Selecting Optimal Representative Training Sets

Apurva Bhargava(ab8687)
Center for Data Science - New York University

Abstract

Training classifiers for text-as-data applications requires high quality annotated or labeled data. Hand-labeling is costly and prone to errors. Scaling up the amount of data only exacerbates the problem. In this paper, I propose novel unsupervised methods for selecting a representative subset for hand-labeling, and training the first instance of the classification model optimally.

1 Introduction

To leverage statistical and machine learning models for supervised tasks on text data, having high-quality labeled training data is imperative. This training data is used to allow a model to learn making predictions for unforeseen examples. Compromising the quality may lead to a 'garbage-in, garbage-out' scenario. While there are certain situations where labels or outcomes for every example in the training set can be acquired from the same source as the data (reviews accompanied with ratings that can be used for sentiment labeling), for the most part, it is required that the data be labeled or annotated by human domain-experts. This makes the process of labeling or annotating the data both error prone and expensive.

The manual procedure of reading the text and consulting some guidelines for assigning labels is also time consuming and prone to inconsistencies if done by different people with different subjective interpretations of guidelines. All these issues exacerbate when scaling up the labeling task for a large amount of data. To give an idea, in an improved annotation scheme, annotations were gathered for 133,479 verb mentions in 64,018 sentences across 3 domains. With 2 validators, the annotation took 9 days on Amazon Mechanical Turk. 1,165 unique workers participated, annotating a total of 299,308 questions. Of these, 89% were considered valid by both validators. The total cost was $43,647.33, for an average of 32.7c per verb mention, 14.6c per question, or 16.5c per valid question [1].

In order to resolve the issues caused by scalability, it is helpful to identify a smaller subset of data that is representative of the dataset as a whole and can be used for training the models after labeling. The primary goal is to select a subset in a way that for the n selected training samples, the performance of the model is better than all other possible combinations of n samples, and that for a sufficiently large n, the performance is comparable to that when training the whole dataset. There isn't a way to efficiently and concretely establish the success of the former part of the primary goal without labeling the dataset and training ${}^{N}C_{n}$ models, where N is the total number of examples, which is both time-inefficient and against the goal of saving labeling costs. As for the latter part, the optimal value of n depends on the characteristics of the dataset- the variety and the noise in it. In real-time applications, the number of samples to be selected can be decided based on the cost and time budgets, and further samples can be added later using another iteration of the methods described in this paper, semi-supervised learning[2] or active learning[3].

An intelligent sampling scheme for selecting documents to label must ensure that the observations span the covariate space. One such algorithm divides the covariate space into equal-sized subspaces and randomly samples from each. However, not all subspaces will have observations in them due to the curse of dimensionality and even so, this introduces the problem of covariate shift. An alternate algorithm instead finds the n observations such that the sum of the distances between any two observations is maximized. In high dimensional problems like text, however, this means that most of

the observations will be outliers, rendering the model prone to overfitting still without solving the covariate shift problem. To optimize this trade-off and to the end of finding n representative samples, I propose a two-step method for (i) representing documents (textual data) in a low-dimensional space and (ii) identifying the documents that best span that space. The secondary goal is to compare the performance of the different combinations of my low-dimensional representations and subset selection algorithms against the D-optimal design method on topics model or Taddy [4], which is an existing well-performing method in literature. All the methods used employ unsupervised learning. For evaluation of the methods, however, labels are used in order to calculate the accuracy, f1-score, area under ROC. Another evaluation metric that does not require labels is the average distance from complete uncertainty (probability=0.5). I found combinations of low-dimensional text representations and subset selection methods that outperform the k-topics factor model combined with D-optimal design method.

2 Related Work

Natural Language Processing (NLP) applications include several supervised tasks such as sentiment analysis, corpus tagging, spam filtering, POS-tagging, named entity recognition, etc., that requires data to be labeled, tagged or annotated. In order to select a minimal subset that is representative of the data in an unsupervised fashion, a multitude of techniques exist- active learning is one where some small number of examples are sampled either randomly or using some distance-based heuristic to form an initial training set T before hand-labeling them and fitting a model to them. This model is then used on the remaining data U to generate the prediction confidence values for each example. The examples with most confident predictions for a label are assigned that label and added to the training set T, and removed from U. The model is trained and fitted to this new training data, and prediction confidences are generated on U, and the process repeats until T has the desired number of samples [5].

Another version of active learning starts similarly by choosing an initial T but then selects the most uncertain predictions (with probabilities close to 0.5) from U for further hand-labeling and adding to T. The latter version somewhat mitigates the issue of propagating initial sample bias, however, in either version, the entire process is iterative (and thus, computationally expensive) as it requires training a model on instance of training set and fitting it to making predictions on the remaining data. The number of variables to be handled include both the hyperparameters of the model as well as those controlling the selection from unlabeled dataset. My methods are non-iterative and do not depend on the choice of initial training examples, example selection procedures or performance of multiple machine learning models [3].

