Batch-Mode Active Learning for Technology-Assisted Review*

Tanay Kumar Saha, Mohammad Al Hasan
Department of Computer Science
Indiana University Purdue University Indianapolis, IN
Email: {tksaha, alhasan}@cs.iupui.edu

Chandler Burgess, Md Ahsan Habib, Jeff Johnson
iControl$^{ESI®}$, 16479 N. Dallas Parkway
Addison, TX, 75001
Email: {cburgess, mhabib, jjohnson}@icontrolesi.com

***Abstract*—In recent years, technology-assisted review (TAR) has become an increasingly important component of the document review process in litigation discovery. This is fueled largely by dramatic growth in data volumes that may be associated with many matters and investigations. Potential review populations frequently exceed several hundred thousands documents, and document counts in the millions are not uncommon. Budgetary and/or time constraints often make a once traditional linear review of these populations impractical, if not impossible—which made "predictive coding" the most discussed TAR approach in recent years. A key challenge in any predictive coding approach is striking the appropriate balance in training the system. The goal is to minimize the time that Subject Matter Experts spend in training the system, while making sure that they perform enough training to achieve acceptable classification performance over the entire review population. Recent research demonstrates that Support Vector Machines (SVM) perform very well in finding a compact, yet effective, training dataset in an iterative fashion using batch-mode active learning. However, this research is limited. Additionally, these efforts have not led to a principled approach for determining the stabilization of the active learning process. In this paper, we propose and compare several batch-mode active learning methods which are integrated within SVM learning algorithm. We also propose methods for determining the stabilization of the active learning method. Experimental results on a set of large-scale, real-life legal document collections validate the superiority of our method over the existing methods for this task.**

I. Introduction

The sheer size of electronically stored documents, and the cost, in money and time, of their review in connection with litigation and regulatory proceedings drive the need for technology-assisted review (TAR) and the development of "predictive coding" software. In a traditional linear review, an attorney who is expert in the subject matter trains a group of contract attorneys or junior associates so that they can churn through the documents for the weeks or months that it may take to complete the review. This process is lengthy and inefficient because a significant portion (generally a majority) of the attorneys' time is spent reviewing non-relevant documents. The objective of predictive coding is to design a machine-learning based system that labels documents as relevant or non-relevant to a specific issue or issues, and hence, minimizes the review-cost and time by maximizing the focus on the relevant documents. The system still requires expert human review, but it significantly reduces the time (and money) required to complete the review process. Initially, predictive coding software and processes were met with reluctance and suspicion around their accuracy, result reproducibility, and defensibility. In recent years, the courts have become more supportive of predictive coding and often advocate for its use in the legal discovery processes. In one specific and frequently referenced case, Global Aerospace, Inc. vs. Landow Aviation, L.P., the court agreed with the defendant that a predictive coding methodology was appropriate, even though that methodology was estimated to achieve 75% recall [1]. For years to come, predictive coding, and TAR in general, will be a mainstay in litigation-driven document review.

By definition, the document review task is akin to a supervised classification task in machine learning—given a huge collection of documents, the primary objective of predictive coding is to use a sub-collection of human-labeled documents to build a classification model to discriminate the remaining documents in the collection as relevant (also known as, "responsive" in the legal domain) or non-relevant (non-responsive). However, another key objective of predictive coding is to maximize the discovered relevant documents as measured by recall while minimizing the human labeling efforts as measured by the number of documents which are labeled by the attorneys. A principled machine learning approach for fulfilling this objective is *active learning*, in which the classification model is updated through an iterative process—the prevailing learning model of an iteration is utilized for selecting an unlabeled training instance to query its label, and then the selected instance becomes part of the training set of the subsequent iterations [2]–[4]. A more practical variant of active learning is *batch-mode active learning* [5], which extends the training set by adding a fixed-size batch of training instances instead of only one instance in each iteration. Numerous research articles in the machine-learning domain have shown that, for achieving a given performance, an active learning approach requires a much smaller number of training instances in comparison to its non-active variants, resulting in reduced labeling effort. This is the reason batch-mode active learning is becoming the most popular learning paradigm for the document review task in the litigation discovery domain.

A significant challenge in predictive coding is the selec-

*Mohammad Hasan's research is supported by NSF CAREER Award: IIS-1149851

tion of a learning algorithm which works well for highly imbalanced distributions of responsive and non-responsive documents in a document collection. This is important because in real-world document review populations, the number of responsive documents is typically a very small percentage (typically between 1 and 10%) of the total documents in the collection. Recent publications on document review [6] report SVM as a robust learning algorithm that works well for highly imbalanced legal datasets. It is also one of the best learning algorithms for large-scale text categorization. Additionally, SVM provides easily computable metrics which can be used for choosing a new batch of training instances in a batch-mode active learning setup. However, in existing machine learning literature, batch-mode active learning using SVM has not received much attention. Also, large-scale studies of the performance of batch-mode active learning on real-life high-dimensional legal data is not currently available.

Another challenge which applies to active learning-based document review is in identifying whether the learning model has stabilized such that no further training is necessary. This is important from the standpoint of real-world usability of predictive coding. A premature learning model almost certainly requires the inclusion of an overly inflated portion of the document population in order to achieve an acceptable level of recall of relevant documents. On the other hand, excessive training beyond stabilization wastes attorney time during the training phase. So, a model stabilization criterion is required for signaling the potential termination of the learning stage. To the best of our knowledge, none of the existing publications on batch-mode active learning have studied the stabilization behavior of the models on real-world litigation discovery data. Finally, the computation of labeling effort of active learning in litigation-driven predictive coding is different than that of other domains, which makes it challenging to use existing active learning methodologies for predictive coding projects. For instance, in many (but not all) real-world document review projects, the attorney review team conducts a "second pass" review, after the calibration and application of the model to the entire population. During this additional review, the attorneys examine each of the "predicted responsive" documents, prior to any production of those documents to a receiving party. Again, none of the existing publications on active learning present the model efficiency considering the complex dual phase labeling effort of the users.

In this paper, we propose two novel methods for batch-mode active learning using SVM. The novelty of the proposed methods is manifested in the way they choose the new batch of unlabeled instances for extending the prevailing training dataset. We compare the performance of the proposed methods with the best of the existing methods by implementing them in a commercial system which is deployed in iControl$^{ESI®}$'s e-discovery platform, Recenseo®. For comparison, we use multiple real-world large case datasets, which fall within the category of big data in any reasonable categorization. Our experiments over the deployed active learning system use a setup that is identical to the setup implemented by iControl$^{ESI®}$, and successfully utilized by the company's clients. These experiments validate that the active learning methods that we propose in this publication achieve a higher recall than that of the existing counterparts. We also study the stabilization behavior of proposed batch-mode active learning methods on these real-world discovery datasets and discuss our recommendation for choosing the stabilization point of the learning model in a deployed system. In addition, we present the performance of the system considering the two-phase review effort of attorneys. Finally, we provide an in-depth discussion of various design choices that we have made in our deployed system during different stages of the predictive coding process—no such insight on real data is yet available in the existing literature.

