The Structure of Predictive Coding: A Guide for the Perplexed

R. T. Oehrle & E. A. Johnson
Chief Linguist Executive Director
Ernst & Young LLP[1]

The fact that 'predictive coding' (*PC*) is not a well-defined term hasn't impeded the popularity of whatever it refers to. Since Legal Tech 2012, *PC* is everywhere: in the press, in the courts, and especially, in press releases and marketing materials. Whatever it is, people want it.

One way to think of *PC* is as a response to the convergence of dynamic market forces:

- volume continues to skyrocket, with no sign of stalling;
- estimates of the quality of human linear review are trending downward;
- comparisons of human linear review to technology assisted review have been favorable to the technology side;
- courts are beginning to recognize the value of *PC* technology in review, in a way related to the recognized importance of 'proportionality' of costs to value.

From this perspective, the core attribute of *PC* is the de-linking of volume and cost, while maintaining or improving quality. But there are other valuable perspectives on *PC*. At first glance, the range of available *PC* systems and solutions is bewildering. A closer look reveals that the landscape is governed by a small number of structural choices—choices posed by the problems that predictive coding confronts. The goal of this paper is to isolate these structural choices and to show how particular responses to them provide a characterization of both the predictive coding systems themselves and also their range of applications. One justification for this goal is simple: characterizing the structural choices of *PC* solutions yields a clearer delineation of the range of systems available in principle and a clearer understanding of the different advantages and disadvantages of the particular systems available in the current e-Discovery marketplace. A core reason to focus on the range and compatibility of structural choices rather than particular technologies is that some of the relevant technology components play different structural roles in different structural configuations.

A note on terminology: the terms 'predictive coding', 'technology assisted review' (TAR), and sometimes 'computer assisted review' (CAR) are at times conflated. In what follows, we use the term 'predictive coding' in a specific sense to denote review systems and protocols in which one or more documents are coded or labeled with review-sensitive information ('responsive' (R), 'non-responsive' (NR), etc.) without direct document-particular human supervision. (Of course, valuable systems of predictive coding will apply non-individual document coding to many more documents than just one. But we can draw a principled line at a single document.) In contrast, in human linear review—non-predictive coding—every label on every document is supplied through direct human judgment. In this terminology, then, 'predictive coding' stands in direct contrast to human linear review. If we take 'technology assisted review' and 'computer assisted review' in their natural interpretations, these phrases apply to any form of review assisted or aided by technology or computation—including many systems that do not apply any form of predictive coding. (Anyone using MD5 hashes for review deduplication is using TAR/CAR. But they're not necessarily using *PC*.)

[1]The views expressed in this paper are those of the authors and do not necessarily represent the views of Ernst & Young LLP.

