Supplementary Information for Improving Probabilistic Models in Text Classification via Active Learning[*]

Mitchell Bosley[†][‡] Saki Kuzushima[†][§] Ted Enamorado[¶]
Yuki Shiraito[‖]

First draft: September 10, 2020
Final submission: April 22, 2024

Contents

[*] We thank Ken Benoit, Yaoyao Dai, Chris Fariss, Yusaku Horiuchi, Kosuke Imai, Walter Mebane, Daichi Mochihashi, Kevin Quinn, Luwei Ying, audiences at the 2020 Annual Meeting of the American Political Science Association, the 2021 Annual Meeting of the Midwest Political Science Association, the 11th Annual Conference on New Directions in Analyzing Text as Data, the 2022 Summer Meeting of the Japanese Society for Quantitative Political Science, and the 40th Annual Summer Meeting of the Society for Political Methodology, and seminar participants at the University of Michigan and members of the Junior Faculty Workshop at Washington University in St. Louis for useful comments and suggestions. We also appreciate detailed and constructive comments from four anonymous reviewers of the journal. Finally, we are extremely grateful to the editor, Michelle Dion, for guiding us through the rigorous review process of *APSR*.

[†] These authors have contributed equally to this work.

[‡] Ph.D. Candidate, Department of Political Science, University of Michigan. Email: mcbosley@umich.edu. ORCID: 0000-0002-9172-966X.

[§] Ph.D. Candidate, Department of Political Science, University of Michigan. Email: skuzushi@umich.edu. ORCID: 0000-0003-3014-5203.

[¶] Assistant Professor, Department of Political Science, Washington University in St. Louis. Siegle Hall, 244. One Brookings Dr. St Louis, MO 63130-4899. Phone: 314-935-5810, Email: ted@wustl.edu, URL: www.tedenamorado.com. ORCID: 0000-0002-2022-7646.

[‖] Assistant Professor, Department of Political Science, University of Michigan. Center for Political Studies, 4259 Institute for Social Research, 426 Thompson Street, Ann Arbor, MI 48104-2321. Phone: 734-615-5165, Email: shiraito@umich.edu, URL: shiraito.github.io. ORCID: 0000-0003-0264-1138.

A Using Machine Learning for Text Classification

In this section we discuss the process of encoding text data into matrix form for the purpose of classifying e.g., whether a document refers to politics or not. We describe Supervised and Unsupervised Learning. In addition, this discussion extends to the selection between discriminative and generative models, shedding light on the advantages and trade-offs of each. Finally, we discuss the issue of model evaluation in text classification tasks, with a focus on using a validation dataset (test data) and evaluation metrics such as accuracy, precision, recall, and the F1 score.

A.1 Encoding Text in Matrix Form

Suppose that a researcher has a collection of social media text data, called a corpus, and wishes to classify whether each text in a corpus is political (e.g., refers to political protest, human rights violations, unfavorable views of a given candidate, targeted political repression, etc.) or not solely based on the words used in a given observation. Critically, the researcher does not yet know which of the texts are political or not at this point.

The researcher must first choose how to represent text as a series of *tokens*, and decide which tokens to include in their analysis. This involves a series of sub-choices, such as whether each token represents an individual word (such as "political") or a combination of words (such as "political party"), whether words should be stemmed or not (e.g., reducing both "political" and "politics" their common stem "politic"), and whether to remove stop-words (such as "in", "and", "on", etc.) that are collectively referred to as *pre-processing.*[1]

The researcher must then choose how to encode information about these tokens in matrix form. The most straightforward way to accomplish this is using a *bag-of-words* approach, where the corpus is transformed into a document-feature matrix (DFM) $\mathbf{D}$ with n rows and m columns, where n is the number of documents and m is the number of tokens, which are more generally referred to as features.[2] Each element of the DFM encodes the frequency that a token occurs in a given document.[3] Once the researcher chooses how to encode their

[1] For a survey of pre-processing techniques and their implications for political science research, see Denny and Spirling (2018).

[2] Note that in the machine learning literature, the concept typically described by the term "variable" is communicated using the term "feature."

[3] An alternative to the bag-of-words approach is to encode tokens as *word embeddings*, where in addition to the matrix summarizing the incidences of words in each document, neural network models are used to create vector representations of each token. In this framework, each token is represented by a vector of some arbitrary length, and tokens that are used in similar contexts in the corpus (such as "minister" and "cabinet") will have similar vectors. While this approach is more complicated, it yields considerably more information about the use of words in the corpus than the simple count that the bag-of-words approach does. For an accessible introduction to the construction and use of word embeddings in political science research, see Rodriguez and Spirling (2022). For a more technical treatment, see Pennington et al. (2014).

corpus as a matrix, she is left with a set of features corresponding to each document **D** and an unknown vector of true labels **Z**, where each element of **Z** indicates whether a given document is political or not. Then, we can repose the classification question as follows: given **D**, how might we best learn **Z**, that is, whether each document is political or not?

A.2 Supervised, Unsupervised, and Semi-supervised Learning

Supervised and unsupervised learning are two fundamental approaches in machine learning for text-as-data (Grimmer et al., 2022). In the supervised approach, the researcher follows these steps: (1) acquiring accurate labels for a subset of the documents through human coding, where e.g., the researcher determines that a news headline such as "Biden and Trump clinch nominations, heading to a general election rematch" refers to politics due to its focus on a political figures (Presidents Biden and Trump) and their role in the upcoming general election; (2) establishing a connection between the textual features of each document in the corpus as represented by a matrix **D** and the true labels represented by a vector **Z** for the labeled documents. This involves understanding how terms like "Biden," "Trump," "clinch," "nominations," "general," "election," and "rematch" are important in determining the political nature of the headline; and (3) utilizing the acquired understanding of the relationship between the text data and the known labels to later predict whether the remaining unlabeled documents in the corpus are political or not (Hastie et al., 2009).

On the other hand, an unsupervised approach does not require the use of labeled data. Instead, the researcher employing an unsupervised approach would select a model that groups documents in the corpus based on shared patterns in the features as represented by the matrix **D**. After assigning documents to clusters, the researcher would determine which cluster corresponds to the desired outcome of interest, namely whether a document is political or not. However, it is important to note that there is no guarantee that there will be a direct connection between clusters and the outcomes of interest (Knox et al., 2022).

Semi-supervised learning combines supervised and unsupervised approaches (Miller and Uyar, 1996; Nigam et al., 2000), making it particularly useful when there is a large amount of unlabeled data and labeling is expensive. In a semi-supervised model, the relationship between the text data matrix **D** and the classification outcome **Z** is learned using both labeled and unlabeled data. Although **Z** is not known for unlabeled data, it still provides information about the joint distribution of the features in **D**. Thus, by incorporating patterns recovered from the unlabeled data and using the labeled data as a foundation for measurement, semi-supervised learning produces more accurate and robust predictions compared to purely supervised or unsupervised methods (Nigam et al., 2000).

A.3 Discriminative vs. Generative Models

In addition to choosing a supervised, unsupervised, or semi-supervised approach as described in Section "Machine Learning Approaches to Text Classification," a researcher must also choose whether to use a discriminative or generative model. As noted by Ng and Jordan (2001) and Bishop and Lassarre (2007), when using a discriminative model (e.g., logistic regression, SVM, etc.), the goal is to directly estimate the probability of the classification outcomes $\mathbf{Z}$ given the text data $\mathbf{D}$ i.e., directly estimate $p(\mathbf{Z}|\mathbf{D})$. In contrast, when using a generative model (e.g., Naive Bayes), learning the relationship between the $\mathbf{Z}$ and $\mathbf{D}$ is a two-step process. In the first step, the likelihood of the matrix of text data $\mathbf{D}$ and outcome labels $\mathbf{Z}$ is estimated given the data and a set of parameters θ that indicate structural assumptions about how the data is generated. In other words, $p(\mathbf{D}, \mathbf{Z}|\theta)$ is directly estimated. In the second step, the researcher uses Bayes' rule to calculate the probability of the outcome vector given the features and the learned distribution of the parameters i.e., $p(\mathbf{Z}|\mathbf{D}; \theta)$.

In addition to allowing for the use of unlabeled data (which reduces labeling costs), one of the main benefits of a generative rather than a discriminative model is that the researcher can include information they know about the data generating process by choosing appropriate functional forms.[4] This can help prevent overfitting when the amount of data in a corpus is small.[5] Conversely, because it is not necessary to model the data generating process directly, the main benefit of a discriminative rather than generative model is simplicity (in general it involves estimating fewer parameters). Discriminative models are therefore appropriate in situations where the amount of data in a corpus is very large, and/or when the researcher is unsure about the data-generating process, which could lead to mis-specification (Bishop and Lassarre, 2007).[6]

A.4 Model Evaluation

A researcher must also decide when she is satisfied with the predictions generated by the model. In most circumstances, the best way to evaluate the performance of a classification algorithm is to reserve a subset of the corpus for validation, which is sometimes referred to

[4] This is particularly true when e.g., the researcher knows that the data has a complicated hierarchical structure since the hierarchy can be incorporated directly into the generative model.

[5] Overfitting occurs when a model learns to predict classification outcomes based on patterns in the training set (i.e., the data used to fit the model) that does not generalize to the broader universe of cases to be classified. A model that is overfitted may predict the correct class with an extremely high degree of accuracy for items in the training set, but will perform poorly when used to predict the class for items that the model has not seen before.

[6] Another benefit of generative models is that they can yield better estimates of how certain we are about the relationship between the outcome and the features. This is the case when a researcher uses an inference algorithm like Markov Chain Monte Carlo (MCMC) that learns the entire distribution for each of the parameters, rather than only point estimates.

as validation and/or test set. At the very beginning of the classification process, a researcher puts aside and label a set of randomly chosen documents that the active learning algorithm does not have access to.[7] Then, after training the model on the remainder of the documents (often called the training set), the researcher should generate predictions for the documents in the validation set using the trained model. By comparing the predicted labels generated by the model to the actual labels, the researcher can evaluate how well the model does at predicting the correct labels.

A common tool for comparing the predicted labels to the actual labels is a *confusion matrix*. In a binary classification setting, a confusion matrix will be a 2 by 2 matrix, with rows corresponding to the actual label, and the columns corresponding to the predicted label. Returning to our running example, imagine that the classification is to predict whether documents are political or not, Table A.1 shows the corresponding confusion matrix. In this scenario, True Positives (TP) are the number of documents that the model predicts to be about politics and that is in fact labeled as such. Correspondingly, True Negatives (TN), are the number of documents that the model predicts to be non-political and is labeled as such in the validation set. A False Negative (FN) occurs when the model classifies a document as non-political, but the document is about politics. Similarly, a False Positive (FP) occurs when the model classifies as political a document that is non-political.

		Predicted Label	
		Political	Non-political
Actual Label	Political	True Positive (TP)	False Negative (FN)
	Non-political	False Positive (FP)	True Negative (TN)

Table A.1: **Confusion Matrix: Comparison of the Predictions of a Classifier to Documents' True Labels**

Using the confusion matrix, the researcher can calculate a variety of evaluation statistics. Some of the most common of these are accuracy, precision, and recall. Accuracy is the proportion of documents that have been correctly classified. Precision is used to evaluate the false positivity rate and is the proportion of the model's positive classifications that are true positives. As the number of false positives increases (decreases), precision decreases (increases). Recall is used to evaluate the false negativity rate, and is the proportion of the actual positive documents that are true positives. As the number of false negatives increases, recall decreases, and *vice-versa*. Accuracy, precision, and recall can be formally calculated

[7] It is important to use a set-aside validation set for testing model performance, rather than a subset of the documents used to train the model, to avoid *overfitting*.

as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}; \qquad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

When the proportion of political and non-political documents in a corpus is balanced, accuracy is an adequate measure of model performance. However, it is often the case in text classification that the corpus is unbalanced, and the proportion of documents associated with one class is low. When this is the case, accuracy does a poor job at model evaluation. Consider the case when 99 percent of documents are non-political, and 1 percent are about politics. A model which simply predicts that all documents belong to the non-politics class would have an accuracy score of 0.99, but would be poorly suited to the actual classification task. In contrast, the precision and recall rates would be 0, which would signal to the researcher that the model does a poor job at classifying documents as political. Precision and recall are not perfect measures of model performance, however. There is a fundamental trade-off involved in controlling the false positivity and false negativity rates: you can have few false positives if you are content with an extremely high number of false negatives, and you can have few false negatives if you are content with an extremely high number of false positives.

Recognizing this trade-off, researchers often combine precision and recall scores to find a model that has the optimal balance of the two. One common way of combining the two is an F1 score, which is the harmonic mean of precision and recall. Formally, the F1 score is calculated as:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score assigns equal importance to precision and recall, and so a high F1 score would indicate that both the false negativity and false positivity rate are low. It is worth noting these evaluation measures (accuracy, precision, recall, and the F1 score) are computed using labeled data ("ground truth"), which in practice, are available only for a limited subset of the records.

B Supplemental Information for "Validation Performance"

We describe the data, explain decisions regarding pre-processing steps, and present additional results supplementing those presented in the Section "Validation Performance."

