DHQ: Digital Humanities Quarterly
2021
Volume 15 Number 2

Interpretable Outputs: Criteria for Machine Learning in the Humanities

James Dobson <james_dot_e_dot_dobson_at_dartmouth_dot_edu>, Dartmouth College

Those disciplines and fields that understand themselves as primarily humanistic have long had a special interest in exploring nuance, uncertainty, and ambiguity in our objects. While data-driven research might frequently be assumed to be in opposition to such concepts, this is not the case. Data may, in fact, enable the discovery and foregrounding of ambiguity. Yet as computational methods, especially those designed to operate on large-scale textual archives, increasingly enter into the work of such humanists there is a risk that access to those ambiguous dimensions of data will be lost. This risk is as much present for the researcher using these tools to interpret their objects as it is for the skeptical reader or reviewer of such interpretations. Visually appealing graphic renderings of data, high classification accuracy and confidence scores, and impressive summary statistics can easily be rhetorically structured to appeal to preconceived notions and commonly accepted understandings. Effective and academically responsible argumentation and interpretation in the humanities requires not just openness, which is to say unfettered access to the methods and data used to generate visualizations and to demonstrate accuracy, but that the selected datasets and models are interpretable. When features are derived from text, as in the case of much of the machine learning taking place in the humanities, these features need to be exposed along with the data that helps readers understand their significance. Humanistic work that is primarily argumentative — it should be acknowledged that of course not all digital humanities work is argumentative — depends upon certain frameworks by which readers can evaluate claims. These frameworks generally operate within established interpretive communities of scholars working on similar problems or using similar approaches [Fish 1980]. While data and statistics have had a minimal presence within the humanities prior to the twenty-first century, contemporary computational approaches, if presented properly, are highly compatible with existing humanistic argumentative frameworks [Dobson 2019]. While some computational fields have been undergoing a reproducibility crisis, the present discussion is not primarily an argument about access, an argument that would critique the use of so-called black box algorithms or advocate for the digital humanities to join in the movement toward open science, but rather this essay argues for the importance of listening to rather than hiding noise, for exposing complexities and ambiguities, and for interpreting the nuances within data derived from our objects. It will do so by explicating the role of hermeneutics in the digital humanities before turning to three critical sites within computational workflows that will serve as case studies: the creation of data objects, topic modeling methods, and classification algorithms. These three sites, with their increasing complexity, enable us to see key common limits to interpretability across the software “stack” of tools commonly used in the digital humanities today. 1

Joining the recent injunction “against cleaning” made by Katie Rawson and Trevor Muñoz, who argue that data cleaning, standardization, and “tidiness [privilege] the structure of a container, rather than the data inside it” [Rawson and Muñoz 2019, 290], this essay turns to basic computational objects used in text-based machine learning, data structures, model formats, and classification algorithms, as a case study for the preservation of nuance, diversity, and interpretability. Similar to the way in which Rawson and Muñoz call data cleaning a “diversity-hiding trick,” selecting certain models and data formats as containers can suppress evidence and minimize the presence of the unexpected, the minor, and the troubling, leading to a misrepresentation of the contents and unwarranted claims. The following critique also shares the ethical force of the request for researchers to establish and follow clear documentation procedures for producing and using machine learning datasets and models made by Margaret Mitchell, Timnit Gebru, and their co-authors [Mitchell et al. 2019] [Gebru et al. 2018]. The combination of prioritizing methodological transparency and foregrounding the complexity and diversity of data is necessary in order to preserve scholarly standards and to enable, within the computational environment, the dialogical forms of argumentation common to the humanities. 2

Data, if we construe data broadly to mean evidence presented to readers or listeners in support of an argument, has long existed within the argumentative apparatus used within humanistic fields. Forms of data include textual evidence 3

put forward in a close reading or material culture gathered from an archive. Evidence is made explicit by authors in academic arguments. Evidence can be variously interpreted by readers and observers. It is another feature of the academic argumentation that frames and focalizes evidence. Warrants, to borrow a term from the philosophy of logic — in particular the work of Stephen Toulmin, function within argumentation as the implicit link between evidence and claims [Toulmin 1969]. Strong arguments involve making explicit that which implicitly provides the foundation for the interpretive claim, stating the warrants alongside the arguments that leverage them. I invoke the language of formal argumentation in order to help frame some emerging problems within computational work within the digital humanities.

4 But we first need to disentangle explanation from interpretation. Recently, there has been a growing demand for explainable models within artificial intelligence research and especially within those social scientific fields making use of AI techniques. These efforts have primarily asserted that explanation and interpretation are the same [Miller 2019]. Explanation, however, does not have the same meaning as interpretation because an explanation can still function without resolving uncertainty and ambiguity found within data while an interpretation can and frequently does take these up as the basis for exploration or argumentation [Burckhardt 1968]. Interpretation arises from an encounter with evidence. In the humanities, interpretation tends to foreground uncertainty and ambiguity within this encounter; the interpreter may locate these sites within the primary object or gesture toward other evidence. Interpretation thus needs to be in relation to presented evidence, whether this evidence is an image, a passage of text, or information derived from data.

5 Computational models tends to reduce uncertainty by only registering that which is classifiable according to the logical structuring of the model as a container. While Julia Flanders and Fotis Jannidis are concerned with text-markup models and frameworks for complex reference systems, their assertion that “most modeling systems at their core require uncertainty to be represented in the content (i.e. through qualifying notations) rather than in the modeling itself” applies equally to numerical models of textual sources and the typical ways in which these are used in the application of machine learning in the humanities [Flanders and Jannidis, 21]. Data-driven arguments in the humanities need to present interpretable outputs because in order to accept arguments as plausible, readers need to be able to understand if the claims are warranted by evidence. The rhetorical situation within existing communities of interpretation is structured according to the logics of plausibility and therefore interpretive arguments cannot only be evaluated according to the governing logic of the correct and rigorous interpretation of data. From the basis of a humanistic approach toward tools and data, interpretable outputs from computational models derived from textual sources are simultaneously qualitative and quantitative. This is to say that plausibility and the qualitative are frequently registered in quantitative features and attributes including edge cases, outliers, anomalies, ill-fitting values as well as cases of noise taken as signal and vice versa.