The remainder of this paper is organized as follows. In Section II, we discuss related works. In Section III, we present the proposed batch-mode active learning methodologies. Section IV presents detailed experimental results. Finally, a discussion of various design choices for commercial deployment of predictive coding is given in Section V.

II. Related Work

A. Active Learning

There are two paradigms of active learning, which differ in the concept by which they choose a new instance to label. They are (1) relevance feedback sampling [7] and (2) uncertainty sampling [8]. Relevance feedback-based methods use keyword search or random sampling for selecting the initial set of responsive documents, and they build a learning model that is updated iteratively after a fraction of the top-scoring documents in each iteration are labeled as relevant. The process stops after a sufficient number of documents are coded as relevant. Due to the positive feedback in favor of responsiveness, such methods yield high recall, but their learning models more often suffer from the *self-fulfilling prophecy* and the performance of such methods depends strongly on the initial batch of instances. On the other hand, uncertainty sampling-based methods select the instance for which the prevailing learning model is the most uncertain. Such a method does not suffer from the problem of relevance feedback-based methods because the decision to add an instance to the training sample depends on an instance's feature values rather than the label, which is predicted by the existing learning model.

Within the uncertainty sampling-based methodologies, a large number of uncertainty metrics have been proposed; examples include entropy [9], smallest-margin [10], least confidence [11], committee disagreement [12], and version space reduction [2], [13]. Different metrics are a good fit for different learning algorithms. For instance, the authors in [11] use conditional random field as the learning method and least confidence as the uncertainty metric for active learning, whereas the authors in [2], [3] use SVM as the learning algorithm with version space reduction as the uncertainty metric. Tong et al. [3] prove that version space reduction provides the optimal solution for active learning under some optimality criterion. They further show that the distance of an instance from the SVM hyperplane approximates the version space reduction metric, and hence, the instance that is the closest to the hyperplane should be chosen in an active learning iteration. Batch-mode active learning methods that we propose in this work use SVM as the learning algorithm, and distance from the hyperplane as the uncertainty metric.

B. Batch-Mode Active Learning

Initial theoretical works on active learning were limited to a batch size of one (one sample at a time) [2], [14], [15]. But, due to practical consideration, it is unreasonable to retrain the classifier at every iteration with only one additional training sample. So, a larger batch size (typically between 20 to 100) is considered in all real-life systems. The majority of the existing works on batch-mode active learning simply apply the single-instance uncertainty metric over the unlabeled instances and choose a batch of k instances with the best metric values. For instance, Tong et al. [3] proposes SVM_{active}, in which they construct the batch with the set of instances that are the closest to the hyperplane. Note that such a method is sub-optimal because it chooses the batch instances by considering the metric value of a single instance independently instead of designing a metric for a group of instances. Brinker [5] attempts to overcome the limitation by proposing an approach that constructs a batch of new training examples by ensuring that selected samples are nearest to the hyperplane and also maximally diverse (through cosine angle) from all other samples that are selected in the current batch. There are other methods [13], [16], [17] that are based on Generalized Binary Search [16] and submodular optimization [18]–[20]. But, they are generally costly, and their applicability in the legal domain where the dimension of the feature space is substantially large (several million) is unpromising.

C. Active Learning Stopping Criteria

The objective of a stopping criterion is to determine when an active learning based classification model reaches the point of maximum effectiveness [8]. Designing such a criterion is a difficult task. A number of heuristic methods have been proposed for stopping active learning. The learning process stops: (1) when all of the remaining unlabeled examples are outside of margin of existing model's hyperplane [14]; (2) when the number of support vectors saturates [21], [22]; (3) when the max confidence, min-error, overall uncertainty, or a combination of these three reaches a certain threshold [23]–[25]; (4) when the entropy of each selected sample or error on prediction is less than a threshold [26]; and (5) when the variance of the confidence score reaches the global peak [27]. In [28], the authors have shown that most of the stopping methods tend to behave too conservatively (taking a large number of samples) except inter-model Kappa agreement [29] method, which although stops early, does not lose performance in terms of F-measure. Subsequently, the authors of [30] analyze how the F-measure changes if the Kappa agreement between two models exceeds a certain threshold. For predictive coding, we have been looking for methods that will stop early

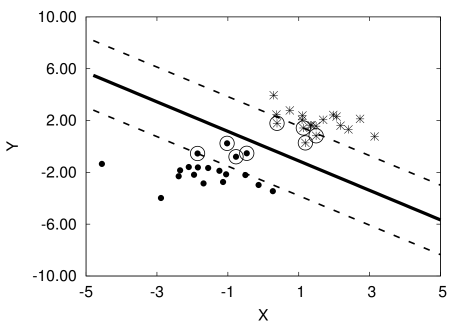


Fig. 1: Separating hyperplane and the margins for a linear soft-margin SVM

without sacrificing performance. Kappa agreement [29] seems quite suitable for our purpose as it measures the model stability and gives us a reasonable stopping point. To the best of our knowledge, we are the first to apply stopping criterion for analyzing active learning models in the legal domain.

D. Predictive Coding in the Legal Domain

A number of studies [1], [31], [32], [32]–[34] have been conducted to show the challenges and promises of "predictive coding". A study conducted by [34] shows that TAR methods can be more effective and efficient than traditional e-discovery practice, which typically consists of a keyword or Boolean searching, followed by manual review of the search results. According to [6], the TAR tools referred to as "predictive coding" in legal marketplace follow one of the three protocols: (1) Simple Active Learning ("SAL"), (2) Simple Passive Learning ("SPL"), and (3) Continuous Active Learning ("CAL"). Note that SAL includes uncertainty sampling-based active learning methodologies, SPL covers non-active supervised learning methodologies, and CAL includes relevance sampling. [6] simply compares CAL and SAL methodologies without proposing any novel active learning method. Existing works on the legal domain also discuss the need of stabilization metrics; typically, some statistical evaluation metrics are proposed [35]. However, these are offline metrics that are not integrated within the learning framework.