In order to leverage all the data without labeling all of it, semi-supervised learning frameworks [5] can be used to assign unlabeled data pseudo labels in order to use them for the task. Either of the following three assumptions must be true for the application of semi-supervised learning: points that are close to each other are more likely to share a label, or, the data tend to form discrete clusters and points in the same cluster are more likely to share a label, or, the data lie approximately on a manifold of much lower dimension than the input space [6]. The problem of selecting initial subset for hand-labeling still persists, the performance of semi-supervised models depends on the structure of the labeled data. This problem is resolved in my approach where the methods look at the entire data at once to make decisions.

My two-step method for optimal training subset selection idea is based on the approach of converting text documents to vectors through a mixed-membership weighting of multinomial topic factors from k-topic model[7], and then using topic D-optimal design algorithm[4] to select examples that are at the edges of the topic space. I used this unsupervised method as the baseline and found combinations of low-dimensional representations and representative subset selection methods that perform better than this baseline, and are executed in significantly lesser time.

3 Problem Definition and Algorithms

3.1 Task

The objective was to devise unsupervised methods to automatically select the most informative or the most representative documents for hand-labeling and training a classifier for text-as-data applications,

leading to building classifiers at a lower cost. In solving the aforementioned problem, I had two major considerations- (i) the data is not supposed to have any label information. As such, the method for text representation and subset selection had to be a completely unsupervised and discriminative approach; (ii) I would want to save time and computing resources. To this end, I would want to create non-iterative methods (unlike active learning) and work with low-dimensional representations (unlike BoW or Bag of Words representation with tens of thousands of features).

The process pipeline is described below.

(i) **Pre-processing:** The input is a corpus where each document is a training example. This is cleaned by removal of punctuation, accents, non-words, meaningless symbols and extra white spaces. Numbers may be removed or converted to words. Separator tokens may be used in case of employing methods where each sentence must be identified separately. All the text is converted to lower-case.

(ii) **Low-dimensional Representation Generation:** Each document is converted to a vector with low-dimensionality (10-768 features).

(iii) **Optimal Representative Subset Selection:** The documents that best span the low-dimensional space are identified using a discriminative approach. The number of documents has to be specified by the user.

(iv) **Hand-labeling/ Fast training:** The selected documents are labeled in accordance with the classification task and used to train the classification model.

3.2 Algorithms

3.2.1 Low-dimensional Representation

(i) **Multinomial Topic Factor-based weight vectors (from the baseline):** A K-topic model represents each document (given as vector of token counts) in the dataset as a sample from multinomial distribution

$$x_i \sim \text{MN}(w_{i1}\theta_1 + \ldots + w_{iK}\theta_K, m_i),$$

where $\theta_j \in \mathbb{R}^p$ is the j-th topic vector (same for all the documents), p is the number of tokens in the vocabulary, $w_{ij} \in \mathbb{R}$ is the weight of j-th topic in i-th document, m_i is the length of x_i

Parameters θ_j and w_{ij} are estimated using MAP with Dirichlet priors:

$$w_{ij} \sim \text{Dir}(1/K)$$
$$\theta_j \sim \text{Dir}(1/(Kp))$$

Vector of topic weights $w_i = (w_{i1}, \ldots, w_{iK})$ is used as low-dimensional representation of the document x_i[7].

(ii) **Sentence-level RoBERTa:** RoBERTa-base model was used to encode the documents into dense vectors of size 768 each. The model is trained on NLI and STSb datasets, uses mean pooling over words and encodes 830 sentences per second on V100 GPU [9].

(iii) **Sentence-level DistilBERT:** DistilBERT-base model was used to encode the documents into dense vectors of size 768 each. The model is trained on NLI and STSb datasets, uses mean pooling over words and encodes 4000 sentences per second on V100 GPU [9].

(iv) **Sentence-level GloVe6B:** GloVe6B model was used to encode the documents into dense vectors of size 300 each. It is trained on Wikipedia 2014 and Gigaword 5 (6 billion tokens, 400k vocabulary), uses mean pooling over words and encodes 34000 sentences per second on V100 GPU [9].

(v) **Universal Sentence Encoder:** It encodes sentences into embedding vectors that specifically target transfer learning to other NLP tasks, thus, the embeddings generated can be used for multiple-tasks. There are two encoder models - transformer with self-attention and deep averaging network that computes unigram and bigram embeddings first and then averages them to get a single embeddings. The encoded vectors have 512 dimensions [10].

(vi) **t-SNE dimension reduction on BoW:** t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction algorithm. First it constructs a distribution over pairs of objects in the initial high-dimensional space in such a way that a pair of close objects has higher probability than a pair of less similar objects. Then, it constructs a similar distribution in the low-dimensional space

and minimizes the KL divergence between these two distributions. The objects that were close in the initial space remain close in the lower-dimensional space [8].