6 There are two major categories in which we might place most computational humanities work at present. The first category is more affiliated with the practices of literary criticism and it draws upon the long history of understanding criticism not as secondary to the creative work that it interprets but itself as a creative act [Hartman 1980, 189–213]. We can understand this mode of criticism as involving a collaborative relation between critic and reader that is playful yet built on shared norms. Creative computational criticism does not explicitly depend upon scientific rigor associated with statistics and computation such as validity, falsifiability, and reproducibility. Rather, algorithmic transformations of text are framed as creative or even speculative renderings. Criticism in this vein can be argumentative but the grounds for the critical work are not subject to a mode of critique that would prove it wrong, obsolete, or invalid. Such forms of computation are primary playful; these readings transform and deform input text. This form of computational work might best be associated with the work of Stephen Ramsay and Nick Montfort, although many others frame their work in similar terms [Ramsay 2014] [Montfort 2018]. The second category involves the use of computation for research in order to make empirical claims about data. These practices and methods share a desire to demonstrate the significance of any findings with the social sciences. Researchers working in this area thus produce arguments that require the methods to be statistically sound. Interpretations of their results are warranted by a series of assumptions that can be tested and verified. These scholars make hermeneutical claims about computational models that are grounded in their analysis of data. While they might make use of similar algorithms as those that I have classed as creative computational critics, they are less interested in engaging deformations than refining models to produce more significant results.

Though it provides some level of access, the explicit presentation of data in support of an argument, a discovery, or a finding does not necessarily mean greater interpretability. One increasingly common example, machine learning classification data, are frequently presented in the form of accuracy scores and the confusion matrix. Parsing f-scores, and precision and recall values enables a certain degree of understanding of the overall performance of the classifier and the function of the workflow as a whole, but this leaves aside many other post-classification metrics that can be directly applied to the underlying data model. These data can tell us much about our input datasets and the criteria by which the classifier made its classifications. For applications in the humanities, in which there is a scholarly community that cares deeply about ambiguity, nuance, and the historicity of language, models that can provide extended interpretable output features, for example coefficients, probability values, and weights used in classification are necessary. The exposure of these extended interpretable features enables some shifting of scale — a shifting that matches the needs of humanistic interpretation and recognizes that while meaning can be made at the level of individual signifier, the word or token and its frequencies, it is in those larger units, for example the sentence, paragraph, chapter, and volume, that much of the activity of meaning making is to be found [Algee-Hewitt et al. 2017][Allison et al. 2017]. Just as decontextualized individual signifiers can all too easily hide ambiguous meanings and data diversity, the absence of interpretable features frequently leaves readers of computational work with many unknowns as lists are leveled, distributions smoothed, and complex geometries collapsed.

7

Tool criticism is an emergent category of analysis that provides a humanities-centric framework for understanding the use of computer-aided methods in both the humanities and the sciences. Tool criticism seeks to foreground the tools used in humanities research that make use of computation. In so doing, it foregrounds methodology for computational researchers and for their critics. Karin van Es, who has done a great deal to develop and mobilize this concept, asks researchers to understand the epistemic impact of their tools. Tool criticism framework requires access to tools and data and produces inquiry into the affordances of computational methods:

8

> Tool criticism is the critical inquiry of knowledge technologies considered or used for various purposes. It reviews the qualities of the tool in light of, for instance, research activities and reflects on how the tool (e.g. its data source, working mechanisms, anticipated use, interface, and embedded assumptions) affects the user, research process and output. [van Es et al. 2018, 26]

Understanding computational methods through the language of tool criticism helps foreground the need for a robust inspection of input data, the workflow, and algorithmically generated output data. In many ways, the preference for interpretable data is the output version of input solutions offered by scholars interested in better descriptions of input data. In Katherine Bode's notion of the data-rich object, we find a strong argument for better descriptions of input data, especially when these data concern book history. The data-rich object, in Bode's account, provides a conceptual framework through which we might bridge the gap between scholarly work in bibliographical studies and computational text analysis in literary studies [Bode 2017]. While some shared repositories provide a set of reference texts and features extracted from these texts that have been designed for computational work (see the HathiTrust Extracted Features Dataset for an example of this type of repository) once these input objects enter into scholarly workflow, much information has been stripped from the input data objects making it much more difficult to do cross-study comparisons and to apply these insights to ongoing scholarly debates within literary critical communities [Capitanu et al. 2016].

In many computational fields it is a norm that researchers, especially academic researchers undergoing peer review and working with complex technological instruments, make their methods transparent and include their data, models, as well as the code necessary to implement their research. Academic researchers are making demands of each other that they open up the black box of their experimental apparatus, much in the way that critics and activists have made similar demands of corporate and government entities deploying proprietary and secretive algorithmically-driven decision-making tools. The "open science" movement champions such a model of methodological transparency as a solution and remedy for what has come to be recognized as a replication crisis in several fields, most notably in the psychological and brain sciences. Open access and transparency into methodology, however, do not ensure that reviewers and others will be able to understand the data-driven claims and the argumentative grounds supporting such claims. Mike Ananny and Kate Crawford critique what they call the "transparency ideal" for its emphasis on openness as the single solution to accountability in technological systems (Ananny and Crawford 2018). Ananny and Crawford understand accountability

9

in terms not just of the operation of a single element but rather through the social critique of systems and in particular the workings of power throughout the complex social systems in which algorithms and other computer-based technological tools are embedded. Johannes Paßmann and Asher Boersma argue in “Unknowing Algorithms: On Transparency of Unopenable Black Boxes” for another mode of interpretation applicable to computational models that might be considered supplementary or an alternative practice to the demands for openness driven by the transparency ideal, a practice “that is not so much concerned with positive knowledge, but that deals with skills which help dealing with those parts of an artefact that one still does not know” [Paßmann 2017, 140]. They cite situations like medical procedures in which an actor — an actor who importantly is not the primary actor but perhaps another physician — might gather unspecified and potentially useless data as a form of an unopenable black box; the collection of data and experience cannot be explained or defended in methodological terms but it might provide useful information for later procedures. This example provides them with a way of describing negative knowledge, an understanding of the known unknown. Paßmann and Boersma use such a scene to defend a notion of transparency that is distant yet still rooted in phenomenological experience. They offer, via Maurice Merleau-Ponty’s account of the way in which a woman wearing a hat equipped with an extended feather navigates space, a tool-based mode of exploratory analysis, “a carefully paced out unknowing” that can limn the space of known unknown [Paßmann 2017, 145]. This amounts to what they call “feather knowledge,” an alternative to direct contact that might be capable of providing knowledge about the workings of computational models when such direct access is not available or not possible.

10 Paßmann and Boersma’s concept of distance and mediated access to black boxes should not be taken as an endorsement of a refusal of access but an interpretive strategy that enables researchers to probe and feel their way around propriety technology. There are situations within academic research in which some models might need to be protected, for example if they might leak identifiable training data about human subjects, but in the vast majority of cases, open access is essential to understand the meaning of the data and subsequent data-based claims. At the same time, feather knowledge might contribute to the understanding of computational models in ways that are fundamentally different and perhaps even more crucial than disembodied observation of models.

