Extreme Multi-label Text Classification with Pseudo Label Descriptions

Anonymous ACL submission

Abstract

Extreme multi-label text classification (XMTC) is the task of tagging each document with the relevant labels in a large predefined label space, where the label frequency distribution is often highly skewed. That is, a large portion of labels (namely the tail labels) have very few positive instances, posing a hard optimization problem for training the classification models. The severe data sparse issue with tail labels is more announced in recent neural classifiers, where the embeddings of both the input documents and the output labels need to be jointly learned, and the success of such learning relies on the availability of sufficient training instances. This paper addresses this tough challenge in XMTC by proposing a novel approach that combines the strengths of both traditional bag-of-words (BoW) classifiers and recent neural embedding based classifiers. Specifically, we use a trained BoW model to generate a pseudo description for each label, and apply a neural model to establish the mapping between input documents and target labels in the latent embedding spaces. Our experiments show significant improvements of the proposed approach over other strong baseline methods on benchmark datasets, especially on tail label prediction. We also provide a theoretical analysis for relating BoW and neural models w.r.t. performance lower bound.

1 Introduction

Extreme multi-label text classification (XMTC) is the task of tagging each document with the relevant category labels in a very large predefined label space, in which the number of categories can be from a few thousands to more than a million. It has a wide range of potential applications, such as assigning subject topics to articles, tagging keywords for online shopping recommendation, classifying products for tax purposes, and so on.

The label frequencies in XMTC often follow a power law. That is, a small portion of the labels have a dominating number of training instances (namely head labels), whereas the majority of the labels have very few training instances (namely as tail labels), as illustrated in figure 1. The severe data sparse issue with the tail labels poses a harder optimization problem for XMTC than the classification tasks where the number of categories are much smaller and the label distributions are not as skewed.

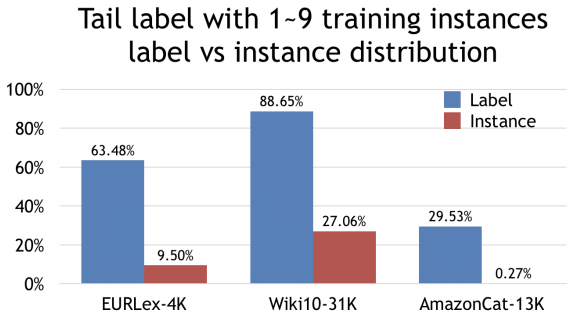

Figure 1: In the skewed distribution of XMTC, tail labels with less than 10 training instances cover a large portion, if not the majority, of the label space, but their training instances cover a only small percentage of the training set.

Traditional classifiers typically use a bag-of-words (BoW) vector to represent each document, whose dimension is the the entire document vocabulary. Each word in the document is treated as an one-hot vector weighted by the tf-idf (Kuang and Xu, 2010) value. Summing over the one-hot vectors of the within-document words yields the BoW representation of the document. Since the BoW vectors of documents are typically high-dimensional and very sparse, we call the classification models based on such representation as BoW classifiers or *sparse classifiers*. Models like DiSMEC (Babbar and Schölkopf, 2017), ProXML (Babbar and Schölkopf, 2019) and PPDSparse (Yen et al., 2017) are recent examples of this kind. BoW classifiers enjoy their simplicity and effectiveness in utilizing

local and global frequencies of words in unlabeled data, such as TF (the within-document frequency of each word) and IDF (the within-corpus inverted document frequency of each word), which is arguably a strength in addressing the severe data sparse issue in tail label prediction (see the rest of this paper for deeper insights). As for their weaknesses or limitations, the word independence assumption in the BoW presentations is clearly unjustified, as word location, ordering and semantic dependencies in context are ignored.