(vii) **PCA dimension reduction on BoW:** PCA or principal component analysis identifies patterns in data based on the correlation between features. PCA aims to find the directions of maximum variance in high-dimensional data (along orthogonal axes called principal components) and projects it onto a new subspace with equal or fewer dimensions than the original one. This new subspace includes the top-k principal components if the dimensions after reduction are chosen to be k.

(viii) **UMAP dimension reduction on BoW:** Uniform Manifold Approximation and Projection (UMAP) is a dimensionality reduction technique that can be used for general non-linear dimension reduction. The manifold is modeled with a fuzzy topological structure. The reduced embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure.

(ix) **NMF dimension reduction on BoW:** The logic for dimensionality reduction with Non-negative matrix factorization (NMF) is to take the $m \times n$ data and to decompose it into two matrices of dimensions $m \times features$ and $features \times n$ respectively. The $features$ will be the reduced dimensions.

3.2.2 Representative Subset Selection

(i) **D-Optimal Design Subset Selection (from baseline):** D-Optimal design is an iterative greedy algorithm that on each step selects one more observation that maximizes the increase of information determinant $D_t = |X_t^T X_t|$, where $X_t = (x_{i1}, \ldots, x_{it})^T$ is the matrix of selected observations[4].

Information determinant on the step $t+1$ can be written as

$$D_{t+1} = |X_t^T X_t + x_{t+1}^T x_{t+1}| = D_t \left(1 + x_{t+1^T} \left(X_t^T X_t\right)^{-1} x_{t+1}\right)$$

The observation on step $t+1$ is selected as

$$x_{t+1} = \arg\max_{x \notin X_t} x^T \left(X_t^T X_t\right)^{-1} x$$

(ii) **Random Pick:** As the name suggests, in this method the required number of documents are selected at random [11].

(iii) **K-means Clustering-based selection:** Given a set of documents encoded into vectors $(x_1, x_2, \ldots, x_n)$ where each document is a d-dimensional vector, the k-means clustering algorithm aims to partition the n documents into k clusters $C_1, C_2, \ldots, C_k$ by minimizing the intra-cluster distances[12]. The algorithm starts at $t = 1$ with k centroids (means) $(m_1^{(t)}, m_2^{(t)}, \ldots, m_k^{(t)})$ and proceeds by alternating between two steps:

Assignment step: Each document is assigned to the cluster with the nearest centroid (with Euclidean distance as the distance measure).

$$C_i^{(t)} = \{x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \; \forall j, 1 \leq j \leq k\}$$

Update step: The centroids are recalculated based on the documents in the cluster.

$$m_i^{(t+1)} = \frac{1}{\left|C_i^{(t)}\right|} \sum_{x_j \in C_i^{(t)}} x_j$$

The algorithm converges when the assignments no longer change. Next, from each cluster, one document is selected in order to get a set of k distant documents.

(iv) **Greedy Farthest Point Sampling using Kullback-Liebler Divergence or KL Divergence:** KL Divergence or relative entropy is a measure of how one probability distribution is different from another. The idea is to identify the documents for which the probability distributions across features are maximally different from each other. This method is inspired from the relative entropy-based subset selection method [11]. In my case, the vectors are dense, continuous, and real-valued. Hence,

the algorithm from the paper is not applicable as is and has very long execution time. The algorithm proceeds as follows:

- Fit a known distribution to each observation. In my case, since most of the representations have normal distribution, I fit the same to each observation using `scipy` module's `norm.fit` method and obtain corresponding mean and standard deviation.
- Calculate KL Divergence between each pair of observations using:

$$KL(p,q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

  KL divergence is asymmetric, but from observation, $KL(p,q)$ is close to $KL(q,p)$. In order to use KL-divergence as a distance measure (and for creating distance matrix), I sum $KL(p,q)$ and $KL(q,p)$.
- Use greedy farthest point sampling, as described in Algorithm 1, using the distance matrix obtained in the previous step as the input.

Algorithm 1: Greedy Farthest Point Sampling

Input: Distance matrix 'D[0...n-1, 0...n-1]' of size nxn, number of observations k;
Result: Array 'indices[0...k-1]' of size k
Initialization: indices[0...k-1] = 0;
//select two points at maximum distance;
indices[0], indices[1] = $\arg\max_{i,j} D[i,j]$;
while *i<k* **do**
    // select the point which maximizes the minimum distance from already selected points;
    indices[i] = $\arg\max_{j \notin indices}(\min_{idx \in indices}(D[idx, j]))$;
    i = i + 1
end

(v) **Greedy Farthest Point Sampling using two-sample Kolmogorov-Smirnov statistic:** The two-sample Kolmogorov–Smirnov statistic (KS-statistic) quantifies a distance between the empirical distribution functions of two samples. The null distribution of this statistic is calculated under the null hypothesis that the samples are drawn from the same distribution (in the two-sample case). The two input embeddings are sorted and converted to two cumulative diftribution functions $F_1(x)$ and $F_2(x)$ and the KS-statistic is calculated as:

$$KS(F_1, F_2) = \sup_x |F_1(x) - F(x)|$$

Using above, a pairwise distance matrix is obtained. Then, greedy farthest point sampling, as described in Algorithm 1 can be used to obtain the most distant points that are representative of the subset.