11 Even if it were possible to achieve the goals of the transparency ideal, certain methods require other forms of inspection than observation of function in order to be understood and explainable. Direct access to parameters and features supplied as training data for classification algorithms can enable queries to examine the distribution of certain words or phrases in order to understand if these are indeed representative of the modeled event or phenomenon. For example, counterfactual testing of a model and its output can be done simply by using the same model with other samples, with data known to the counterfactual investigator. While counterfactual testing fits into the general design of a tool or workflow, sometimes this might not provide enough useful information to evaluate the model and to expose the diversity of data within the model. Developing an explanation of how some computational functions work can involve treating the algorithm or model as something to be tricked through adversarial techniques. Adversarial techniques might involve observing reactions to unexpected input, for example, and rather than supplying expected textual features one might supply randomized data or even alternate numerical input. This method of probing the functioning of an algorithm can leak or reveal decision criteria or training features if these are not available. Needing to follow such procedures might be considered undesirable due to the unreliability of such methods of probing. By making use of highly interpretable data and algorithms, the digital humanities community can avoid needing to turn to adversarial techniques to expose nuanced and diverse data and aid in the verification and interpretation of computational models.

Data Objects

12 It is the containers that reshape and normalize data, the data objects themselves, in which we find the largest potential risks to interpretability. Because data models and formats are the foundation for higher level transformations and operations, they are crucial to improving interpretability. In order to better understand this problem, I will turn now to examine the affordances and limitations in two alternative data models found in the popular and widely-used Scikit-learn package for the Python programming ecosystem. Scikit-learn bundles together a number of fast and reliable machine learning algorithms along with associated data models, tools for preprocessing text, and facilities for connecting these together in a reproducible workflow. While Scikit-learn provides the foundations for developing, using, and distributing complex models for a number of different applications and research domains, it is especially well suited to the analysis

of textual sources and widely used within the computational humanities community. The major data models and algorithms implemented by Scikit-learn will provide case studies through which we can examine the stakes of the choice between these models for the interpretability of the models themselves.

13 The ability to inspect and interpret models is a crucial capability for understanding the meaning of both their output and the function of the models themselves. Models, after all, model something. The assumptions that inform the design and operation of models are sometimes explicitly coded but more often these assumptions are implicitly registered in the details of their construction and in their operation. These details can span a number of levels, from the aesthetic features and arbitrary choices made during the coding, to the selection of specific algorithms and input parameters. What makes "model work" in the humanities distinct is the critical interest in the possibilities opened up with and foreclosed by each of these levels of interpretation. To put it another way, modeling in the humanities is subject to forms of critique that aim at both explicit and implicit assumptions while in other disciplines the primary mode of critique operates according technical criteria. Because of this, it is necessary for humanists, in selecting their tools and data objects, to use those objects and methods that afford the greatest levels of inspection and interpretability.

14 We can understand the stakes of these differences in terms of inspection by examining the various attributes found in two Scikit-learn feature extraction tools that produce models of textual sources. These two tools perform the exact same function and are typically used for differing scales of analysis. Crucially, they make available dramatically different levels of inspection and interrogation. Feature extraction, for text analysis, involves the preprocessing and fragmentation of the text into a document-term matrix (dtm). Preprocessing, for most applications, means normalization of accented text, removal of stopwords, and selection criteria for inclusion (thresholds for feature count). The fragmentation of the text or document involves splitting or tokenizing input text into words or n-gram (multiword) phrases.

15 The conversion of a document (text) or a set of documents into a document-term matrix or other vector space model provides what we might call a remediated representation of the text. The vectorization of data, of course, involves more than just shortcuts for the transposition or manipulation of data. The ability to apply a transformation to an entire row, column, or matrix without iteration marks, as Adrian Mackenzie argues, an epistemic shift away from the tabular representation of data and toward new pliable movements through data [Mackenzie 2017, 69]. This is one of the primary transformations of the text in most computational workflows used by humanists. Vectorizing texts produces an alternate representation, a data model of a text collection that makes it possible to compare one text to another and the entire collection to other collections, as well as making it possible to perform a simultaneous vector operation across the entire model. Word order is lost but depending on key choices made by the researcher there can still be much information available that may serve interpretive goals within the resulting model about the size and distribution of the vocabulary and its relative frequency.

16 There are several vectorizers included within the Scikit-learn package, the two that are most pertinent to this discussion are the HashingVectorizer and the CountVectorizer. The Scikit-learn documentation explains that there are three limitations or "cons," as they put it, to using the HashingVectorizer to generate a document-term matrix instead of other alternatives including the CountVectorizer:

- there is no way to compute the inverse transform (from feature indices to string feature names) which can be a problem when trying to introspect which features are most important to a model.
- there can be collisions: distinct tokens can be mapped to the same feature index. However in practice this is rarely an issue if n_features is large enough (e.g. 2 ** 18 for text classification problems).
- no IDF weighting as this would render the transformer stateful. [Scikit-Learn 2020]).

17 The second limitation is related to the typical use-case of the HashingVectorizer: it can vectorize very large numbers of documents and with a lower use of computational resources, primarily random access memory, but there is a chance of some data corruption (in the form of collisions) in which the same index in the document-term matrix is used for distinct features (words or phrases). The first limitation presents the largest problem for researchers wanting to examine or "introspect," to use the language of the Scikit-learn developers, the document-term matrix and to use the feature index to extract meaningful features that have contributed to classification accuracy. The HashingVectorizer uses hashing, a

non-cryptographic one-way encoding scheme that saves space and increases access to encoded data but comes at a cost to interpretability in that you can no longer match those features identified as statistically significant to their indexed names.

The short segment of Python code in Figure 1 demonstrates what we might want to call the different affordances of the CountVectorizer and the HashingVectorizer. While both will extract features and produce a document-term matrix usable for different tasks, the HashingVectorizer does not enable direct inspection of the vocabulary. The HashingVectorizer might enable scholars to vectorize large-scale archives into reasonable compact document-term matrices but it has important limitations that make it much more difficult to inspect and interpret. With the CountVectorizer-produced matrix, we can query the vocabulary with the column index number of a term of interest, for example, the term “woman” in Figure 1, and then determine the number of times it appears in each text, leading to knowledge about the relative representation of this term across the collection. More importantly, the existence of the mapping between feature index (columns) and names (vocabulary) in the CountVectorizer makes it possible to determine, in the example of a classification task, the vocabulary terms that were the most important features for that task. Here, as in the other examples found in this essay, “important” has a technical meaning, the degree to which a particular feature is statistically influential to the model rather than the relative importance of these terms within a text. The sharing of a model produced with the HashingVectorizer might satisfy the demand for openness, which is to say transparency in relation to methods, but the model is not as interpretable as one produced with CountVectorizer. Without these crucial feature indexing elements, models become much harder to interpret. This may render such a model inappropriate for applications in the digital humanities.

