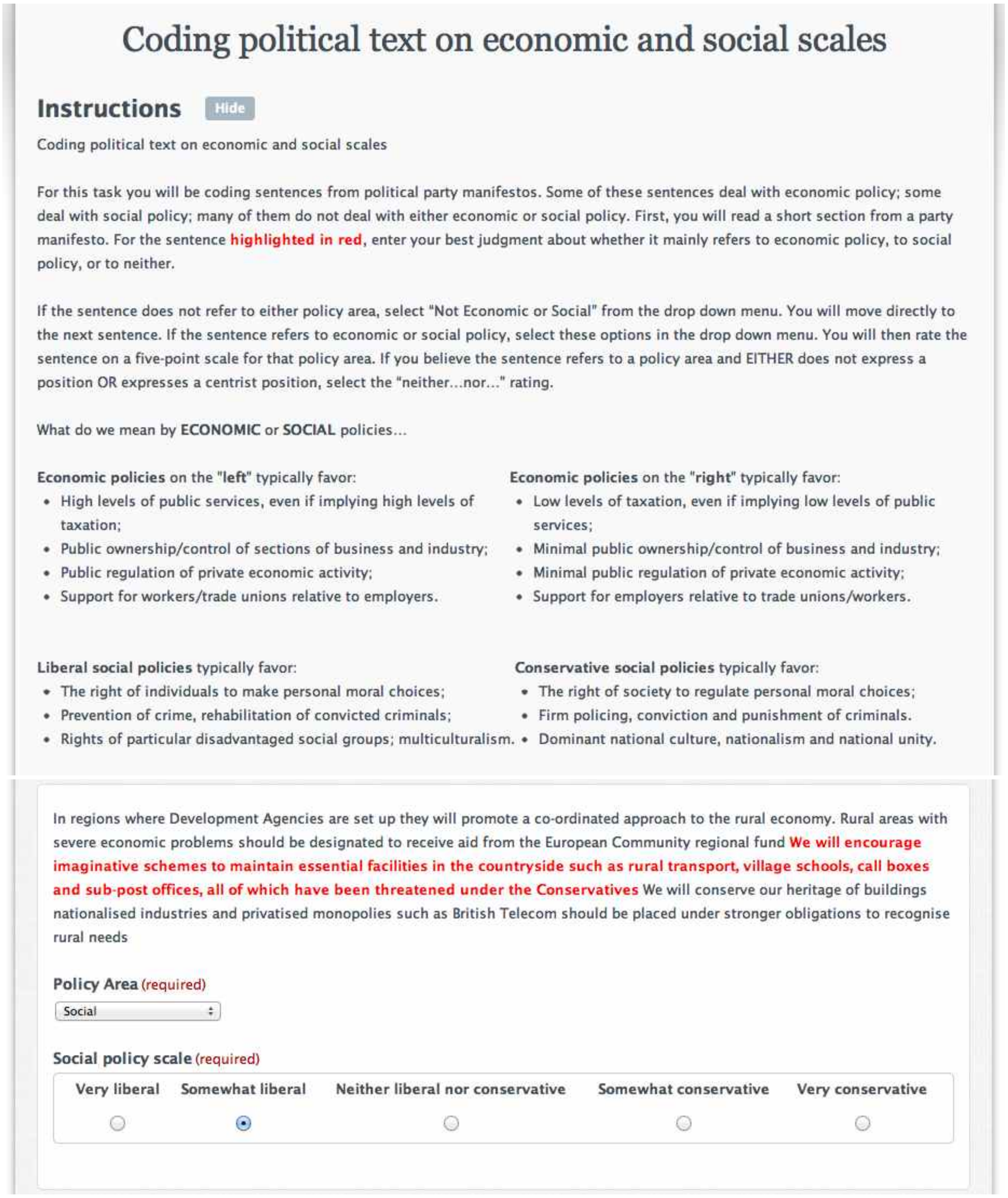
Odds ratios (95% confidence intervals), *** p<0.01, ** p<0.05, * p<0.1

Table A3: Coder misjudgment (within scale)