(vi) **Greedy Farthest Point Sampling using cosine distance:** Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined to equal the cosine of the angle between them, which is also the same as the inner product of the same vectors normalized to both have length 1. Cosine distance is calculated by subtracting cosine similarity from 1. The cosine distance between two vectors or embeddings A and B is given by

$$D_{cos}(A,B) = 1 - \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| \times ||\vec{B}||}$$

Using above, a pairwise distance matrix is obtained. Then, greedy farthest point sampling, as described in Algorithm 1 can be used to obtain the most distant points that are representative of the subset.

(vii) **Reconstruction Loss Minimization with Sparsity Regularization:** This is based on a similar intuition as using an auto-encoder for feature selection[13]. Here, I attempt to reconstruct the entire dataset using a weight matrix while applying sparsity regularization on the weights in order to limit

the amount of data that can be used for reconstruction of the original dataset (effectively disallowing learning of identity function). This allows identification of the most relevant observations that can reconstruct the entire dataset. The steps are given below.

- Set X = transpose of complete dataset. It has d features and N observations The shape of Y is $d \times N$.
- Minimize reconstruction objective with sparsity regularization[14] (regularization term is the sum of L2 norms of rows of weight matrix W (size: $N \times N$) multiplied by λ- the regularization coefficient):

$$\phi(W) = \frac{1}{2}||Y - YW||_F^2 + \lambda||C||_{2,1}$$

- Select rows/observations with higher L2 norm ($||C_i||_2$).

4 Experimental Evaluation

4.1 Data

In order to evaluate and compare the different methods, I used two datasets: executive orders[15] and 20 News Groups[16]. These are described in Table 1.

Table 1: Dataset Description

Characteristic	Executive Orders	20 News Groups
Classification Type	Binary	Multi-label (20 labels)
Class Imbalance	17.9% positive	Roughly balanced
Same distribution of labels in train & eval sets?	Yes	Yes
Full training set size	8447	14682
Evaluation set size	2100	3674

4.2 Methodology

Selecting optimal subsets: Each dataset was partitioned into training and test set in the ratio 80:20 with stratified sampling in order to ensure same distribution of labels. Each low-dimension document representation from Section 3.2.1 was used in combination with every subset selection method from Section 3.2.2 (except Random Pick) to select training examples from the training sets. The number of training samples selected for each method combination are 100, 400, 800, 1200, 1800, 2400, 3000, 4000, and 5000. The test set is not seen by the subset selection algorithms.

Model, training and evaluation: For both datasets, Multinomial Naive Bayes was chosen as the model for comparison experiments owing to its time-efficiency and good performance. The training and test data consisted of count vectors with 10000 (unigram, bigram, trigram) features in the case of executive orders dataset and 30000 unigram features in the case of 20NewsGroups dataset. Embeddings generated earlier were not used for training, only for subset selection in order to allow for simplistic comparison. After fitting the model on training data, it was evaluated on the test set.

Evaluation metrics: I used accuracy and F1 score for both datasets and additionally, area under ROC for executive orders dataset. These evaluation measures require label-information, and are helpful for comparison.

4.3 Results and Discussion

For comparing the methods, I plotted AUC (Area under ROC) scores against the number of training samples (sample_size or n). Figure 1 shows AUC for the simple random sample approach, the Taddy approach, and the two strongest word embedding-based methods, DistilBERT minimizing Reconstruction loss and RoBERTa maximizing KL Divergence. Table 2 shows the full table of all methods and their AUCs for each sample size. Figure 2 summarizes the tables, showing which representations and which distance metrics perform best in general. In all cases, it is observed that

the random method, used by nearly every applied researcher, under-performs. The selection methods that generally perform the best are maximizing KL divergence, minimizing reconstruction error and k-means clustering.

In order to understand the differences between the methods, I looked for overlap in the observations selected by each method for the training set when sample size = 1000, shown in a heatmap in Figure 3. It can be observed that very few of the methods produce any significant overlap with any other, indicating that they are all producing meaningfully different training sets.