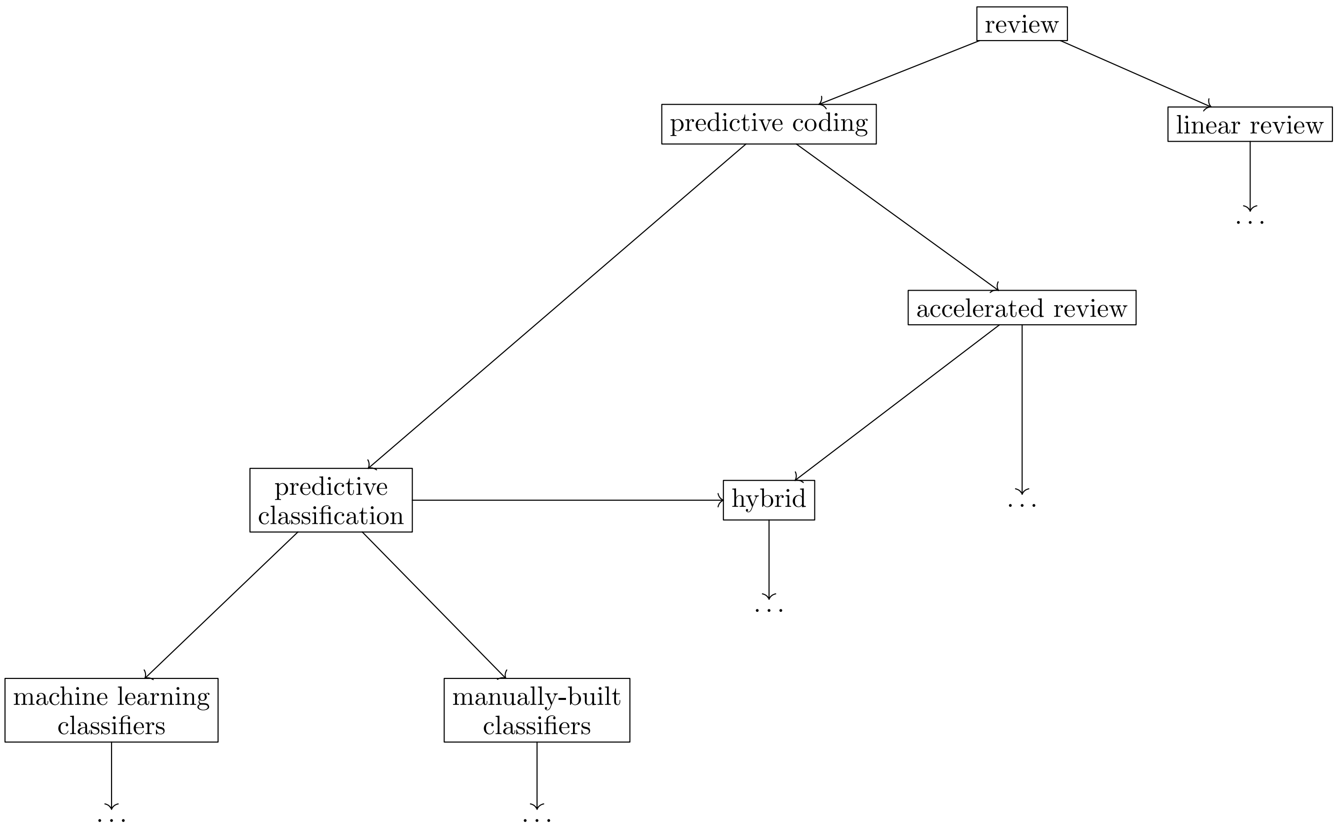
review
predictive coding
linear review
accelerated review
predictive classification
hybrid
machine learning classifiers
manually-built classifiers
...

Figure 1: the basic structural landscape of review

1 The Basic Landscape

The EDRM has released a draft of a Computer Assisted Review Reference Model, aiming to document the steps in a generic vision of technology assisted review.[2] But the focus on the steps in the process glosses over the critical nature of what the possible steps are and what constrains their combinations. In our sketch of the Basic Landscape in Figure 1, we emphasize these underlying structural choices.

The basic problem of E-Discovery is to identify within a universe of potentially responsive documents the subset that is responsive to a specification of some kind (a request for production, a subpoena, a 2nd request, ...). Thus, we have two basic models: a model of the documents and a model of the request for production (or: '*RFP*'). Review and production depends on both models. A single document collection can be the universe of potentially responsive documents for a variety of different matters. If the *RFP*'s of these matters are different, productions are likely to be different as well. In general, production does not depend only on the document collection; it depends equally on the *RFP*. Not surprisingly, then, the way in which these two models—the document model and the *RFP* model—are brought into contact in review structurally illuminates many intuitive and familiar distinctions in the review landscape.

[2]http://www.edrm.net/resources/carrm.

1.1 linear review

In linear review, reviewers who have absorbed the information contained in the request for production, either directly or through a review manual, label each document as responsive or not responsive to the *RFP*. No label is predicted. By our definition above, this is not predictive coding.