18

```
In [1]: from glob import glob
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import HashingVectorizer

        input_files = glob("data/na-slave-narratives/data/texts/*txt")

        cvec = CountVectorizer(input='filename',
                               stop_words='english',
                               lowercase=True)

        hvec = HashingVectorizer(input='filename',
                                 stop_words='english',
                                 lowercase=True)

        cvec_vectors = cvec.fit_transform(input_files)
        hvec_vectors = hvec.fit_transform(input_files)

In [2]: # display number of documents and number of features in document-term matrix
        print(hvec_vectors.shape)
        print(cvec_vectors.shape)

        (294, 1048576)
        (294, 68002)

In [3]: cvec_vocab = cvec.vocabulary_
        print(len(cvec_vocab))

        68002

In [4]: # find feature number for the term "woman"
        print(cvec_vocab['woman'])

        67072

In [5]: # with the CountVectorizer generated DTM we can query mean number of
        # invocations of the term "woman" found within the document collection
        import numpy as np
        np.mean(cvec_vectors.getcol(67072))

Out[5]: 16.421768707482993
```

Figure 1. Comparing Scikit-Learn’s HashingVectorizer and CountVectorizer using the latter to map feature indices back to vocabulary terms.

Data models, especially vectorized data models derived from text, are quite varied. There are some data models that can be easily modified for exploratory counterfactual testing while others cannot. Vectorizing new texts with the intention of hypothesis testing, which would involve adding new rows to the matrix, can be accomplished when the document-text matrix has been produced with either all existing word/n-grams or a restricted and known set of excluded or stopwords.

19

If new words found in these additional texts are added to the vocabulary then new columns will be added and zero-count values for existing documents can be added. If terms were excluded previously for not meeting a minimum threshold of appearances or other criteria, then the newly modified model may not be correct. Using methods to vectorize documents that preserve all detected vocabulary and recording selection criteria alongside the model are strategies that can enable such modes of counter factual understanding and increase interpretability. More complex, higher-level methods making use of a slightly different variety of vectors, such as the word embedding models found in word2vec or fastText, can sometimes function in similar ways. One form of counterfactual testing of word2vec-produced vectors would include aligning or fitting another, well-known model to the one on hand and querying the resulting model. Perhaps the more common method at present takes the original model as the initial vector for additional training with another set of inputs. It is important to keep in mind that this method is limited to the extracted and trained vocabulary of the initial model. Comparisons of the original and the resulting model can make visible previously unavailable aspects of the model [Hamilton et al. 2016]. Digital humanists need not only produce transparent and inspectable but also interpretable data models, models that in their containing function preserve as many points of access into the complexity and diversity of the data. Selecting interpretable data models is crucial because they are the foundation for transformations and operations. This is important because limits in interpretable data at this lower level are compounded by those found at the higher levels of many workflows.

Topic Models

Topic modeling, one of the earliest computational models used for exploratory data analysis in the humanities, continues to be a dominant method. The topic word or feature lists — much more than the frequencies by which these topics are associated with documents — have attracted the attention of humanists. The result of an unsupervised model, the "topics" produced as one of the outputs of a topic modeling algorithm are unnamed and presented to the interpreter as an interpretive object. There is much variability to found in the output of topic modeling, depending upon the construction of workflow including the algorithms chosen, the parameters used, and the preprocessing performed. Nan Da argues that this variability makes topic modeling "extremely sensitive to parametricization, prone to overfitting, and… fairly unstable" [Da 2019, 625]. The presentation of individual topics from a topic model with a wordcloud makes topic models particularly difficult to interpret. The viewer has to first *infer* the relationship between the words, typically presented with different font sizes for the depicted words or less frequently with color-coding, to understand how to read these words in relation to each other. Frequently critics making use of topic models will label their model with what they take to be the meaning of the topic. 20

In David M. Blei's review article that popularized topic modeling for the humanities, Blei displayed the partial inference of a 100-topic latent Dirichlet allocation (LDA) model from 17,000 articles published in *Science*. This method is one of several used in topic modeling and remains the most popular due to its widespread availability and simplicity. Blei's presentation of these topics and his explanation of his inferring of the topics helped established a pattern of what we might think of as the display of data without data. The terms, of course, are data but the values attached to these terms and their relative importance to the topics are hidden. The way in which Blei assigns labels to his topics also leads to some additional confusion by introducing another explicitly subjective layer of interpretation. We understand that words listed first in each column of a table labeled "top 15 most frequent words from the most frequent topics" [Blei 2012, 79] might be the most important to determining the meaning of the list but how important really are these words to the topic? The meaning of important in a model like LDA means statistically influential or having a relatively high bearing on the statistically calculated outcome. The statistical importance of these terms is difficult to interpret without numerical values by which we can determine the relations among the items in these lists. While Evolution, Disease, and Computers are relatively straightforward labels of topics with the most frequently appearing words, the Genetics topic required moving down to the fourth most frequent term to find the correct label. The point here is not that these topics are mislabeled but that the method by which labels are determined is opaque — they are assigned in an ad-hoc manner by the operator — and the information needed to determine the relationships among these words — which is to say that other than ranked order there are no numerical values presented — is not available. 21

“Genetics”	“Evolution”	“Disease”	“Computers”
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
genome	evolutionary	host	models
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Table 1. Topic model displaying words without data and with operator-assigned “topic” headings [Blei 2012].

22 Scikit-learn, which provides several algorithms used to generate topic models, makes available methods to access the data used to assign words to the topic groups with LDA models. These values can be queried with the “n_components” attribute of Scikit-learn’s implementation of the LDA model. These “components” or “variational parameters for topic word distribution” can provide either what the package calls a pseudocount, which is the number of times that each of the vocabulary terms has been assigned to the topic, or a number that represents the normalized distribution of words within each topic, the: “Since the complete conditional for topic word distribution is a Dirichlet, `components_[i, j]` can be viewed as pseudocount that represents the number of times word `j` was assigned to topic `i`” [Scikit-Learn 2020]. The pseudocount figure is especially useful in determining the relative significance of each term within the topic list or container.

23 There are several methods that use the output from standard LDA models to provide higher-level statistics and visualizations that enable additional degrees of inspection of both the topics and the individual features included within the topics. Despite a history of calls for greater clarity with the presentation of topic models in the humanities, these methods to enable greater inspection have not seen much use by humanists [Schmidt 2012]. The LDAvis package for R and Python provides several metrics for examining and understanding the contribution of single terms to the topics. One of the metrics used by LDAvis is known as termite and uses seriation to insert extracted terms back into some greater textual context, i.e., the appearance of the word of interest prior to vectorization [Chuang et al. 2012]. This restored context preserves word order and enhances the understanding of topics and words but it is only available if the pre-vectorized sources, in other words plain-text sources with the original word order intact, are available. The selection of interpretable data objects, in this case the presence of complete textual sources, with associated bibliographic information, rather than just the vectorized sources as might be distributed by scholars working with copywritten or otherwise restricted textual datasets, directly informs the interpretability of higher-level models.