Lastly, I compared how long each method takes to produce a training set, as shown in Figure 4. It can be seen that there is quite large variation in how long a given method takes to produce a training set, and that the Taddy method performs close to the median. It is around the 53rd percentile for 100 samples and around the 47th percentile for 4000 samples. The method combining DistilBERT and minimizing reconstruction error is about 33% slower than Taddy, though it performs much better. On the other hand, the method combining RoBERTa and minimizing KL Divergence, which similarly outperforms Taddy, requires only 55% as much time.

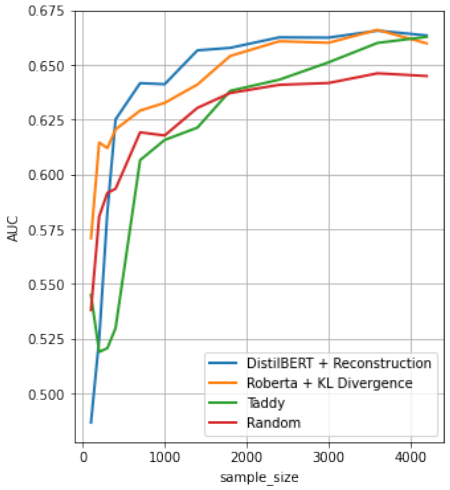

Figure 1: The top two word embedding-based methods substantially outperform both the random sampling and the Taddy approches, achieving both accuracy gains and cost reductions.

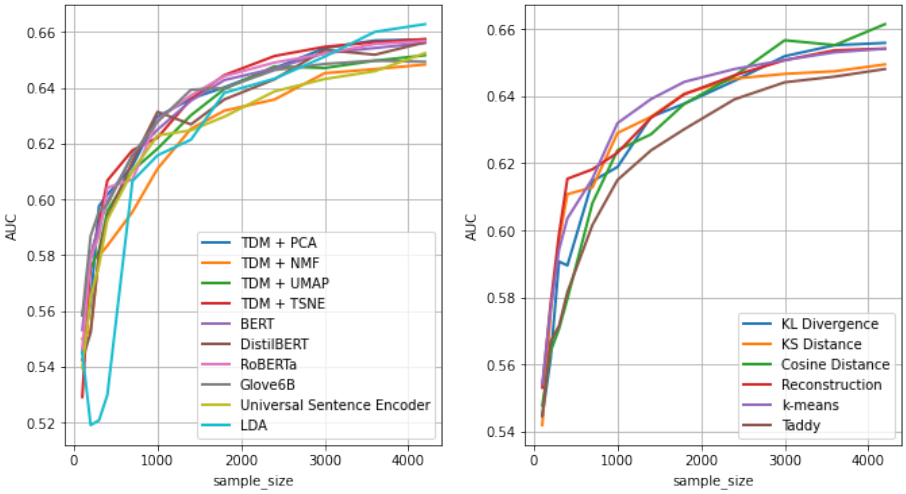

Figure 2: Average AUCs, for each value of sample size, for each (a) representation and (b) subset selection method

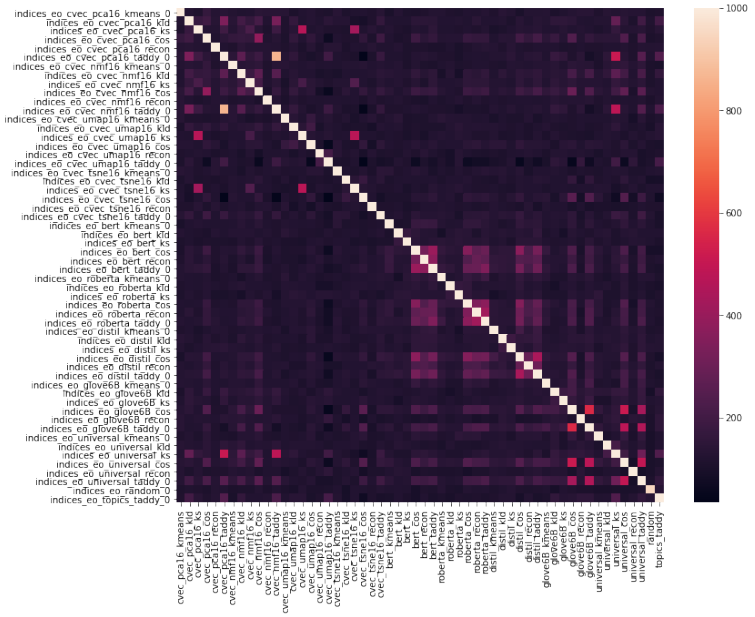

Figure 3: After selecting 1000 training observations, the average number of in-common observations between any two methods is 140 and the median is 126.

Figure 4: For training sets of 100, 500, 2000, and 4000 observations, the 56 methods vary widely in how long they require to select observations. Taddy performs near the median.