Let D represent the input set of documents, P the production set, and $\bar{P}$ the non-produced subset of D. We can represent the abstract form of this basic process as follows, where the ellipsis signs represent unknown or unspecified sources or targets of one or more maps, starting from the document collection D and ending at the disjoint union $P \sqcup \bar{P}$.[3]

$$\begin{tikzcd} D \arrow[r] & \ldots \arrow[r] & P \sqcup \bar{P} \end{tikzcd}$$

From this point of view, we can model human linear review as a two-step process. In the first step, the documents D are mapped to the disjoint union $D_R \sqcup D_{NR}$—a partition of D into two disjoint sets whose union is D. We label the first step *h,rfp* in honor of rfp-informed human review and the second step p for 'production':

$$\begin{tikzcd} D \arrow[r, "h{,}rfp"] & D_R \sqcup D_{NR} \arrow[r, "p"] & P \sqcup \bar{P} \end{tikzcd}$$

In the step labeled *h,rfp*, reviewers combine their knowledge of the request for production and their understanding of a document to assign one or more labels to it. Both knowledge of the request and document understanding involve sophisticated capacities of human cognition. The step labeled p is much simpler: any document in D_R goes to P; any document in D_{NR} goes to $\bar{P}$. Since any document of D belongs to exactly one of D_R or D_{NR}, composing the maps provides a path from D to the production set:

$$\begin{tikzcd} D \arrow[r, "h{,}rfp \circ p"] & P \sqcup \bar{P} \end{tikzcd}$$

1.2 predictive coding

In predictive coding, by our definition, it is not the case that every label on every document is supplied by human reviwers: the arrow labeled *h,rfp* above is to be replaced by automated steps in some way. But how? We would like to be able to specify all the possibilities: this is the solution space to this problem.

There is a useful and relevant distinction in machine learning ([Mohri et al., 2012], for example) which separates *examples* (individual data instances), *features* (the relevant set of properties by which examples are characterized), and *labels* (additional values associated with examples). In the context of e-discovery, it is easy to see what the examples are (individual members of the document set D) and easy to see what the labels are (*responsive*, *non-responsive*, *privileged*, ...). It is less easy to see what corresponds to the

[3]A *map* or *function* from D to P, say, associates to each element in D a unique element in P. For present purposes, a disjoint union is the union of two disjoint sets—sets with no elements in common.

features, first, because the choice of features is not determined by the document set, second, because its not determined by the the production requirements, and third, because not all forms of PC fit squarely within the machine learning paradigm. We will start with the how the standard labels (*responsive, non-responsive,* ...) are assigned to documents, but as our discussion continues, the structural importance of features will become more and more apparent.

The most important structural property in document review is how labels (in the sense of the previous paragraph) are associated with documents. In *linear review* in its pure form, documents are labeled by human reviewers one by one, so that the only way a document can be labeled *responsive* or *non-responsive* is through the action of a human reviewer. In predictive coding, this individual coding gives way in a variety of ways. In the immediate paragraphs that follow in this section, we briefly describe what we take to be the structural features of the landscape. In subsequent sections, we take up a number of the issues that arise from this perspective, and examine in more detail the structural details of some of the key predictive coding types we distinguish.

Focusing on how documents are labeled provides a first, principled distinction among current review methods. One of the miracles of the marketing of predictive coding is that regardless of which side of this distinction your favorite method sits on, you can still use predictive coding methods to make it better. Or faster. Or cheaper.

1.2.1 accelerated review

One way to cut down the number of touches in document labeling, while still relying on human review as the labeling mechanism, is to label sets of documents rather than each of the individual documents contained in these sets. Suppose there is a way (actually, there are many ways) to cluster similar documents together, so that if documents d_1 and d_2 are sufficiently similar in some respect, they may belong to the same cluster k in the set of clusters D_K. Intuitively, we want each document in D to belong to exactly one element of D_K. Thus, D_K is a *partition* of D: every member of D belongs to exactly one element of D_K and the union of the elements of D_K is D. The passage from D to D_K is structure enhancing. In this case, the structural enrichment consists of the implicit association of elements in M that belong to the same cluster. It is our expectation—well, at least our hope—that elements belonging to the same cluster will have other properties in common—properties like responsiveness.

$$\begin{tikzcd} D \arrow[r, "k"] & D_K \arrow[r, "{h,rfp}"] & D_R \sqcup D_{NR} \arrow[r, "p"] & P \sqcup \bar{P} \end{tikzcd}$$

There are many different approaches to clustering, involving, among other things, different measures of distances and different goals in grouping. Without describing these in detail, we make a few basic points below about the role of clustering in review:

- clustering by itself doesn't entail that documents in the same cluster are label-equivalent: a perfectly sound method of clustering is to assign documents that have the same number of occurrences of *the* in their contents, but it is unlikely that this method will produce label-equivalent clusters;
- clustering should not be dominated by irrelevant material such as signature lines, footers, embedded messages, and so on;

- if bulk coding is based on statistical sampling, the protocol should be described and adhered to; if it isn't based on statistical sampling, what is it based on?
- the bulk coding protocol should characterize the criteria by which a cluster is associated with a label, as well as the procedure to be followed when no label can be consistently applied to the members of a cluster.