B.1 Data Description

The following four datasets are used to validate the performance of *activeText*:

BBC News: The BBC News Dataset is a collection of 2,225 documents from 2004 to 2005 available at the BBC news website (Greene and Cunningham, 2006). This dataset is divided equally into five topics: business, entertainment, politics, sport, and technology. For example, the binary classification exercise is to correctly predict whether or not an article belongs to the 'politics' topic.

Wikipedia Toxic Comments: The Wikipedia Toxic Comments dataset is a dataset made up of conversations between Wikipedia editors in Wikipedia's internal forums. The dataset was made openly available as part of a Kaggle competition,[8] and was used as a principle dataset of investigation by Miller et al. (2020). The basic classification task is to label a given speech as toxic or not, where toxicity is defined as including harassment and/or abuse of other users.[9] The complete dataset is comprised of approximately 560,000 documents, where roughly 10 percent of which are labeled as toxic.

Supreme Court Cases: The Supreme Court Rulings dataset is a collection of the text of 2000 US Supreme Court rulings between 1946 and 2012. We use the majority opinion of each case and the text was obtained through Caselaw Access Project.[10] For the classification label, we use the categories created by the Supreme Court Database.[11] The classification exercise here is to correctly identify rulings that are categorized as 'criminal procedure', which is the largest category in the corpus (26% of all rulings).

Human Rights Allegation: Human Rights Allegation dataset contains more than 2 million sentences of human rights reports in 196 countries between 1996 and 2016, produced by Amnesty International, Human Rights Watch and the US State Department (Cordell et al., 2021). The classification goal is to identify sentences with physical integrity rights

[8] See `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

[9] While the dataset also contains finer gradation of 'types' of toxicity, we like Miller et al. (2020) stick to the binary toxic-or-not classification task.

[10] `https://case.law`

[11] For a full list of categories, see `http://www.supremecourtdatabase.org/documentation.php?var=issueArea`.

allegation (16% of all reports). Example violations of physical integrity rights include torture, extrajudicial killing, and arbitrary arrest and imprisonment.

B.2 Pre-processing Steps

We employ the same pre-processing step for each of the four datasets using the *R* package *Quanteda*.[12] For each dataset, we construct a *document-feature matrix* (DFM), where each row is a document and each column is a feature. Each feature is a stemmed unigram. We remove stopwords, features that occur extremely infrequently, as well as all features under 4 characters.

To generate datasets with a fixed proportion p (e.g. 5% or 50%) of the main class of interest, we randomly sample documents from the original dataset so that it achieves the proportion p. Let's use the BBC dataset as an example. Suppose the number of documents in the BBC dataset is N with N_1 and N_0 representing the total number of documents about politics and non-politics in the original data, respectively. We compute $M_1 = \text{floor}(Np)$ and $M_0 = N - M_1$ as the number of sampled politics and non-politics documents, respectively.[13] When $M_1 > N_1$ or when $M_0 > N_0$, we decrement M_1 and M_0 keeping the politics proportion (p) to its target value. When $M_1 < N_1$ and $M_0 < N_0$, we sample M_1 politics documents and M_0 non-politics documents from the original dataset. Finally, combine the sampled politics and non-politics documents to obtain the final dataset. This sampling scheme is applied to all the other datasets when generating the validation datasets with a fixed proportion p.

[12] See `https://quanteda.io`

[13] The floor function takes as input a real number x and returns the greatest integer larger or equal to x. For example, the floor(3.1415) is 3.

Note that in the main text, to facilitate exposition, we use the terms political and non-political to describe the problem of binary classification. Without loss of generality, in the sections below, we use the (more generic) positive vs. negative class dichotomy instead.

In addition, note that for a matrix $\mathbf{X}$, we denote its ath row by $\mathbf{X}_{a\cdot}$. In other words, we use the subscript $a\cdot$ to denote the ath row of $\mathbf{X}$. Similarly, we use the subscript $\cdot b$ to refer to the bth column of $\mathbf{X}$.

C Binary Classification: EM Algorithm

We now describe how we estimate the parameters of the probabilistic model behind our approach. As mentioned in the main text, our approach is built upon the work of Nigam et al. (2000), which showed that probabilistic classifiers can be enhanced by integrating information from both labeled and unlabeled data.

Let's consider the task of classifying documents into one of two classes (e.g., positive vs. negative). However, as we discuss below, our approach can be extended to accommodate: 1) a multi-class classification setting where $K > 2$ and each document is assigned to one of K classes, such as classifying news articles into politics, business, and sports (refer to SI D), and 2) modeling more than two classes while maintaining a binary final classification output (refer to SI E).

Under the same model structure defined in Section "The Method" of the main text, let $\mathbf{D}^{lp}$, $\mathbf{D}^{ln}$ and $\mathbf{D}^{u}$ be the document feature matrices for documents with positive labels, documents with negative labels, and unlabeled documents, respectively. Also let N^{lp}, N^{ln}, and N^{u} be the number of documents with positive labels, negative labels, and documents without labels.[14] Likewise, $\mathbf{C}^{lp}$ and $\mathbf{C}^{ln}$ be the vectors of positive and negative labels. Then, the observed-data likelihood is:

[14] Consequently, the number of labeled documents N^{l} is equal to $N^{lp} + N^{ln}$.

$$
\begin{aligned}
&p(\pi, \boldsymbol{\eta}|\mathbf{D}, \mathbf{C}^{lp}, \mathbf{C}^{ln}) \\
&\propto p(\pi)p(\boldsymbol{\eta})p(\mathbf{D}^{lp}, \mathbf{C}^{lp}|\pi, \boldsymbol{\eta})p(\mathbf{D}^{ln}, \mathbf{C}^{ln}|\pi, \boldsymbol{\eta})\Big[p(\mathbf{D}^{u}|\pi, \boldsymbol{\eta})\Big]^{\lambda} \\
&= p(\pi)p(\boldsymbol{\eta}) \times \prod_{i=1}^{N^{lp}} p(\mathbf{D}^{lp}_{i\cdot}|Z_i = 1, \boldsymbol{\eta}_{\cdot 1})p(Z_i = 1|\pi) \times \prod_{i=1}^{N^{ln}} \Big\{p(\mathbf{D}^{ln}_{i\cdot}|Z_i = 0, \boldsymbol{\eta}_{\cdot 0})p(Z_i = 0|\pi)\Big\} \\
&\quad \times \left[\prod_{i=1}^{N^{u}} \Big\{p(\mathbf{D}^{u}_{i\cdot}|Z_i = 1, \boldsymbol{\eta}_{\cdot 1})p(Z_i = 1|\pi) + p(\mathbf{D}^{u}_{i\cdot}|Z_i = 0, \boldsymbol{\eta}_{\cdot 0})p(Z_i = 0|\pi)\Big\}\right]^{\lambda} \\
&\propto \underbrace{\big\{(1-\pi)^{\alpha_0 - 1}\prod_{v=1}^{V} \eta_{v0}^{\beta_{v0}-1}\big\} \times \big\{\pi^{\alpha_1 - 1}\prod_{v=1}^{V} \eta_{v1}^{\beta_{v1}-1}\big\}}_{\text{prior}} \times \underbrace{\prod_{i=1}^{N^{lp}} \Big\{\prod_{v=1}^{V} \eta_{v1}^{D_{iv}} \times \pi\Big\}}_{\text{positive labeled doc. likelihood}} \\
&\quad \times \underbrace{\prod_{i=1}^{N^{ln}} \Big\{\prod_{v=1}^{V} \eta_{v0}^{D_{iv}} \times (1-\pi)\Big\}}_{\text{negative labeled doc. likelihood}} \times \underbrace{\left[\prod_{i=1}^{N^{u}} \Big\{\prod_{v=1}^{V} \eta_{v0}^{D_{iv}} \times (1-\pi)\Big\} + \Big\{\prod_{v=1}^{V} \eta_{v1}^{D_{iv}} \times \pi\Big\}\right]^{\lambda}}_{\text{unlabeled doc. likelihood}}
\end{aligned}
\tag{1}
$$

As described in Section "The Method" of the main text, we weigh the part of the observed likelihood that refers to the unlabeled document with $\lambda \in [0, 1]$. This is done because we typically have many more unlabeled documents than labeled documents. By downweighting the information from the unlabeled document (i.e., setting λ to be small), we can use more reliable information from labeled documents than from unlabeled documents.

We estimate the parameters using EM algorithm Dempster et al. (1977). Note that by taking the expectation of the complete-data log-likelihood function (Q function), we have that:

$$
\begin{aligned}
&Q \equiv \mathbb{E}_{\mathbf{Z}|\pi^{(t)}, \boldsymbol{\eta}^{(t)}, D, C}[\log p(\pi, \boldsymbol{\eta}, \mathbf{Z}|\mathbf{D}, \mathbf{C})] \\
&\quad = (\alpha_0 - 1)\log(1 - \pi^{(t)}) + (\alpha_1 - 1)\log \pi^{(t)} + \sum_{v=1}^{V} \Big\{(\beta_{v0} - 1)\log \eta_{v0}^{(t)} + (\beta_{v1} - 1)\log \eta_{v1}^{(t)}\Big\} \\
&\qquad + \sum_{i=1}^{N^{lp}} \Big\{\sum_{v=1}^{V} D_{iv} \log \eta_{v1}^{(t)} + \log \pi^{(t)}\Big\} + \sum_{i=1}^{N^{ln}} \Big\{\sum_{v=1}^{V} D_{iv} \log \eta_{v0}^{(t)} + \log(1 - \pi^{(t)})\Big\} \\
&\qquad + \lambda \left[\sum_{i=1}^{N^{u}} p_{i0}\Big\{\sum_{v=1}^{V} D_{iv} \log \eta_{v0}^{(t)} + \log(1 - \pi^{(t)})\Big\} + p_{i1}\Big\{\sum_{v=1}^{V} D_{iv} \log \eta_{v1}^{(t)} + \log \pi^{(t)}\Big\}\right]
\end{aligned}
\tag{2}
$$

where p_{ik} is the probability of a document i being assigned to the kth class, $k = \{0, 1\}$, given data and the parameters at t th iteration. If a document has been hand-coded and has a positive label, we have that $p_{i1} = 1$. If a document has been hand-coded and has been assigned a negative label, we have that $p_{i0} = 0$. If a document has no label, then:

$$\begin{aligned} p_{i0} &= 1 - p_{i1} \\ p_{i1} &= \frac{\prod_{v=1}^{V} \eta_{v1}^{D_{iv}} \times \pi}{\prod_{v=1}^{V} \left\{\eta_{v0}^{D_{iv}} \times (1-\pi)\right\} + \prod_{v=1}^{V} \left\{\eta_{v1}^{D_{iv}} \times \pi\right\}} \end{aligned} \tag{3}$$

Equation 3 also works as the prediction equation. The predicted class of a document i is k that maximizes this probability.

In the M-step, we maximize the Q function, and obtain the updating equations for π, η_{v1}, and η_{v0}. The updating equation for π is the following:

$$\hat{\pi}^{(t+1)} = \frac{\alpha_1 - 1 + N^{lp} + \lambda \sum_{i=1}^{N^u} p_{i1}}{\left(\alpha_1 - 1 + N^{lp} + \lambda \sum_{i=1}^{N^u} p_{i1}\right) + \left(\alpha_0 - 1 + N^{ln} + \lambda \sum_{i=1}^{N^u} p_{i0}\right)} \tag{4}$$

The updating equations for each η_{vk} for $k \in \{0, 1\}$ and $v \in \{1, \ldots, V\}$, are as follows:

$$\begin{aligned} \hat{\eta}_{v0}^{(t+1)} &\propto (\beta_{v0} - 1) + \sum_{i=1}^{N^{ln}} D_{iv} + \lambda \sum_{i=1}^{N^u} p_{i0} D_{iv}, \quad v = 1, \ldots, V \\ \hat{\eta}_{v1}^{(t+1)} &\propto (\beta_{v1} - 1) + \sum_{i=1}^{N^{lp}} D_{iv} + \lambda \sum_{i=1}^{N^u} p_{i1} D_{iv}, \quad v = 1, \ldots, V \end{aligned} \tag{5}$$

We continue cycling between the E-step and M-step until we reach convergence. In Figure C.1, we graphically present the mixture model at the core of our active learning algorithm. In this representation, hyperparameters are shown in shaded rectangles, random variables and parameters in unshaded circles, and observed data in shaded circles. The larger unshaded rectangular plates are used to indicate the scope of indices for various types of data and parameters.

In Section "An Active Learning Algorithm," we explain how we integrate this model into our algorithm. In each iteration, we follow these steps: first, we fit the model, then we learn its parameters, next, we label the most uncertain cases, and finally, we refit the model with the newly labeled data. We repeat this cycle until a stopping rule is met. We present the implementation of our active learning algorithm as pseudocode in Algorithm 1.