	(4)	(5)	(6)	(7)
	Coded [-1, 0, 1]		Coded [-2, -1, 0, 1, 2]	
Independent Variable	Economic	Social	Economic	Social
Con 1987	8.541***	158.7***	9.939***	286.8***
	(4.146 - 17.60)	(79.86 - 315.4)	(4.050 - 24.39)	(87.86 - 936.4)
Lab 1987	0.867	0.902	1.066	2.268
	(0.386 - 1.946)	(0.409 - 1.993)	(0.444 - 2.556)	(0.478 - 10.77)
Con 1997	5.047***	4.248***	4.385***	10.80***
	(2.485 - 10.25)	(1.754 - 10.29)	(2.063 - 9.320)	(2.919 - 39.97)
LD 1997	0.953		1.089	
	(0.493 - 1.841)		(0.546 - 2.171)	
Lab 1997	3.274***	328.0***	4.554***	1,004***
	(1.623 - 6.604)	(146.1 - 736.5)	(2.087 - 9.941)	(246.1 - 4,099)
Context	0.386***	1.113	0.389***	1.218
	(0.218 - 0.685)	(0.719 - 1.724)	(0.211 - 0.719)	(0.408 - 3.637)
Context * Con 1987	2.675**	0.834	3.425**	0.972
	(1.225 - 5.841)	(0.414 - 1.682)	(1.258 - 9.327)	(0.270 - 3.497)
Context * Lab 1987	0.62	2.772**	0.373**	3.184
	(0.263 - 1.463)	(1.114 - 6.895)	(0.144 - 0.968)	(0.592 - 17.12)
Context * Con 1997	3.734***	1.106	3.713***	0.805
	(1.806 - 7.719)	(0.422 - 2.900)	(1.716 - 8.036)	(0.193 - 3.362)
Context * LD 1997	2.785***		2.645***	
	(1.395 - 5.557)		(1.280 - 5.468)	
Context * Lab 1997	1.008	0.855	0.846	0.713
	(0.487 - 2.088)	(0.425 - 1.721)	(0.378 - 1.894)	(0.184 - 2.763)
Title	0.506***	0.857	0.557**	0.87
	(0.331 - 0.773)	(0.585 - 1.256)	(0.346 - 0.896)	(0.326 - 2.320)
Title * Con 1987	1.920**	1.133	2.309**	1.252
	(1.114 - 3.306)	(0.614 - 2.089)	(1.105 - 4.825)	(0.393 - 3.983)
Title * Lab 1987	1.211	0.672	1.16	0.954
	(0.639 - 2.295)	(0.350 - 1.293)	(0.510 - 2.639)	(0.299 - 3.041)
Title * Con 1997	1.891**	2.080*	1.446	2.492
	(1.086 - 3.292)	(0.971 - 4.457)	(0.778 - 2.690)	(0.734 - 8.459)
Title * LD 1997	1.35		1.205	
	(0.793 - 2.299)		(0.675 - 2.149)	
Title * Lab 1997	1.439	0.618	1.236	0.549
	(0.826 - 2.505)	(0.347 - 1.101)	(0.676 - 2.260)	(0.169 - 1.787)
Sequential	0.842	0.84	0.843	0.802
	(0.680 - 1.044)	(0.639 - 1.104)	(0.658 - 1.080)	(0.529 - 1.218)
Observations	2,370	1,481	2,370	1,481

Note: LD 1987 is base category. Odds ratios (95% CIs), *** p<0.01, ** p<0.05, * p<0.1

Figure A1: Screenshots of text coding platform, implemented in CrowdFlower



Coding political text on economic and social scales

Instructions

Coding political text on economic and social scales

For this task you will be coding sentences from political party manifestos. Some of these sentences deal with economic policy; some deal with social policy; many of them do not deal with either economic or social policy. First, you will read a short section from a party manifesto. For the sentence **highlighted in red**, enter your best judgment about whether it mainly refers to economic policy, to social policy, or to neither.

If the sentence does not refer to either policy area, select "Not Economic or Social" from the drop down menu. You will move directly to the next sentence. If the sentence refers to economic or social policy, select these options in the drop down menu. You will then rate the sentence on a five-point scale for that policy area. If you believe the sentence refers to a policy area and EITHER does not express a position OR expresses a centrist position, select the "neither…nor…" rating.

What do we mean by **ECONOMIC** or **SOCIAL** policies…

Economic policies on the "**left**" typically favor:
- High levels of public services, even if implying high levels of taxation;
- Public ownership/control of sections of business and industry;
- Public regulation of private economic activity;
- Support for workers/trade unions relative to employers.

Economic policies on the "**right**" typically favor:
- Low levels of taxation, even if implying low levels of public services;
- Minimal public ownership/control of business and industry;
- Minimal public regulation of private economic activity;
- Support for employers relative to trade unions/workers.

Liberal social policies typically favor:
- The right of individuals to make personal moral choices;
- Prevention of crime, rehabilitation of convicted criminals;
- Rights of particular disadvantaged social groups; multiculturalism.

Conservative social policies typically favor:
- The right of society to regulate personal moral choices;
- Firm policing, conviction and punishment of criminals.
- Dominant national culture, nationalism and national unity.