III. Methods

Given a collection of n documents $\mathcal{D} = \{D_i\}_{1 \le i \le n}$, which are potentially related to a legal issue, the objective of a review task is to discover the responsive documents with the least amount of effort from the expert attorneys. In TAR, this task is modeled as a 2-class classification problem where each document D_i is represented as a tuple $\langle \mathbf{x}_i, y_i \rangle$; $\mathbf{x}_i$ is a d-dimensional feature vector representation of document D_i, and $y_i \in \{+1, -1\}$ is a binary label denoting whether the document D_i is responsive $(+1)$ or non-responsive (-1). For the collection of documents, the feature vectors, $\mathbf{x}_i$'s can be built using standard IR methodologies that convert each document to a d-dimensional vector by considering it as a bag of words/phrases and then selecting an appropriate weight (such as tf-idf) for each word/phrase. On the other hand, y_i is initially unknown for all of the documents but an expert attorney with the knowledge of the legal issue can review the document D_i and assign y_i manually. For a dataset $\mathcal{D}$, we use the matrix $\mathbf{X}_{\mathcal{D}}$ to represent all of the feature vectors $\mathbf{x}_i$'s; similarly, we use $\mathbf{y}_{\mathcal{D}}$ to represent all the labels, y_i's. TAR methods use a supervised classification model to predict the label of each document with the least amount of labeling effort by the attorneys.

In this work, we use soft-margin SVM for the classification task. SVM is a supervised classification algorithm which uses a set of labeled data instances and learns a maximum-margin separating hyperplane $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$ by solving a quadratic optimization problem. $\mathbf{w}$ controls the orientation of the hyperplane and b is the bias which fixes the offset of the hyperplane in d dimensional space. Separating hyperplanes of SVM are linear, but by using non-linear kernels, SVM can also learn a non-linear hyperplane, if necessary. Hyperplane $h(\mathbf{x})$ splits the original d-dimensional space into two half-spaces such that if a test instance $\mathbf{x}_i$ falls on the positive side of the hyperplane (i.e., $h(\mathbf{x}_i) \geq 0$), $\mathbf{x}_i$ is predicted as $+1$, and otherwise, it is predicted as -1. In Figure 1, we show a separating hyperplane obtained using a linear SVM; the solid line represents the decision boundary and the dashed lines represent the margin.

For batch-mode active learning using SVM, given an unlabeled document-set $\mathcal{D}$, and a batch size k, we uniformly sample k instances from $\mathcal{D}$ to construct the first batch for which attorneys provide the label. Using this as the training data for SVM, we construct the initial hyperplane $h_c(\mathbf{x})$. In every subsequent iteration of active learning, we use the current hyperplane $h_c(\mathbf{x})$ to actively select a new batch of unlabeled instances $\mathcal{B}_c$. The instances of this batch become part of the training data once their labels are obtained. Using the extended training data, we update the current hyperplane. The process continues until a stopping criterion is met. We formally describe the active learning process in Algorithm 1. In Line 1, we obtain the initial hyperplane h_c using a randomly selected size-k batch of training instances. The **while** loop in Line 2-7 is one iteration of active learning where a new batch of training instances is added and an updated hyperplane is obtained using the extended training dataset.

Algorithm 1: Batch Mode Active Learning Algorithm using SVM

```
Input  : $\mathcal{D}$, unlabeled dataset; $k$, batch size
Output: Learned hypothesis, $h$
1 $h_c \leftarrow$ ObtainInitialHyperplane $(\mathcal{D}, k)$
2 while Stopping Criteria not met do
3     $\mathcal{B}_c \leftarrow$ SelectABatch $(\mathcal{D}, h_c, k)$
4     $\mathbf{y}_{\mathcal{B}_c} \leftarrow$ QueryLabels $(\mathcal{B}_c)$
5     $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{B}_c$
6     $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup \mathcal{B}_c$
7     $h_c \leftarrow$ Train $(\mathcal{D}_c)$
8 $h \leftarrow h_c$
9 return $h$
```

A. Proposed Active Learning Methods

We propose two novel methods, namely Diversity Sampler (DS) and Biased Probabilistic Sampler (BPS), for selecting a batch of k documents at each iteration using the prevailing SVM hyperplane, h_c. Like the existing SVM-based active learning methods, both DS and BPS use an uncertainty metric, which selects the instances closest to the separating hyperplane of SVM. But unlike existing works, we introduce the concept of exploration and exploitation of reinforcement learning in our methodologies, which we discuss in the following paragraphs.

In an active learning setup, the existing hyperplane represents the current knowledge regarding the decision boundary between the two classes. However, this hyperplane is obtained by training over the existing training set $\mathcal{D}_c$, and hence, it can be substantially different than the optimal hyperplane the entire dataset $\mathcal{D}$. Many of the existing active learning methods, such as SVM_{active}, select a batch of k instances that are nearest to the current hyperplane h_c. Such an action is similar to the concept of full exploitation as the selection of instances are made entirely based on the existing knowledge of the environment. Such methods fail to shift the initial hyperplane towards the ideal hyperplane because every iteration selects instances that are closest to the prevailing hyperplane without any exploration. Thus, they perform poorly if the initial hyperplane is far-off from the ideal hyperplane. Specifically, for TAR datasets that have very small prevalence (the proportion of relevant documents is very small), a uniform random selection at initialization most often returns a hyperplane which is far-off from the optimal hyperplane. So, such methods perform poorly on such datasets.

An alternative to full exploration can be a mix of exploration and exploitation, where instances are not only selected by their distance from the hyperplane, but also by a diversity criterion. Based on our observations of a large number of real-life TAR datasets, we found that many documents are substantially similar to each other, so we enforce diversity among the instances selected in a batch. Both DS and BPS facilitate diversity, but they differ in the way they select an instance—the selection of DS is deterministic, whereas the selection of BPS is probabilistic. For the DS method, we first sort all of the available documents in a non-decreasing order of their distance from the current hyperplane h_c and filter all of the documents (we do not select them in the current batch) that are similar to the last instance selected in the current batch. For BPS, we construct a probability vector and use it to select a document in inverse proportion to its distance from the current hyperplane. Using probabilistic selection, BPS uses an idea that is similar to the concept of the randomized weighted majority (RWM) algorithm [36] used for no-regret online learning. Considering the documents in increasing order of their distance from the hyperplane ensures exploitation, and filtering similar documents thus enables the selection of documents which otherwise would not have been selected, ensuring exploration.