In the form of clustering that we have described, the key characteristic is that every *cluster* is labeled by human review. But if a label assigned to a cluster propagates automatically to each member of the cluster, regardless of whether that member has been individually reviewed, this is not linear review. There are other paths to accelerated review—various forms of prioritized review or ranked retrieval. We address a number of these shortly below in the section on *hybrid review.*

1.2.2 predictive classification

One especially important form of predictive coding offers an alternative to human labeling: the automated or semi-automated construction of a document model M, in which the document set is mapped to a structured set of document subsets corresponding to the labels, for example M_R, M_{NR}, and possibly other sets overlapping these representing potentially privileged documents, documents tagged for particular issues, and so on. We call these forms of predictive coding *predictive classification.* Abstracting away from how the labeled model is constructed, we have the structural diagram below:

$$\begin{tikzcd} D \arrow[r] & \ldots \arrow[r] & M_R \sqcup M_{NR} \arrow[r] & \ldots \end{tikzcd}$$

On the right hand, downstream, side, there is still a critical role for human review—but this role consists of validation of predicted labeling. And the number of touches or document interactions in validation is smaller—often orders of magnitude smaller—than in human linear review, where each document is touched. When validation is successful, we can extend the diagram above as shown below.[4]

$$\begin{tikzcd} D \arrow[r] & \ldots & {} \arrow[r, "\mu"] & M_R \sqcup M_{NR} \arrow[r, "val"] & D_R \sqcup D_{NR} \arrow[r] & P \sqcup \bar{P} \end{tikzcd}$$

There is a structural property that distinguishes all forms of predictive classification from non-predictive classification. In non-predictive classification—both linear review and non-hybrid forms of accelerated review—the properties of the *RFP* come into play during the human coding process (as in the maps labeled above *h,rfp*). In predictive classification, properties of the *RFP* must enter before model validation, thus prior to the completion of the model construction step represented above by the map μ. We will explore exactly how the properties of the *RFP* enter when we discuss machine learning and manual classificaion below.

[4]We return below to the case when validation comes up short.

1.2.3 hybrid review: partial or implicit classifiers

Suppose that a possibly high-precision NR subset of a document set D is identified, so that D can be partitioned into the disjoint union $D_{NR} \sqcup (D \setminus D_{NR})$.[5] D_{NR} is thus a model (possibly a partial model) of the non-responsive subset of D. The hypothesis that it is an adequate model may be tested by sampling and validated. If the validation test succeeds, then every document in D_{NR} may be labeled NR, just as in predictive classification. If the remainder $D \setminus D_{NR}$ is linearly reviewed, then the result is a hybrid form of review: part linear, part predictive. In a diagram:

$$\begin{tikzcd}
 & & D \setminus D_{NR} \arrow[dr, "h{,}rfp"] & \\
D \arrow[r] & D_{NR} \sqcup (D \setminus D_{NR}) \arrow[ur] \arrow[dr] & & P \sqcup \bar{P} \\
 & & D_{NR} \arrow[ur, "val"] &
\end{tikzcd}$$

There are many variants of this hybrid architecture. One of them involves *ranked retrieval*, in which the document set is ranked in decreasing order of projected responsiveness, and linear review starts at the top and continues until it reaches a point where the density of responsive documents is so low that the remaining documents may be regarded as a possibly high-precision NR subset and subjected to validation testing as a whole by sampling, rather than linear review.