Classification Algorithms

24 Humanists are eager to have interpretable output that can be parsed according to familiar interpretative and framing strategies. Classification algorithms that promise to reify through empirical methods longstanding categories of humanistic analysis operate from what might be thought of as the opposite interpretive position: while topic models

demand hermeneutical interpretation in which featured words are read in relation to algorithmically proposed “topics,” classification algorithms are generally used to produce numerical probabilities of the likelihood of individual texts as belonging to this or that category. Classification algorithms tend to draw their power through numerous features. These features are usually not inspected through the hermeneutical circle that would interpret features in relation to categories and categories in relation to features. There are numerous uses for classification algorithms in the humanities and literary studies, in particular. Major classification categories include genres, topoi, literary periodization, focalization, narrative structures, and authorial identity. Equipped with labeled training and testing data, humanists have developed models that can, with various degrees of accuracy, classify input texts or sections of texts into such categories.

25 What we might want to call the “data science” method of reporting results frequently takes the form of displaying overall model accuracy along with what is known as the confusion matrix, a visualization that can help determine the number and nature of misclassifications. Much of the published work in computational literary studies provides such descriptions and visualizations of these models and their reported metrics as evidence of successful modeling. Yet we cannot assess the usefulness and ultimately the meaning of classification models without access to the features, weights, and parameters used in the classification task. The selection of classification algorithms is usually made according to the model with the highest reported accuracy. The scholarship concerning the evaluation of these algorithms highlights this dimension. Bei Yu evaluates and compares the two classification algorithms invoked in the present essay, Naïve Bayes and Support Vector Machines, for use in literary studies, but her evaluation criteria consists of performance metrics, the relative accuracy of these algorithms in correctly classifying (verification involves the comparison of machine and human assigned labels applied to the objects) a selection of Emily Dickinson’s poems as either erotic or non-erotic and chapters of early American novels as either sentimental or non-sentimental [Yu 2008]. While Matthew L. Jockers and Daniela M. Witten’s comparison of algorithms and exploration of the results of different classification models used in authorship attribution includes a table of the fifty most important words and two-word phrases for one classifier, the ability of these algorithms and models to produce such data are not emphasized as much as the performance (accuracy) of the algorithms in correctly identifying the author of a text [Jockers and Witten 2010]. This preference for high accuracy on testing datasets, while the best practice in many fields, does not necessarily lead to the most interpretable model. In order to be appropriate for humanities scholarship, the selection criteria for a classification algorithm needs to depend as much on the ability to return interpretable features as its classification accuracy on known testing data. It is an issue of balance, to be sure, but one needs a model that can successfully discriminate among texts while also making available the criteria used to produce these decisions.

26 Different classification algorithms applied to the same exact data can produce very similar classification results and accuracies but the features used by these algorithms to determine class membership can be substantially different. This complexity has proved to be a significant problem for humanistic engagement. What might it mean that one algorithm assigns more importance to specific subset of an extracted feature set than another? If both of these algorithms can determine a significant distinction at the level of word frequencies or other extracted features between the classes that make up the provided labeled dataset, does that mean that the distinction within the dataset is real? And if these are to be determined with a different set of features, why do the individual word features matter? Handling these questions and objections is not straightforward. An algorithm tuned for high accuracy will find distinctive features that fit into the model of distinction prioritized by that particular algorithm.

27 The dataset of Text-Encoding Initiative (TEI) tagged texts from the Text Creation Partnership (TCP) from the Early English Books Online (EEBO) collection enables some trivial experimentation with different classifiers to examine the features used for classifying short lyrical objects. After extracting plain-text segments from the XML mark-up files that were tagged as these lyrical categories (ballad, prayer, song) and vectorized using CountVectorizer, the objects can be split into training and testing datasets for examining the ability of several classification algorithms to generate interpretable output. The Support Vector Machines (SVM) machine learning classifier on one iteration of this classification task produces a 92.3% overall accuracy in classification across the three classes: precision recall f1-score support ballad 0.80 0.41 0.54 116 prayer 0.99 0.95 0.97 1151 song 0.85 0.97 0.90 733 avg/total 0.92 0.92 0.92 2000 The confusion matrix for this model, using the testing dataset described above, demonstrates excellent classification accuracy for prayers and songs, some ability to classify ballads, and many misclassifications of ballads as songs

(leading to the recall score of 41%).

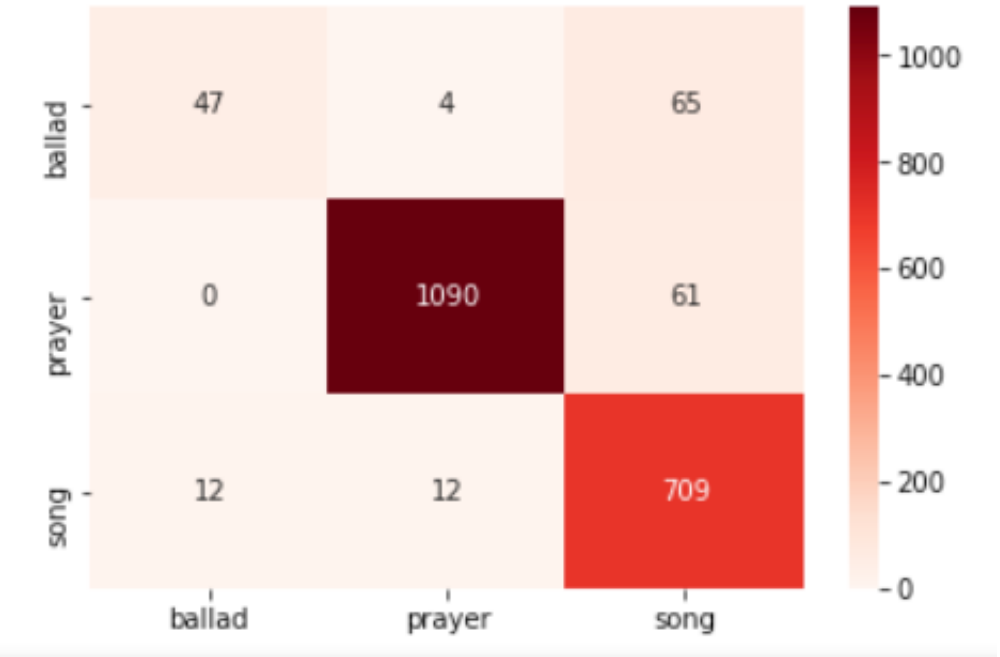

Figure 2. SVM Confusion Matrix

At the present time, some classification algorithms such as the SVM classifier are much more preferable than others, for example the k-nearest neighbors (kNN) algorithm, which is well-known as simple to understand but difficult to interpret. The k-nearest neighbors algorithm can return similarities between objects but does not readily supply the criteria by which these data elements were determined to be similar. Additional probing of the data can return feature similarity between the neighbors discovered via the k-nearest neighbors algorithm but these are not initially provided as output. At present, the level of computational skill required and critical faculty for such probing is rather high. The ability to engage in data exploration is necessary for the evaluation of results presented as meaningful. Yet determining the meaning of textual data requires both computational knowledge that can determine significance within the model and domain-specific contextual knowledge that can be applied to the understanding of these features. If features presented as significant are in fact meaningful within the terms of the selected textual dataset, the information needs to framed in an understandable fashion.