In contrast, recent neural classifiers use neural networks to extract latent features with learnable parameters, and obtain a lower dimensional dense vector as document embedding. Typically, neural classifiers learn the latent label embeddings jointly with the document embeddings in an end-to-end training process. Since the learned document and label representations are both dense vectors, we call such models *dense classifiers* in this paper. Recently, large pre-trained Transformer-based models such as BERT (Devlin et al., 2018) and XL-Net (Yang et al., 2019) have been adapted to XMTC problems to extract expressive contextualized features. Examples include X-Transformer (Chang et al., 2020), APLC-XLNet (Ye et al., 2020) and LightXML (Jiang et al., 2021), which achieved state-of-the-art (SOTA) performance on several benchmark datasets for XMTC evaluation. Despite the desirable expressiveness of such dense classifiers, how much does their successes depend on the availability of a large quantity of labeled training data, and how much does their performance suffers under severe data sparse conditions? These questions have not investigated in sufficient depth so far. Furthermore, the dense classifiers do not make explicit use of the unsupervised statistics like tf-idf term weights, which maybe a potential weakness.

This paper aims to improve the prediction power of XMTC classifiers especially with respect to tail label prediction. Specifically, we propose a framework that combines the strengths of both BoW sparse classifiers and neural dense classifiers. The key idea is 1) using a trained BoW model to select keywords for each label, which yields the pseudo label description, and 2) training a neural model based on the training pairs of input documents and the pseudo descriptions. Step 1 leverages the strength of Bow models in using unsupervised TF-IDF statistics, which would be particularly important under severe data sparse conditions (see section 3 for details) , and Step 2 takes advantage of a neural encoder in learning expressive representations (embeddings) of documents and labels. Our experiments show significant improvements of the proposed approach over other strong baseline methods on benchmark datasets, especially on tail label prediction. A theoretical analysis is also provided for relating BoW and neural models w.r.t. performance lower bound.

2 Related Work

Sparse Classifier: Traditional sparse classifiers rely on the bag-of-words features such as one-hot vector with tf-idf weights, which capture the word importance in a document. An early example is the one-vs-all SVM (Babbar and Schölkopf, 2017; Yen et al., 2017). Later methods leverage the tree structure of the label space for more effective or scalable learning (Prabhu et al., 2018; Jain et al., 2019). Since tf-idf features rely on surface-level word matching, sparse classifiers tend to miss the semantic matching among lexical variants or related concepts in different wording.

Dense Neural Models: Neural models learn to capture the high level semantics of documents with dense feature embeddings. Typically, these methods employ a neural feature extractor such as CNN (Liu et al., 2017) or deep pre-trained Transformer models (Chang et al., 2020; Jiang et al., 2021; Ye et al., 2020) to encode the input document into a fixed vector. Another type of method applies a label-word attention mechanism (You et al., 2018) to calculate label-aware document embeddings, but it requires more computational cost proportional to the document length. In the above neural models, the feature extractor and the label embedding (randomly initialized) are jointly optimized via supervised signals. As we will show later, neither the feature extractor nor the label embedding can be sufficiently optimized for the tail labels whose supervision signals are mostly negative.

Hybrid Approach: X-Transformer (Chang et al., 2020) complements neural model with sparse feature by concatenating tf-idf with the learned cluster-level neural embedding. The prediction is mathematically equivalent to aggregating the scores of the sparse and dense classifiers. However, the two classifiers are independent to each other. Recent works in retrieval combines the sparse and dense features into a unified system for enhanced performance. SPARC (Lee et al., 2019) learns contex-

tualized sparse feature indirectly via Transformer attention. COIL (Gao et al., 2021) leverages lexical matching of contextualized BERT embeddings, and CLEAR (Gao et al., 2004) designs a residual-based loss function for the neural model to learn hard examples from a sparse retrieval model. While we also combine the sparse feature with neural model, the classification setting does not assume predefined label description as in the retrieval setting.

Label Description: When both the document text and label descriptions are available, the ranked-based multi-label classification is similar to the retrieval setting, where the dual encoder models (Gao and Callan, 2021; Xiong et al., 2020; Luan et al., 2020; Karpukhin et al., 2020) have achieved SOTA performance in information retrieval on large benchmark datasets with millions of passages. The Siamese network (Dahiya et al., 2021) for classification encodes both input documents and label descriptions under the assumption that high quality label descriptions are available. Chai et al. (2020) tries a generative model with reinforcement learning to produce extended label description with predefined label descriptions for initialization and uses cross attentions between input text document and output labels. Although their ideas of utilizing label descriptions are attractive, the performance of those systems crucially depends on availability of predefined high-quality label descriptions, which is often difficult to obtain in real-world applications. Instead, the realistic label descriptions are often short, noisy and insufficient for lexicon-matching based label prediction for input documents. How to generate informative label descriptions without human efforts is thus an important problem, for which we offer an algorithmic solution in this paper.