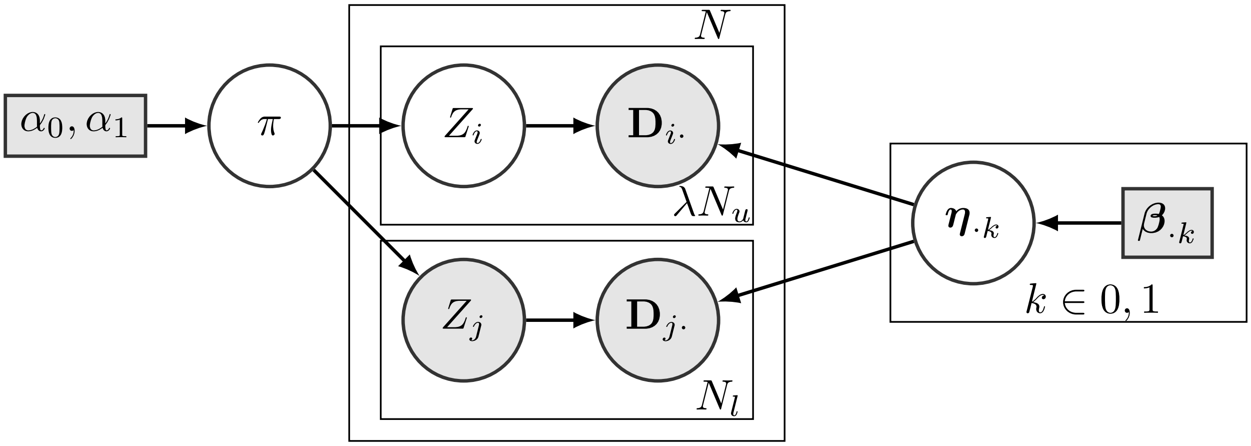

Figure C.1: **Graphical Representation of the Mixture Model with Two Classes.** Hyperparameters are represented by shaded rectangles, while unobserved random variables and parameters are depicted by unshaded circles. Observed data is displayed within shaded circles. Larger unshaded rectangular plates are used to denote the range of indices for different types of data and parameters.

Algorithm 1: EM algorithm to classify text

```
Result: Maximize $p(\pi^{(t)}, \boldsymbol{\eta}^{(t)} \mid \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u, \boldsymbol{\alpha}, \boldsymbol{\beta})$
if In the first iteration of Active learning then
    Initialize $\pi$ and $\boldsymbol{\eta}$ by Naive Bayes;
        $\pi^{(0)} \leftarrow \mathrm{NB}(\mathbf{D}^l,\ Z^l, \boldsymbol{\alpha})$;
        $\boldsymbol{\eta}^{(0)} \leftarrow \mathrm{NB}(\mathbf{D}^l,\ \mathbf{Z}^l, \boldsymbol{\beta})$;
else
    Inherit $\pi^{(0)}$ and $\boldsymbol{\eta}^{(0)}$ from the previous iteration of Active learning;
end
while $p(\pi^{(t)}, \boldsymbol{\eta}^{(t)} \mid \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u, \boldsymbol{\alpha}, \boldsymbol{\beta})$ does not converge do
    (1) E step: obtain the probability of the class for unlabeled documents;
        $p(\mathbf{Z}^u \mid \pi^{(t)}, \boldsymbol{\eta}^{(t)}\mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u) \leftarrow$ E step($\mathbf{D}^u$, $\pi^{(t)}$, $\boldsymbol{\eta}^{(t)}$);
    (2) Combine the estimated classes for the unlabeled docs and the known classes
     for the labeled docs;
        $p(\mathbf{Z} \mid \pi^{(t)}, \boldsymbol{\eta}^{(t)}, \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u) \leftarrow$ combine($\mathbf{D}^l$, $\mathbf{D}^u$, $Z^l$, $p(Z^u \mid \pi^{(t)}, \boldsymbol{\eta}^{(t)}, \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u)$);
    (3) M step: Maximize $Q \equiv \mathbb{E}[p(\pi, \boldsymbol{\eta}, \mathbf{Z}^u \mid \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u, \boldsymbol{\alpha}, \boldsymbol{\beta})]$ w.r.t $\pi$ and $\boldsymbol{\eta}$;
        $\pi^{(t+1)} \leftarrow \mathrm{argmax}\ Q$;
        $\boldsymbol{\eta}^{(t+1)} \leftarrow \mathrm{argmax}\ Q$;
    (4) Check convergence: Obtain the value of $p(\pi^{(t+1)}, \boldsymbol{\eta}^{(t+1)} \mid \mathbf{D}^l, \mathbf{Z}^l, \mathbf{D}^u, \boldsymbol{\alpha}, \boldsymbol{\beta})$;
end
```

D Multiclass Classification

The model outlined in Section "The Method," assumes a binary outcome for the classification task. However, consider a scenario where researchers aim to categorize news articles into various classes such as political news, business news, entertainment news, sports news, and technology news. This section introduces a solution specifically designed for this multi-class classification task (where $K \geq 2$) and each document is assigned to one of K classes.

D.1 Model

Let K be the number of the classes and is equal to the number of classes to be classified, with $K \geq 2$. In other words, the model presented below is a generalization of the model presented in the main text. The multiclass classification model can be summarized by the following equations:

$$\begin{aligned} \boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) && (6) \\ Z_i &\overset{\text{i.i.d}}{\sim} \text{Categorical}(\boldsymbol{\pi}) && (7) \\ \boldsymbol{\eta}_{\cdot k} &\overset{\text{i.i.d}}{\sim} \text{Dirichlet}(\boldsymbol{\beta}_{\cdot k}), \quad k = \{1, \ldots, K\} && (8) \\ \mathbf{D}_{i\cdot} | Z_i = k &\overset{\text{i.i.d}}{\sim} \text{Multinomial}(n_i, \boldsymbol{\eta}_{\cdot k}) && (9) \end{aligned}$$

In this model setup, we consider a probability vector denoted by $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$, where each π_k represents the proportion of documents in the corpus belonging to class k. This vector $\boldsymbol{\pi}$ is sampled from a Dirichlet distribution with parameters defined by the K-dimensional vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_K)$. Then, for each document i, we generate its features from a multinomial distribution based on the total word count n_i and the vector $\boldsymbol{\eta}_{\cdot k}$, which represents the kth column of the $V \times K$ matrix $\boldsymbol{\eta}$. Here, each entry η_{vk} of $\boldsymbol{\eta}_{\cdot k}$ is the probability of observing word v given that the document belongs to class k. The prior distribution of $\boldsymbol{\eta}_{\cdot k}$ is specified by a Dirichlet distribution with hyperparameter vector $\boldsymbol{\beta}_{\cdot k}$ of length V. The word counts of document i are represented by the row vector $\mathbf{D}_{i\cdot}$ of length V. Given that document i belongs to class k (i.e., $Z_i = k$), $\mathbf{D}_{i\cdot}$ is drawn from a multinomial distribution with parameters n_i and $\boldsymbol{\eta}_{\cdot k}$.

In the case where document i is labeled, the sampling of Z_i is not based on a Bernoulli distribution as in the scenario with unlabeled data. Instead, its value is manually assigned. However, apart from this difference, the model structure remains the same for both labeled and unlabeled data. Notably, to give more weight to the information from labeled documents, we introduce a downweighting factor λ when incorporating information from unlabeled documents.

Figure D.1 graphically presents the multiclass mixture model. In this representation, hyperparameters are shown in shaded rectangles, random variables and parameters in unshaded circles, and observed data in shaded circles. The larger unshaded plates are used to indicate the scope of indices for various types of data and parameters.

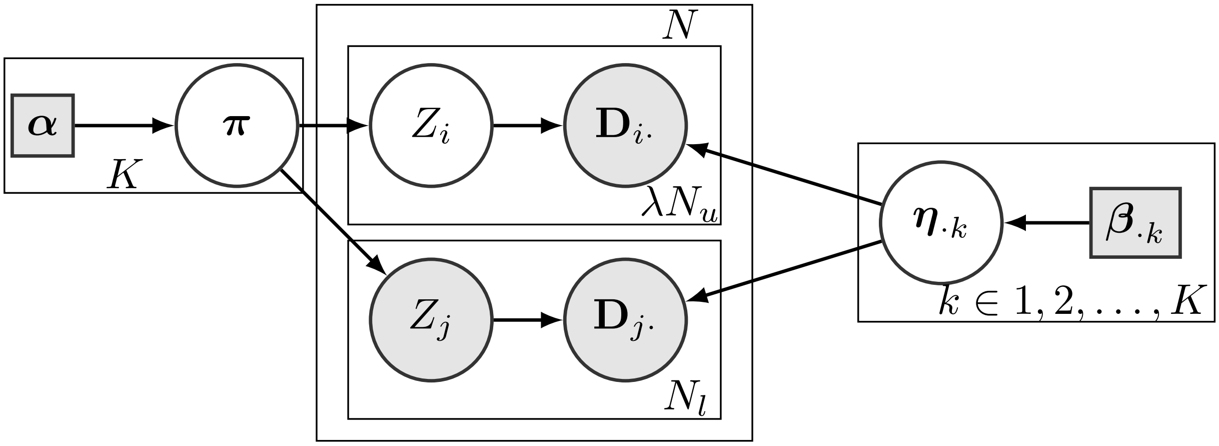

Figure D.1: **Graphical Representation of the Model for Multiclass Classification.** Hyperparameters are represented by shaded rectangles, while unobserved random variables and parameters are depicted by unshaded circles. Observed data is displayed within shaded circles. Larger unshaded rectangular plates are used to denote the range of indices for different types of data and parameters.

D.2 EM Algorithm

Equations (6) to (9) result in the following observed likelihood:

$$
\begin{aligned}
p(\boldsymbol{\pi},\boldsymbol{\eta}|\mathbf{D},\mathbf{C}^l) \propto{}& p(\boldsymbol{\pi})p(\boldsymbol{\eta})p(\mathbf{D},\mathbf{C}|\boldsymbol{\pi},\boldsymbol{\eta})\Big[p(\mathbf{D}^u|\boldsymbol{\pi},\boldsymbol{\eta})\Big]^{\lambda} \\
={}& p(\boldsymbol{\pi})p(\boldsymbol{\eta}) \times \prod_{k=1}^{K}\prod_{i=1}^{N^k} p(\mathbf{D}^l_{i\cdot}|Z_i = k, \boldsymbol{\eta}_{\cdot k})p(Z_i = k|\boldsymbol{\pi}) \\
&\times \left[\prod_{i=1}^{N^u}\sum_{k=1}^{K}\Big\{p(\mathbf{D}^u_{i\cdot}|Z_i = k, \boldsymbol{\eta}_{\cdot k})p(Z_i = k|\boldsymbol{\pi})\Big\}\right]^{\lambda} \\
\propto{}& \underbrace{\prod_{k=1}^{K}\left\{\pi_k^{\alpha_k - 1}\prod_{v=1}^{V}\eta_{vk}^{\beta_{vk}-1}\right\}}_{\text{prior}} \times \underbrace{\prod_{k=1}^{K}\prod_{i=1}^{N^k}\Big\{\prod_{v=1}^{V}\eta_{vk}^{D_{iv}} \times \pi_k\Big\}}_{\text{labeled doc. likelihood}} \times \underbrace{\left[\prod_{i=1}^{N^u}\sum_{k=1}^{K}\Big\{\prod_{v=1}^{V}\eta_{vk}^{D_{iv}} \times \pi_k\Big\}\right]^{\lambda}}_{\text{unlabeled doc. likelihood}}
\end{aligned}
\tag{10}
$$

The Q function (the expectation of the complete-data log-likelihood) is

$$
\begin{aligned}
Q &\equiv \mathbb{E}_{\mathbf{Z}|\boldsymbol{\pi}^{(t)},\boldsymbol{\eta}^{(t)},D,C}[\log p(\boldsymbol{\pi},\boldsymbol{\eta},\mathbf{Z}|\mathbf{D},\mathbf{C})] \\
&= \sum_{k=1}^{K}\left[(\alpha_k - 1)\log \pi_k^{(t)} + \sum_{v=1}^{V}\left\{(\beta_{vk}-1)\log \eta_{vk}^{(t)}\right\}\right] \\
&\quad + \sum_{k=1}^{K}\sum_{i=1}^{N^k}\left\{\sum_{v=1}^{V} D_{iv}\log \eta_{vk}^{(t)} + \log \pi_k^{(t)}\right\} \\
&\quad + \lambda\left[\sum_{i=1}^{N^u}\sum_{k=1}^{K} p_{ik}\left\{\sum_{v=1}^{V} D_{iv}\log \eta_{vk}^{(t)} + \log \pi_k^{(t)}\right\}\right]
\end{aligned}
\tag{11}
$$

For the unlabeled documents, the probability of $Z_i = k$, p_{ik}, is given by

$$
p_{ik} = \frac{\prod_{v=1}^{V}\eta_{vk}^{D_{iv}} \times \pi_k}{\sum_{k=1}^{K}\left[\prod_{v=1}^{V}\eta_{vk}^{D_{iv}} \times \pi_k\right]} \tag{12}
$$

In the M step, the updating equation for π is the following:

$$
\hat{\pi}_k \propto \alpha_k - 1 + N^k + \lambda \sum_{i=1}^{N^u} p_{ik} \tag{13}
$$

The updating equation for each η_{vk} is the following:

$$
\hat{\eta}_{vk} \propto (\beta_{vk} - 1) + \sum_{i=1}^{N^k} D_{iv} + \lambda \sum_{i=1}^{N^u} p_{ik} D_{iv} \tag{14}
$$

D.3 Results

To evaluate how well *activeText* performs in classifying out-of-sample in a multiclass scenario, we utilize two datasets: BBC news articles and Supreme Court Cases (see SI B for more details about these datasets). In the BBC dataset, the classes are “Politics,” “Entertainment,” “Business,” “Sports,” and “Technology.” Here, “Politics” represents 5% of the entire dataset, while the remaining 95% is evenly distributed among the other four classes. In the Supreme Court Cases dataset, the classes are “Criminal Procedure” (32.4% of the corpus), “Civil Rights” (21.4%), “Economic Activity” (22.2%), “Judicial Power” (15.4%), and “First Amendment” (8.6%).