Ballad	Prayer	Song
pdf (3.644), page (2.346), ballad (2.279), tune (1.785), ſame (1.302), engliſh (1.113), pope (1.103), ſecond (1.091), dog (1.091), popery (1.08), young (1.063), did (1.023), tryal (1.019), ſaid (0.983), husband (0.982), england (0.97), tanner (0.952), cripple (0.929), finis (0.928), gillian (0.917), ſir (0.901), ſée (0.896), monſters (0.893), country (0.889), ile (0.873), trading (0.871), farewel (0.854), true (0.839), ſwéet (0.834), live (0.798)	amen (5.614), prayer (4.456), iesu (2.414), spirit (2.091), unto (1.964), god (1.862), deus (1.823), thy (1.806), praier (1.742), sins (1.646), yn (1.624), point (1.604), bleſſed (1.529), ps (1.529), jesus (1.52), father (1.51), lord (1.488), oremus (1.482), speach (1.48), vnto (1.404), things (1.382), vpon (1.376), pſal (1.369), merciful (1.365), meate (1.359), com (1.356), qui (1.331), beseech (1.33), ein (1.328), vs (1.306)	song (7.892), ij (2.749), repeat (2.355), psalme (2.263), voc (2.134), doth (2.113), ſupra (1.817), lawes (1.649), sing (1.611), prayſe (1.581), ſong (1.549), bassvs (1.482), cantvs (1.473), ii (1.43), mr (1.402), oh (1.383), chorus (1.382), heau (1.311), ſing (1.31), gods (1.305), ll (1.297), allison (1.259), heav (1.23), duplicate (1.211), foes (1.167), gate (1.142), st (1.138), ſtyll (1.137), tenor (1.12), eu (1.112)

Table 2. Most statistically important features from each class for the SVM classifier

If we want to understand why the word “pdf” was the most statistically important feature for classification of ballads, we can explore the supplied data, provided we have used a data model that makes the vocabulary available for inspection. In Figure 3 we can see a sample function that queries the vocabulary for the column number for a word of interest, extracts total count, and then examines the rows belonging to the supplied classes. The SVM classifier learned from labeled training data that “pdf” was the most informative feature for those included texts marked ballads and the document-term matrix for this training data shows that by far the majority of the three hundred and eighty two appearances of the term “pdf” appeared in those documents. The word “pdf,” the shortened name for a portable document file, was mistakenly included within the extracted tagged text in an earlier preprocessing stage. It functions as a metadata description, an indicator of the location of the material on scanned pages, and was recorded in the XML as a comment (an example of this comment appears as <! — PDF PAGE 1 — >). Because this comment text appeared more frequently in those texts in the training dataset identified as ballads, it has become statistically important to the classifier in recognizing this class of lyrical text. This level of inspection, querying the labeled data object directly, enables checks for obvious errors in preprocessing or coding (as in this case) as well as a form of hypothesis testing directed at understanding the meaning of the extracted features. Attempting to explain the classification from a single term, as this example would imply, will most likely not be possible in most cases. But words or other features of interest can be queried, and the distribution of the vocabulary over the classes can help determine if the classification is in fact meaningful to the classes or an artifact of the particular training data selected or of the model and its parameters.

```
In [12]: def term_debug(term):

             # obtain index
             if term not in vectorizer.vocabulary_:
                 return("Not in vocab")

             idx = vectorizer.vocabulary_[term]
             tc = int(sum(train_data_vectors.getcol(idx).toarray()))
             print("{0} appears in {1} documents".format(term, tc))

             class_dict = dict()
             for class_label in clf.classes_:
                 class_dict[class_label] = 0

             for i, v in enumerate(train_data_vectors.getcol(idx).toarray()):
                 if v != 0:
                     class_label = train_labels[i]
                     class_dict[class_label] += int(v)

             return(class_dict)

In [13]: term_debug("pdf")

         pdf appears in 382 documents

Out[13]: {'ballad': 234, 'elegy': 73, 'ode': 0, 'prayer': 8, 'song': 67, 'sonnet': 0}
```

Figure 3. Code to extract token count for each class in training data

Interpreting individual features becomes increasingly complex when comparing multiple classification algorithms. Workflows and pipelines used in many machine learning tasks iterate through different parameters and algorithms in search of the greatest accuracy score. As previously mentioned, the algorithms reporting highest accuracy might not necessarily be the most interpretable choices and the features selected might not be meaningful in the available interpretive contexts. The features determined as key for classification can be dramatically different even if the overall performance on the classification task is roughly similar. While this might suggest that we do not need to spend valuable time interpreting the individual features that were indicated as important, this is simply not true. The Naïve Bayes classification algorithm, when applied to the exact same training data and features as those used in the Support Vector Machines classification task displayed above, results in a slightly higher accuracy score, 94% total accuracy across the three classes. The following accuracy report was produced by Scikit-Learn’s multinomial Naïve Bayes classifier: precision recall f1-score support ballad 0.66 0.58 0.62 116 prayer 0.99 0.98 0.98 1151 song 0.91 0.94 0.92 733 avg/total 0.94 0.94 0.94 2000