1.2.4 predictive classifiers: machine learning models and manually-built models

Predictive classifiers construct a model (often iteratively) of how the document set D is implicitly structured by the *RFP*. Regardless of how the model is constructured, it's quality can be quantitatively assessed and validated by statistical methods. If the quantitative assessment surpasses our validation standards, the results can be produced; if they don't, we improve the model and assess the improved result, continuing until we pass.[6] We distinguish (structurally—via the role of the *RFP* information) two basic methods of model construction.

In *machine learning classification,* human reviewers, relying on their knowledge and understanding of the *RFP*, label a sample of documents, resulting in a partition of D consisting of D_L and $D \setminus D_L$). The partition is intended to be asymmetrical: D_L is much smaller than $D \setminus D_L$. A machine-learning algorithm α then projects the sampling results represented by D_L across the rest of the data—that is, $D \setminus D_L$—in a way that creates a model in D responding to the requests in the *RFP*.

$$D \xrightarrow{h,rfp} D_L \sqcup (D \setminus D_L) \xrightarrow{\alpha} M_R \sqcup M_{NR} \xrightarrow{val} P \sqcup \bar{P}$$

There are many machine-learning algorithms available to play the role of α, surveyed below.

[5] The operator '\' is the relative complement operator: $D \setminus D_{NR}$ consists of the elements of D not in D_{NR}.

[6] Note that iterative model improvement does not entail iteration of the model construction process—another form of hybridization that we return to below.

In *manual classification,* a classifier to build the model-structure $M_R \sqcup M_{NR}$ is constructed manually. As an example, suppose the classifier consists of a set of ontology-based searches. Whether the ontologies are selected from off-the-shelf libraries or designed by hand to deal with the particular properties of D, the resulting ontology-based searches incorporate the relevant properties of the *RFP*—a fact we represent in the map below by '$\Omega(rfp)$'. Structurally, then, we have:

$$D \xrightarrow{\Omega(rfp)} M_R \sqcup M_{NR} \xrightarrow{val} P \sqcup \bar{P}$$

Just as there are many machine-learning algorithms available to play the role of α, there are also many other forms of search-based classification that can play the structual of $\Omega(rfp)$ above.

1.3 summary

The basic structural landscape contains familiar landmarks. Key point: these landmarks are distinguishable by a small number of structural oppositions.

2 predictive coding: data structures, algorithms, process

The structural view of predictive coding offered above abstracts from a broad range of particulars: particular data structures and data representations, particular algorithms applying to these structures and representations, and essential aspects of the predictive coding process involving iterative training, testing, and validation. The reason for separating structural properties from these particulars is simple: any given particular may be compatible with multiple, distinct structures; as a result, focusing on the particulars at the outset obscures the underlying structural distinctions and the clarity they provide on the predictive coding landscape as a whole. In what follows, we survey a variety of these particulars and consider their roles in the structural configurations discussed above.[7]

2.1 data structures: formal models of document collections

Electronically stored information comes in many forms, including a variety of formats for representing text. (See [Büttcher et al., 2010, pp. 9ff.] for an insightful overview.) Contemporary corporate document collections typically contain documents in multiple formats representing texts in multiple languages. In preparation for and during review, the documents in such collections must be processed in multiple ways. The preparation side includes normalizing each document in the form of a sequence of characters (such as utf-8). Subsequent steps involve basic decisions about stop words, stemming, and tokenization. These may seem inconsequential, but they can have large downstream impacts, because distinct words can have the same normal form. While these initial preparatory processing decisions set the stage for all that follows, they are basically orthogonal to the distinctions between the points of prominence in the structural landscape above. As a result, we won't pursue them here, except to note first, that these decisions already represent a move away from the documents themselves to a tailored representation of the documents, and second, that this

[7]Two cogent classifications of these particulars due to Herb Roitblat and Doug Oard have been presented and discussed by Ralph Losey in his e-discovery blog at `http://e-discoveryteam.com/2013/03/03/the-many-types-of-legal-search-software-in-the-car-market-today/`.

move directly involves forms of TAR/CAR, but have nothing directly to do with predictive coding. But predictive coding systems also rest on assumptions about the representation of document collections as a formal model or computational data structure which is compatible with the methods selected for predictive clustering (as in *accelerated review*) or *predictive classification* and often has additional virtues involving the time and space requirements of these methods.