Algorithm 2: SelectABatch

```
Input  : $h_c$, current hyperplane; $\mathcal{D}$, available instances;
         $k$, batch size; and similarity threshold, $t$
Output: A batch of $k$ documents to be included in
         training
1  if Strategy is DS then
2      $\mathcal{B}_c \leftarrow$ EmptySet()
3      $I \leftarrow$ ArgSort (Distance$(h_c, \mathcal{D}), order = increase$)
4      while Size ($\mathcal{B}_c$) $< k$ do
5          Insert$(\mathcal{B}_c, I[1])$
6          $S \leftarrow$ GetSimilar$(I[1], I, \mathcal{D}, t, similarity = cosine)$
7          $I \leftarrow$ Remove$(I, S)$
8  else if Strategy is BPS then
9      $\mathbf{w} \leftarrow 1.0/($Distance$(h_c, \mathcal{D})^2$
10     $\mathbf{w} \leftarrow$ Normalize$(\mathbf{w})$
11     $I \leftarrow$ List$(\mathcal{D})$
12     while Size ($\mathcal{B}_c$) $< k$ do
13         $c \leftarrow$ Choose$(I, prob = \mathbf{w}, num = 1)$
14         Insert$(\mathcal{B}_c, c)$
15         $S \leftarrow$ GetSimilar$(c, I, \mathcal{D}, t, similarity = cosine)$
16         $I \leftarrow$ Remove$(I, S)$
17         $\mathbf{w} \leftarrow$ Normalize$(\mathbf{w}[I])$
18 return $\mathcal{B}_c$
```

In Algorithm 2, we formally describe both of our batch selection algorithms. In addition to the current hyperplane h_c and available dataset $\mathcal{D}$, both of these methods also have a user-defined parameter $t \in [0, 1]$, which denotes a cosine similarity threshold. Lines 2-7 describe the DS method and Lines 9-17 describe the BPS method. For DS, in Line 2, we first sort the documents in increasing order based on their absolute distance from the prevailing hyperplane h_c and get the sorted indices of the available documents, $\mathcal{D}$ in I. In Line 5, we choose the nearest one deterministically and insert it into the current batch set, $\mathcal{B}_c$. We then get the indices of the documents that have cosine angle $\geq t$ with the currently selected document, $I[1]$ (including $I[1]$). We then remove all of those indices from the I and repeat Lines 5, 6 and 7 until $\mathcal{B}_c = k$. For the probabilistic sampler, distance is calculated over the unlabeled documents. For some documents, the distance value can be 0 (falling over the hyperplane); in those cases, we set a minimum absolute distance as the distance of those documents from the hyperplane. We need to do this as in Line 9, we have an inverse operation. In Line 10, we normalize the weight vector to convert it into probability. In Line 13, we choose one document, c, using the weight, $\mathbf{w}$, calculated in Line 10. We then perform the same operations we did in Lines 5, 6 and 7. Finally, we re-normalize the weight, $\mathbf{w}$ as some of the documents have been removed from index list, I in Line 16.

Note that our proposed method DS is somewhat similar to Brinker's method [5], but the latter has a trade-off parameter that determines the importance between the distance from hyperplane and diversification of the instances. Such a parameter is hard to choose. We will show experimental results which validate that both DS and BPS perform better than Brinker's method on real-life legal datasets.

Computational Complexity In this section, we analyze the computational complexity of our proposed methods as described in Algorithm 2. For DS, sorting in Line 3 takes $O(|\mathcal{D}| \log |\mathcal{D}|)$ time after the $O(|\mathcal{D}|)$ distance calculation operation. For a small batch size, k (in our case, k = 64), the insert operation takes a constant time, $O(1)$. Line 6 takes $O(k \cdot |I|)$ time. We actually do not remove any elements from I; we just unset (not available for choosing) a flag for those documents, and when we choose in Line 4, we keep a pointer to the first document where the flag is set (available for choosing). Also, when we get similar documents in Line 6, we do it in the order of I. So, in Line 7, we just unset $|S|$ flags which takes $O(|S|)$ time. Hence, the computational complexity of DS is $O(|\mathcal{D}| \log |\mathcal{D}|)$ as $O(k \cdot |I|)$ is much less than $O(|\mathcal{D}| \log |\mathcal{D}|)$. For the probabilistic sampler, initial distance computation and normalization takes $O(|\mathcal{D}|)$ operations. The main cost incurs from k "choose" operations which take $O(k \cdot |I| \cdot \log |I|)$ time.

B. Stopping Condition

The primary motivation for having a stopping condition is to stop training as early as possible (training is costly). However, we also want to confirm that the final hyperplane h is stable (i.e., the prediction model will not change considerably if we add more training documents). For tracking stability, we use Cohen's Kappa agreement [29], which is a metric used in computational linguistics for measuring inter-coder agreement. Let's say that, after the batch update operation in Line 8 of Algorithm 1, the hyperplane is h and before the update, it was h'. Kappa agreement measures how much these two hyperplanes agree on their prediction of labels on a carefully chosen test set. If h and h' agree on a instances out of n instances, the fraction $\frac{a}{n}$ represents the *observed agreement*, A_o. However, the observed agreement needs to be scaled with respect to the *chance agreement*, A_e, which measures the agreement that is expected between h and h' purely by chance. For calculating A_e, Cohen's Kappa computes the likelihood by which the hyperplanes h and h' agree with each other even if they are independent. Mathematically,

$$A_e = P(+1|h)P(+1|h') + P(-1|h)P(-1|h')$$

where $P(+1|h)$ is the probability that hyperplane h labels an instance as being $+1$, which is estimated based on the proportion of observed instances that h labels as $+1$. Similarly, the proportion of observed instances that h labels as -1 provides $P(-1|h)$. The same pair of expressions can also be obtained for h' and can be used in the above equation to measure A_e. Once we have A_o and A_e, Cohen's Kappa, κ

is computed as follows:

$$\kappa = \frac{A_o - A_e}{1 - A_e} \tag{1}$$

The value $A_o - A_e$ quantifies the agreement between h and h' that is found beyond chance which is normalized by the maximum possible quantity for this value $(1 - A_e)$. Even though the ratio in the range $[0.8, 1.0]$ is considered good, we want a much stronger guarantee for a legal dataset. For all datasets, we stop training when κ reaches ≥ 0.991 for several consecutive iterations.