In regions where Development Agencies are set up they will promote a co-ordinated approach to the rural economy. Rural areas with severe economic problems should be designated to receive aid from the European Community regional fund **We will encourage imaginative schemes to maintain essential facilities in the countryside such as rural transport, village schools, call boxes and sub-post offices, all of which have been threatened under the Conservatives** We will conserve our heritage of buildings nationalised industries and privatised monopolies such as British Telecom should be placed under stronger obligations to recognise rural needs

Policy Area (required)

Social

Social policy scale (required)

Very liberal	Somewhat liberal	Neither liberal nor conservative	Somewhat conservative	Very conservative
○	●	○	○	○

Figure A2: Example of CML used to generate coding tasks

```
<p>
{{pre_sentence}}  <strong><font color="red">
{{sentence_text}}</font></strong>
{{post_sentence}}</p>
<cml:select label="Policy Area" class="" instructions="" id=""
validates="required" gold="true" name="policy_area">
  <cml:option label="Not Economic or Social" id=""
name="1"></cml:option>
  <cml:option label="Economic" value="2" id=""></cml:option>
  <cml:option label="Social" value="3" id=""></cml:option>
</cml:select>

 <cml:ratings class="" from="" to="" label="Economic policy
scale" points="5" name="econ_scale" only-if="policy_area:[2]"
gold="true">
  <cml:rating label="Very left" value="-2"></cml:rating>
  <cml:rating label="Somewhat left" value="-1"></cml:rating>
  <cml:rating label="Neither left nor right"
value="0"></cml:rating>
  <cml:rating label="Somewhat right" value="1"></cml:rating>
  <cml:rating label="Very right" value="2"></cml:rating>
</cml:ratings>

<cml:ratings class="" from="" to="" label="Social policy scale"
name="soc_scale" points="5" only-if="policy_area:[3]"
gold="true">
  <cml:rating label="Very liberal" value="-2"></cml:rating>
  <cml:rating label="Somewhat liberal" value="-1"></cml:rating>
  <cml:rating label="Neither liberal nor conservative"
value="0"></cml:rating>
  <cml:rating label="Somewhat conservative"
value="1"></cml:rating>
  <cml:rating label="Very conservative" value="2"></cml:rating>
</cml:ratings>
```

REFERENCES

Alonso, O., and R. Baeza-Yates. 2011. "Design and Implementation of Relevance Assessments Using Crowdsourcing." In *Advances in Information Retrieval*, ed. P. Clough, C. Foley, C. Gurrin, G. Jones, W. Kraaij, H. Lee and V. Mudoch: Springer Berlin / Heidelberg.

Alonso, O., and S. Mizzaro. 2009. Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. Paper read at Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation.

Ariely, D., W. T. Au, R. H. Bender, D. V. Budescu, C. B. Dietz, H. Gu, and G. Zauberman. 2000. "The effects of averaging subjective probability estimates between and within judges." *Journal of Experimental Psychology: Applied* 6 (2):130-47.

Benoit, Kenneth, and Michael Laver. 2006. *Party Policy in Modern Democracies*. London: Routledge.

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.

Berinsky, A., G. Huber, and G. Lenz. 2012a. "Evaluating Online Labor Markets for Experimental Research: Amazon.com's Mechanical Turk." *Political Analysis*.

Berinsky, A., M. Margolis, and M. Sances. 2012b. "Separating the Shirkers from the Workers? Making Sure Respondents Pay Attention on Internet Surveys." In *NYU CESS 5th Annual Experimental Political Science Conference*. NYU.

Bohannon, J. 2011. "Social Science for Pennies." *Science* 334:307.

Carpenter, B. 2008. "Multilevel Bayesian models of categorical data annotation."

Clemen, R., and R. Winkler. 1999. "Combining Probability Distributions From Experts in Risk Analysis." *Risk Analysis* 19 (2):187-203.

Däubler, Thomas, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2011. "Natural sentences as valid units for coded political text." *British Journal of Political Science* Forthcoming.

Dawid, A., and A. Skene. 1979. "Maximum likelihood estimation of observer error-rates using the EM algorithm." *Applied Statistics* 28 (1):20-8.

Eickhoff, C., and A. de Vries. 2012. "Increasing cheat robustness of crowdsourcing tasks." *Information Retrieval* 15:1-17.

Galton, F. 1907. "Vox Populi." *Nature* 75:450-1.

Goodman, Joseph, Cynthia Cryder, and Amar Cheema. forthcoming. "Data Collection in a Flat World: Strengths and Weaknesses of Mechanical Turk Samples." *Journal of Behavioral Decision Making*.

Hooghe, Liesbet, Ryan Bakker, Anna Brigevich, Catherine de Vries, Erica Edwards, Gary Marks, Jan Rovny, Marco Steenbergen, and Milada Vachudova. 2010. " Reliability and Validity of Measuring Party Positions: The Chapel Hill Expert Surveys of 2002 and 2006." *European Journal of Political Research.* 49 (5):687-703.

Horton, J., D. Rand, and R. Zeckhauser. 2011. "The online laboratory: conducting experiments in a real labor market." *Experimental Economics* 14:399-425.