3 Rethinking Sparse and Dense XMTC

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{N_{\text{train}}}\}$ be the training data where $\mathbf{x}_i$ is the input text and $\mathbf{y}_i \in \{0,1\}^L$ are the binary ground truth labels. Given an instance $\boldsymbol{x}$ and a label l, a neural classification system jointly learns the feature embedding $\phi_n(\boldsymbol{x};\theta)$ (parameterized by θ) and the label embedding $\boldsymbol{w}_l$. The system calculates a relevance score (logits) $s_l = \langle \phi_n(\boldsymbol{x}), \boldsymbol{w}_l \rangle$, which is then normalized by the sigmoid function to produce $p_l = \sigma(s_l)$, indicating the probability of the label being true. The probability is optimized by the binary cross entropy (BCE) loss:

$$\mathcal{L}_{\text{BCE}} = -\sum_{l=1}^{L} y_l \log p_l + (1-y_l)\log(1-p_l)$$

The derivative of $\mathcal{L}_{\text{BCE}}$ w.r.t the logits is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}}{\partial s_l} = \begin{cases} p_l - 1 & \text{if } y_l = 1 \\ p_l & \text{otherwise} \end{cases}$$

We analyze the difficulty of both the document and label embedding optimization in the skewed label distribution from the gradient perspective, and show that: 1) the learned document feature lacks for the representation of tail label. 2) the tail label embedding is hard to encode meaningful information by supervised signals alone. We then provide empirical observations together to shed light on our proposed framework.

3.1 Analysis of Document Feature Learning

In multi-label classification, the document feature needs to reflect all the representations of relevant labels. In fact, the gradient describes the relation between feature $\phi_n(\boldsymbol{x})$ and label embedding $\boldsymbol{w}_l$. By the chain rule, the gradient of $\mathcal{L}_{\text{BCE}}$ w.r.t the document feature is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}(y_l, p_l)}{\partial \phi_n(\boldsymbol{x})} = \begin{cases} (p_l - 1)\boldsymbol{w}_l & \text{if } y_l = 1 \\ p_l \boldsymbol{w}_l & \text{otherwise} \end{cases}$$

By optimizing parameters θ of feature extractor, the document feature is encourage to remove the information of a negative label, that is:

$$\phi_n(\boldsymbol{x};\theta') \leftarrow \phi_n(\boldsymbol{x};\theta) - \eta p_l \boldsymbol{w}_l$$

where η is the learning rate. Since a tail label has an overwhelming number of negative instances and θ is shared for all the data, the feature extractor is inclined to remove the tail label information, which will be missing in the document representation. In comparison, the sparse feature like tf-idf is unsupervised from corpus statics, which does not suffer from this problem. The feature may still maintain the representation power to separate the tail labels.

3.2 Analysis of Label Feature Learning

When the labels are treated as indices in a classification system, they are randomly initialized and learned from supervised signals. The gradients of $\mathcal{L}_{\text{BCE}}$ w.r.t the label feature is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}(y_l, p_l)}{\partial \boldsymbol{w}_l} = \begin{cases} (p_l - 1)\phi_n(\boldsymbol{x}) & \text{if } y_l = 1 \\ p_l \phi_n(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

The label embedding is updated by:

$$\boldsymbol{w}_l' = \boldsymbol{w}_l + \frac{\eta}{N_{\text{train}}} \sum_{i:y_{il}=1} (1 - p_{il})\phi_n(\boldsymbol{x}_i) - \frac{\eta}{N_{\text{train}}} \sum_{i:y_{il}=0} p_{il}\phi_n(\boldsymbol{x}_i)$$

After the optimization, the label embedding tends to include features from positive instances and exclude features from negative instances. As most of the instances are negative for a tail label, the update is inundated with the aggregation of negative features, making it hard to encode distinctive feature reflecting its identity. Therefore, learning the tail label embedding from supervised signals alone can be very distracting. Although previous works leverage negative sampling to alleviate the problem (Jiang et al., 2021; Chang et al., 2020), we argue that it is important to initialize the label embedding with the label side information.