We compare the performance of four models: 1) *activeText*, 2) a passive learning variant of *activeText*(Random Mixture), 3) Active Learning with Support Vector Machines (Active SVM), and 4) Passive Learning with Support Vector Machines (Random SVM). Our analysis, illustrated in Figure D.2, demonstrates that *activeText* outperforms the other models across both datasets, even with just 100 labeled documents. Not only does *activeText* exhibit superior classification performance in our validated datasets, but this result is confirmed in the reanalysis of Gohdes (2020) and Park et al. (2020) – both empirical applications involve multi-class classification.

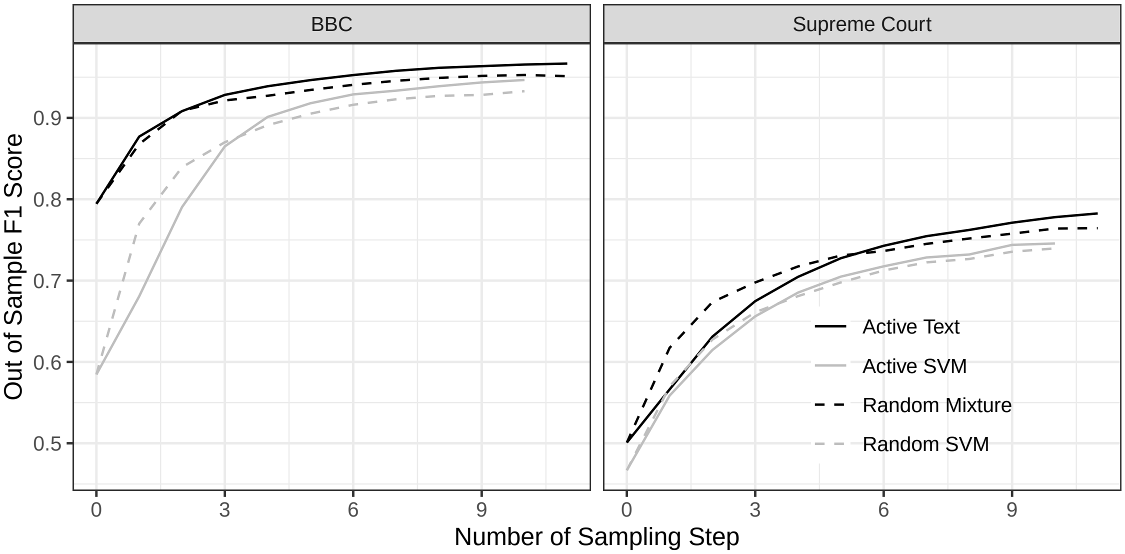

Figure D.2: **Multiclass Classification Results.**
The darker lines show the results with *activeText* and the lighter lines show the results with SVM. The solid lines use active sampling to decide the next set of documents to be labeled, and the dashed lines use random (passive) sampling. The y-axis indicates the out-of-sample F1 score and the x-axis show the number of sampling steps. The left column shows the results on BBC corpus, where the target classes are "Politics," "Entertainment," "Business," "Sports," and "Technology." "Politics" class has 5% of the total dataset, and the rest 95% is evenly split across the rest of classes. The right column shows the results on the Supreme Court corpus, where the target classes are "Criminal Procedure" (32.4% of the corpus), "Civil Rights" (21.4%), "Economic Activity" (22.2%), "Judicial Power" (15.4%), "First Amendment (8.6%)." In our model, we set the number of classes to be the same as the classification categories and linked each class to one classification category. *activeText* performs the best across the four specifications on both corpora.

E Binary Classification With Multiple Classes

The model outlined in the main text (see Section "The Method") assumes that there are two classes (a positive and negative class). However, this assumption can be relaxed to link multiple classes to the negative class. In the world of mixture models, the simplest setup is to let $K = 2$ since the classification goal is binary, and we can link each class to the final classification categories. A more general setup is to use $K > 2$ even when a goal is a binary classification. If $K > 2$, but our focus is to uncover the identity of one class, we can choose one of the class to be linked to the "positive" class and let all other classes be linked to the "negative" class (see e.g., Larsen and Rubin 2001 for a similar idea in the realm of record linkage). In other words, we collapse the $K - 1$ classes into one class for the classification purpose. Using $K > 2$ makes sense if the "negative" class consists of multiple sub-categories. For instance, suppose researchers are interested in classifying news articles into political news or not. Then, it is reasonable to assume that the non-political news category consists of multiple sub-categories, such as technology, entertainment, and sports news. Note that differently than in SI D, in this section, we impose hierarchies between classes and the end goal is binary classification of documents. However, as described below, the model structure is the same, with the collapsing of classes happens at the estimation stage.

E.1 Model

This section presents a model and inference algorithm when we use more than 2 classes in estimation but the final classification task is binary. In other words, we impose a hierarchy where many classes are collapsed into the negative class. In contrast, the positive class is made out of just one class. The model presented is as follows:

$$\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) && (15)\\
Z_i &\overset{\text{i.i.d}}{\sim} \text{Categorical}(\boldsymbol{\pi}) && (16)\\
\boldsymbol{\eta}_{\cdot k} &\overset{\text{i.i.d}}{\sim} \text{Dirichlet}(\boldsymbol{\beta}_{\cdot k}), \quad k = \{1, \dots, K\} && (17)\\
\mathbf{D}_{i\cdot} | Z_i = k &\overset{\text{i.i.d}}{\sim} \text{Multinomial}(n_i, \boldsymbol{\eta}_{\cdot k}) && (18)
\end{aligned}$$

It is important to note that if a document is labeled, then we have that the value assigned to Z_i is determined by hand-coding. The model structure is identical to the one presented in SI D. The difference, as described below is that the collapsing of the $K-1$ classes that make the negative class happens during the estimation stage. As before, note that we downweight the information from the unlabeled documents by λ, to utilize more information from labeled documents.

E.2 EM Algorithm

Let k^* be the class index linked to the positive class. Then, under the structure imposed by equations (15) to (18), the observed likelihood is the following:

$$
\begin{aligned}
&p(\boldsymbol{\pi}, \boldsymbol{\eta} | \mathbf{D}, \mathbf{C}^{lp}, \mathbf{C}^{ln}) \\
&\propto p(\boldsymbol{\pi}) p(\boldsymbol{\eta}) p(\mathbf{D}^{lp}, \mathbf{C}^{lp} | \boldsymbol{\pi}, \boldsymbol{\eta}) p(\mathbf{D}^{ln}, \mathbf{C}^{ln} | \boldsymbol{\pi}, \boldsymbol{\eta}) \Big[p(\mathbf{D}^{u} | \boldsymbol{\pi}, \boldsymbol{\eta}) \Big]^{\lambda} \\
&= p(\boldsymbol{\pi}) p(\boldsymbol{\eta}) \times \prod_{i=1}^{N^{lp}} p(\mathbf{D}^{lp}_{i\cdot} | Z_i = k^*, \boldsymbol{\eta}_{vk^*}) p(Z_i = k^* | \boldsymbol{\pi}) \\
&\quad \times \prod_{i=1}^{N^{ln}} \sum_{k \neq k^*} \Big\{ p(\mathbf{D}^{ln}_{i\cdot} | Z_i = k, \boldsymbol{\eta}_{v,k}) p(Z_i = k | \boldsymbol{\pi}) \Big\} \times \left[\prod_{i=1}^{N^u} \sum_{k=1}^{K} \Big\{ p(\mathbf{D}^{u}_{i\cdot} | Z_i = k, \boldsymbol{\eta}) p(Z_i = k | \boldsymbol{\pi}) \Big\} \right]^{\lambda} \\
&\propto \underbrace{\prod_{k=1}^{K} \left\{ \pi_k^{\alpha_k - 1} \prod_{v=1}^{V} \eta_{vk}^{\beta_{vk} - 1} \right\}}_{\text{prior}} \times \underbrace{\prod_{i=1}^{N^{lp}} \Big\{ \prod_{v=1}^{V} \eta_{vk^*}^{D_{iv}} \times \pi_k \Big\}}_{\text{positive labeled doc. likelihood}} \\
&\quad \times \underbrace{\prod_{i=1}^{N^{ln}} \sum_{k \neq k^*} \Big\{ \prod_{v=1}^{V} \eta_{vk}^{D_{iv}} \times \pi_k \Big\}}_{\text{negative labeled doc. likelihood}} \times \underbrace{\left[\prod_{i=1}^{N^u} \sum_{k=1}^{K} \Big\{ \prod_{v=1}^{V} \eta_{vk}^{D_{iv}} \times \pi_k \Big\} \right]^{\lambda}}_{\text{unlabeled doc. likelihood}}
\end{aligned}
\tag{19}
$$

The Q function (the expectation of the complete log likelihood) is:

$$
\begin{aligned}
Q &\equiv \mathbb{E}_{\mathbf{Z} | \boldsymbol{\pi}^{(t)}, \boldsymbol{\eta}^{(t)}, D, C} [\log p(\boldsymbol{\pi}, \boldsymbol{\eta}, \mathbf{Z} | \mathbf{D}, \mathbf{C})] \\
&= \sum_{k=1}^{K} \left[(\alpha_k - 1) \log \pi_k^{(t)} + \sum_{v=1}^{V} \Big\{ (\beta_{vk} - 1) \log \eta_{vk}^{(t)} \Big\} \right] \\
&\quad + \sum_{i=1}^{N^{lp}} \Big\{ \sum_{v=1}^{V} D_{iv} \log \eta_{vk^*}^{(t)} + \log \pi_{k^*}^{(t)} \Big\} + \sum_{i=1}^{N^{ln}} \sum_{k \neq k^*} p_{ik} \Big\{ \sum_{v=1}^{V} D_{iv} \log \eta_{vk}^{(t)} + \log \pi_k^{(t)} \Big\} \\
&\quad + \lambda \left[\sum_{i=1}^{N^u} \sum_{k=1}^{K} p_{ik} \Big\{ \sum_{v=1}^{V} D_{iv} \log \eta_{vk}^{(t)} + \log \pi_k^{(t)} \Big\} \right]
\end{aligned}
\tag{20}
$$

For the unlabeled documents, the probability of $Z_i = k$, p_{ik}, is:

$$p_{ik} = \frac{\prod_{v=1}^{V} \eta_{vk}^{D_{iv}} \times \pi_k}{\sum_{k=1}^{K} \left[\prod_{v=1}^{V} \eta_{vk}^{D_{iv}} \times \pi_k \right]} \tag{21}$$

In the M step, the updating equation for π_k is the following:

$$\hat{\pi}_k \propto \begin{cases} \alpha_k - 1 + \sum_{i=1}^{N^{ln}} p_{ik} + \lambda \sum_{i=1}^{N^u} p_{ik} & \text{if } k \neq k^* \\ \alpha_k - 1 + N^{lp} + \lambda \sum_{i=1}^{N^u} p_{ik^*} & \text{if } k = k^* \end{cases} \tag{22}$$

The updating equation for η_{vk} is the following.

$$\hat{\eta}_{vk} \propto \begin{cases} (\beta_{vk} - 1) + \sum_{i=1}^{N^{ln}} p_{ik} D_{iv} + \lambda \sum_{i=1}^{N^u} p_{ik} D_{iv} & \text{if } k \neq k^* \\ (\beta_{vk} - 1) + \sum_{i=1}^{N^{lp}} D_{iv} + \lambda \sum_{i=1}^{N^u} p_{ik^*} D_{iv} & \text{if } k = k^* \end{cases} \tag{23}$$

E.3 Results

In this section, we first evaluate the out-of-sample performance of the model for binary classification, with multiple classes. Then, we evaluate the combined performance of the multiple class approach for binary classification and keyword upweighting.

Figure E.1 shows the results of a model with just two classes vs. a model with 5 classes. In both instances, the final output of the classification task are two classes (positive vs. negative). The darker lines show the results with 5 classes and the lighter lines show the results with 2 classes. Overall, the model with 5 classes performs better or as well as the model with 2 classes. The gain from using 5 classes is the highest when the proportion of positive labels is small and when the size of labeled data is small.

Figure E.2 shows the results when the multiple class approach for binary classification and keyword upweighting approaches are combined. We find that the model with 5 classes yields performance comparable to or slightly better than using 2 classes. The most significant performance improvement occurs with the BBC corpus, comprising 5 news topic categories. Thus, our findings suggest that the classification performance of our method is improved when employing multiple classes, particularly when the number of classes aligns with the data generating process. Finally, regardless of the number of classes, our mixture models incorporating keywords demonstrate superior performance compared to models without keywords. The most notable improvement is observed in the human rights corpus, characterized by the smallest number of words per document.