Table 2: Full AUC results for all 56 methods

sample_size	100	200	300	400	700	1000	1400	1800	2400	3000	3600	4200
Random Sample	0.54	0.58	0.59	0.59	0.62	0.62	0.63	0.64	0.64	0.64	0.65	0.64
Taddy	0.55	0.52	0.52	0.53	0.61	0.62	0.62	0.64	0.64	0.65	0.66	0.66
roberta_kld	0.57	0.61	0.61	0.62	0.63	0.63	0.64	0.65	0.66	0.66	0.67	0.66
distil_recon	0.49	0.52	0.58	0.63	0.64	0.64	0.66	0.66	0.66	0.66	0.67	0.66
cvec_pca16_kld	0.53	0.54	0.60	0.57	0.61	0.63	0.63	0.63	0.65	0.66	0.66	0.66
cvec_nmf16_kld	0.59	0.56	0.57	0.56	0.60	0.60	0.62	0.62	0.62	0.63	0.64	0.65
cvec_umap16_kld	0.54	0.55	0.57	0.58	0.58	0.61	0.64	0.64	0.65	0.66	0.66	0.66
cvec_tsne16_kld	0.52	0.57	0.57	0.60	0.62	0.62	0.64	0.65	0.65	0.66	0.66	0.66
bert_kld	0.49	0.51	0.59	0.60	0.62	0.63	0.63	0.63	0.64	0.65	0.65	0.65
distil_kld	0.59	0.57	0.60	0.60	0.62	0.60	0.61	0.63	0.64	0.65	0.66	0.66
glove6B_kld	0.57	0.63	0.64	0.62	0.63	0.63	0.65	0.65	0.66	0.65	0.65	0.66
universal_kld	0.49	0.52	0.56	0.57	0.62	0.63	0.64	0.64	0.64	0.65	0.65	0.65
cvec_pca16_ks	0.50	0.54	0.63	0.64	0.61	0.64	0.65	0.65	0.66	0.65	0.65	0.65
cvec_nmf16_ks	0.53	0.60	0.60	0.62	0.62	0.63	0.64	0.64	0.63	0.63	0.64	0.64
cvec_umap16_ks	0.60	0.62	0.61	0.61	0.61	0.63	0.63	0.65	0.65	0.64	0.65	0.65
cvec_tsne16_ks	0.51	0.57	0.60	0.62	0.63	0.62	0.64	0.65	0.65	0.65	0.65	0.66
bert_ks	0.58	0.58	0.59	0.58	0.62	0.63	0.64	0.66	0.67	0.66	0.65	0.65
roberta_ks	0.52	0.57	0.60	0.60	0.62	0.64	0.64	0.65	0.65	0.65	0.65	0.66
distil_ks	0.57	0.56	0.59	0.63	0.61	0.63	0.63	0.64	0.64	0.65	0.65	0.65
glove6B_ks	0.55	0.60	0.60	0.62	0.60	0.64	0.64	0.63	0.64	0.64	0.64	0.64
universal_ks	0.53	0.54	0.54	0.57	0.60	0.61	0.59	0.61	0.63	0.64	0.64	0.64
cvec_pca16_cos	0.56	0.56	0.58	0.61	0.62	0.65	0.64	0.65	0.66	0.66	0.66	0.67
cvec_nmf16_cos	0.52	0.50	0.54	0.55	0.56	0.61	0.63	0.63	0.65	0.66	0.66	0.66
cvec_umap16_cos	0.60	0.58	0.58	0.62	0.64	0.59	0.62	0.64	0.66	0.66	0.65	0.66
cvec_tsne16_cos	0.53	0.57	0.61	0.61	0.62	0.62	0.62	0.65	0.66	0.67	0.66	0.66
bert_cos	0.62	0.62	0.61	0.57	0.64	0.64	0.65	0.64	0.63	0.66	0.65	0.66
roberta_cos	0.50	0.54	0.54	0.58	0.59	0.60	0.61	0.62	0.63	0.64	0.65	0.67
distil_cos	0.51	0.53	0.54	0.53	0.58	0.65	0.62	0.63	0.63	0.65	0.64	0.66