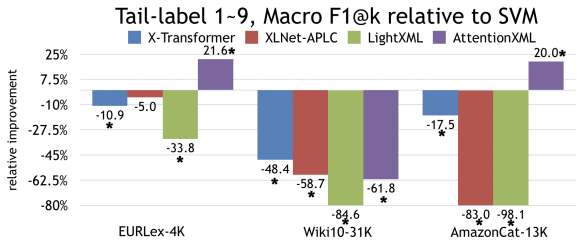

Figure 2: The evaluation of tail label prediction for SOTA Transformer-based models with macro-averaging F1@k, where $k = 19$ for Wiki10-31K and $k = 5$ for the other datasets. The figure shows the relative improvement to the SVM baseline with $*$ indicating the significance for $p < 0.01$.

3.3 Observations and Proposed Idea

Obs. 1: sparse classifier has an advantage on tail label prediction. We evaluate the classification performance of tail labels with less than 10 training instances by the macro-averaging F1 metric described in section 6. In figure 2, the dense models are compared with a linear SVM baseline trained from tf-idf feature, and the relative improvement is reported. The 3 deep Transformer models except for AttentionXML underperform the sparse classifier on all datasets, with $*$ indicating the significance (Yang and Liu, 1999) for $p < 0.01$. AttentionXML performs the best in two datasets, but it utilizes the more expensive label-word attention that doesn't encode the document into a fixed representation, which we don't seek to improve it in this paper.

Obs. 2: provided label descriptions are noisy. The provided label descriptions are shown in the left of table 1, which are usually very short, noisy and insufficient to be related to the document with lexical matching. As an example in Wiki10-31K, the label text *phase4* (with only 1 training instance) is hard even for human to understand the meaning without more context or definitions specific to the document. With generated pseudo label description (explained in the next part), we can understand it is about medical testing phases with keywords *trial, clinical, drug ...* extracted. Although not all the keywords can provide rich semantics to complement the original label text, they may serve as a context for the label to make it more distinguishable, i.e. *responsibility, reconciliation* for label *child care*.

Proposed Idea From the analysis of document feature learning and observation 1, the sparse classifier has a strength in tail label prediction. Additionally, the learned label embeddings from a sparse classifier can be interpreted as importance of words in the corpus vocabulary for the label. In this way, we can extract the top k keywords from the label embedding as the pseudo label description. From the analysis of label feature learning, it is important to provide additional label side information for label embedding. In observation 2, the neural model may benefit from the extracted keywords that leverage the knowledge from the sparse classifier and generalize better with neural embeddings of the keywords.

In section 4, we will explain the proposed framework that leverages the pseudo label description; in section 5, we provide theoretical analysis to relate the performance of our model with respect to the sparse classifier; in section 6, we demonstrate the generalization power of our model by experiments on benchmark datasets.

4 Keyword-selected Dual Encoder

In this section, we proposed a dual encoder framework that leverages the pairs of input document with the pseudo label descriptions extracted from a sparse classifier. We call it the Keyword-selected Dual Encoder (KDE).

Sparse classifier: We train a linear SVM model with tf-idf feature $\phi_t(\boldsymbol{x})$ as the sparse classifier:

$$f_{\text{sparse}}(\boldsymbol{x}, l) = \sigma(\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_l \rangle)$$

The learned word embedding w_{li} denotes the im-

Dataset	Label Text	Top Keywords
EURLex-4K	child care (1) scientific profession (3) mining extraction(5)	care child parent upbringing family responsabilitie occupational affordable reconciliation leave reconcile responsibility arise enable need service effective talent ambition charter science confidential 452 access society scientific purposes whose researchers bodies bank 831 issues may central list recalling data recognises 412 extractive industries extracting drilling contract soda awarded mineral mining ash 531 corporation through dialogue tailings communicate tenneco 1600 webb value
Wiki10-31K	phase4 (1) eco-construction (3) bookmaking (5)	trials clinical protection personal directive processed data trial drug phase eu processing patients sponsor controller legislation regulation art investigator study solar building cob passive glazing cordwood thermal timber sewage natural autonomous roof clay window insulation cistern septic shade bale reduce tex book artist spine binding bind bookbinding bookbinder nickname scroll signature someone raise glue sew texas fold cloth endpaper typeset
Amazoncat-13K	booklet envelopes (1) sweater dresses (3) farming (5)	cabinetmaking booklet upcoming excerpts building centers cabinet book joinery kreg mark basics develop provided content produce allow entire customer covers favorable slung impress dye sweater worn multi dress pair space super sure palette knit jean low favorite jacket brown soft crops acres farmers sustainable hobby grit profitable livestock farms discusses small issue national gardening soil readers designed magazine traditional homegrown