IV. Experiments

We implement our proposed methods, Diversity Sampler (DS) and Biased Probabilistic Sampler (BPS) in a commercial system which is deployed in iControl$^{\text{ESI®}}$'s e-discovery platform, Recenseo®. Under the same platform, we also implement two of the existing methods, SVM_{active} [3] and Brinker [5]. For SVM, we use LibLinear, an open-source linear SVM implementation which is well-known for its good performance with large-scale text classification. We perform a set of experiments for evaluating the performance of the proposed methods, DS and BPS, on a number of legal system datasets and publicly available datasets. Experimental results also include comparison between the proposed methods with SVM_{active} and Brinker's method. Our method has only one user-defined parameter and that is a similarity threshold value. Our experiments on a large number of datasets show that a reasonable value for similarity is between 0.50 and 0.95, and within this range, the performance of the model differs only marginally. For all of our experiments, we fix this value to be 0.85. Brinker's method also has a parameter which is the relative importance between hyperplane distance and diversity; we use 0.85 for this parameter also. We run all of the experiments on a computer with a quad-core Intel XEON E5-2665, 2.4Ghz processor running CentOS 6.6 operating system. For the largest dataset that we use, the time to run each iteration of batch-mode active learning is about 1 minute for SVM_{active}, and approximately 5 minutes for the remaining three methods.

A. Datasets

We use seven matters for our experiments. For each of these matters, we present the statistics of the corresponding dataset in Table I. For each matter, we partition the dataset into Train and Test; the active learning is performed over the Train partition, and the Test partition is only used for evaluation.

The first two matters, ACQ and MONEY-FX, are from the publicly available Reuters Dataset[1]. This dataset has a total of $21,578$ documents. Matters D1-D4 correspond to documents that a review team examined for responsiveness in two distinct product liability lawsuits. The team consisted of approximately 50 attorneys, and they conducted the review in a traditional linear fashion over the course of several months, in 2013 and 2014. The reviewers designated each document as responsive to lawsuit 1 only (D1), lawsuit 2 only (D2), lawsuit 3 only (D3) and lawsuit 1, 2, and 3 (D4). There are $788,875$ documents in D1-D4, after filtering out files without extractable text. Matter C comes from another dataset of $366,999$ documents which was reviewed by 30 attorneys for a particular lawsuit. This dataset has a higher prevalence (25.98%) than the other datasets because we ran a keyword search on the documents and filtered out a larger number of non-responsive documents (initially, the prevalence was around 3%). From Table I, it is evident that our collection is rich in terms of prevalences—we have a collection with low (1.20), medium (6.20, 11.24) and high prevalence (25.98) scores. Finally, the prevalence score for test documents reveals that they are a true representative of the training documents.

B. Performance Metrics

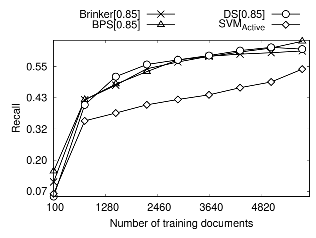
We use recall for measuring an algorithm's performance because this is used in real-life TAR tasks in the legal domain. Note that recall is computed over the held-back test dataset using the final model, which is learned using active learning. If on a test dataset, $\mathcal{R}_p$ is the number of documents that are marked as responsive by a prediction method and $\mathcal{R}_t$ is the number of true responsive documents, recall is defined as $\frac{\mathcal{R}_p}{\mathcal{R}_t}$. However, recall does not provide any indication of the attorneys' effort for labeling the train dataset. To determine this, a different approach is used, which we discuss below.

In active learning-based training in the legal domain, there are two phases of review by the attorneys. The first phase goes along with the active learning process, in which attorneys provide feedback on the batches of documents that are selected by the active selection strategy. However, when the model stabilizes and the final model is obtained, the "Second Pass" review of the remaining part of the training data (which has not been used for training the model) begins. Let's call this set of instances $\mathcal{D}_r$. To minimize reviewers' efforts, the goal of the second pass is to choose a small subset of documents from $\mathcal{D}_r$, of which the majority will be responsive. For this, the documents in $\mathcal{D}_r$ are ranked based on their likelihood to be responsive. When using linear SVM, this ranking can easily be done by finding the signed distance of a document from the separating hyperplane returned by the final model. The more positive the distance, the more likely it is that the document is responsive. So, the documents in $\mathcal{D}_r$ are sorted in the non-increasing order of their signed distance, and a fraction of documents from the beginning of this sorted list are considered for the second pass review. The yield curve shows the relationship between recall and the minimum fraction of documents that must be reviewed to achieve that recall value. The steeper the yield curve, the better the model, and the fewer number of documents are needed for the second pass review. Also note that, in real-life TAR, yield curve is used to determine the required bias of the classifier for obtaining a desired recall metric (aka model calibration). We will also use yield curve to show the reviewers' efforts using our proposed model.

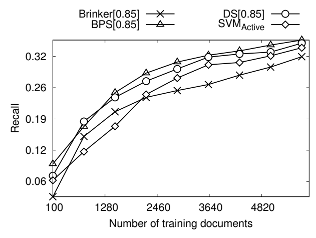
[1]http://archive.ics.uci.edu/ml/machine-learning-databases/reuters21578-mld/reuters21578.tar.gz

TABLE I: Dataset Statistics

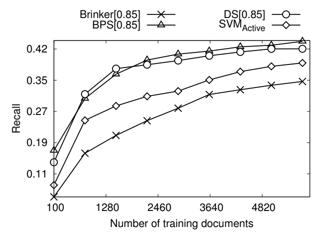
Matter	Total no. of features	Train Documents			Test Documents		
		Total	Positive	Prevalence (%)	Total	Positive	Prevalence (%)
ACQ	$\sim 41k$	$14,668$	1650	11.24	$6,910$	798	11.54
MONEY-FX			539	3.67		262	3.70
C	$\sim$ 2.6 millions	$358,903$	$93,256$	25.98	$8,096$	$2,094$	25.88
D1	$\sim$ 6.7 millions	$772,491$	$14,726$	1.20	$16,384$	364	2.22
D2			$48,355$	6.20		$1,142$	6.97
D3			$95,857$	12.40		$2,199$	13.42
D4			$158,938$	20.75		$3,284$	20.04



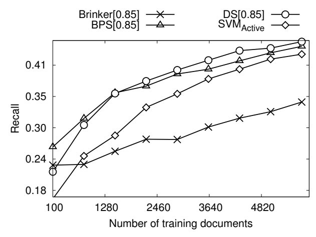
(a) Recall (D1)



(b) Recall (D2)



(c) Recall (D3)



(d) Recall (D4)

Fig. 2: Classification assessment results for Matters D1-D4 (Batch size = 64)

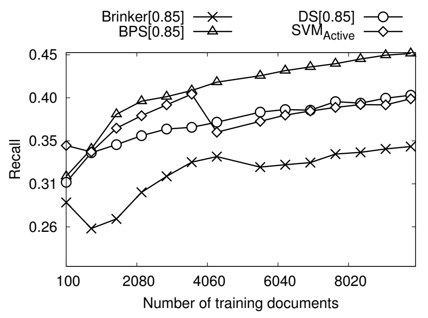
C. Performance Comparison of DS and BPS with the Existing Methods

For each of the matters in this experiment, we train four distinct linear SVM models using the following four active learning methods: DS, BPS, SVM_{active}, and Brinker. We compare the performance of these methods by tracking the recall of their trained model over test data across all training stage iterations until the model stabilizes. In Figures 2 and 3, we show seven plots, one for each of the matters. In each plot, we show the number of training instances along the x-axis of the plot, and the recall value along the y-axis. Within each plot there are 4 curves, one for each of the active learning methods. The method for which the recall value is the highest for a given number of training instances is the best.