2.1.1 term distribution: inverted indexes

Large volume search-based classification systems often depend for efficiency on an inverted index, comparable to the index in the back of a book or a concordance. Structurally, an inverted index consists of a list of the vocabulary of the collection as a whole, together with a *postings list* that pairs individual words with a list of documents the word occurs in, possibly along with other information, such as the position or positions the word occurs in (in order to support phrasal search and proximity queries). The various kinds of inverted indexes and the functionality they support—Boolean search (with operators *and*, *or*, *not*), proximity searches, various kinds of wild-card searches—are lucidly described in [Manning et al., 2008, Büttcher et al., 2010]. In addition, inverted indexes can also support ontology-based classification, when it is implemented through searches of these kinds. Inverted indexes also support weighted search and corresponding forms of ranked retrieval. But they lack a direct model of documents: documents are only represented indirectly in the global properties of the postings list.

2.1.2 vector models

Another familiar standard model, with many variations, represents each individual document as a vector in a high-dimensional vector space V. One can label the dimensions of V with many different kinds of data. In the simplest case, the dimensions correspond to the terms of the dictionary of the collection. The value for a document d at the coordinate corresponding to term t might be 1 if t occurs in d or 0 if not; alternatively, the value might be a weight based on the frequency of the term in the document; or it might be a weight based on the term frequency and the inverse document frequency (tf-idf); or it might be a pair consisting of the weight, and a list of the positions of t in d. These are all *unigram* models, because they are based on properties of one terms at a time, and each document is associated with a specific vector. There is no reason to stop at unigrams. In a *bigram* language model, the basic unit involves two word sequences. And we label the dimensions of the vector space with weights associated with bigrams. Many other choices are possible. Vector models of this kind often lose information: in the unigram vector model, for example, the vector associated with the 3 word document 'Me. Not you.' is the same as the vector associated with the 3 word document 'You. Not me.' (assuming that punctuation is discarded and words are normalized to lower-case). But one of the strong attractions of vector-based models with numerical weights is that they support an intrinsic measure of the distance between documents.

2.1.3 matrix models

The matrix-based model of Latent Semantic Analysis (LSA) combines the document-centric and term-centric points of view in a single formal model. The model is built in two stages—M_1 and M_2. Both stages consist of a $t \times d$ matrix, where t is the number of terms in the collection and d is the number of documents in the collection. To map the first stage M_1 to the second stage M_2, we factor M_1 into a product of three terms, T_0 ($t \times m$, with orthonormal columns, so that $T_0^T T_0 = 1$), S_0 ($m \times m$ diagonal, with decreasing entries, so that $S_0(i,i) \geq S_0(i+1, i+1)$), and D_0 ($m \times d$ with orthonormal columns, so that $D_0^T D_0 = 1$) using the technique in Linear Algebra known as Singular Value Decomposition (*SVD*) (a technique with connections to Principal Component Analysis in statistics). Then there is a delicate step in which we (manually) choose the top k elements on the diagonal of S_0 and replace the rest with 0. Call this new diagonal matrix S. Multiplying the factors T_0SD_0, we get a new matrix of the same dimensionality as the first, but with different entries—entries which will better reflect (we hope) the hypothesized latent semantic structure, since we've focused on the k principal components and eliminated (we hope) the noise attributable to the $m-k$ missing diagonal entries. Like the vector models above, matrix models support an instrinsic notion of distance between documents and as a result are readily adaptable to clustering applications based on distance.

2.1.4 topic models

For a third class of models, take the probabilistic model of Probabilistic Latent Semantic Analysis (PLSA), an approach which starts from the problems motivating LSA, but constructs a solution on completely different lines—using a probabilistic generative topic model. PLSA is one of a growing family of such generative topic models, which now includes Latent Dirichlet Allocation (LDA) and a large and growing set of extensions to LDA. (`http://www.cs.princeton.edu/~blei/topicmodeling.html`).Their underlying assumptions are elegantly displayed as graphical models, where the observable documents in the dataset are represented as probabilistically generated from a variety of latent, nonobservable structures.[8]

2.1.5 summary

The models just described are models of document sets and are independent of any document request (*RFP*). They can all be used in systems of *accelerated review*: the index system needs to be coupled with a form of search-based classification (such as ontologies), while the vector, matrix, and topic models need to be coupled with one of the variety of measures of similarity or distance that they intrinsically support. More ambitious and effective systems combine these models of document sets with information related to the *RFP*.