Table 1: Example of provided label descriptions with training frequency in parenthesis for the benchmark datasets, and the top 20 extracted label keywords from the sparse classifier. For illustration purpose, we manually highlight keywords that can enrich the original label text.

portance of the word i for label l. The top k important keywords selected for label l is denoted as $\boldsymbol{z}_l$, where k is a hyperparameter.

KDE: We first train a base neural classifier with feature extract $\phi_n(\boldsymbol{x})$ (initialized by BERT) and the label embeddings $\boldsymbol{u}_l$ (randomly initialized):

$$f_{\text{dense}}(\boldsymbol{x}, l) = \sigma(\langle \phi_n(\boldsymbol{x}), \boldsymbol{u}_l \rangle) \tag{1}$$

Then we fine-tune the model on pseudo label descriptions by sharing the encoder $\phi_n(.)$ for both the text ($\boldsymbol{x}$) and keywords ($\boldsymbol{z}_l$):

$$f_{\text{KDE}}(\boldsymbol{x}, l) = \frac{\sigma(\langle \phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_l) \rangle) + f_{\text{dense}}(\boldsymbol{x}, l)}{2} \tag{2}$$

The predicted probability is an average of two terms, where $\sigma(\langle \phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_l) \rangle)$ leverages the document and label semantic matching that benefits the tail label prediction, and $f_{\text{dense}}(\boldsymbol{x}, l)$ is a dense classifier that is better optimized for head labels with sufficient training data.

Inference: we can directly use f_{KDE} or a weighted sum of sparse and KDE classifiers:

$$f_{\text{final}}(\boldsymbol{x}, l) = (1 - \lambda) f_{\text{sparse}}(\boldsymbol{x}, l) + \lambda f_{\text{KDE}}(\boldsymbol{x}, l)$$

We set $\lambda = \frac{1}{2}$ as a simple design choice.

Learning: f_{dense} and f_{sparse} are optimized with the BCE loss. For $f_{\text{KDE}}(\boldsymbol{x}, l)$, calculating $\phi_n(\boldsymbol{z}_l)$ for all labels both expensive and prohibitive by memory limit, so we use negative sampling for in-batch optimization. Specifically, the label representations are calculated over a subset of labels $\mathbb{S}_b = \mathbb{P}_b \cup \mathbb{N}_b$, where $\mathbb{P}_b$ contains all the positive labels for the instances in the batch and $\mathbb{N}_b$ are the negative labels with hard negatives and uniform random sampling. For each instance, the hard negative labels are the false positive predictions by the SVM model. The objective for dual encoder is:

$$\min \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{p \in \boldsymbol{y}_i^+} \log f_{\text{KDE}}(\boldsymbol{x}, p) + \sum_{n \in \mathbb{S}_b \setminus \boldsymbol{y}_i^+} \log(1 - f_{\text{KDE}}(\boldsymbol{x}, n)) \right) \tag{3}$$

where $\boldsymbol{y}_i^+$ is the set of positive labels for instance i.

5 Theoretical Analysis on Performance

In the theoretical analysis, we demonstrate that KDE achieves a lower bound performance as the sparse classifier, given the selected keywords are important and the sparse classifier can separate the positive from the negative instances with non-trivial margin.

Let $\phi_t(\boldsymbol{x})$ be the normalized tf-idf feature vector of text with $\|\phi_t(\boldsymbol{x})\|_2 = 1$. The sparse label embeddings $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L\}$ satisfies $\|\boldsymbol{w}_l\|_2 \leq 1, w_{li} > 0$. In fact, label embeddings can be transformed to satisfy the condition without changing the prediction rank. Let $\boldsymbol{z}_l$ be the selected keywords and $\boldsymbol{v}_l$ be the

sparse keyword embedding with $v_{li} = w_{li}$ if i is keyword and 0 otherwise. We define the keyword importance and error margin which are major terms affecting the performance bound.