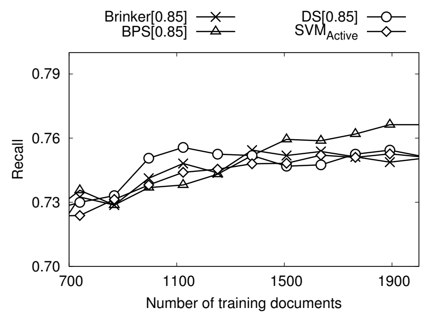
As we observe from these plots, for all of the methods and all of the datasets, as we increase the training data, the recall of the model improves, which is expected. For most datasets, BPS has a higher recall than the remaining three methods at all stages of training. One exception is the ACQ dataset (Figure 3 (b)) for which the DS is the best during the initial part of the training but as the model stabilizes, BPS comes back to the best position and retains that position by a good margin from the remaining methods. Another exception is the D4 dataset, for which both BPS and DS have the best performance with a marginally higher recall for DS. Overall, DS is the second best method, and its performance is almost the same as BPS for the D2, D3, and D4 datasets. Both our proposed methods have a higher slope in their curves at the beginning part of the training. This proves their ability on selecting good instances early when the prevailing hyperplane may not be close to the ideal one. Except for the D1 dataset, Brinker's method generally performed the worst for all our proprietary datasets. For some datasets (D3, D4, and C), the performance of Brinker's method is worse than all of the our proposed methods by a good margin across the entire training period. The performance of SVM_{active} is somewhat between Brinker and our proposed methods. In summary, the plots in these figures clearly demonstrate that our active learning methods are substantially superior to the existing methods.

D. Yield Curve Results on BPS

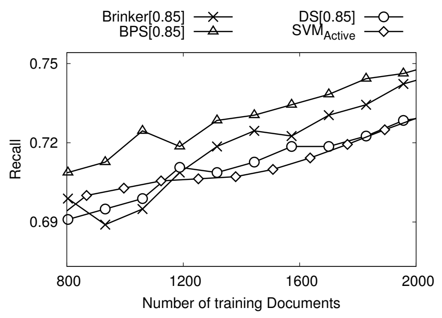
We also study the yield curves for both of our DS and BPS methods across all the datasets. But, due to space limitations, in Figure 4, we show these curves only for the BPS method for six datasets. The trend is similar for the remaining dataset and also for the DS method. Remember that the yield curve is built from the ranking of unused training data obtained from the stabilized model; and using the yield curve, we can determine the percentage of documents that is required for the second pass review for achieving a specific recall value for a particular dataset. For all of our experiments, we use $\kappa = 0.991$ for deciding the stabilization of the model. For all of the yield curves in Figure 4, we show along x-axis, the percentage of remaining training documents as are considered from the rank order built by the final model; and along the y-axis, we show the corresponding recall. Similar to an ROC curve, the steeper the curve, the better the performance. Also, as we consider all of the documents, the recall value becomes 1.0. So, what is interesting is to see the percentage of documents that the attorneys need to review for achieving an acceptable recall in real life (usually 0.75). As we can see, for the least prevalent dataset (D1), the second pass review only needs to consider 3.38% of the documents for achieving 0.75 recall, which is a 96.62% savings in terms of attorney effort. For datasets that have high prevalence, these values are naturally larger; for instance, for dataset D2-D4, 0.75 recall is reached by reviewing roughly one-third of the documents, resulting in



(a) Recall (C)

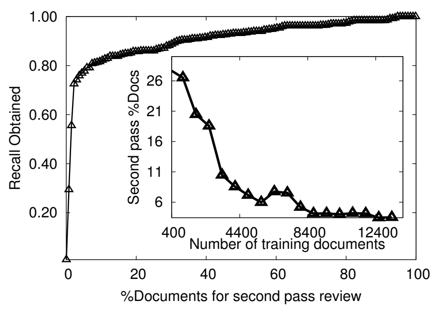


(b) Recall (ACQ)

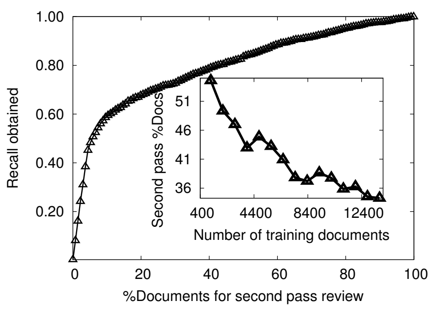


(c) Recall (MONEY-FX)

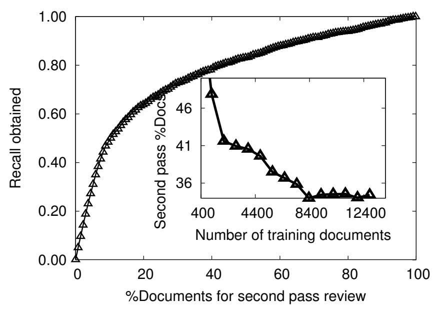
Fig. 3: Classification assessment results for C, ACQ, and MONEX-FX matters (Batch size = 64)



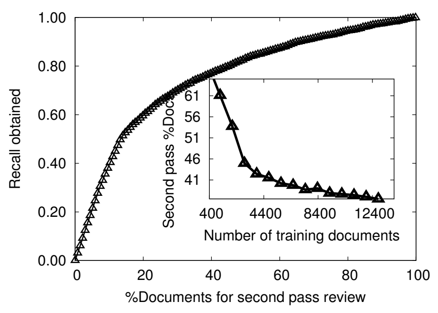
(a) Yield (D1)



