X-Class: Text Classification with Extremely Weak Supervision

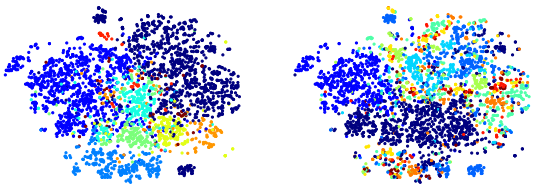
Zihan Wang **Dheeraj Mekala** **Jingbo Shang**
University of California San Diego
{ziw224, dmekala, jshang}@ucsd.edu

Abstract

In this paper, we explore to conduct text classification with *extremely weak supervision*, i.e., only relying on the surface text of class names. This is a more challenging setting than the seed-driven weak supervision, which allows a few seed words per class. We opt to attack this problem from a representation learning perspective—ideal document representations should lead to very close results between clustering and the desired classification. In particular, one can classify the same corpus differently (e.g., based on topics and locations), so document representations must be adaptive to the given class names. We propose a novel framework X-Class to realize it. Specifically, we first estimate comprehensive class representations by incrementally adding the most similar word to each class until inconsistency appears. Following a tailored mixture of class attention mechanisms, we obtain the document representation via a weighted average of contextualized token representations. We then cluster and align the documents to classes with the prior of each document assigned to its nearest class. Finally, we pick the most confident documents from each cluster to train a text classifier. Extensive experiments demonstrate that X-Class can rival and even outperform seed-driven weakly supervised methods on 7 benchmark datasets.

1 Introduction

Weak supervision has been recently explored in text classification to save human effort. Typical forms of weak supervision include a few labeled documents per class (Meng et al., 2018; Jo and Cinarel, 2019), a few seed words per class (Meng et al., 2018, 2020a; Mekala and Shang, 2020; Mekala et al., 2020), and similar open-data (Yin et al., 2019).Though much weaker than a fully annotated corpus, these forms still require non-trivial, corpus-specific knowledge from experts. For example, nominating seed words requires experts to consider their relevance to not only the desired classes but also the input corpus; To acquire a few labeled documents per class, unless the classes are balanced, one needs to sample and annotate a much larger number of documents to cover the minority class.



(a) NYT-Topics (b) NYT-Locations

Figure 1: Visualizations of News using Average BERT Representations. Colors denote different classes.

In this paper, we focus on *extremely weak supervision*, i.e., only relying on the surface text of class names. This setting is much more challenging than the ones above, and can be considered an almost-unsupervised text classification.

We opt to attack this problem from a representation learning perspective—ideal document representations should lead to very close results between clustering and the desired classification. Recent advances in contextualized representation learning using neural language models have demonstrated the capability of clustering texts to domains with high accuracy (Aharoni and Goldberg, 2020). Specifically, a simple average of token representations is sufficient to group documents about the same domain together. However, the same corpus could be classified using various criteria, such as topics, locations, and sentiments. As visualized in Figure 1, such class-invariant representations separates topics well but mixes up locations. It becomes a necessity to make document representations adaptive to the user-specified class names.

We propose a novel framework X-Class to con-

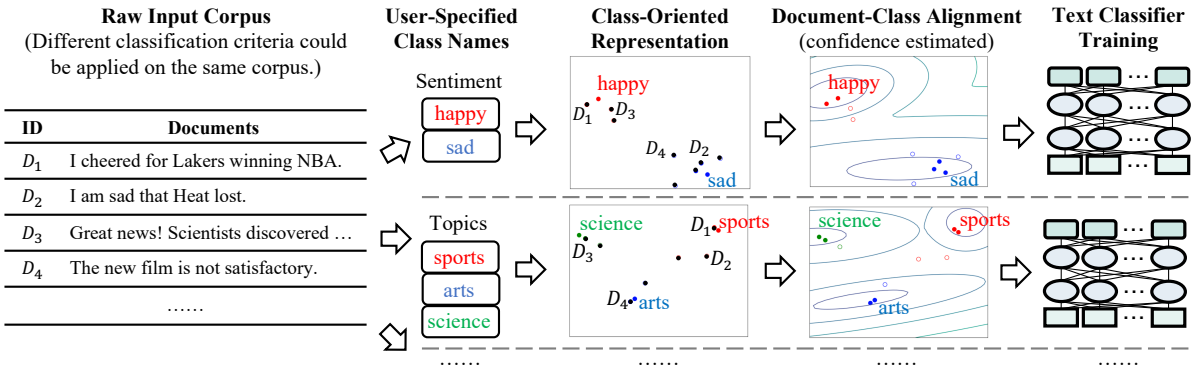


Figure 2: An overview of our X-Class. Given a raw input corpus and user-specified class names, we first estimate a class-oriented representation for each document. And then, we align documents to classes with confidence scores by clustering. Finally, we train a supervised model (e.g., BERT) on the confident document-class pairs.

duct text classification with extremely weak supervision, as illustrated in Figure 2. Specifically, we first estimate comprehensive class representations by incrementally adding the most similar word to each class and recalculating its representation. Following a tailored mixture of class attention mechanisms, we obtain the document representation via a weighted average of contextualized word representations. These representations are based on pre-trained neural language models, and they are supposed to be in the same latent space. We then adopt clustering methods (e.g., Gaussian Mixture Models) to group the documents into K clusters, where K is the number of desired classes. The clustering method is initialized with the prior knowledge of each document assigned to its nearest class. In this way, we can easily align the final clusters to the classes. In the end, we pick confident documents from each cluster to form a pseudo training set, based on which we can train any document classifier. In our implementation, we use BERT as both the pre-trained language model and the text classifier. It is also worth mentioning that on 7 benchmark datasets, X-Class rivals and even outperforms existing weakly supervised methods, which have access to at least 3 seed words per class.

Our contributions are summarized as follows.

- We advocate an important but not-well-studied problem of text classification with extremely weak supervision.
- We develop a novel framework X-Class to attack this problem from a representation learning perspective. It estimates high-quality, class-oriented document representations based on pre-trained neural language models so that the confident clustering examples could form pseudo training set for any document classifiers to train on.
- We show that on 7 benchmark datasets, X-Class achieves comparable and even better performance than existing weakly supervised methods that require more human effort.

Reproducibility. We will release both datasets and codes on Github[1].

2 Preliminaries

In this section, we formally define the problem of text classification with extremely weak supervision. And then, we brief on some preliminaries about BERT (Devlin et al., 2019), Attention (Luong et al., 2015) and Gaussian Mixture Models.

Problem Formulation. The extremely weak supervision setting confines our input to only a set of documents $D_i, i \in \{1, ..., n\}$ and a list of class names $c_j, j \in \{1, ..., k\}$. The class names here are expected to provide hints about the desired classification objective, considering that different criteria (e.g., topics, sentiments, and locations) could classify the same set of documents. Our goal is to build a classifier to categorize a (new) document into one of the classes based on the class names.

Compared with seed-driven weakly supervised text classification, our setting here is much more challenging. The seed-driven weak supervision requires carefully designed label-indicative keywords. Keywords can concisely define what a class represents, however, they require human experts to understand the corpus extensively. One of our motivations is to relax such a strict requirement for human effort. Interestingly, our proposed method using extremely weak supervision can offer com-

[1] https://github.com/ZihanWangKi/XClass

parable and even better performance than the seed-driven methods in our experiments.

BERT. BERT is a pre-trained masked language model with a transformer structure. It takes one or more sentences as input, breaks them up into word-pieces, and generates a contextualized representation for each word-piece. To handle long documents in BERT, we apply a sliding window technique. To retrieve representations for words, instead of word-pieces, we average a word's word-pieces representation. It has been widely adopted in a large variety of NLP tasks as backbones. Therefore, in our work, we will utilize BERT for two purposes: (1) representations for words in the documents and (2) the supervised text classifier.

Attention. Attention mechanisms assign weights to a sequence of vectors, given a context vector (Luong et al., 2015). It first estimates a hidden state $\tilde{h}_j = K(h_j, c)$ for each vector h_j, where K is a similarity measure and c is the context vector. Then, the hidden states are transformed into a distribution via a softmax function. In our work, we use attentions to assign weights to representations, which we then average them accordingly.

Gaussian Mixture Model. Gaussian Mixture Model (GMM) is a traditional clustering algorithm. It assumes that each cluster is generated through a Gaussian process. Given an initialization of the cluster centers and the co-variance matrix, it iteratively optimizes the point-cluster memberships and the cluster parameters following an Expectation–Maximization framework. Unlike K-Means, it does not restrict clusters to have a ball-like shape. Therefore, we will apply GMM to obtain clusters based on our document representations.

3 Our X-Class Framework

As shown in Figure 2, our X-Class framework contains three modules: (1) class-oriented document representation estimation, (2) document-class alignment through clustering, and (3) text classifier training based on confident labels. Algorithm 1 is an overview, and we will introduce them in detail in further sections.

3.1 Class-oriented Document Representation

Ideally, we wish to have some document representations such that clustering algorithms can find k clusters very similar to the k desired classes. Aharoni and Goldberg (2020) has demonstrated that contextualized token representations generated by BERT can preserve the domain (i.e., topic) information of documents. Specifically, it averages contextualized token representations in each document as document representations, which they observed very similar among the documents from the same topic. This observation motivates us to "classify" documents by topics in an unsupervised way.

Algorithm 1: Class-Oriented Document Representation Estimation

Input: n documents D_i, k class names c_j, max number of iterations T, and attention mechanism set $\mathcal{M}$
Output: Document representations $\mathbf{E}_i$.
Compute $\mathbf{t}_{i,j}$ (contextualized token rep.)
Compute $\mathbf{s}_w$ for all words (Eq. 1)
// class rep. estimation
for $j = 1 \dots k$ **do**
  $\mathcal{K}_j \leftarrow \langle\, c_j \,\rangle$
  for $i = 2 \dots T$ **do**
    Compute $\mathbf{x}_j$ based on $\mathcal{K}_j$ (Eq. 2)
    $w = \arg\max_{w \notin \mathcal{K}_j} sim(\mathbf{s}_w, \mathbf{x}_j)$
    Compute $\mathbf{x}'_j$ based on $\mathcal{K}_j \oplus \langle w \rangle$
    // consistency check
    if $\mathbf{x}'_j$ *changes the words in* $\mathcal{K}_j$ **then**
      break
    else
      $\mathcal{K}_j \leftarrow \mathcal{K}_j \oplus \langle w \rangle$
// document rep. estimation
for $i = 1 \dots n$ **do**
  for *attention mechanism* $m \in \mathcal{M}$ **do**
    Rank $D_{i,j}$ according to m
    $r_{m,j} \leftarrow$ the rank of $D_{i,j}$
  Rank $D_{i,j}$ according to $\prod_m r_{m,j}$
  $r_j \leftarrow$ the final rank $a_j \leftarrow 1/r_j$
  $\mathbf{E}_i \leftarrow \frac{\sum_j a_j \cdot \mathbf{t}_{i,j}}{\sum_j a_j}$

Unfortunately, topics are never the only criterion to classify documents. For example, as shown in Figure 1, such document representations work well for topics (i.e., sports, arts, and science) but work poorly for locations (i.e., Canada, France, and Italy). Hence, we want to incorporate information from the given class names and obtain *class-oriented document representations*.

We propose to estimate the document representations and class representations based on pre-trained neural language models. In our implementation, we use BERT as an example. For each document, we want its document representation similar to the class representation of its desired class. We break up this module into two parts, (1) class represen-

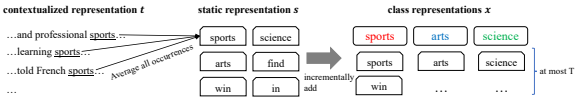


Figure 3: Overview of Our Class Rep. Estimation.

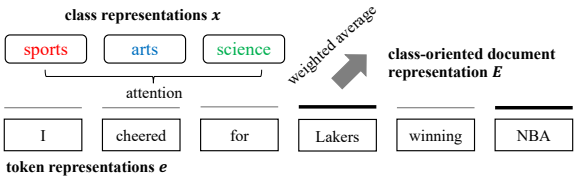


Figure 4: Overview of Our Document Rep. Estimation.

tation estimation and (2) document representation estimation.

Class Representation Estimation. To understand the semantics of the user-specified classes, inspired by those seed-driven weakly supervised methods, we argue that a number of keywords per class would be enough. Intuitively, the class name could be the first keyword we can start with. We propose to incrementally add new keywords to each class to enrich our understanding. Figure 3 shows an overview of our class representation estimation.

First, for each word, we obtain its *static representation* via averaging the contextualized representations of all its occurrences in the input corpus. Formally, we define the static representation of a word w, $\mathbf{s}_w$, as

$$\mathbf{s}_w = \frac{\sum_{D_{i,j}=w} \mathbf{t}_{i,j}}{\sum_{D_{i,j}=w} 1} \tag{1}$$

where $D_{i,j}$ is the j-th token in the document D_i and $\mathbf{t}_{i,j}$ is its contextualized token representation. Ethayarajh (2019) adopted a similar strategy of estimating a static representation using BERT. Such static representations are used as anchors to initialize our understanding of the classes.

A straightforward way to enrich the class representation is to take a fixed number of words similar to the class name and average them to get a class representation. However, it suffers from two issues: (1) setting the same number of keywords for all classes may hurt the minority classes, and (2) a simple average may shift the semantics away from the class name itself. As an extreme example, when the 99% of documents are talking about *sports* and the rest 1% are about *politics*, it is not reasonable to add as many keywords as *sports* to *politics*—it will diverge the *politics* representation.

To address these two issues, we propose to iteratively find the next keyword for each class and recalculate the class representation each iteration by a weighted average on all the keywords found. We will stop this iterative process when the new representation is not consistent with the previous one. In this way, different classes will have a different number of keywords adaptively. Specifically, we define a comprehensive representation $\mathbf{x}_c$ for a class c as a weighted average representation based on a ranked list of keywords $\mathcal{K}_c$. The top-ranked keywords are expected to have more similar static representations to the class representation. Assuming that the similarities follow Zipf's laws distribution (Powers, 1998), we define the weight of the i-th keyword as $1/i$. That is,

$$\mathbf{x}_c = \frac{\sum_{i=1}^{|\mathcal{K}_c|} 1/i \cdot \mathbf{s}_{\mathcal{K}_{c,i}}}{\sum_{i=1}^{|\mathcal{K}_c|} 1/i} \tag{2}$$

For a given class, the first keyword in this list is always the class name. In the i-th iteration, we will retrieve the out-of-list word with the most similar static representation to the current class representation. We then calculate a new class representation based on all the $i+1$ words. We will stop this expansion if we already have enough (e.g., $T = 100$) keywords, or the new class representation cannot yield the same set of top-i keywords in our list. In our experiments, some classes indeed stop before reaching 100 keywords.

Document Representation Estimation. Intuitively, the content of each document should stick to its underlying class. For example, in the sentence "I cheered for Lakers winning NBA", its content covers *sports* and *happy* classes, but never spans over *arts*, *politics*, and *sad*. Therefore, we assume that each word in a document is similar to its desired class's representation or unrelated to all classes. Based on this assumption, we upgrade the simple average of contextualized token representations (Aharoni and Goldberg, 2020) to a weighted average. Specifically, we follow the popular attention mechanisms to assign weights to the tokens based on their similarities to the class representations. Figure 4 shows an overview of our document representation estimation.

We propose to employ a mixture of attention mechanisms to make it more robust. For the j-th token in the i-th docuemnt $D_{i,j} = w$, there are two possible representations: (1) the contextualized token representation $\mathbf{t}_{i,j}$ and (2) the static representation of this word $\mathbf{s}_w$. The contextualized

representations disambiguate words with multiple senses by considering the context, while the static version accounts for outliers that may exist in documents. Therefore, it is reasonable to use either of them as the token representation $\mathbf{e}$ for attention mechanisms. Given the class representations $\mathbf{x}_c$, we define two attention mechanisms:

- **Significance**: $h_{i,j} = \max_c\{\cos(\mathbf{e}, \mathbf{x_c})\}$. It captures the maximum similarity to one class. This is useful for detecting words that are specifically similar to one class, such as *NBA* to *sports*.
- **Relation**: $h_{i,j} = \cos\left(\mathbf{e}, \text{avg}_c\{\mathbf{x}_c\}\right)$ which is the similarity to the average of all classes. This ranks words by how related it is to the general set of classes in focus.

Combining 2 choices of $\mathbf{e}$ and 2 choices of attention mechanisms totals 4 ways to compute each tokens's attention weight. We further fuse these attention weights in an unsupervised way. Instead of using the similarity values directly, we rely on the rankings. Specifically, we sort the tokens decreasingly for each attention mechanism based on similarities to obtain a ranked list. Following previous work (Mekala and Shang, 2020; Tao et al., 2018), we utilize the geometric mean of these ranks for each token and then form a unified ranked list. Like class representation estimation, we follow Zipf's law and assign a weight of $1/r$ to a token ranked at the r-th position in the end. Finally, we can obtain the document representation $\mathbf{E}_i$ from $\mathbf{t}_{i,j}$ following these weights.

Detailed steps in class-oriented document representations can be found in Algorithm 1.

3.2 Document-Class Alignment

One straightforward idea to align the documents to classes is simply finding the most similar class based on their representations. However, document representations not necessarily distribute ball-shaped around the class representation—the dimensions in the representation can be correlated freely.

To address this challenge, we leverage the Gaussian Mixture Model (GMM) to help capture the co-variances for the clusters. Specifically, we set the number of clusters the same as the number of classes k and initialize the cluster parameters based on the prior knowledge that each document D_i is assigned to its nearest class L_i, as follows.

$$L_i = \arg\max_c cos(\mathbf{E}_i, \mathbf{x}_c) \tag{3}$$

We use a tied co-variance matrix across all clusters since we believe classes are similar in granularity. We cluster the documents while remembering the class each cluster is initialized to. In this way, we can align the final clusters to the classes.

Considering the potential redundant noise in these representations, we, following the experience in topic clustering (Aharoni and Goldberg, 2020), also apply principal component analysis (PCA) for dimension reduction. By default, we fix the PCA dimension $P = 64$.

3.3 Text Classifier Training

The alignment between documents and classes can produce high-quality pseudo labels for the documents in the training set. To generalize such knowledge to unseen text documents, we can train a supervised model based on (parts of) these pseudo labels. This is a classical noisy training scenario (Angluin and Laird, 1987; Goldberger and Ben-Reuven, 2017). Since we know how confident we are on each instance (i.e., the posterior probability on its assigned cluster in GMM), we further select the most confident ones to train a text classifier (e.g., BERT). By default, we set a confidence threshold $\delta = 50\%$, i.e., the top 50% instances are selected for classifier training.

4 Experiments

We conduct extensive experiments to show and ablate the performance of X-Class.

4.1 Compared Methods

We compare with two seed-driven weakly supervised methods. **WeSTClass** (Meng et al., 2018) generates pseudo documents via word embeddings of keywords and employees a self-train module to get the final classifier. **ConWea** (Mekala and Shang, 2020) utilizes pre-trained neural language models to make the weak supervision contextualized. In our experiments, we will feed at least 3 seed words per class to these two.

We also compare with **LOTClass** (Meng et al., 2020b), which could work under the extremely weak supervision setting. The original paper experiments mostly rely on class names but requires a few keywords to elaborate on some difficult classes. In our experiments, we only feed the class names to LOTClass.

We denote our method as **X-Class**. To further understand the effects of different modules, we have two ablation versions. **X-Class-Rep** refers to the prior labels L_i derived based on class-oriented

Table 1: An overview of our 7 benchmark datasets. They cover various domains and classification criteria. We also estimate the imbalance factor of a dataset by the ratio of its largest class's size to the smallest class's size.

	AGNews	**20News**	**NYT-Small**	**NYT-Topic**	**NYT-Location**	**Yelp**	**DBpedia**
Corpus Domain	News	News	News	News	News	Reviews	Wikipedia
Class Criterion	Topics	Topics	Topics	Topics	Locations	Sentiment	Ontology
# of Classes	4	5	5	9	10	2	14
# of Documents	120,000	17,871	13,081	31,997	31,997	38,000	560,000
Imbalance	1.0	2.02	16.65	27.09	15.84	1.0	1.0

Table 2: Evaluations of Compared Methods and X-Class. Both micro-/macro-F_1 scores are reported. WeSTClass and ConWea consume at least 3 seed words per class. Supervised provides a kind of upper bound. We are not able to re-run WeSTClass and ConWea on DBpedia due to the large size.

Model	**AGNews**	**20News**	**NYT-Small**	**NYT-Topic**	**NYT-Location**	**Yelp**	**DBpedia**
Supervised	93.99/93.99	96.45/96.42	97.95/95.46	94.29/89.90	95.99/94.99	95.7/95.7	98.96/98.96
WeSTClass	82.3/82.1	71.28/69.90	91.2/83.7	68.26/57.02	63.15/53.22	81.6/81.6	81.1/ N/A
ConWea	74.6/74.2	75.73/73.26	95.23/90.79	**81.67/71.54**	85.31/83.81	71.4/71.2	N/A
LOTClass	**86.89/86.82**	73.78/72.53	78.12/56.05	67.11/43.58	58.49/58.96	87.75/87.68	86.66/85.98
X-Class	84.8/84.65	**81.36/80.6**	**96.67/92.98**	80.6/69.92	**90.5/89.81**	**88.36/88.32**	**91.33/91.14**
X-Class-Rep	77.92/77.03	75.14/73.24	92.13/83.94	77.85/65.38	86.7/87.36	77.87/77.05	74.06/71.75
X-Class-Align	83.1/83.05	79.28/78.62	96.34/92.08	79.64/67.85	88.58/88.02	87.16/87.1	87.37/87.28

document representation. **X-Class-Align** refers to the labels obtained after document-class alignment.

We present the performance of supervised models, serving as a kind of upper-bound for X-Class. Specifically, **Supervised** refers to a BERT model cross-validated on the training set with 2 folds (matching our confidence selection threshold).

4.2 Datasets

Weak supervision for text classification has been extensively explored recently. However, different datasets were used for model comparison, even for the same researcher. Therefore, in this paper, we pool the most popular datasets to establish a benchmark on weakly supervised text classification. Table 1 provides an overview. Our 7 selected datasets cover different text sources (e.g., news, reviews, and Wikipedia articles) and different criteria of classes (e.g., topics, locations, and sentiment).

- **AGNews** (Zhang et al., 2015) (used in WeSTClass and LOTClass) is for topic categorization in news articles from AG's corpus.
- **20News** (Lang, 1995)[2] (used in WeSTClass and ConWea) is for topic categorization in the 20 Newsgroups dataset.
- **NYT-Small** (Meng et al., 2018) (used in WeSTClass and ConWea) is for topic categorization in New York Times news articles.
- **NYT-Topic** (Meng et al., 2020a) (used in (Meng et al., 2020a)) is another (larger) dataset collected from New York Times news articles for topic categorization.
- **NYT-Location** (Meng et al., 2020a) (used in (Meng et al., 2020a)) is the same corpus as NYT-Topic but for location classification.
- **Yelp** (Zhang et al., 2015) (used in WeSTClass) is for sentiment analysis in Yelp reviews.
- **DBpedia** (Zhang et al., 2015) (used in LOTClass) is for topic classification based on titles and descriptions in DBpedia.

4.3 Experimental Settings

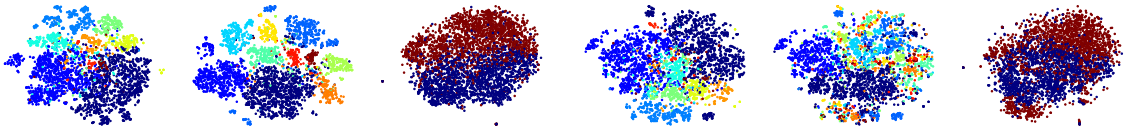
For all X-Class experiments, we report the performance under on fixed random seed. By default, we set $T = 100, P = 64, \delta = 50\%$. For contextualized token representations $\mathbf{t}_{i,j}$, we use the `BERT-base-uncased`. We use the `BERT-base-cased` for better document understanding for supervised model training and follow BERT fine-tuning (Wolf et al., 2019) and leave all hyper-parameters unchanged.

For both WeSTClass and ConWea, we have tried our best to find keywords for the new datasets. For LOTClass, we tune their hyper-parameter *match_threshold* and report the best performance during their self-train process.

4.4 Performance Comparison

From Table 2, one can see that X-Class performs the best among all compared methods. It is only

[2] http://qwone.com/~jason/20Newsgroups/



(a) Our Class-Oriented Document Representations (b) Simple Average of BERT Representations

Figure 5: T-SNE Visualizations of Representations. From left to right: NYT-Topics, NYT-Locations, Yelp.

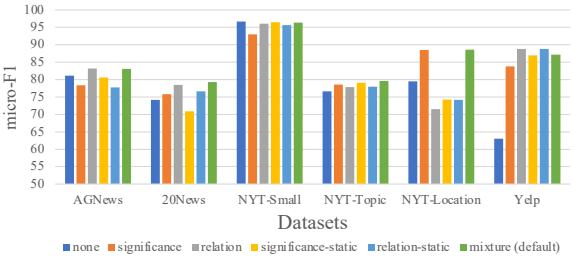


Figure 6: Effects of Different Attention Mechanisms. We report the performance of X-Class-Align to explore their direct effects. "None" refers to the unweighted case.

slightly worse than LOTClass on AGNews and ConWea on NYT-Topics. Note that, WeSTClass and ConWea consume at least 3 carefully designed keywords per class.

The great performance of X-Class-Rep shows success of our representation estimations. The performance on NYT-Topics and NYT-Locations can also justify that our document representations are indeed class-oriented. The improvement of X-Class-Align over X-Class-Rep demonstrates the usefulness of our clustering module. It is also clear that the classifier training is beneficial by comparing X-Class and X-Class-Align.

It is noteworthy that X-Class can approach the supervised upper bound to a small spread, especially on the NYT-Small dataset.

4.5 Necessity and Effect of Attention

In Figure 5, we visualize our class-oriented document representations and the unweighted variants using T-SNE (Rauber et al., 2016). We conjecture that this is because, by default, the BERT representations' most significant feature is topic information. We have also tried using different attention mechanisms in X-Class, and from the results in Figure 6, one can see that using a single mechanism, though not under-performing much, is less stable than our proposed mixture. The unweighted case works well on all four datasets that focus on news topics but not good enough on locations and sentiments.

4.6 GMM vs. K-Means

Table 3 shows that K-Means does not perform as well as GMM on most datasets. This matches our previous analysis as K-Means assumes that all the classes have similar variances in the representation space, while GMM models it as a clustering parameter.

4.7 Hyper-parameter Sensitivity in X-Class

Figure 7 visualized the performance trend w.r.t. to the three hyper-parameters in X-Class, i.e., T in class representation estimation, P in document-class alignment, and δ in text classifier training.

Intuitively, a class won't have too many highly relevant keywords. One can confirm this in Figure 7(a) as the performance of X-Class is relatively stable unless T goes too large to 1000.

Choosing a proper PCA dimension could prune out redundant information in the embeddings and improve the running time. However, if P is too small or too large, it may hurt due to information loss or redundancy. One can observe this expected trend in Figure 7(b) on all datasets except for NYT-Locations.

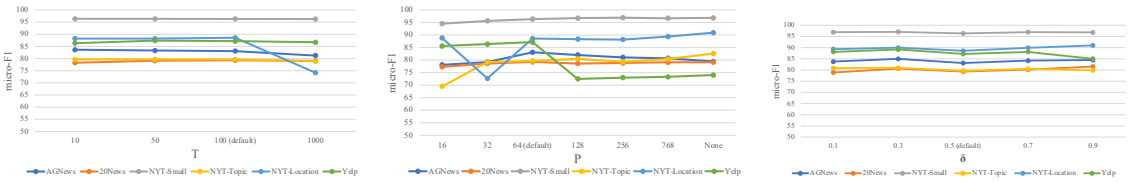
Typically, we want to select a reasonable number of confident training samples for the text classifier training. Too few training samples (i.e., too large δ) would lead to insufficient training data. Too many training samples (i.e., too small δ) would lead to too noisy training data. Figure 7(c) shows that $\delta \in [0.3, 0.9]$ is a good choice on all datasets.

5 X-Class for Hierarchical Classification

There are two straightforward way to extend X-Class for hierarchical classification (1) **X-Class-End**: We can give all fine-grained class names as input to X-Class and conduct classification in an end-to-end manner; and (2) **X-Class-Hier**: We can first give only coarse-grained class names to X-Class and obtain coarse-grained predictions. Then, for each coarse-grained class and its predicted documents, we further create a new X-Class classifier based on the fine-grained class names.

Table 3: GMM vs. K-Means in X-Class.

Cluster Method	AGNews	20News	NYT-Small	NYT-Topic	NYT-Location	Yelp
KMeans	81.33/81.24	71.61/72.04	91.95/84.78	70.67/61.5	92.93/91.66	79.44/79.41
GMM	83.1/83.05	79.28/78.62	96.34/92.08	79.64/67.85	88.58/88.02	87.16/87.1



(a) T in Class Rep. Estimation. (b) P in Document-Class Alignment. (c) δ in Text Classifier Training.

Figure 7: Hyper-parameter Sensitivity in X-Class. For T and P, we report the performance of X-Class-Align to explore their direct effects.

We experiment with hierarchical classification on the NYT-Small dataset, which has annotations for 26 fine-grained classes. We also introduce **WeSHClass** (Meng et al., 2019), the hierarchical version of WeSTClass, for comparison. LOTClass is not investigated here due to its poor coarse-grained performance on this dataset. The results in Table 4 show that X-Class-Hier performs the best, and it is a better solution than X-Class-End. We conjecture that this is because the fine-grained classes' similarities are drastically different (a pair of fine-grained classes can much similar than another pair). Overall, we show that we can apply our method to a hierarchy of classes.

Table 4: Fine-grained Classification on NYT-Small. Compared methods use 3 keywords per class. WeSHClass does not have a non-hierarchical version.

Model	Coarse (5 classes)	Fine (26 classes)
WeSTClass	91/84	50/36
WeSHClass	N/A	87.4/63.2
ConWea	95.23/90.79	91/79
X-Class-End	96.67/92.98	86.07/75.30
X-Class-Hier	96.67/92.98	92.66/80.92

6 Related Work

We discuss related work from two angles.

Weakly supervised text classification. Weakly supervised text classification has attracted much attention from researchers (Tao et al., 2018; Meng et al., 2020a; Mekala and Shang, 2020; Meng et al., 2020b). The general pipeline of such work is to generate a set of document-class pairs and then train a supervised model above them. Most previous work utilizes keywords to find such pseudo data for training, which requires an expert who understands the corpus well. In this paper, we show that it is possible to reach a similar, and often better, performance on various datasets without such guidance from experts.

A recent work (Meng et al., 2020b) also studied the same topic — extremely weak supervision on text classification. It follows a similar idea of (Meng et al., 2020a) and further utilizes BERT to query replacements for class names to find keywords for classes, identifying potential classes for documents via string matching. Instead of hard matching, we attack the problem from a representation learning perspective—our learned representations enable a soft manner for X-Class to cluster and align the documents to classes.

BERT for clustering. Aharoni and Goldberg (2020) studied BERT embeddings and showed that clustering document representations obtained from averaging token embeddings from BERT can differentiate different domains. We borrow this idea to improve our document representations through clustering. Our work differs from theirs in that our document representations are guided by the given class names.

7 Conclusion

We propose our method X-Class for extremely weak supervision on text classification, which is text classification with only class names as supervision. X-Class leverages BERT representations to generated class-oriented document presentations, which we then cluster to form document-class pairs, and in the end, fed to a supervised model to train on. We further set up benchmark datasets for this task that covers different data (news and reviews) and various class types (topics, locations, and sentiments). Through extensive experiments, we show the strong performance and stability of our method.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7747–7763. Association for Computational Linguistics.

Dana Angluin and Philip D. Laird. 1987. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 55–65. Association for Computational Linguistics.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Hwiyeol Jo and Ceyda Cinarel. 2019. Delta-training: Simple semi-supervised text classification using pretrained word embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3456–3461. Association for Computational Linguistics.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 331–339. Morgan Kaufmann.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics.

Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 323–333. Association for Computational Linguistics.

Dheeraj Mekala, Xinyang Zhang, and Jingbo Shang. 2020. Meta: Metadata-empowered weak supervision for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Zihan Wang, Chao Zhang, Yu Zhang, and Jiawei Han. 2020a. Discriminative topic mining via category-name guided text embedding. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2121–2132. ACM / IW3C2.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 983–992. ACM.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6826–6833. AAAI Press.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020b. Text classification using label names only: A language model self-training approach.

David M. W. Powers. 1998. Applications and explanations of zipf's law. In *Proceedings of the Joint Conference on New Methods in Language Processing and Computational Natural Language Learning, NeMLaP/CoNLL 1998, Macquarie University, Sydney, NSW, Australia, January 11-17, 1998*, pages 151–160. ACL.

Paulo E. Rauber, Alexandre X. Falcão, and Alexandru C. Telea. 2016. Visualizing time-dependent data using dynamic t-sne. In *18th Eurographics Conference on Visualization, EuroVis 2016 - Short Papers, Groningen, The Netherlands, June 6-10, 2016*, pages 73–77. Eurographics Association.

Fangbo Tao, Chao Zhang, Xiusi Chen, Meng Jiang, Tim Hanratty, Lance M. Kaplan, and Jiawei Han. 2018. Doc2cube: Allocating documents to text cube without labeled data. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 1260–1265. IEEE Computer Society.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3912–3921. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.