Definition 1. *For label l and $\delta \geq 0$, the sparse keyword embedding is δ-bounded if $\langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_l \rangle \geq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_l \rangle - \delta$.*

Definition 2. *For two labels p and n, the error margin is the difference between the predicted scores $\mu(\phi(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle$.*

We state the theorem below with additional assumptions and proofs in appendix A.

Theorem 3. *Let $\phi_t(\boldsymbol{x})$ and $\phi_n(\boldsymbol{x})$ be the sparse and dense (dimension d) feature, $\boldsymbol{w}_l$ be the label embedding and $\boldsymbol{z}_l$ be the δ-bounded keywords. For a positive label p, let $\mathbb{N}_p = \{n_1, \ldots, n_{M_p}\}$ be a set of negative labels ranked lower than p. The error margin $\epsilon_i = \mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_{n_i})$ and $\epsilon = \min(\{\epsilon_1, \ldots, \epsilon_{M_p}\})$. An error event $\mathcal{E}_i$ occurs when $\mu(\phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_p), \phi_n(\boldsymbol{z}_{n_i})) \leq 0$. The probability of any such error happening satisfies*

$$P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq 4M_p \exp(-\frac{(\epsilon - \delta)^2 d}{50})$$

When $(\epsilon - \delta) \geq 10\sqrt{\frac{\log M_p}{d}}$, the probability is bounded by $\frac{1}{M_p}$.

Discussion: An error event occurs when the sparse model makes a correct prediction but the neural model doesn't. If the neural model avoids all such errors, the performance should be at least as good as the SVM model, and Theorem 3 gives a bound of that probability.

The term δ measures the importance of selected keywords (smaller the more important), the term ϵ term measures the error margin of the correctly predicted positive and negative pairs by the sparse model. The theorem states that the model achieves a lower bound performance as sparse classifier if the keywords are informative and error margin is non-trivial.

Theoretical analysis vs. empirical experiments: 1) the bound in the theoretical analysis is very loose as the proof doesn't consider contextualized semantic embedding of Transformer models, which could produce more meaningful features based on the context and generalize better. 2) we select top k keywords for every label as a hyper-parameter for efficient batch encoding of label keywords instead of the δ-bounded keywords. 3) by mining hard negatives as inputs to KDE , the neural model may generalize better to avoid the mistakes by sparse classifier.

6 Experiments

6.1 Experimental Setting

Datasets We conduct our experiments on 3 benchmark datasets: EURLex-4K (Loza Mencía and Fürnkranz, 2008), Wiki10-31K (Zubiaga, 2012) and AmazonCat-13K (McAuley and Leskovec, 2013). The statistics of the datasets are shown in Table 2. For the tail label evaluation, we consider the labels with $1 \sim 9$ training instances, because we assume the absolute number of training instance reflects the difficulty of optimization across datasets. The tail labels covers 63.48%, 88.65% and 30% of training labels for the 3 datasets respectively. We obtain the datasets from the Extreme classification Repository[1] and a unstemmed version of EURLex-4K from the APLC-XLNet github[2].

Implementation Details For the sparse model, since the public available BoW feature doesn't have a vocabulary dictionary, we generate the tf-idf feature by ourselves. We tokenize and lemmatize the raw text with the Spacy (Honnibal and Montani, 2017) library and extract the tf-idf feature with the Sklearn (Pedregosa et al., 2011) library, with unigram whose idf is $>= 2$ and $<= 70\%$. For the dense model, we fine-tune a 12 layer BERT-base model with different learning rates for the BERT encoder, BERT pooler and the classifier. The learning rates are $(1e{-}5, 1e{-}4, 1e{-}3)$ for Wiki10-31K and $(5e-5, 1e-4, 2e-3)$ for the rest datasets. We used learning rate $1e-5$ for fine-tuning the KDE model. For the pseudo label descriptions, we concatenate the provided label description with the generated the top 20 keywords. The final length is truncated up to 32 after BERT tokenization.