glove6B_cos	0.55	0.58	0.55	0.56	0.63	0.63	0.65	0.64	0.65	0.66	0.66	0.66
universal_cos	0.53	0.58	0.59	0.59	0.60	0.62	0.63	0.63	0.65	0.65	0.66	0.67
cvec_pca16_recon	0.55	0.57	0.60	0.61	0.63	0.64	0.63	0.64	0.64	0.65	0.66	0.66
cvec_nmf16_recon	0.53	0.54	0.60	0.61	0.62	0.62	0.62	0.63	0.63	0.64	0.64	0.64
cvec_umap16_recon	0.58	0.58	0.61	0.62	0.63	0.63	0.63	0.64	0.64	0.64	0.64	0.65
cvec_tsne16_recon	0.54	0.58	0.57	0.61	0.62	0.63	0.64	0.65	0.65	0.65	0.65	0.66
bert_recon	0.58	0.62	0.60	0.61	0.61	0.56	0.60	0.62	0.64	0.65	0.66	0.66
roberta_recon	0.57	0.61	0.62	0.63	0.57	0.62	0.64	0.65	0.65	0.66	0.65	0.65
glove6B_recon	0.56	0.57	0.60	0.62	0.61	0.62	0.64	0.64	0.64	0.65	0.65	0.65
universal_recon	0.57	0.60	0.60	0.61	0.63	0.64	0.64	0.64	0.65	0.65	0.65	0.65
cvec_pca16_kmeans	0.54	0.59	0.60	0.62	0.61	0.64	0.64	0.63	0.64	0.65	0.65	0.65
cvec_nmf16_kmeans	0.56	0.56	0.60	0.60	0.62	0.63	0.64	0.64	0.65	0.65	0.65	0.65
cvec_umap16_kmeans	0.56	0.59	0.58	0.58	0.61	0.62	0.64	0.65	0.65	0.65	0.66	0.65
cvec_tsne16_kmeans	0.53	0.58	0.62	0.60	0.61	0.62	0.64	0.64	0.65	0.65	0.66	0.66
bert_kmeans	0.53	0.58	0.58	0.60	0.62	0.63	0.63	0.65	0.65	0.66	0.65	0.66
roberta_kmeans	0.57	0.57	0.56	0.61	0.61	0.64	0.65	0.64	0.65	0.65	0.65	0.65
distil_kmeans	0.56	0.56	0.60	0.59	0.61	0.64	0.64	0.64	0.64	0.66	0.65	0.66
glove6B_kmeans	0.57	0.59	0.60	0.61	0.62	0.63	0.64	0.65	0.64	0.65	0.65	0.65
universal_kmeans	0.55	0.57	0.60	0.61	0.61	0.64	0.63	0.64	0.65	0.65	0.65	0.66
cvec_pca16_taddy	0.58	0.58	0.56	0.57	0.58	0.58	0.61	0.63	0.63	0.65	0.66	0.65
cvec_nmf16_taddy	0.56	0.57	0.58	0.57	0.55	0.58	0.60	0.63	0.64	0.66	0.64	0.64
cvec_umap16_taddy	0.47	0.54	0.54	0.56	0.60	0.62	0.61	0.62	0.63	0.63	0.64	0.65
cvec_tsne16_taddy	0.54	0.59	0.58	0.60	0.60	0.62	0.64	0.64	0.65	0.64	0.65	0.65
bert_taddy	0.55	0.57	0.59	0.60	0.64	0.65	0.65	0.65	0.65	0.65	0.65	0.65
roberta_taddy	0.55	0.57	0.57	0.59	0.62	0.64	0.65	0.65	0.65	0.66	0.66	0.65
distil_taddy	0.53	0.56	0.58	0.59	0.61	0.63	0.60	0.61	0.64	0.65	0.65	0.65
glove6B_taddy	0.55	0.55	0.58	0.56	0.61	0.62	0.63	0.63	0.64	0.64	0.63	0.64
universal_taddy	0.57	0.58	0.57	0.60	0.61	0.61	0.62	0.61	0.62	0.63	0.63	0.64