2.2 from data structures to models

Ideally, responding to an *RFP* implicitly defines a model of the *RFP* within the document source collection: each request or subrequest is associated with the set of documents that correctly respond to it. Concretely, the models we define in the course of responding to an *RFP* are hypotheses. One of the fundamental advantages of technology-assisted model construction—compared to the possibly haphazard course of linear review—is that hypotheses can be easily tested by standard statistical sampling techniques and the results can be used to inform the construction of better hypotheses. In the following paragraphs, we describe two distinct methods for combining information about a document set with information from an *RFP*. While one is more automated than the other, we will see that they have many similarities and that there are common workflows which draw on both methods.

2.2.1 from data structures to models: algorithms

From the perspective of machine-learning, hypothesis construction in response to an *RFP* may be regarded as an instance of *supervised learning*, consisting of:

[8]For background on graphical models, see [Koller and Friedman, 2009, Barber, 2012] or Daphne Koller's Coursera offering at `https://www.coursera.org/course/pgm`. Blei et al. [Blei et al., 2003] provide an insightful discussion of the differences among unigram models, PLSA, and LDA from this point of view.

examples: individual items from the document set;

features: properties of examples; note that the different document models discussed above make available different properties, even though they may be based on the same document collection.

labels: indicators of how examples are classified with respect to the *RFP* model

In supervised learning, a relatively small subset of the data is labeled (in some way), the labeled data is split into a *training set* and a *test set*, and correlations between labels and features in the training set are generalized and projected across the universe of examples as a whole, and performance is measured against the labeled test set. In the present setting, where the labels provide the information asssociated with the *RFP*, this classification of the example space is a hypothesis with regard to the implicit model of the *RFP* within the data set. This projection cannot be done on the basis of the labels alone nor on the basis of the features alone: it must be based on their interaction. Consequently, the feature space must be rich enough to support label discrimination: that is, we would like to minimize the situation in which contradictory labels are associated with two different documents that are represented identically in the feature space.

Statistical learning theory offers many algorithms that can be used in the projection step. From our structural perspective adopted here, we may think of them as modules serving a particular structural role (though of course, different algorithms may make different assumptions about the properties of the data they operate on). For a quick introduction, with computational examples in the statistical programming languages **S** and **R**, see [Venables and Ripley, 2002]. More extensive treatments may be found in [Hastie et al., 2009, Mohri et al., 2012, Murphy, 2012]. Examples: linear regression, logistic regression, support vector machines (SVM's), principal component analysis (PCA), clustering (many subtypes), Bayesian classifiers, classification and regression trees (CART's), ...

Since there are many different kinds of data and different algorithms perform differently as the underlying data changes, it is not to be expected that a single one of these algorithms will provide optimal performance across the range of electronic discovery datasets. But every method of statistical classification generates a testable hypothesis regarding the implicit document model associated with any given *RFP*. The cost of obtaining such hypotheses is remarkably low and scales amazingly well (especially compared to *linear review* and forms of *accelerated review* at the inefficient end). At the same time, the quality of such hypotheses can be remarkably high. This is the promise and potential of algorithmic predictive classification.

2.2.2 from data structures to models: manual classifiers

In the machine learning paradigm, the hypothesized model is constructed by statistical algorithms that project correlations between labels applied by reviewers sensitive both to the legal requirements of the *RFP* and to the properties of documents in a review sample. An alternative is to construct a hypothetical model using conceptually-based search techniques (such as ontology-based search). It might be hoped that a large and comprehensive library of ontologies would be sufficient to craft an adequate model for any document set and *RFP*. In our experience, some ontologies (emotional expression) are broadly applicable; some are applicable to particular sectors (think of sector-specific newsletters and trade associations, which are likely, but not necessarily, to be non-responsive to *RFP* requests); some are applicable to particular organizations (brand names and product codes); and others depend on idiosyncrasies that depend on the *RFP* and the document set. This means that in most projects, hypothesis construction based on ontologies is likely to be resource-intensive and difficult to scale.