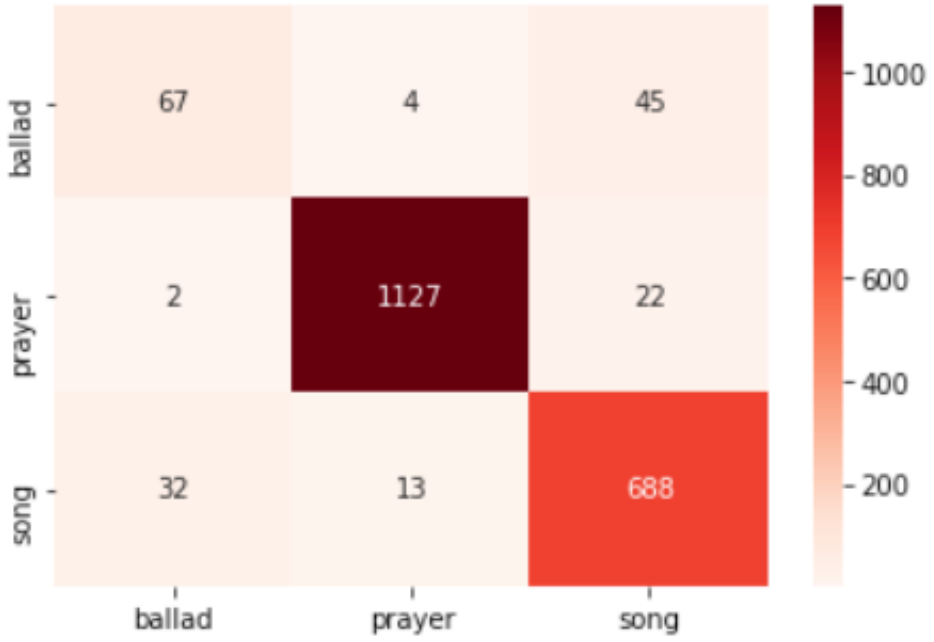

Figure 4. Naïve Bayes Confusion Matrix

The Scikit-learn multinomial implementation of Naïve Bayes uses a mechanism similar to the package's SVM implementation to access and display what the algorithm deems the most important features for making classification decisions among input data. It is one of the few Naïve Bayes classifiers in Scikit-learn that can provide access to these data. For Hoyt Long and Richard Jean So, the probability values used by Naïve Bayes classifiers and the ability to learn the criteria used in misclassifications were key to their selection of this method for finding statistical patterns in English-language haiku, although they used a different implementation of Naïve Bayes [Long and Jean So 2016]. Values for individual features are stored as log probability values (available via classifier's `feature_log_prob_` or formerly the `coef_` property) for each trained class. In addition, the Naïve Bayes classifier has a property (`feature_count_`) that can be accessed to get raw word or feature counts or weighted counts for the training data as found within labeled data. Table 3 displays the probability values for the features within each class or category of labeled data. The multinomial Naïve Bayes classifier's log probability values are negative numbers because these are result of the logarithm of numbers between 0 and 1 and values that are closer to zero have a greater probability value and thus are more statistically important to the classification. While this Naïve Bayes classifier has a similar and even slightly higher accuracy across the model, it is not able to classify ballads as well as the SVM classifier. One reason might be the significantly less important role features such as "pdf" and "page" play in determining classification of these texts. The Naïve Bayes classifier has trouble distinguishing between ballads and songs, although far fewer songs are misclassified as ballads than ballads misclassified as songs. There are many features shared between the classes and interpreting the significance of these terms collectively across the classes and within each class requires investigation of both quantitative and semantic meaning.

Ballad	Prayer	Song
did (-4.916), thy (-5.467), thou (-5.634), ſo (-5.747), man (-5.749), good (-5.78), ſhe (-5.895), like (-5.962), men (-5.996), let (-6.027), king (-6.086), doth (-6.089), love (-6.092), god (-6.13), make (-6.168), haue (-6.174), come (-6.258), ſhall (-6.258), day (-6.301), great (-6.329), lord (-6.349), thee (-6.351), doe (-6.385), tune (-6.411), true (-6.431), tis (-6.539), heart (-6.577), hath (-6.59), wife (-6.644), came (-6.65)	thy (-3.224), thou (-4.004), thee (-4.052), lord (-4.347), vs (-4.589), god (-4.605), vnto (-5.185), let (-5.185), haue (-5.339), good (-5.391), life (-5.483), father (-5.486), holy (-5.488), thine (-5.525), hast (-5.545), amen (-5.622), christ (-5.682), grace (-5.717), art (-5.751), make (-5.761), shall (-5.809), vpon (-5.87), unto (-5.872), death (-5.883), things (-5.892), mercy (-5.892), great (-5.904), world (-5.954), come (-5.972), mee (-5.989)	la (-3.76), thy (-4.688), repeat (-4.821), thou (-4.996), loue (-5.215), thee (-5.251), did (-5.279), ij (-5.305), doth (-5.391), let (-5.425), fa (-5.464), love (-5.569), lord (-5.638), like (-5.684), shall (-5.727), come (-5.743), song (-5.753), god (-5.758), haue (-5.85), make (-5.937), ſo (-5.945), hath (-5.955), man (-5.991), doe (-5.994), ii (-6.06), day (-6.062), men (-6.087), good (-6.089), great (-6.122), vs (-6.153)

Table 3. Most important features within each class of lyrical text for the Naïve Bayes classifier

Conclusion

From a humanist perspective, we might want to think of data models created from textual sources as alternative representations of supplied texts, and transformations of these might be, as Katherine Bode argues, performative materializations of the text sources [Bode 2020]. These might be creative performances, in line with the notion of computational transformation as deformance, or argumentative acts. Even as an act or performance, humanists would best serve their audience by selecting, as the basic building blocks of their work, data models and algorithms that enable the greatest degree of interpretability. Understanding the significance of a particular instantiation of the execution or performance requires an attentive act by the audience. To pay attention means to enter into collaborative meaning making with the critical/creative work. This collaborative meaning-making activity is licensed by access to a shared vocabulary and the potential space for playful manipulation. If argumentative claims are put forward, these are then evaluated and warranted by shared assumptions and the ability to test and verify that the data are indeed comprehensible according to the norms of the shared interpretative community. Inspection and interpretation thus function together to animate the relationship between researcher and reader. Making computational work interpretable is essential to preserving two distinct threads within the digital humanities: upholding the standards of responsible scholarship, as articulated by the move toward open and reproducible workflows, and enabling the shared meaning-making activity between critic and reader that characterizes much humanistic interpretation. 32

Throughout this essay we have seen the limits to interpretability found in the selection and use of common vector-based data models, topic modeling algorithms and parameters, and in classification algorithms. Computational workflows are composite and modular. Alternative procedures and choices exist at almost every level within a workflow. This is both an asset and a liability. As a mode of conducting research, this flexibility enables digital humanists to select and link together the best models and algorithms for their purposes but as abstraction and complexity increases so too do the risks to interpretability and access to underlying data diversity. The ability for humanists, in particular, to examine and interpret the texts and methods used to make a model is as important if not more important than the reporting of the overall best results of a model — we saw this in the topic modeling case study — or the accuracy of a particular classification model. It is for all these reasons that humanists making use of computational methods to conduct their research need to select and privilege those models and methods that most enable inspection, exploration, and interpretation of diverse data and parameters. 33

Acknowledgments

Much of my thinking about interpretive issues related to computation in the humanities has been the result of conversations and debates with Aden Evens. I would also like to thank the many students who have taken my Cultural Analytics courses and worked with me as research assistants, especially Julianna Thomson and Catherine Parnell, for 34

their probing questions and for putting so much of the theory into practice.