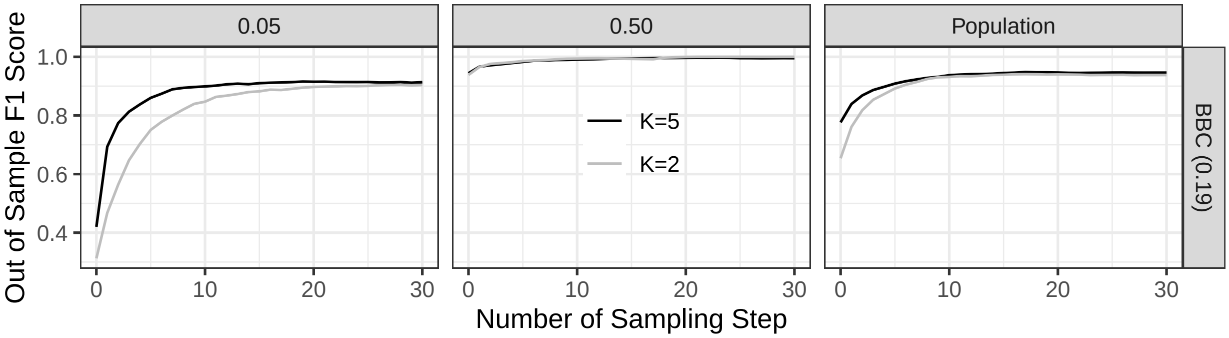

Figure E.1: **Binary Classification Results with 2 and 5 Classes.**
The darker lines show the results with 5 classes and the lighter lines show 2 classes. The columns correspond to various proportions of positive labels in the corpus. The y-axis indicates the out-of-sample F1 score and the x-axis show the number of sampling steps. For binary classification, using multiple classes improves the classification performance when the number of classes matches the data generating process.

F A LURE Approach to Remove The Bias of Active Learning

As described in Section "The Bias of Active Learning" in the main text, active learning introduces statistical biases in training. This problem arises because active learning selects non-i.i.d. data by choosing the most "informative" data points. Farquhar et al. (2021) propose a framework to address this bias, especially in scenarios where each observation has a chance of being actively sampled. Farquhar et al. (2021) proposal, the Levelled Unbiased Risk Estimator (LURE), addresses this bias by adjusting the empirical risk computation using weighted averages based on sampling probabilities of actively selected documents. More specifically, the LURE estimator is defined as follows: $\widetilde{R}_{\text{lure}} = \frac{1}{M}\sum_{m=1}^{M}[v_m L(Z_m - \widehat{Z}_m)]$, where L is a loss function (e.g., the L2 norm), and v_m is calculated as $1 + \frac{N-M}{N-m}\left(\frac{1}{(N-m+1)p(\text{index}_m|\text{index}_{1:m-1}, \mathbf{D}^u)} - 1\right)$. Here, $p(\text{index}_m|\text{index}_{1:m-1}, \mathbf{D}^u)$ represents the likelihood that the index associated with the mth document is actively sampled, given that the indices of $m-1$ documents have already been actively sampled and $\mathbf{D}^u$ represents the pool of unlabeled documents. In essence, the LURE estimator for the empirical risk computes a weighted average, where the weights are adjusted to account for the sampling probabilities of all actively selected documents.

In this section, we first describe how to incorporate such weights in training the model and recover model parameters that are not subject to these issues. Then, we provide addition details the simulation setting used to test the in-sample and out-of-sample performance of

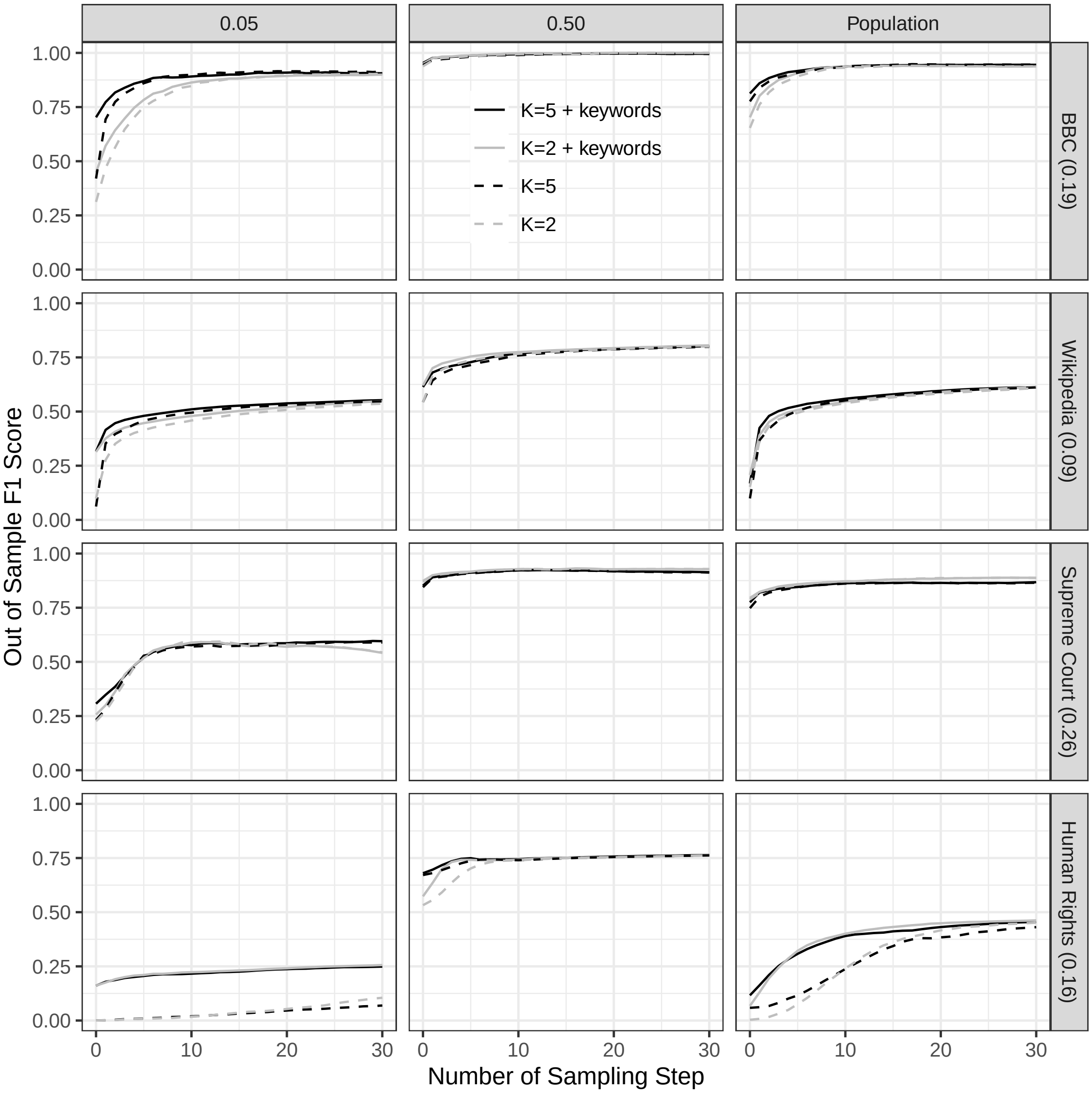

Figure E.2: **Binary Classification Results with Multiple Classes and Keywords.** The rows correspond to different datasets and the columns correspond to various proportions of positively labeled documents in the corpus. The y-axis indicates the out-of-sample F1 score and the x-axis show the number of sampling steps. The linetype show whether keywords are supplied: the solid lines show the results with keywords and the dashed lines without keywords. The colors show the number of classes in the mixture model: the darker lines show the results with 5 classes and the lighter lines with 2 classes. Using 5 classes leads to as good or slightly better performance than using 2 classes. The performance improvement is the largest with the BBC corpus, which consists of 5 news topic categories. Likewise, our mixture models with keywords leads to as good or better performance than the models without keywords. The improvement is the largest with the human rights corpus, where the number of words per document is the smallest.

*activeText*and *activeText*+LURE.

F.1 An *activeText* Model with (LURE) Weights

We introduce an extension to *activeText* that enables the use of weights. Let's revisit the binary classification problem, where the model structure aligns with the one described in Section "The Method" in the main text and in SI C. The key distinction here is that each labeled observation carries a predetermined weight w_i. Consequently, integrating the LURE weights entails expressing our observed-data quasi-likelihood as follows:

$$
\begin{aligned}
&p(\pi, \boldsymbol{\eta} | \mathbf{D}, \mathbf{C}^{lp}, \mathbf{C}^{ln}, w) \\
&\propto p(\pi)p(\boldsymbol{\eta})p(\mathbf{D}^{lp}, \mathbf{C}^{lp} | \pi, \boldsymbol{\eta}, w)p(\mathbf{D}^{ln}, \mathbf{C}^{ln} | \pi, \boldsymbol{\eta}, w)\Big[p(\mathbf{D}^{u} | \pi, \boldsymbol{\eta})\Big]^{\lambda} \\
&= p(\pi)p(\boldsymbol{\eta}) \times \prod_{i=1}^{N^{lp}} p(\mathbf{D}^{lp}_{i\cdot} | Z_i = 1, \boldsymbol{\eta}_{\cdot 1}, w_i)p(Z_i = 1 | \pi, w_i) \times \prod_{i=1}^{N^{ln}} \Big\{p(\mathbf{D}^{ln}_{i\cdot} | Z_i = 0, \boldsymbol{\eta}_{\cdot 0}, w_i)p(Z_i = 0 | \pi, w_i)\Big\} \\
&\quad \times \left[\prod_{i=1}^{N^{u}} \Big\{p(\mathbf{D}^{u}_{i\cdot} | Z_i = 1, \boldsymbol{\eta}_{\cdot 1})p(Z_i = 1 | \pi) + p(\mathbf{D}^{u}_{i\cdot} | Z_i = 0, \boldsymbol{\eta}_{\cdot 0})p(Z_i = 0 | \pi)\Big\}\right]^{\lambda} \\
&\propto \underbrace{\Big\{(1-\pi)^{\alpha_0 - 1} \prod_{v=1}^{V} \eta_{v0}^{\beta_{v0}-1}\Big\} \times \Big\{\pi^{\alpha_1 - 1} \prod_{v=1}^{V} \eta_{v1}^{\beta_{v1}-1}\Big\}}_{\text{prior}} \times \underbrace{\prod_{i=1}^{N^{lp}} \Big\{ \prod_{v=1}^{V} \eta_{v1}^{D_{iv} w_i} \times \pi^{w_i} \Big\}}_{\text{positive labeled doc. quasi-likelihood}} \\
&\quad \times \underbrace{\prod_{i=1}^{N^{ln}} \Big\{ \prod_{v=1}^{V} \eta_{v0}^{D_{iv} w_i} \times (1-\pi)^{w_i} \Big\}}_{\text{negative labeled doc. quasi-likelihood}} \times \underbrace{\left[\prod_{i=1}^{N^{u}} \Big\{ \prod_{v=1}^{V} \eta_{v0}^{D_{iv}} \times (1-\pi) \Big\} + \Big\{ \prod_{v=1}^{V} \eta_{v1}^{D_{iv}} \times \pi \Big\}\right]^{\lambda}}_{\text{unlabeled doc. likelihood}}
\end{aligned}
\tag{24}
$$

We estimate the parameters π and each η_{vk} using EM algorithm. In the E-step, we take the expectation of the log complete likelihood function (Q function),

$$
\begin{aligned}
Q &\equiv \mathbb{E}_{\mathbf{Z}|\pi^{(t)},\boldsymbol{\eta}^{(t)},\mathbf{D},\mathbf{C},w}[\log p(\pi,\boldsymbol{\eta},\mathbf{Z}|\mathbf{D},\mathbf{C},w)] \\
&= (\alpha_0-1)\log(1-\pi^{(t)}) + (\alpha_1-1)\log\pi^{(t)} + \sum_{v=1}^{V}\Big\{(\beta_{v0}-1)\log\eta_{v0}^{(t)} + (\beta_{v1}-1)\log\eta_{v1}^{(t)}\Big\} \\
&\quad + \sum_{i=1}^{N^{lp}}\Big\{\sum_{v=1}^{V} w_i\big(D_{iv}\log\eta_{v1}^{(t)} + \log\pi^{(t)}\big)\Big\} + \sum_{i=1}^{N^{ln}}\Big\{\sum_{v=1}^{V} w_i\big(D_{iv}\log\eta_{v0}^{(t)} + \log(1-\pi^{(t)})\big)\Big\} \\
&\quad + \lambda\left[\sum_{i=1}^{N^u} p_{i0}\Big\{\sum_{v=1}^{V} D_{iv}\log\eta_{v0}^{(t)} + \log(1-\pi^{(t)})\Big\} + p_{i1}\Big\{\sum_{v=1}^{V} D_{iv}\log\eta_{v1}^{(t)} + \log\pi^{(t)}\Big\}\right]
\end{aligned}
\tag{25}
$$

Again, p_{ik} is the probability of a document i being assigned to the kth cluster, $k = \{0, 1\}$, given data and the parameters at t th iteration. If a document has a positive label, $p_{i0} = 0$ and $p_{i1} = 1$. If a document has no label,

$$
\begin{aligned}
p_{i0} &= 1 - p_{i1} \\
p_{i1} &= \frac{\prod_{v=1}^{V}\eta_{v1}^{D_{iv}} \times \pi}{\prod_{v=1}^{V}\big\{\eta_{v0}^{D_{iv}} \times (1-\pi)\big\} + \prod_{v=1}^{V}\big\{\eta_{v1}^{D_{iv}} \times \pi\big\}}
\end{aligned}
\tag{26}
$$

In the M-step, we maximize the Q function, and obtain the updating equations for π and each η_{vk} for $k \in \{0, 1\}$. The updating equation for π is the following:

$$
\pi^{(t+1)} = \frac{\alpha_1 - 1 + \sum_{i=1}^{N^{lp}} w_i + \lambda\sum_{i=1}^{N^u} p_{i1}}{\Big(\alpha_1 - 1 + \sum_{i=1}^{N^{lp}} w_i + \lambda\sum_{i=1}^{N^u} p_{i1}\Big) + \Big(\alpha_0 - 1 + \sum_{i=1}^{N^{ln}} w_i + \lambda\sum_{i=1}^{N^u} p_{i0}\Big)}
\tag{27}
$$

The updating equation for each η_{vk} are as follows:

$$
\hat{\eta}_{v0}^{(t+1)} \propto (\beta_{v0}-1) + \sum_{i=1}^{N^{ln}} w_i D_{iv} + \lambda\sum_{i=1}^{N^u} p_{i0}D_{iv}, \quad v = 1,\ldots,V \tag{28}
$$

$$
\hat{\eta}_{v1}^{(t+1)} \propto (\beta_{v1}-1) + \sum_{i=1}^{N^{lp}} w_i D_{iv} + \lambda\sum_{i=1}^{N^u} p_{i1}D_{iv}, \quad v = 1,\ldots,V \tag{29}
$$

As before, we downweight the information from unlabeled document by λ, to utilize more reliable information from labeled documents.

F.2 Simulation Setup to Evaluate the Bias of Active Learning

We now describe in more detail the simulation setting used to produce Figure 7 in the main text.

The process of generating simulated data is based on the mixture model that we have defined above (See Section "The Method"). This allows us to systematically vary many dimensions of datasets and examine the performance of our classifier under different settings. We vary the following parameters:.

1. **The number of documents in the corpus** (N): We consider 1000 documents. 800 documents are used for training and 200 documents are used for out-of-sample testing.

2. **The size of vocabulary** (V): We consider two setups, 500 and 5000 words.

3. **The number of words per document**: (n_i): We generate n_i for $i = 1, \ldots, N$ from Poisson distribution with three different means (10, 50, and 100). This means that the average number of words per document is 10, 50, and 100, respectively.

4. **Proportion of documents in the positive class** (π): We consider two setups, 5% and 10%.

5. **Difficulty of classification** ($\tilde{\boldsymbol{\beta}}$): To control the difficulty of classification, we change the way the vector containing the probabilities of observing words for each class ($\boldsymbol{\eta}_{\cdot 0}$ and $\boldsymbol{\eta}_{\cdot 1}$ for negative and positive class, respectively) are generated from the original model. Note that simulating real corpora poses a challenge due to the presence of words that occur in both positive and negative classes, as well as words that are more prevalent in one class compared to the other. For instance, words like "election" and "democracy" are expected to appear frequently in documents labeled as "political" compared to those labeled as "non-political."

   To mimic this scenario, we first draw a "base" parameter vector of length V (the size of the vocabulary), denoted as $\tilde{\boldsymbol{\eta}}$, from a Dirichlet distribution with parameter vector $\tilde{\boldsymbol{\beta}} = (\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_V)$. We initially set $\boldsymbol{\eta}_{\cdot 0} = \boldsymbol{\eta}_{\cdot 1} = \tilde{\boldsymbol{\eta}}$. Subsequently, we exchange the highest values with those associated with the lowest values in the vector $\boldsymbol{\eta}_{\cdot 1}$ while leaving the vector $\boldsymbol{\eta}_{\cdot 0}$ unchanged. This results in $\boldsymbol{\eta}_{\cdot 1}$ and $\boldsymbol{\eta}_{\cdot 0}$ being nearly identical, except for the swapped elements.

   Consequently, the swapped words act as keywords since their values in $\boldsymbol{\eta}_{\cdot 1}$ are entirely distinct from their values in $\boldsymbol{\eta}_{\cdot 0}$. This allows us to control the level of informativeness of the keywords when manipulating the values in $\tilde{\boldsymbol{\beta}}$. For instance, lower values in

$\tilde{\boldsymbol{\beta}}$ concentrate the values in $\tilde{\boldsymbol{\eta}}$ towards the edges of the probability space, thereby making the keywords more informative. Conversely, higher values in $\tilde{\boldsymbol{\beta}}$ distribute the values in $\tilde{\boldsymbol{\eta}}$ more evenly, resulting in less informative keywords and a more challenging classification task. We have two parameters to control for the difficulty of classification: The number of keywords and the hyperparameter vector $\tilde{\boldsymbol{\beta}}$. First, for the number of keywords, or the number of words swapped, we consider three setups: 1, 5, and 20. Second, for the hyperparameter vector $\tilde{\boldsymbol{\beta}}$, we investigate three different values: 0.1, 0.5, and 0.9. As $\tilde{\boldsymbol{\beta}}$ is a vector with a length of V, for each simulation setting we fix all elements of $\tilde{\boldsymbol{\beta}}$ to the same value e.g., $\forall v, \tilde{\beta}_v = 0.1$. Figure F.1 illustrates how the values in the word-class matrix $\boldsymbol{\eta} = [\boldsymbol{\eta}_{\cdot 0}; \boldsymbol{\eta}_{\cdot 1}]$ are generated under the different values of $\tilde{\boldsymbol{\beta}}$. As the figure reveals, smaller (larger) values for $\tilde{\boldsymbol{\beta}}$ result in more (less) informative keywords, and, as a result, easier (more difficult) classification tasks.

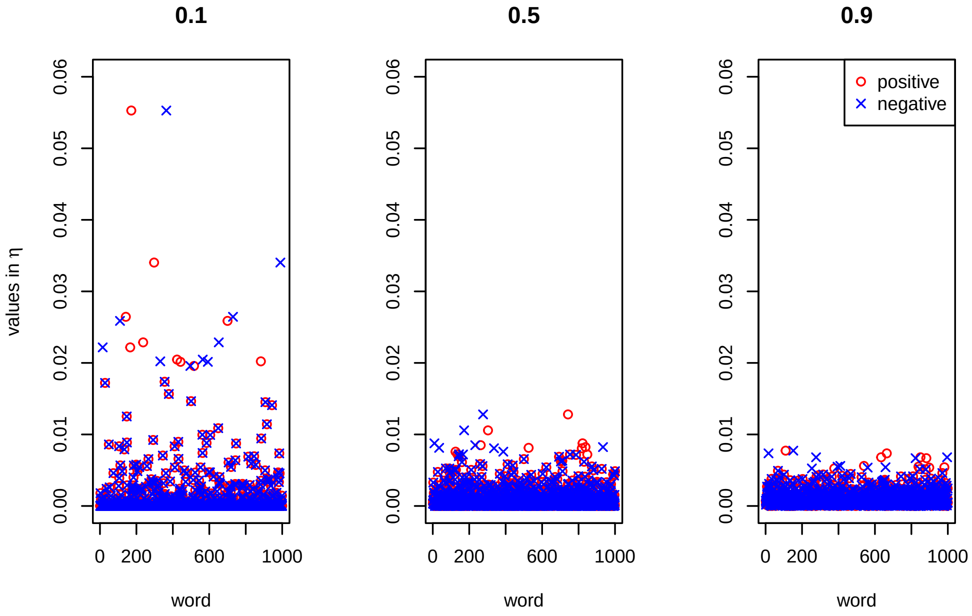

Figure F.1: **Values in the Word-class Matrix ($\boldsymbol{\eta}$) across Three Different Simulation Setups by $\tilde{\boldsymbol{\beta}}$.**
The figure displays the word indices on the horizontal axis and the corresponding $\boldsymbol{\eta}$ values for each word on the vertical axis. For each v, three values of $\tilde{\boldsymbol{\beta}}$ are used: 0.1 (left), 0.5 (middle), and 0.9 (right). In the graph, circles represent values associated with the positive class, while crosses represent values associated with the negative class. The vocabulary consists of 1000 unique words, and we swap the 10 highest values in $\boldsymbol{\eta}_{\cdot 1}$ with the 10 lowest values in $\boldsymbol{\eta}_{\cdot 1}$. From the figure, it is evident that smaller (larger) values of $\tilde{\boldsymbol{\beta}}$ lead to more (less) informative keywords. Consequently, the classification tasks become easier (more difficult).

In our study, we examine a total of 108 distinct simulation setups, obtained by combining different values for five parameters: $V = \{500, 5000\}$, $n_i = \{10, 50, 100\}$, $\pi = \{0.05, 0.1\}$, the number of swapped words $= \{1, 5, 20\}$, and $\tilde{\boldsymbol{\beta}} = \{0.1, 0.5, 0.9\}$. For each setup, we generate 100 datasets.[15] We then train the model using *activeText* and *activeText* with LURE weights. Because the LURE weights can only be computed when all documents have non-zero probability of being labeled, we slightly modified our sampling scheme using the epsilon-greedy algorithm as in Farquhar et al. (2021). In short, we select the document with the highest probability of being labeled with probability $1 - \epsilon$ and select a document at random with probability ϵ. We set $\epsilon = 0.1$. This ensures that the modified algorithm is almost identical to the entropy-based sampling scheme we used throughout the paper, but allows us to compute the LURE weights.

We report two quantities: the in-sample bias of the empirical risk and the out-of-sample average classification metrics, following the approach of Farquhar et al. (2021). For the in-sample bias, we first obtain a sequence of labeled document selected according to our active learning algorithm. After obtaining the sequence of labeled documents, we calculate the empirical risk at each active step both with and without LURE weights. The empirical risk is evaluated at a fixed parameter that do not depend on the training data. This ensures that the bias is only due to the fact that more difficult documents are labeled more likely. We then calculate the bias as the difference between the population risk and the empirical risk with and without LURE weights, respectively.[16] For the out-of-sample average classification metrics, we calculate the average F1 score for each simulation setup using the model trained with and without LURE weights.

Figure F.2 presents additional results from the simulation study aside from Figure 7 in the main text. While Figure 7 in the main text presents the results for the easy classification setting, Figure F.2 presents the results for the difficult classification setting. Specifically, while the dataset used in Figure 7 in the main text swap 5 keywords, the dataset used in Figure F.2 swap only 1 keyword. Other simulation settings are the same as the ones used in Figure 7 in the main text. Similar to Figure 7 in the main text, the results show that the out-of-sample F1 score is higher without LURE weights.

Finally, Figure F.3 illustrates the results of applying the LURE bias correction to *activeText* for editors involved in our validation dataset using toxic comment discussions on Wikipedia's internal forum. We divided the data into 80% for training and 20% for testing, conducting 100 Monte Carlo simulations. The left panel displays the in-sample bias of $\widetilde{R}$ and $\widetilde{R}_{lure}$, while the right panel shows their respective out-of-sample F1 scores. The figure confirms our simulation-based findings. Initially, $\widetilde{R}$ shows a positive bias during the labeling

[15] Please refer to Supplementary Appendix for the Simulation Studies for the full set of results.

[16] This is the same approach as Figure 2 in Farquhar et al. (2021)

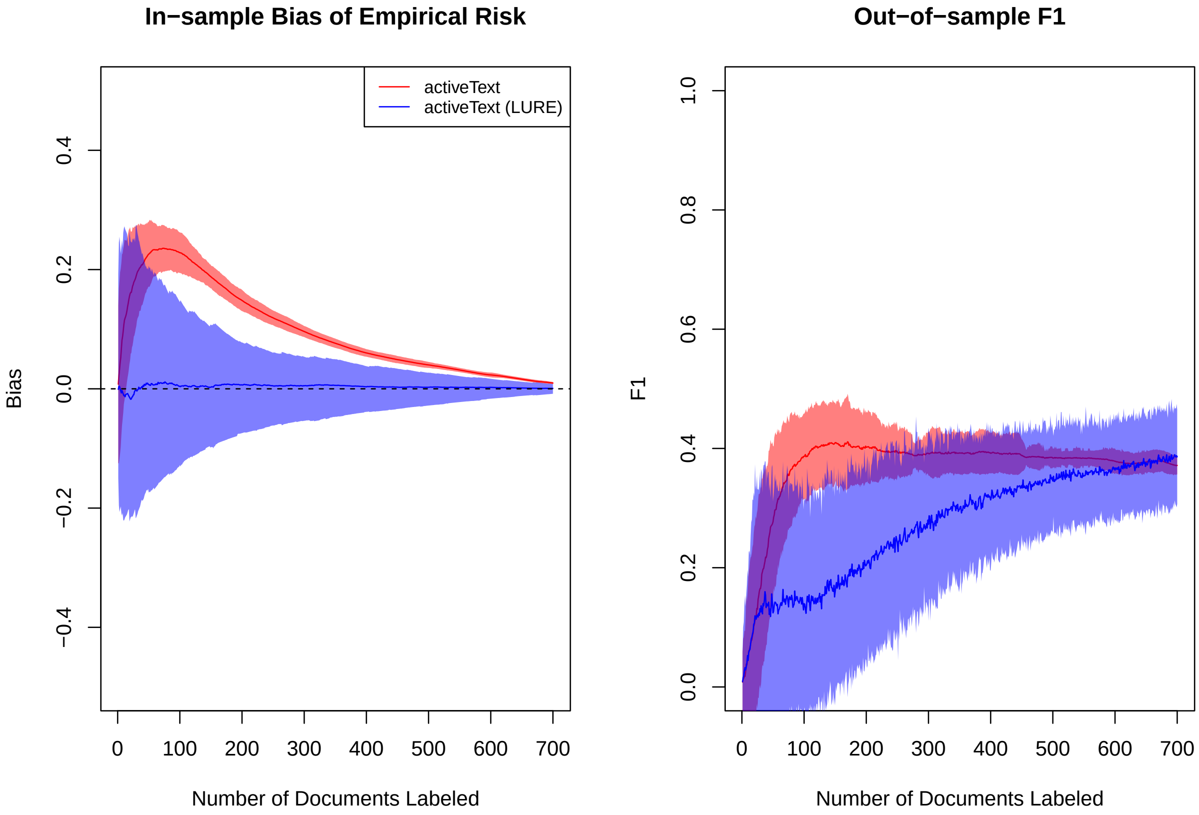

Figure F.2: Bias of the Empirical Risk for Labeled Data (left panel) and Out-of-sample Classification Performance (right panel) of *activeText*, *activeText*+LURE. For each panel, the x-axis represents the number of documents labeled, and the y-axis represents the average bias and average out-of-sample F1 score across 100 Monte Carlo simulations. Shaded areas represent the 95% confidence intervals across Monte Carlo simulations.

process, which LURE effectively eliminates. However, the unadjusted *activeText* outperforms its bias-corrected version in out-of-sample classification, as depicted in the right panel of Figure F.3.Thus, our validation results suggest that although correcting for in-sample bias improves the performance within the sample, it does not necessarily translate to better out-of-sample classification for *activeText*.

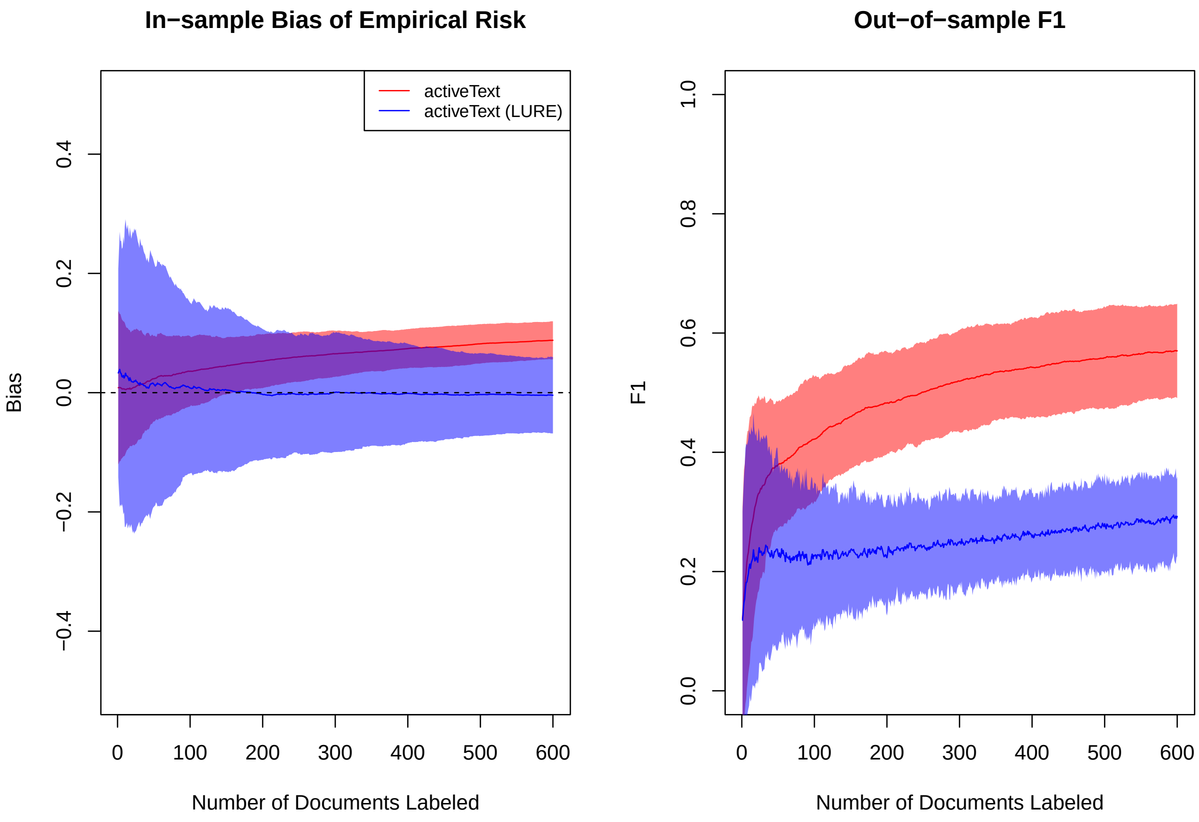

Figure F.3: Wikipedia Toxic Comments: Bias of the Empirical Risk for Labeled Data (left panel) and Out-of-sample Classification Performance (right panel) of *activeText*, *activeText*+LURE. The reference class is Toxic comment (9% of all documents). For each panel, the x-axis represents the number of documents labeled, and the y-axis represents the average bias and average out-of-sample F1 score across 100 Monte Carlo simulations. Shaded areas represent the 95% confidence intervals across Monte Carlo simulations.

G Classification Performance with Mislabels

In this section, we assess the impact of errors introduced during 1) manual document labeling and 2) manual labeling of keywords. To ensure a realistic evaluation, we utilize data derived from our validation studies for binary classification (refer to Section "Validation Performance" in the main text and SI B for data details). The datasets include: internal forum conversations of Wikipedia editors (class of interest: toxic comment), BBC News articles (political topic), the United States Supreme Court decisions (criminal procedure), and Human Rights allegations (physical integrity rights allegation). Again, we use 80% of each dataset for training and and 20% for evaluation. Documents to be labeled are sampled from the training dataset, and documents to test are not included in training. The out-of-sample F1 score is calculated to evaluate model performance. We use $\lambda = 0.001$ to downweight information from unlabeled documents.

G.1 Mislabeled Documents

To introduce label noise, we systematically vary the percentage of incorrectly labeled documents. Specifically, for each dataset and at every active iteration (20 labeled documents per iteration), we randomly select θ% of documents chosen by our algorithm for labeling and switch their labels. For example, if a document's true label is politics, we relabel it as non-political, and vice versa. We explore different levels of 'honest' (random) mistakes represented by θ, with values ranging from $\{0, 0.10, 0.20, 0.30, 0.40, 0.50\}$. Note that this process of adding noise uniformly at random, is often referred to as classical measurement error.

As shown in Figure G.1, the classification performance of *activeText* for binary classification, diminishes with an increasing proportion of mislabels. For example, considering the BBC News articles dataset, the out-of-sample F1 score remains high (approximately 0.90) after labeling around 200 documents with no measurement error (θ equal to 0). However, the F1 score decreases to 0.87 when introducing a small amount of measurement error with θ set to 0.10. Notably, the F1 score experiences a significant drop when θ exceeds 0.20. This observed trend holds consistently across all datasets.

What is the effect of labeling a document incorrectly on downstream analyses? For instance, let's say we want to predict how many documents fall into a specific category, like politics. In our validation studies, we already know the true proportions of documents in the classes we are interested in. For example, in the BBC dataset, we find that 19% of the articles are about politics, while in the Wikipedia corpus, 9% of the documents are toxic. Similarly, 26% of Supreme Court cases deal with criminal procedure, and 16% of Human

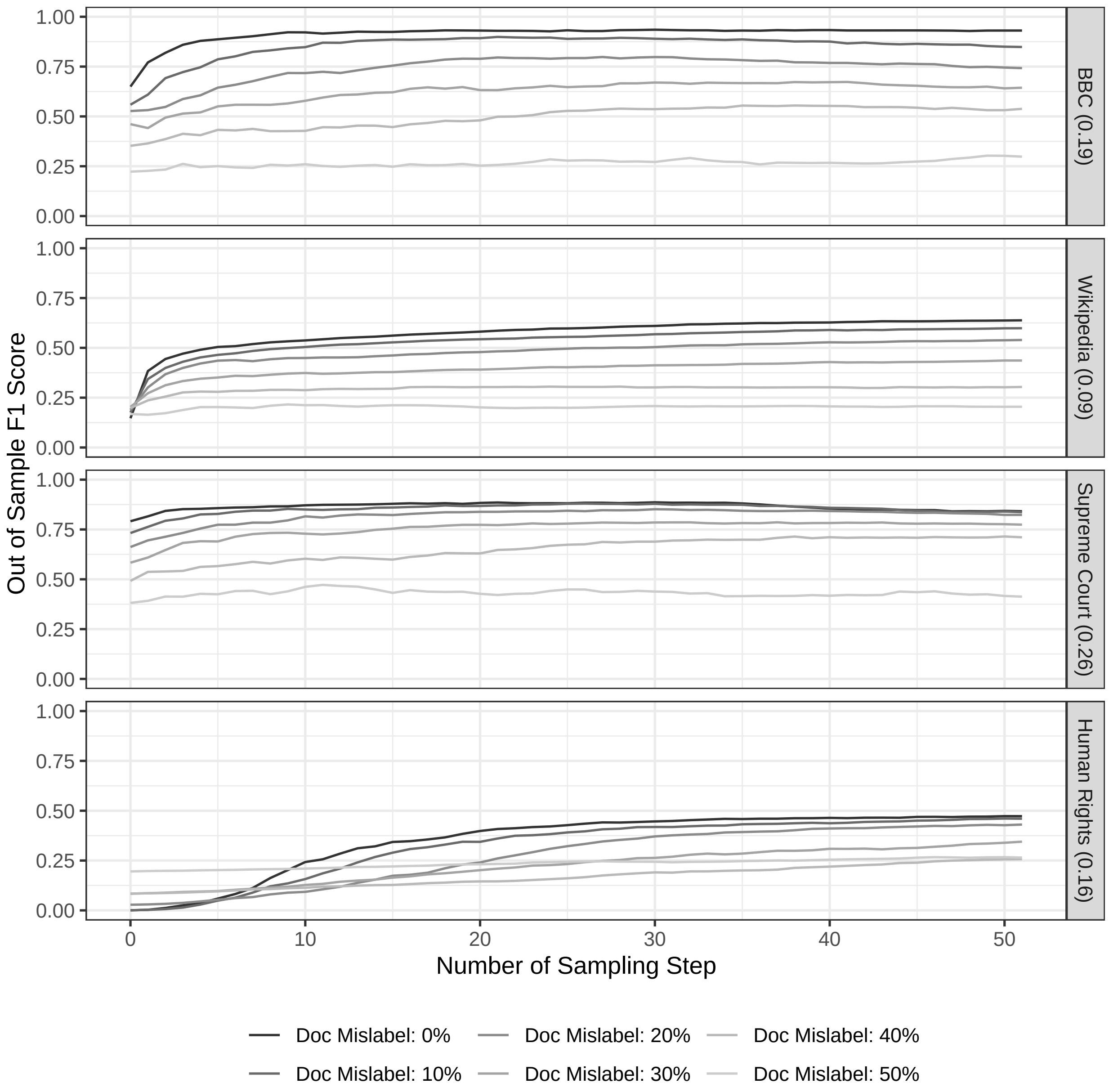

Figure G.1: **Classification Results with Mislabels in Active Document Labeling** The rows correspond to different datasets. The y-axis indicates the out-of-sample F1 score and the x-axis shows the number of sampling steps. 20 documents are labeled at each sampling step. The colors correspond to different levels of mislabels in the labeling of documents. We find that as the proportion of mislabels increases, the classification performance of *activeText* decreases.

Rights reports involve allegations of physical integrity rights violations. To understand the impact of mislabeling, we calculate the absolute value of the bias in the prediction for the proportion of documents in the reference class made by *activeText*. In other words, the out-of-sample magnitude of the bias in the proportion of documents in the class of interest is calculate as: $\left|\pi - \frac{\sum_{i=1}^{N} p_{i1}}{N}\right|$.

As shown in Figure G.2, the magnitude of the bias increases as the proportion of mislabels rises. For example, in the Human Rights allegations dataset, the bias is small if we label around 200 documents accurately (with no measurement error, θ equal to 0). However, introducing classical measurement error with $\theta > 0.30$ leads to an increase in bias by 0.25 units. This trend holds consistently across all datasets, similar to what we observe with the F1 score. In other words, even when there are unintentional (or "random") errors present, the performance of *activeText* is reduced when calculating a simple summary statistic such as a sample mean.

If our predictions are not accurate, using them either as an outcome or as an explanatory variable in popular downstream tasks (such a regression analysis) could introduce bias in our estimates. The extent of this bias varies. In the best-case scenario, in a regression framework, random errors in explanatory variables result in attenuation bias. However, any departure from classical measurement error could lead to bias whose direction is difficult to determine. This underscores the importance of exercising caution when assessing the limitations of methods based on active learning. Nevertheless, refer to Egami et al. (2023) for an apporach on how to adjust quantities of interest in situations where prediction outcomes are used as outcomes in regression.