One way to improve this process is to embed ontology-based hypothesis construction in a statistical sampling framework derived directly from the machine learning paradigm discussed above. This is an iterative process (see below) and on any iteration, an appropriate statistical sample is selected (perhaps random, perhaps stratified—see below), the sample is labeled and the current hypothesis is adjusted to fit, then tested against held back test data.

2.3 sampling: training, testing, validation

Predictive classification relies on iterative sampling. In any iterative round, sampled data is labeled with the relevant *RFP* information and divided into a training set. The classification system projects the training set labeling across the document collection as a whole. If the performance on a held back test set or on cross-validation is good enough, a statistical validation round takes place. If the validation round takes place unsuccessfully or the performance on the test set is not good enough, a new iterative sampling round takes place (on the assumption that improved results that flow from further iteration will justify the time and effort involved). Of the many basic questions that arise in this setting, we take up just, well, a sample.

Ralph Losey, in a series of important and influential blog postings (`http://e-discoveryteam.com`) has focused attention on three kinds of training data—random sampling, stratified sampling based on active learning, and judgmental sampling directly informed by subjective human knowledge—and argued that a 'multimodal' approach using all three (the 'three-cylinder multimodal approach') is optimal, preferred, sine qua non. Who could disagree? Our own view is that it's worthwhile to assess, at each stage of predictive classification, what kind of sampling most effectively advances overall goals. At the outset, a random sample provides basic information about the global distribution of the various document categories of interest—information that is distorted or blurred if the global random sample is mixed with non-global samples. In subsequent rounds, if the non-responsive / responsive ratio is high, then continuing to draw global random samples means that one's resources are allocated to labeling non-responsive documents. On this scenario, then, it makes sense to switch to stratified sampling, focusing on sub-populations whose labeling will be most useful. On Losey's metaphor, our predictive coding engine runs most powerfully when we switch from one cylinder to another. When should the third cylinder fire most effectively? One reason to ask this question is that the third cylinder—the judgmental sampling cylinder—is fueled directly by expensive subject matter experts, high-priced adepts in fact and law. Judgmental sampling at the outset involves a period of exploratory search and investigation, followed by the incorporation of the labeled results in the training data. This is essentially the same methodology that the subjects of the famous 1985 Blair & Maron study [Blair and Maron, 1985] followed, with notoriously poor recall results. It is a strength of random sampling that any significant population of responsive documents will eventually overlap with randomly drawn samples. This suggests that a more efficient and effective use of expensive human review resources might take place closer to the end of the modeling process, where it can take advantage of all the previous coding work, especially clashes between model predictions and expert labeling decisions, to push model performance above the limits of current machine learning paradigms. The possibility of using human resources as a final boost to iterative machine learning rounds offers a useful perspective on how to deal with the situation in which we're faced with more iterations than progress or when an attempted validation round fails: tune the machine learning classifier with manual classification based on human insights.

3 predictive coding structures and predictive coding workflows

The basic landscape of review abstracts from many specific workflows. Many modular technologies can play roles in workflows that count as distinct from the structural perspective adopted here. For example, one can use machine-learning technology in accelerated review and in hybrid accelerated review, as well as in machine-learning classification. Shifting the focus from the technology modules to the simple structural properties stressed above clarifies the properties and potential of all the varieties of predictive coding.

References

[Barber, 2012] Barber, D. (2012). *Bayesian Reasoning and Machine Learning.* Cambridge University Press, Cambridge.

[Blair and Maron, 1985] Blair, D. C. and Maron, M. E. (1985). An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the Association for Computing Machinery*, 28(3):289–299.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[Büttcher et al., 2010] Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines.* MIT Press, Cambridge, Massachusetts.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, New York, second edition.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, Massachusetts.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press, Cambridge.

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Fundamentals of Machine Learning.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, Massachusetts.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* MIT Press, Cambridge, Massachusetts.

[Venables and Ripley, 2002] Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S.* Statistics and Computing. Springer, New York, fourth edition.