Works Cited

Algee-Hewitt et al. 2017 Algee-Hewitt, Mark, Ryan Heuser, and Franco Moretti. “On Paragraphs: Scale, Themes, and Narrative Form.” In *Canon/Archive: Studies in Quantitative Formalism from the Stanford Literary Lab*. Edited by Franco Moretti. New York: n+1 books, 2017, 65-94.

Allison et al. 2017 Allison, Sarah, Marissa Gemma, Ryan Heuser, Franco Moretti, Amir Tevel, and Irena Yamboliev. “Style at the Scale of the Sentence.” In *Canon/Archive: Studies in Quantitative Formalism from the Stanford Literary Lab*. Edited by Franco Moretti. New York: n+1 books, 2017, 33-63.

Ananny and Crawford 2018 Ananny, Mike and Kate Crawford, “Seeing without Knowing: Limitations of the Transparency Ideal and Its Application to Algorithmic Accountability.” *New Media & Society* 20, no. 3 (March 2018): 973–89.

Blei 2012 Blei, David M. “Probabilistic Topic Models.” *Communications of the ACM* 55, no. 4 (April 1, 2012): 77-84.

Bode 2017 Bode, Katherine. “The Equivalence of ‘Close’ and ‘Distant’ Reading; or, Toward a New Object for Data-Rich Literary History.” *Modern Language Quarterly* 78, no. 1 (March 2017): 77–106.

Bode 2020 Bode, Katherine. “Data Beyond Representation: From Computational Modelling to Performative Materiality.” Presented at the Annual MLA Convention, Seattle, WA, January 2020. https://katherinebode.wordpress.com/home/mla-convention-2020/.

Burckhardt 1968 Burckhardt, Sigurd. “Notes on the Theory of Intrinsic Interpretation.” In *Shakespearean Meanings*. Princeton, NJ: Princeton University Press, 1968.

Capitanu et al. 2016 Capitanu, Boris, Ted Underwood, Peter Organisciak, Timothy Cole, Maria Janina Sarol, J. Stephen Downie *The HathiTrust Research Center Extracted Feature Dataset* (1.0) [Dataset]. HathiTrust Research Center. (2016).

Chuang et al. 2012 Chuang, Jason, Christopher D. Manning, and Jeffrey Heer, “Termite: Visualization Techniques for Assessing Textual Topic Models.” In *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI ’12*. The International Working Conference. Capri Island, Italy: ACM Press, 2012, 74-78.

Da 2019 Da, Nan Z. “The Computational Case against Computational Literary Studies.” *Critical Inquiry* 45, no. 3 (March 2019): 601–39.

Dobson 2019 Dobson, James E. *Critical Digital Humanities: The Search for a Methodology.* Urbana: University of Illinois Press, 2019.

Fish 1980 Fish, Stanley. *Is There a Text in This Class? The Authority of Interpretive Communities.* Cambridge, MA: Harvard University Press, 1980.

Flanders and Jannidis Flanders, Julia and Fotis Jannidis. “Data Modeling in a Digital Humanities Context: An Introduction.” In *The Shape of Data in the Digital Humanities: Modeling Texts and Text-based Resources*, edited by Julia Flanders and Fotis Jannidis. New York: Routledge, 2019, 3-25.

Gebru et al. 2018 Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. “Datasheets for Datasets.” *CoRR* abs/1803.09010 (2018). http://arxiv.org/abs/1803.09010

Hamilton et al. 2016 Hamilton, William L., Jure Leskovec, and Dan Jurafsky, “Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change,” *ArXiv Preprint ArXiv:1605.09096*, 2016, https://arxiv.org/abs/1605.09096.

Hartman 1980 Hartman, Geoffrey H. *Criticism in the Wilderness: The Study of Literature Today*. New Haven: Yale University Press, 1980.

Jockers and Witten 2010 Jockers Matthew L. and Daniela M. Witten. “A Comparative Study of Machine Learning Methods for Authorship Attribution.” *Literary and Linguistic Computing* 25, no. 2 (2010): 215–223.

Long and Jean So 2016 Long, Hoyt, and Richard Jean So. “Literary Pattern Recognition: Modernism between Close Reading and Machine Learning.” *Critical Inquiry* 42, no. 2 (January 2016): 235–67.

Mackenzie 2017 Mackenzie, Adrian. *Machine Learners: Archaeology of a Data Practice*. Cambridge: MIT Press, 2017.

Miller 2019 Miller, Tim. “Explanation in Artificial Intelligence: Insights from the Social Sciences.” *Artificial Intelligence* 267

(February 2019): 1–38.

Mitchell et al. 2019 Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. “Model Cards for Model Reporting,” *Proceedings of the Conference on Fairness, Accountability, and Transparency - FAT* ’19* (2019): 220–229. https://doi.org/10.1145/3287560.3287596.

Montfort 2018 Montfort, Nick. *The Truelist*. Denver: Counterpath, 2018.

Paßmann 2017 Paßmann, Johannes and Asher Boersma, “Unknowing Algorithms: On Transparency of Unopenable Black Boxes.” In *The Datafied Society*, edited by Mirko Tobias Schäfer and Karin van Es. Amsterdam: Amsterdam University Press, 2017, 139–46.

Ramsay 2014 Ramsay, Stephen. “The Hermeneutics of Screwing Around; or What You Do with a Million Books.” In *Pastplay: Teaching and Learning History with Technology*, edited by Kevin Kee. Ann Arbor: University of Michigan Press, 2014.

Rawson and Muñoz 2019 Rawson Katie and Trevor Muñoz, “Against Cleaning.” In *Debates in the Digital Humanities 2019*, edited by Matthew K Gold and Lauren F. Klein. Minneapolis: University of Minnesota Press, 2019, 279–92.

Schmidt 2012 Schmidt, Benjamin. “Words Alone: Dismantling Topic Models in the Humanities.” *Journal of Digital Humanities* 2, no. 1 (Winter 2012).

Scikit-Learn 2020 Scikit-Learn. Documentation. (2020). https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html

Toulmin 1969 Toulmin, Stephen Edelston. *The Uses of Argument.* Cambridge University Press, 1969.

Yu 2008 Yu, Bei. “An Evaluation of Text Classification Methods for Literary Study.” *Literary and Linguistic Computing* 23, no. 3 (2008): 327–43.

van Es et al. 2018 van Es, Karin, Maranke Wieringa, and Mirko Tobias Schäfer. “Tool Criticism: From Digital Methods to Digital Methodology.” In *Proceedings of the 2nd International Conference on Web Studies - WS.2 2018*, 24–27. Paris, France: ACM Press, 2018.