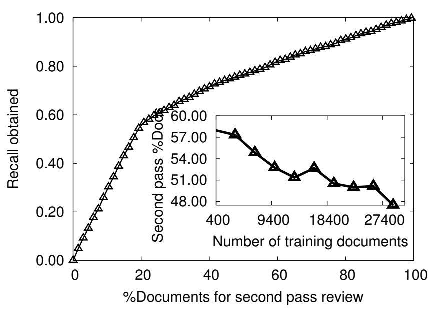
(b) Yield (D2)



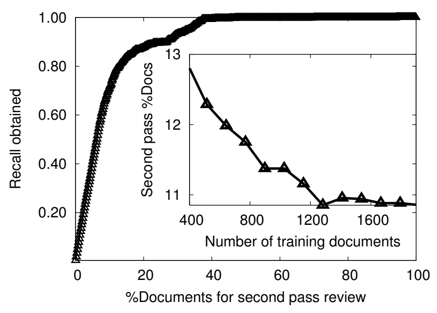
(c) Yield (D3)



(d) Yield (D4)



(e) Yield (C)



(f) Yield (ACQ)

Fig. 4: Yield curve

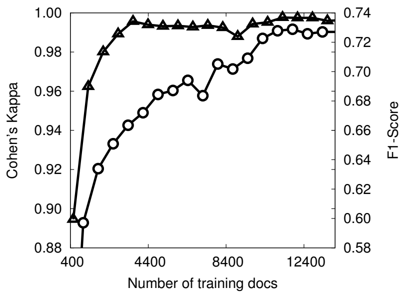
TABLE II: Statistics of Review to Achieve 75% Recall

Matter	%Docs to review to achieve 75% Recall	%Docs used for training
ACQ	10.89	12.66
MONEY-FX	4.51	14.84
C	46.30	8.62
D1	3.38	1.76
D2	33.40	1.80
D3	33.40	1.71
D4	36.76	1.73

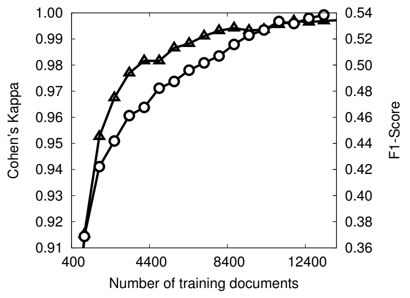
a 67% savings.

In the inset of each of the plots in Figure 4, we show how the number of training documents improves the learning model, which in turn decreases the percentage of documents that are needed for achieving 75% recall in second pass review. The downward trend in the inset plots confirms that as the learning progresses and the Kappa value stabilizes, the percentage of documents for second pass review decreases for all the datasets.

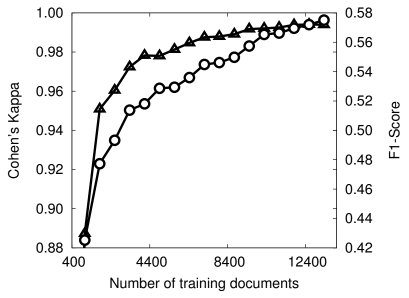
In table II, we summarize the percentage of documents that are needed to achieve 75% recall for the second pass review. We also show the percentage of documents that are used for training the model when it is stabilized. These are the documents for which attorneys provide labels. For matter D1, only 5% of the documents need to be reviewed (95% savings), but for matter C, this value is around 55% (45% savings). The reason for the higher value in matter C is due to the high prevalence of matter C (which is 26%). Note that, for TAR tasks, attorneys are expected to review all of the relevant documents, so for matter C, they are required to review 26% of the documents, but using our model, they need to review 55% of the documents to yield a 0.473 precision, which is considered excellent in the TAR domain.



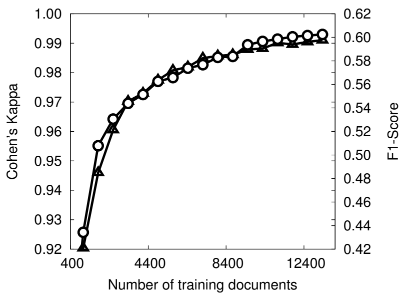
(a) Cohen's Kappa (D1)



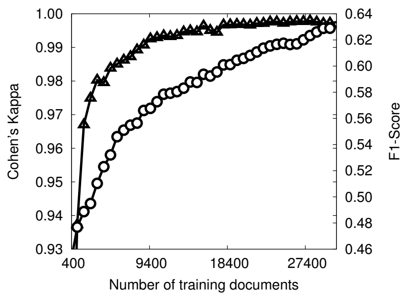
(b) Cohen's Kappa (D2)



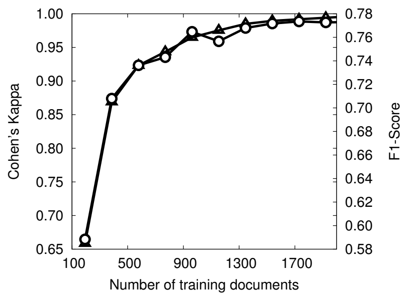
(c) Cohen's Kappa (D3)



(d) Cohen's Kappa (D4)



(e) Cohen's Kappa (C)



(f) Cohen's Kappa (ACQ)

Fig. 5: Stabilization assessment results

E. Stabilization Behavior Study on BPS

In this experiment, we show that Cohen Kappa (κ) is an excellent metric for determining the stabilization point of an active learning method. Again, we show results for the BPS dataset only, as the results for the DS method are almost identical. In Figure 5, each plot shows two curves; one shows the relationship between the Kappa value and the number of training documents, and the other shows the relationship between the F1-score (harmonic mean of precision and recall) of the model and the number of training documents. For all of the datasets, both the Kappa value and the F1-score of the model increase as we increase the number of training samples. However, as we can see in Figure 5, when the Kappa value converges (the curve becomes horizontal), the F1-score also becomes horizontal indicating the model's stabilization. In all previous experiments, we used recall metric, but for tracking stabilization, we use the F1-score because the recall metric in isolation cannot indicate model stabilization. This is due to the fact that a training model can always improve its recall by sacrificing the precision, and this can be done simply by biasing the hyperplane. The F1-score considers both precision and recall together and hence, it is a neutral metric that can be used for the model stabilization purposes. For our entire study, we use a Kappa value of 0.991 for deciding model stabilization.

V. Discussion and Lesson Learned

Choice of Learning Algorithm. As mentioned previously, existing research has shown SVM to be one of the best methods for text classification. In addition to SVM, we have experimented with a number of other algorithms including Naive Bayes, Nearest Neighbor, Logistic Regression, Perceptron, and various ensemble approaches on real-life datasets. In all cases, SVM performed better than, or at least as good as, every other method attempted. SVM also easily lend itself to active learning approaches, and is computationally efficient even for very large numbers of features and examples.