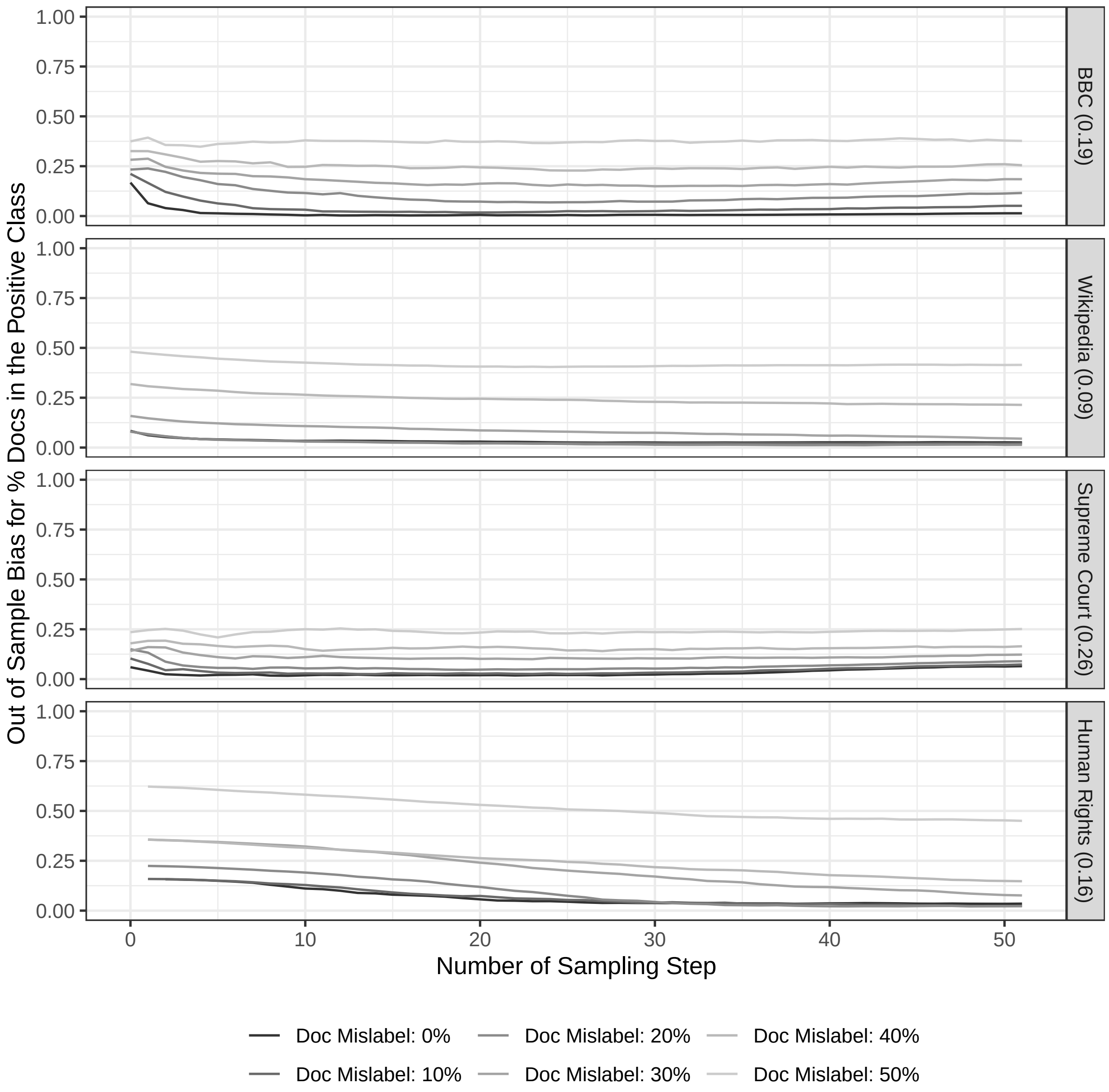

Figure G.2: **Bias in the Proportion of Documents in the Reference Class as a result of Mislabels in Active Document Labeling**

The rows correspond to different datasets. The y-axis indicates the absolute value of the out-of-sample bias and the x-axis shows the number of sampling steps. 20 documents are labeled at each sampling step. The colors correspond to different levels of mislabels in the labeling of documents. We find that as the proportion of mislabels increases, the magnitude of the bias increases as well.

G.2 Mislabeled Keywords

We now evaluate the impact of errors that occur during the manual labeling of keywords. First, we need a definition of a "true" keyword. To do so, as we discussed in Section "Benefits of Keyword Upweighting" in the main text, we use all documents and their corresponding labels to obtain the empirical distributions of observing word v in each class. Then, using this information, a keyword v is considered a true keyword for the positive class if the ratio $\eta_{v,1}/\eta_{v,0}$ exceeds the 90th percentile. Remember, for each η_{vk}, the positive class is denoted by $k = 1$ and the negative class by $k = 0$. Similarly, a true negative keyword is defined based on the ratio $\eta_{v,0}/\eta_{v,1}$ and based on the same percentile cutpoint.

Once we have defined the true keywords for each class, the model settings for training and testing are the same ones used are the same as those described in the previous subsection. At each iteration of *activeText*, 10 candidate keywords are suggested. Noise in labeling keywords is introduced by selecting uniformly at random $p\%$ of the candidate keywords to be mislabeled. In other words, the human introduces an honest (random) mistake in labeling with a probability of $p \in \{0, 0.10, 0.20, 0.30\}$. Specifically, if a candidate keyword v is a true keyword, the human will not label v as a keyword. Conversely, if a candidate keyword v is not a true keyword, the human will label v as a keyword. We also vary the value of γ (the parameter representing the upweight for the keywords) to take two values, 10 (small upweight) and 100 (large upweight).

Figure G.3 presents the results. In contrast to the case where noise is added to the labels of the actively sampled documents, the classification performance of *activeText* for binary classification slightly diminishes with an increasing proportion of mislabels for the keywords (30% or more). This is result is consistent across datasets and values of γ (the weight given to each keyword).

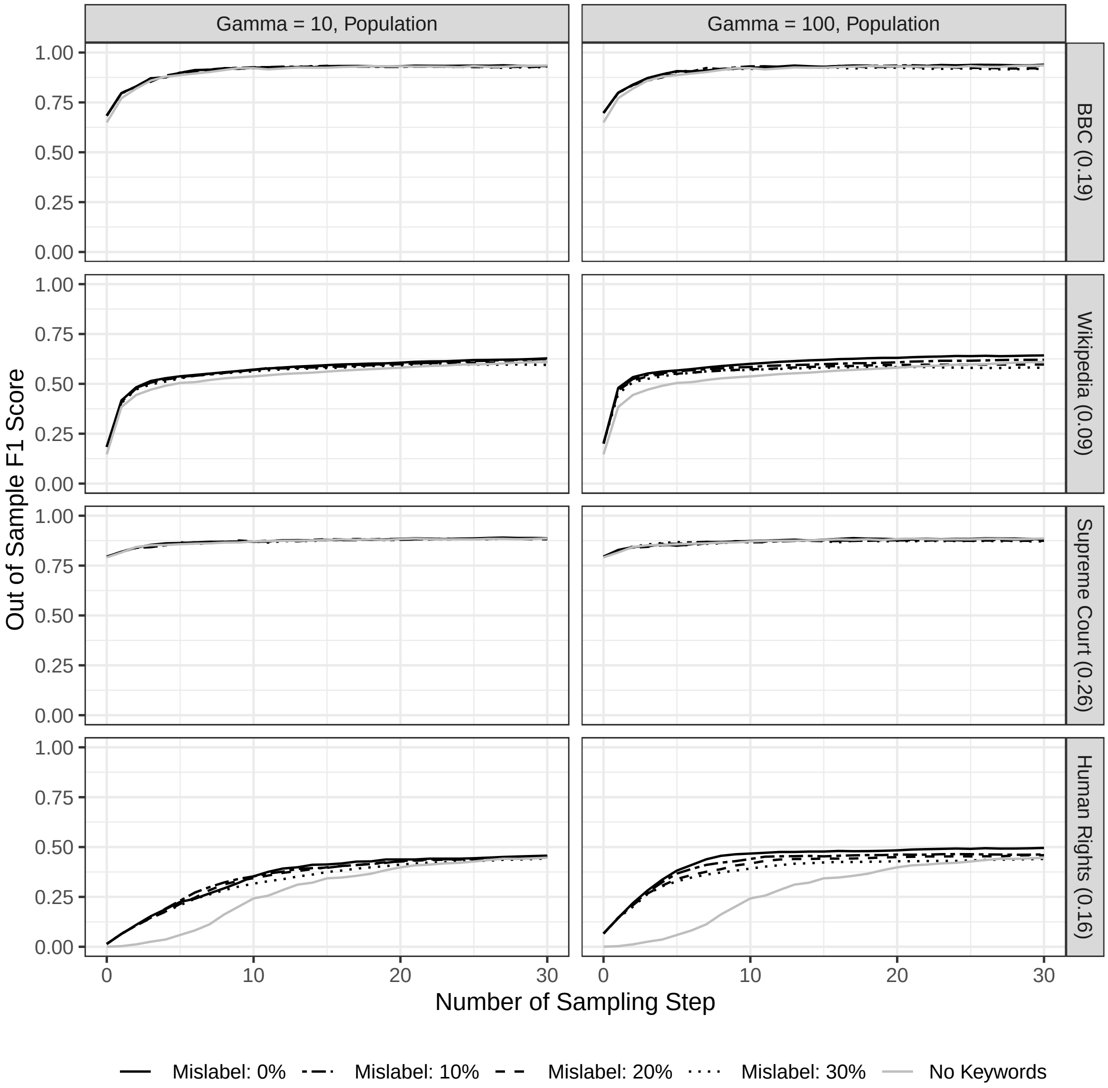

Figure G.3: **Classification Results with Mislabels in Active Keywords**
The rows correspond to different datasets and the columns correspond to two values of γ, which controls the degree of keyword upweighting. The y-axis indicates the out-of-sample F1 score and the x-axis shows the number of sampling steps. The lines correspond to different levels of mislabels at the keyword labeling.

References

Bishop, C. M., and Lassarre, J. (2007), "Generative or Discriminative? Getting the Best of Both Worlds," *Bayesian Statistics*, 8, 3–24.

Cordell, R., Clay, C., Fariss, C., Wood, R., and Wright, T. (2021), "Recording repression: Identifying physical integrity rights allegations in annual country human rights reports," *International Studies Quarterly*, .

Dempster, A., Laird, N., and Rubin, D. (1977), "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1–22.

Denny, M. J., and Spirling, A. (2018), "Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it," *Political Analysis*, 26(2), 168–189.

Egami, N., Hinck, M., Stewart, B., and Wei, H. (2023), Using Imperfect Surrogates for Downstream Inference: Design-based Supervised Learning for Social Science Applications of Large Language Models,, in *37th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1–13.

Farquhar, S., Gal, Y., and Rainforth, T. (2021), On Statistical Bias In Active Learning: How and When to Fix It,, in *International Conference on Learning Representations.*
URL: *https://openreview.net/forum?id=JiYq3eqTKY*

Gohdes, A. (2020), "Repression technology: Internet accessibility and state violence," *American Journal of Political Science*, 64(3), 488–503.

Greene, D., and Cunningham, P. (2006), Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering,, in *Proc. 23rd International Conference on Machine learning (ICML'06)*, ACM Press, pp. 377–384.

Grimmer, J., Roberts, M., and Stewart, B. (2022), *Text as data: A New Framework for Machine Learning and the Social Sciences* Princeton University Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning*, Springer Series in Statistics, New York, NY, USA: Springer New York Inc.

Knox, D., Lucas, C., and Cho, W. K. T. (2022), "Testing Causal Theories with Learned Proxies," *Annual Review of Political Science*, 25(1), 419–441.

Larsen, M. D., and Rubin, D. B. (2001), "Iterative Automated Record Linkage Using Mixture Models.," *Journal of the American Statistical Association.*, 96(453), 32–41.

Miller, B., Linder, F., and Mebane, W. R. (2020), "Active Learning Approaches for Labeling Text: Review and Assessment of the Performance of Active Learning Approaches," *Political Analysis*, pp. 1–20.

Miller, D., and Uyar, H. (1996), A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabelled Data,, in *Advances in Neural Information Processing Systems*, eds. M. Mozer, M. Jordan, and T. Petsche, Vol. 9, MIT Press.

Ng, A., and Jordan, M. (2001), "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes," *Advances in neural information processing systems*, 14.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000), "Text classification from labeled and unlabeled documents using EM," *Machine learning*, 39(2-3), 103–134.

Park, B., Greene, K., and Colaresi, M. (2020), "Human Rights are (Increasingly) Plural: Learning the Changing Taxonomy of Human Rights from Large-scale Text Reveals Information Effects," *American Political Science Review*, 114(3), 888–910.

Pennington, J., Socher, R., and Manning, C. D. (2014), Glove: Global vectors for word representation,, in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Rodriguez, P. L., and Spirling, A. (2022), "Word embeddings: What works, what doesn't, and how to tell the difference for applied research," *The Journal of Politics*, 84(1), 101–115.