5 Conclusions

These analyses have shown that a word embedding approach to training set selection in text-as-data applications introduces substantial gains. It leverages information that researchers have at their disposal but have not utilized: the raw text of the documents. I have also developed an approach for evaluating new proposed methods for the same purpose. My approach here may extend to other data domains as well. Text data applications often require little data compared to, for example, audio or video data, and my approach of testing different representations and subset selection methods may greatly reduce the amount of training data needed to classifying images or signal data (e.g., audio). My analysis falls short, however, in making a singular recommendation. A method that works best for a given dataset or domain may not perform as well for others.

My evaluation is based on the use of labels. In the future, I plan to use metrics that quantify the uncertainty in predictions (closer to 0.5), thus allowing for evaluation of effectiveness of the methods on unlabeled data. Also, my current methods cannot be used to determine the optimal number of examples to be selected from a subset- this will be a consideration in my extension of this work. The chief outcome of the project was the creation of an efficient way to find optimal representative documents and saving the costs incurred in manual selection and hand-labeling, and training a fairly optimal model with very less data.

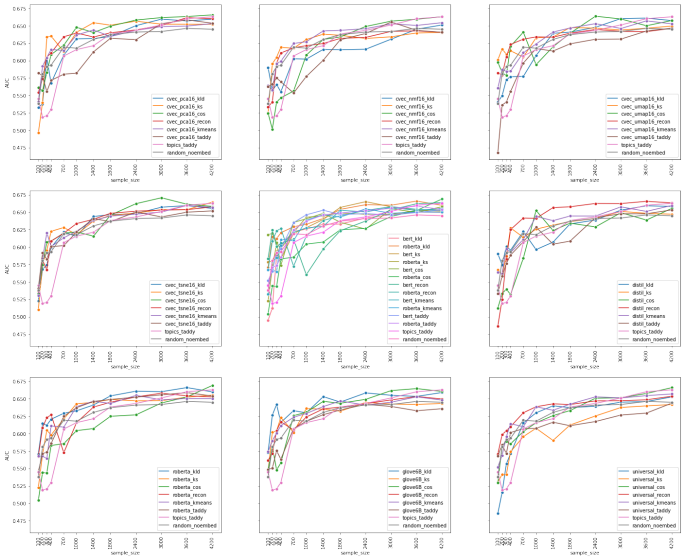

Figure 5: AUC vs sample size for comparing a given representation for all methods

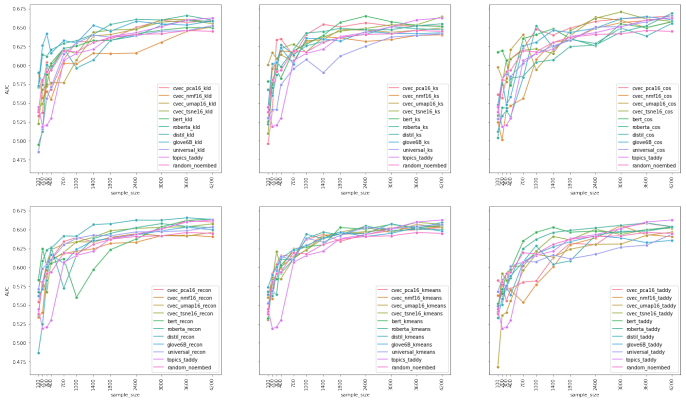

Figure 6: AUC vs sample size for comparing a given method for all representations

References

[1] FitzGerald, Nicholas Michael, Julian He, Luheng Zettlemoyer, Luke. (2018). Large-Scale QA-SRL Parsing. 2051-2060. 10.18653/v1/P18-1191.

[2] Liu, Xiuming Zachariah, Dave Wågberg, Johan. (2018). Robust Semi-Supervised Learning when Labels are Missing at Random.

[3] Blake Miller, Fridolin Linder, Walter R. Mebane Jr. (2020). Active Learning Approaches for Labeling Text: Review and Assessment of the Performance of Active Learning Approaches. Political Analysis (2020) vol. 28:532–551. 10.1017/pan.2020.4

[4] Taddy, Matt. (2012). Measuring Political Sentiment on Twitter: Factor Optimal Design for Multinomial Inverse Regression. Technometrics. 55. 10.1080/00401706.2013.778791.

[5] Alves de Souza, Vinícius Rossi, Rafael Batista, Gustavo Rezende, Solange. (2017). Unsupervised active learning techniques for labeling training sets: An experimental evaluation on sequential data. Intelligent Data Analysis. 21. 1061-1095. 10.3233/IDA-163075.

[6] Chapelle, Olivier; Schölkopf, Bernhard; Zien, Alexander (2006). Semi-supervised learning. Cambridge, Mass.: MIT Press. ISBN 978-0-262-03358-9.

[7] Taddy, Matthew. (2011). On Estimation and Selection for Topic Models.

[8] Van der Maaten, L.J.P.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9:2579-2605, 2008.

[9] Reimers, Nils Gurevych, Iryna. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

[10] Cer, Daniel Yang, Yinfei Kong, Sheng-yi Hua, Nan Limtiaco, Nicole John, Rhomni Constant, Noah Guajardo-Cespedes, Mario Yuan, Steve Tar, Chris Sung, Yun-Hsuan Strope, Brian Kurzweil, Ray. (2018). Universal Sentence Encoder.

[11] Pan, Feng Wang, Wei Tung, Anthony Yang, Jiong. (2005). Finding representative set from massive data. 8 pp.-. 10.1109/ICDM.2005.69.

[12] Daszykowski, Michal Walczak, Beata Massart, D.L.. (2002). Representative subset selection. Analytica Chimica Acta. 468. 91-103. 10.1016/S0003-2670(02)00651-7.

[13] Han, Kai Wang, Yunhe Zhang, Chao Li, Chao. (2018). Autoencoder Inspired Unsupervised Feature Selection. 2941-2945. 10.1109/ICASSP.2018.8462261.

[14] Panda, Rameswar Das, Abir Roy-Chowdhury, Amit. (2016). Embedded sparse coding for summarizing multi-view videos. 191-195. 10.1109/ICIP.2016.7532345.

[15] Available: [https://www.archives.gov/open/dataset-executiveorders.html]

[16] Tom mitchell. Available: [https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups]