CHAPTER 18

Coding a Training Set

Human coding has long been used to organize and quantify texts. Once a researcher has defined categories, human coding is the process of putting these documents into these categories manually. While human coding predates modern approaches to statistical text analysis, it is still extremely useful on its own for small datasets and coding of complex concepts. Learning to write an excellent codebook and train coders is also important in the context of automated methods as manual coding often forms the basis of training and validation sets for statistical classifiers.

Human coding provides a mapping function from the text to the categories of interest, one that the humans produce based on their interpretation of the text. This mapping function is the combination of the codebook, training given to the coders, and their internal thought processes when assigning codes. In comparison to statistical classification, humans can use their substantive and background knowledge to fill in gaps in the coding scheme using the context in the text. Even though human coding by itself is much slower than automated classification, the process of human coding can also be useful for clarifying the categories of interest and how they are communicated.

While manually placing documents into categories might seem at first to be a relatively straightforward task, human coding can be quite complex.[1] Ambiguities in language, limited attention of coders, and nuanced concepts make the reliable classification of documents difficult—even for expert coders (Grimmer and Stewart, 2013). Unfortunately, teams of expert coders are few and far between, and the modal manual content analysis project in the social sciences is, as Schrodt (2006) colorfully put it describing the collection of events data, "legions of bored students flipping through copies of *The New York Times*" (p. 2). Complications arise because of the deeply contextual nature of language that makes it difficult to specify an entire codebook ex ante. For this reason, we recommend an explicit exploratory/discovery phase in which a preliminary and concise codebook is written to guide coders, who then apply the codebook to an initial set of documents. When using the codebook—particularly at first—coders are likely to identify ambiguities in the coding scheme or overlooked categories. This subsequently leads to a revision of the codebook. Only after coders apply the coding scheme to documents without noticing substantial ambiguities is a "final" scheme ready to be applied to a separate set of documents used for analysis. This ensures that ambiguities

[1] Part of this section is drawn from Grimmer and Stewart (2013).

have been sufficiently addressed without risk of overfitting to the document set used to develop the codebook.

In this chapter, we first cover the basics of developing a codebook, selecting coders, and coding a training set. We then also consider other sources of labeled data, including crowd sourcing and supervision with found data. Manual coding is a very important topic, but we cannot give a full treatment of it in this book. Fortunately, manual content analysis is a richly developed field and there are whole books on how to write a good codebook and train coders. We particularly recommend Krippendorff (2012), Neuendorf (2016), and Riffe et al. (2019) for more detailed treatments of the coding process along with practical advice. Barberá et al. (2021) provide an excellent article-length treatment of the whole supervised learning process.

18.1 Characteristics of a Good Training Set

The goal of human coding is to provide a reliable human mapping from the texts of interest to the categories the researcher is interested in. In Part 3: Discovery, we discovered the concepts we want to measure; in coding a training set, we want to *operationalize* these concepts. Many complications arise when creating this mapping, as we will detail more below. Neuendorf (2016, Chapter 1) nicely lays out the following characteristics of a good human coded dataset or training set:

- **Objectivity-Intersubjectivity**: The measurement of the categories is objective, in that the understanding of the categories is not specific to a particular person. Even if it cannot be objective, Neuendorf (2016) argues that at least it should have intersubjectivity, or a shared understanding between researchers.
- **An A Priori Design**: As we have discussed in Part 3: Discovery, defining and redefining categories is an important part of the research process. In the process of human coding, categories may need to be further refined. However, the final training set should be based on a final codebook and coded on fresh data.
- **Reliability**: The mapping of texts to categories should be reliable, in that different human coders should produce the same mapping when working independently.
- **Validity**: The measure should reflect the concept or category of interest. The category label we assign to the measure should reflect what we are indeed measuring.
- **Generalizability**: To create a training set, we will typically code only a sample of a much larger set of data. The mapping we produce with hand coding should be generalizable to the entire dataset and the final population the researcher is interested in.
- **Replicability**: The measure should be able to be replicated in the same data, and ideally different data and different contexts as well.

18.2 Hand Coding

With these goals in mind, we first consider the case where the researcher creates a training set using hand coding. Depending on the categories the researcher intends to measure and the size of the sample set aside for hand coding, the process of hand coding can be very straightforward to a process that takes weeks, months, or even years

(Baumgartner, Jones, and MacLeod, 1998). Regardless, there are a few basic steps and decisions necessary to create a human coded dataset:

18.2.1 1: DECIDE ON A CODEBOOK

The codebook is the instruction manual for the coders to place documents into categories. If the categories have been created by the researcher, she will have to write her own codebook. The codebook not only defines the categories of interest, but also must communicate them effectively in order to ensure reliability, validity, and replicability. This may mean providing example texts for each category and describing edge cases.[2] Flowcharts can also be used to help the coders understand hierarchy of the categories or deal with cases where there are many categories (Krippendorff, 2012, p. 135).

If the researcher would rather not create a custom codebook from scratch, many codebooks exist that could coincide with the goal of the research project. The advantage of adapting an existing codebook is that it has often been refined through multiple rounds of testing. For example, the Comparative Manifestos Project (https://manifesto-project.wzb.eu/) provides several versions of codebooks for categorizing the policy preferences of election manifestos.

18.2.2 2: SELECT CODERS

In order to ensure intersubjectivity, reliability, and replicability, one of the principles of human coding is that coders other than the researcher label the data (Krippendorff, 2012, p. 131). Using a separate set of coders is a test of whether the researcher can effectively communicate the categories to other people.[3]

The next question is how many coders to use to label the data. Best practice is for the researcher to select at least two for creating the training data. This allows the researcher to assess the level of agreement between the coders as a measure of reliability, which we discuss in more detail below. The total number of coders the researcher should select will depend on how precise the coders are in labeling the data. With trained coders with a lot of experience in the research area, researchers may be able to use only two coders total. With a less trained or less attentive labor pool where coders may be more imprecise, many evaluations of each data point may be necessary to reliably label the data. Barberá et al. (2021) provide an excellent overview and characterization of these trade-offs in selecting the number of coders.

18.2.3 3: SELECT DOCUMENTS TO CODE

Next, the researcher must select which and how many documents to code. For generalizability and best performance of the classifier, the training set should be representative of the larger dataset that the researcher intends to code and also of the population about which the researcher intends to draw inference. One straightforward way to ensure this is to draw a simple random sample of the larger set for the training set.

The number of documents that should be used in the training set will depend on the number of categories and the reliability of the coders. The higher the number of categories and the lower the reliability of the coders, the higher the number of documents

[2] See Neuendorf (2016) Chapter 5 for the trade-offs of providing example texts.

[3] We acknowledge that this may not always be possible due to funding considerations or poor access to coders with sufficient expertise to perform the coding.

the researcher will need to create a reliable classifier downstream (Barberá et al., 2021). Downstream validation of the classifier can help reveal whether a sufficient number of documents has been coded to achieve the desired accuracy.

18.2.4 4: MANAGE CODERS

For best performance, researchers should train coders before asking them to label the training set. This involves having the coders carefully read the codebook and ask any questions. It often also involves asking the coders to label a sample of texts and evaluating whether they have understood the instructions, or whether the instructions need to be revised.

Once coders have been trained and the codebook has been finalized, the coders should label the data without any contact with each other, or with resources outside of the codebook itself (Krippendorff, 2012, p. 131). This ensures that the training data produced can be evaluated for reliability. The final training data should be produced after the last revision of the codebook.

18.2.5 5: CHECK RELIABILITY

The last step in creating a human coded training set is checking for intercoder reliability.[4] This involves comparing the labels on the same documents between coders. Several different measures can be used to compare labels; two widely used metrics are Cohen's *kappa* (Landis and Koch, 1977) and Krippendorff's *alpha* (Krippendorff, 2012). We refer the reader to Chapter 6 of Neuendorf (2016) for an excellent overview of the metrics and pitfalls of assessing reliability.

18.2.6 MANAGING DRIFT

Researchers should be aware of two types of "drift" that might occur when human coding. First, the coders themselves may change the way that they assign labels to categories as the process of coding wears on. This could be due to anything from fatigue with coding to new insights gained during the process of coding (Neuendorf, 2016, Chapter 6). For this type of drift, it might be useful for the researcher to assess reliability at multiple points in the coding process.

Human coding can also be affected by data drift—where a content stream is evolving over time, which can create new ambiguities for the codebook. For example, new categories might evolve, collapse, or merge with others as the data changes over time. If the codebook is changed based on the drift, it could be that the newer data is labeled differently than the old data. In this situation, one final coding round can help assess the final reliability of the new codebook (Krippendorff, 2012, 218).

18.2.7 EXAMPLE: MAKING THE NEWS

In *Making the News*, Amber Boydstun considers the important question of what types of policy issues receive the most news coverage and why. Boydstun (2013) argues that the media do not act purely as watchdogs—patrolling all issue areas—or as alarm systems—only responding to sudden events. She instead puts forward an alarm/patrol hybrid model of news generation, one where the media are monitoring many issue

[4]Note that intercoder reliability does not guarantee accuracy, because coders could have similar errors (Barberá et al., 2021). However, without access to "true" labels, true accuracy is difficult to assess.

areas, but sink disproportionate resources into a few when they become "hot." To measure attention to policy issues in news, she collects a dataset of 31,034 articles from *The New York Times* and places them into policy issue areas. Because she is most interested in understanding policy areas that receive the most news attention, she limits her analysis to front page stories.

Boydstun (2013) relies on an existing codebook—the Policy Agendas Project—to label each of the front page *New York Times* articles in her dataset. In order to adapt the codebook to *The New York Times* corpus, she annotated the codebook to provide more explicit instructions to the coders. The codebook consists of many categories—27 major topic categories and 225 subtopics within these 27 major categories.

Boydstun (2013) trained the coders on a sample of articles, requiring the coders to obtain high reliability in this sample before coding the rest of the data. In order to ensure reliability over time, Boydstun (2013) ensured that coders labeled duplicate texts throughout the coding process. Boydstun (2013) reports both Krippendorff's alpha and Cohen's kappa and has made her codebook available online.

Boydstun (2013)'s hand coding of front page *New York Times* articles allows her to study attention to the different policy issues over time. She finds that issue coverage is highly skewed toward international affairs and often driven by events. For example, Figure 18.1 from Boydstun (2013, Chapter 4) shows that the attacks on the World Trade Center in September of 2001 drastically shifted media coverage on the front page of *The New York Times* to defense, crowding out issue coverage of other policy areas.

18.3 Crowdsourcing

In the last ten years, the introduction of online labor markets such as Mechanical Turk has radically altered the landscape of recruitment for annotators, survey respondents, and participants in experiments (Snow et al., 2008; Kittur, Chi, and Suh, 2008; Buhrmester, Kwang, and Gosling, 2011; Berinsky, Huber, and Lenz, 2012; Budak, Goel, and Rao, 2016; Benoit et al., 2016). Labor markets provide access to untrained or lightly trained workers at scales that would be unfathomable in a typical university setting. For tasks that can be easily explained, are relatively straightforward, and can be quickly completed, online labor markets can be a valuable way to collect document annotations.

It is tempting to think about these online labor markets as infinitely large pools of workers but they are (at least at the time of writing) actually much smaller than it might otherwise seem. For example, Amazon's Mechnical Turk is the most popular online labor market for academic research. A recent study estimates that there are less than two hundred thousand unique workers, with about two to five thousand active at any given time (Difallah, Filatova, and Ipeirotis, 2018). These workers tend to have skewed demographic and political characteristics relative to the population as a whole, although certainly not as skewed as the population of students in university settings (Huff and Tingley, 2015).

Differences between crowdsourcing and human coders. There are three big differences between working with a small team of human coders and a huge crowd. First, in the crowd setting fixed costs have to be lower, so intensive training is less feasible. In practical terms this limits the kind of work that can be done by crowd-workers to comparatively simple tasks that don't require expert background. The advantages of low fixed costs lead to the second big difference—it is more feasible to quickly scale up crowd workers to enable high throughput coding. In a university or industry setting,

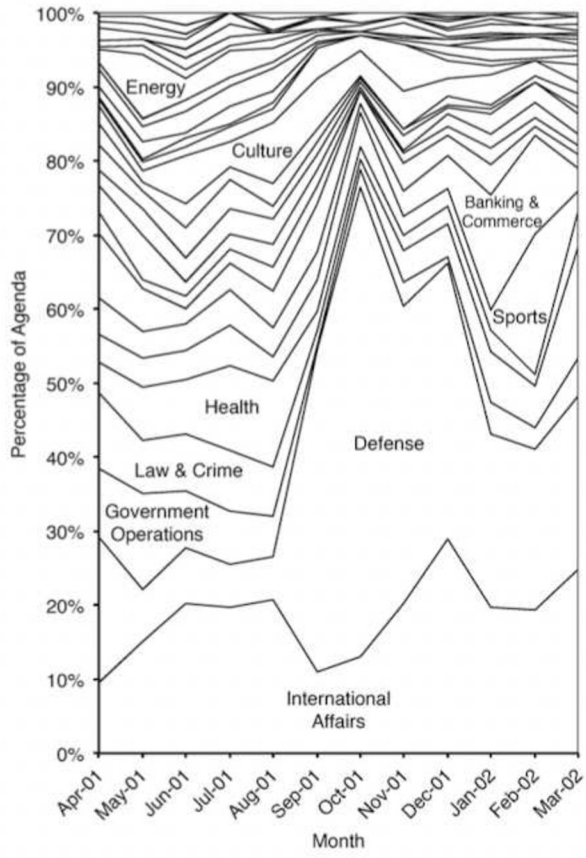
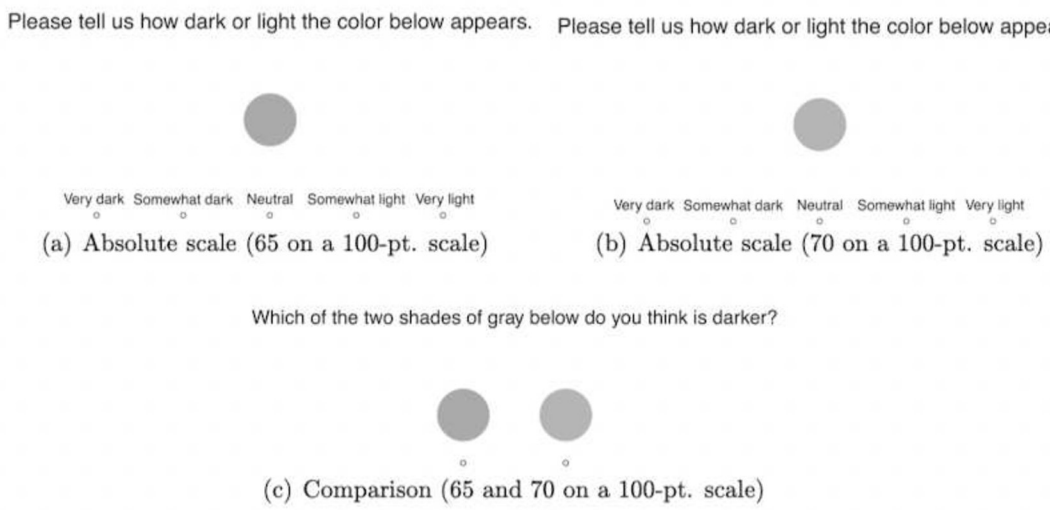


Figure 18.1. Discussion of defense in front page news articles in *The New York Times* increased substantially after September 11, 2001. Figure 4.8 from Boydstun (2013).

hiring new workers takes time, but with crowdsourcing thousands of annotations can be collected for a simple task in just a few hours. The third major difference, is that we have to fully embrace the inevitability of error in annotations. We can conceivably train a small team of annotators until error rates are tolerably low, but in crowdsourcing we have to come up with ways to reconcile conflicting labels. Thankfully this is an old problem which has seen renewed interest in recent years as a result of online labor markets (Dawid and Skene, 1979; Sheng, Provost, and Ipeirotis, 2008; Zhang et al., 2014; Benoit et al., 2016; Barberá et al., 2021). We return to these methods in Chapter 20 on validation.

Some would add a fourth major difference, that crowdsourcing is cheaper. It is certainly true that crowdsourced annotations *can* be obtained cheaper on Mechanical Turk than from undergraduate annotators, but it is questionable whether or not they should be. Because the workers are independent contractors, they often end up being paid effectively below local minimum wage. We note that while for the purposes of statistical text analysis it can be helpful to think of the crowd as a certain type of "algorithm," the workers are very much humans and so ethical considerations about appropriate compensation apply (Fort, Adda, and Cohen, 2011; Mason and Suri, 2012; Shank, 2016).

Changing the Task. In practice, the key to using crowdsourcing has generally been to find the optimal way to divide the coding task into small "bite-sized" chunks. Now that online labor markets have been around for over a decade, researchers have



(a) Absolute scale (65 on a 100-pt. scale)

(b) Absolute scale (70 on a 100-pt. scale)

(c) Comparison (65 and 70 on a 100-pt. scale)

Figure 18.2. A figure from Carlson and Montgomery (2017) explaining how pairwise comparison tasks can be more tractable than absolute scale tasks. Task C is going to be much more consistent than either task A or B.

started to creatively develop new tasks that can more easily be discretized. Carlson and Montgomery (2017) introduce an approach to placing documents on a scale by collecting thousands of responses to quick pairwise comparison tasks.

The core insight is beautifully illustrated by an example from Carlson and Montgomery (2017), which is depicted in Figure 18.2. For many tasks, such as assessing how dark or light a color is, it is quite difficult to assign an absolute scale without some kind of reference point. However, it is very tractable to make a reliable assessment of a comparison between two values. These pairwise comparisons can in turn be used to recover an underlying latent dimension—an idea that has been around in psychometrics since at least Thurstone (1927), and, arguably, traces back to the 1800s.

In Carlson and Montgomery (2017), they collect many such pairwise judgments by recruiting crowd workers through Amazon's Mechanical Turk platform. They show that the system is highly reliable and compares favorably to other approaches. Remarkably, coders seem to need considerably less training to make a pairwise judgment than to accurately assign an absolute scale. The need to make pairwise comparisons causes the number of evaluations needed to scale with the number of texts squared, so this general approach works best with a moderate number of documents (the largest case is less than 2,000 documents).

It is worth emphasizing that the task in Carlson and Montgomery (2017) is not fully supervised in the sense that at no point is any unit labeled with its actual "true" value. Rather we use indirect feedback about the latent dimension (e.g., which gray is darker) to provide insight on the intended measurement. This leaves these styles of measurement as a bit of a middle ground between supervised categorization approaches and unsupervised approaches, and it is worth careful investigation that the continuous dimension that is learned has the intended outcome.

18.4 Supervision with Found Data

The last strategy we describe for building a training set is "supervision with found data," which involves using found data in order to determine the categories and label the documents. This approach uses existing categories produced by individuals outside