Howe, Jeff. 2006. "The Rise of Crowdsourcing." *Wired* June.

Howe, Jeff. 2008. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*: Crown Business.

Hsueh, P., P. Melville, and V. Sindhwani. 2009. Data quality from crowdsourcing: a study of annotation selection criteria. Paper read at Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing.

Ipeirotis, Panagiotis, F. Provost, V. Sheng, and J. Wang. 2010. "Repeated Labeling Using Multiple Noisy Labelers." NYU Working Paper.

Kapelner, A., and D. Chandler. 2010. Preventing satisficing in online surveys: A `kapcha' to ensure higher quality data. Paper read at The World's First Conference on the Future of Distributed Work (CrowdConf 2010).

Kazai, G. 2011. "In Search of Quality in Crowdsourcing for Search Engine Evaluation." In *Advances in Information Retrieval*, ed. P. Clough, C. Foley, C. Gurrin, G. Jones, W. Kraaij, H. Lee and V. Mudoch: Springer Berlin / Heidelberg.

Laver, M. 1998. "Party policy in Britain 1997: Results from an expert survey." *Political Studies* 46 (2):336-47.

Laver, Michael, Kenneth Benoit, and John Garry. 2003. "Estimating the policy positions of political actors using words as data." *American Political Science Review* 97 (2):311-31.

Laver, Michael, and W. Ben Hunt. 1992. *Policy and party competition*. New York: Routledge.

Lawson, C., G. Lenz, A. Baker, and M. Myers. 2010. "Looking Like a Winner: Candidate appearance and electoral success in new democracies." *World Politics* 62 (4):561-93.

Lorenz, J., H. Rauhut, F. Schweitzer, and D. Helbing. 2011. "How social influence can undermine the wisdom of crowd effect." *Proceedings of the National Academy of Sciences* 108 (22):9020-5.

Lowe, Will, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2011. "Scaling policy positions from hand-coded political texts." *Legislative Studies Quarterly* Forthcoming.

Mason, W, and S Suri. 2012. "Conducting Behavioral Research on Amazon's Mechanical Turk." *Behavior Research Methods* 44 (1):1-23.

Mikhaylov, Slava, Michael Laver, and Kenneth Benoit. 2012. "Coder reliability and misclassification in comparative manifesto project codings." *Political Analysis* 20 (1):78-91.

Miller, B., and M. Steyvers. 2011. The Wisdom of Crowds with Communication. Paper read at Proceedings of the 33rd Annual Conference of the Cognitive Science Society, at Austin, TX.

Nowak, S., and S. Rger. 2010. How reliable are annotations via crowdsourcing? a study about inter-annotator agreement for multi-label image annotation. Paper read at The 11th ACM International Conference on Multimedia Information Retrieval, 29-31 Mar 2010, at Philadelphia, USA.

Paolacci, Gabriel, Jesse Chandler, and Panagiotis Ipeirotis. 2010. "Running experiments on Amazon Mechanical Turk." *Judgement and Decision Making* 5:411-9.

Raykar, V. C., S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bo- goni, and L. Moy. 2010. "Learning from crowds." *Journal of Machine Learning Research* 11:1297-322.

Ribeiro, F., D. Florencio, C. Zhang, and M. Seltzer. 2011. crowdMOS: An Approach for Crowdsourcing Mean Opinion Score Studies. Paper read at 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), at Prague.

Sheng, V., F. Provost, and Panagiotis Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. Paper read at Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Snow, R., B. O'Connor, D. Jurafsky, and A. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. Paper read at Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Steyvers, M., M. Lee, B. Miller, and P. Hemmer. 2009. "The wisdom of crowds in the recollection of order information." In *Advances in neural information processing systems, 23*, ed. J. Lafferty and C. Williams. Cambridge, MA: MIT Press.

Surowiecki, J. 2004. *The Wisdom of Crowds*. New York: W.W. Norton & Company, Inc.

Wang, J., Panagiotis Ipeirotis, and F. Provost. 2011. "Managing Crowdsourcing Workers." In *The 2011 Winter Conference on Business Intelligence*. Utah, March 10-12.

Welinder, P., S. Branson, S. Belongie, and P. Perona. 2010. The multidimensional wisdom of crowds. Paper read at Advances in Neural Information Processing Systems 23 (NIPS 2010).

Welinder, P., and P. Perona. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels. Paper read at IEEE Conference on Computer Vision and Pattern Recognition Workshops (ACVHL).

Whitehill, J., P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. Paper read at Advances in Neural Information Processing Systems 22 (NIPS 2009).

Yi, S.K.M., M. Steyvers, M. Lee, and Matthew Dry. forthcoming. "The Wisdom of the Crowd in Combinatorial Problems." *Cognitive Science*:1-19.