Feature Selection and Representation. In most real-life legal datasets, the document collection contains several difficulties that impact the performance of any classification algorithm. There are exact duplicate documents, near duplicate documents, OCR'ed documents with significant amounts of noisy text, spreadsheets with significant numerical data, binary files, etc. Without any preprocessing, these files cause computation times and storage volumes that exceed acceptable levels. We have found that it is absolutely necessary to clean the data before the learning task to achieve the best performance. For our task, first we have removed document types that contain little or no usable text from the predictive coding collection, and we classify documents from those types through other means. Second, we have implemented several algorithms for identifying and removing noisy tokens from the collection. This includes identifying OCR errors and other noisy tokens through a variety of heuristics, filtering stop words, and filtering words that are uncommon in the collection.

Another significant challenge with real-life legal datasets that is unaddressed in the existing literature is that document collections are rarely fixed. Over the course of litigation, new documents can be added to the collection and other documents removed. However, existing literature assumes that the document collection is fixed before the learning task begins, and stays fixed throughout. This has major implications on the selected feature representation. Specifically, any term weighting scheme with a global weighting component (TF-IDF, for example) could result in feature vectors changing over the course of the learning task with unstudied effects on the active learning process. So, global weighting scheme should be avoided as much as possible. For a large collection of real-life litigation datasets, we at iControl$^{\text{ESI®}}$ have performed experiments with TF-IDF, LSA, LDA, and Log-Entropy based feature weighting schemes and found them to perform no better than the standard bag-of-words model. One other potential feature selection scheme is the hashing trick, which would be suitable since it has no global component, but it was also found to perform worse than the bag-of-words model.

VI. Conclusion

In this paper, we present two active learning-based methods for "predictive coding" in the legal domain. Experimental results show that both of our proposed methods achieve better recall than the existing methods that are used in TAR processes. We also show experimental results on the stabilization behavior of our methods and discuss practical recommendations for the various design choices.

References

[1] D. W. Henry, "Predictive coding: Explanation and analysis of judicial impact and acceptance compared to established e-commerce methodology," http://www.dwhenry.com/files/Predictive%20Coding.pdf, [Online;Accessed 23-June-2015].

[2] S. Tong and D. Koller, "Support vector machine active learning with application to text classification," vol. 2, 2001, pp. 45–66.

[3] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *Proceedings of the ninth ACM international conference on Multimedia*. ACM, 2001, pp. 107–118.

[4] S. Dasgupta, "Coarse sample complexity bounds for active learning," in *NIPS*, 2005, pp. 235–242.

[5] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *ICML*, vol. 3, 2003, pp. 59–66.

[6] G. V. Cormack and M. R. Grossman, "Evaluation of machine-learning protocols for technology-assisted review in electronic discovery," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 153–162.

[7] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Readings in information retrieval*, vol. 24, no. 5, pp. 355–363, 1997.

[8] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc., 1994, pp. 3–12.

[9] I. Dagan and S. P. Engelson, "Committee-based sampling for training probabilistic classifiers," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 150–157.

[10] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," in *Advances in Intelligent Data Analysis*. Springer, 2001, pp. 309–318.

[11] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks," in *AAAI*, 2005, pp. 746–751.

[12] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 287–294.

[13] R. Nowak, "Noisy generalized binary search," in *Advances in neural information processing systems*, 2009, pp. 1366–1374.

[14] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *ICML*. Citeseer, 2000, pp. 839–846.

[15] M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmen, "Active learning with support vector machines in the drug discovery process," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 667–673, 2003.

[16] S. Dasgupta, "Analysis of a greedy active learning strategy," in *NIPS*, 2004, pp. 337–344.

[17] Y. Chen and A. Krause, "Near-optimal batch mode active learning and adaptive submodular optimization," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 160–168.

[18] A. Guillory and J. A. Bilmes, "Active semi-supervised learning using submodular functions," in *UAI*, 2011, pp. 274–282.

[19] D. Golovin and A. Krause, "Adaptive submodularity: A new approach to active learning and stochastic optimization." in *COLT*, 2010, pp. 333–345.

[20] A. Asadpour, H. Nazerzadeh, and A. Saberi, "Stochastic submodular maximization," in *Internet and Network Economics*. Springer, 2008, pp. 477–489.

[21] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: active learning in imbalanced data classification," in *Proc. of ACM Int. Conf. on Knowledge Management*, 2007, pp. 127–136.

[22] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," in *SIGIR*, 2007, pp. 823–824.

[23] J. Zhu and E. H. Hovy, "Active learning for word sense disambiguation with methods for addressing the class imbalance problem." in *EMNLP-CoNLL*, vol. 7, 2007, pp. 783–790.

[24] J. Z. H. Wang and E. Hovy, "Learning a stopping criterion for active learning for word sense disambiguation and text classification," in *Third International Joint Conf. on Natural Language Processing*, 2008, p. 366.

[25] J. Zhu, H. Wang, and E. Hovy, "Multi-criteria-based strategy to stop active learning for data annotation," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008, pp. 1129–1136.

[26] F. Laws and H. Schätze, "Stopping criteria for active learning of named entity recognition," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008, pp. 465–472.

[27] M. Ghayoomi, "Using variance as a stopping criterion for active learning of frame assignment," in *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 1–9.

[28] M. Bloodgood and K. Vijay-Shanker, "A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2009, pp. 39–47.

[29] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[30] M. Bloodgood and J. Grothendieck, "Analysis of stopping active learning based on stabilizing predictions," *arXiv preprint arXiv:1504.06329*, 2015.

[31] H. L. Roitblat, A. Kershaw, and P. Oot, "Document categorization in legal electronic discovery: computer classification vs. manual review," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 1, pp. 70–80, 2010.

[32] M. Gabriel, C. Paskach, and D. Sharpe, "The challenge and promise of predictive coding for privilege," in *ICAIL 2013 DESI V Workshop*, 2013.

[33] M. R. Grossman and G. V. Cormak, "Inconsistent responsiveness determination in document review: Difference of opinion or human error," *Pace L. Rev.*, vol. 32, p. 267, 2012.

[34] M. R. Grossman and G. V. Cormack, "Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review," *Rich. JL & Tech.*, vol. 17, p. 1, 2010.

[35] J. Halskov and H. Takeda, "When to stop reviewing documents in ediscovery cases: The lit i view quality monitor and endpoint detector," in *Proc of the Fifth International Conference on Management of Emergent Digital EcoSystems*, ser. MEDES '13, 2013, pp. 227–232.

[36] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212–261, 1994.