Evaluation Metrics We use the micro-averaging $P@k$ metric to evaluate the overall system performance and the macro-averaging $F1@k$ metric to evaluate the tail label prediction, because both precision and recall are important for tail label predictions (and $F1@k$ is a harmonic average of both).
micro-averaging P@k: The micro-averaging P@k metric is widely used to evaluate a ranked list of

[1] http://manikvarma.org/downloads/XC/XMLRepository.html
[2] https://github.com/huiyegit/APLC_XLNet.git

Dataset	N_{train}	N_{test}	F	$\bar{L}_d$	L	p^l_{tail}	p^d_{tail}
EURLex-4K	15,539	3,809	34,932	5.30	3,956	63.48%	9.50%
Wiki10-31K	14,146	6,616	189,795	18.64	30,938	88.65%	27.06%
AmazonCat-13K	1,186,239	306,782	200,000	5.04	13,330	29.53%	0.27%

Table 2: N_{train} and N_{test} are the number of training and testing instances respectively. F is the tf-idf feature size. $\bar{L}_d$ is the average number of labels per document. L is the number of labels. For tail labels with $1 \sim 9$ training instances, p^l_{tail} is percentage of tail labels and p^d_{tail} is the percentage of training instances covered by the tail labels.

	EUR-Lex			Wiki10-31K			AmazonCat-13K		
Methods	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
DisMEC	83.21	70.39	58.73	84.13	74.72	65.94	93.81	79.08	64.06
PfastreXML	73.14	60.16	50.54	83.57	68.61	59.10	91.75	77.97	63.68
Parabel	82.12	68.91	57.89	84.19	72.46	63.37	93.02	79.14	64.51
Bonsai	82.30	69.55	58.35	84.52	73.76	64.69	92.98	79.13	64.46
AttentionXML	85.12	72.80	61.01	86.46	77.22	67.98	95.53	82.03	67.00
X-Transformer	85.46	72.87	60.79	87.12	76.51	66.69	95.75	82.46	67.22
BERT-APLC	85.54	72.68	60.59	88.54	77.21	67.43	94.49	79.74	64.46
XLNet-APLC	86.83	74.34	61.94	88.99	78.79	69.79	94.56	79.78	64.59
LightXML	86.12	73.87	61.67	87.39	77.02	68.21	94.61	79.83	64.45
tf-idf+SVM (ours)	83.44	70.62	59.08	84.61	74.64	65.89	93.20	78.89	64.14
BERT (ours)	84.72	71.66	59.12	87.60	76.74	67.03	94.26	79.63	64.39
KDE	86.13	73.82	62.22	88.52	78.13	68.98	96.13	**82.70**	**67.52**
KDE+SVM	**87.98**	**75.81**	**63.62**	**89.10**	**80.24**	**70.49**	**96.25**	81.90	65.20

Table 3: Comparisons between KDE and the SOTA sparse and dense classifiers. The results are evaluated with the micro-averaging P@5 metrics, with highest values in bold. We report our baseline sparse (tf-idf+SVM) and dense (BERT) classifiers with KDE and KDE+SVM (average of KDE and SVM predicted score).

predicted labels:

$$P@k = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}_{\boldsymbol{y}_i^+}(p_i) \tag{4}$$

where p_i is the i-th label in the predicted ranked list $\boldsymbol{p}$ and $\mathbb{1}_{\boldsymbol{y}_i^+}$ is the indicator function. The metric score is averaged over all the test instances.

macro-averaging F1@k: The $F1$ metric is a harmonic average of prediction (P) and recall (R):

$$\text{F1} = 2\frac{P \cdot R}{P + R} \tag{5}$$

The precision and recall for a predicted ranked list $\boldsymbol{p}$ are computed by $\text{P} = \frac{\text{TP}}{\text{TP+FP}}, \text{R} = \frac{\text{TP}}{\text{TP+FN}}$ according to the confusion matrix in table 4.

	l in $\boldsymbol{y}_i$	l not in $\boldsymbol{y}_i$
l in $\boldsymbol{p}_i$	True Positive(TP^i_l)	False Positive(FP^i_l)
l not in $\boldsymbol{p}_i$	False Negative(FN^i_l)	True Negative(TN^i_l)

Table 4: Confusion Matrix for instance i and label l given the ranked list $\boldsymbol{p}_i$.

Given N_{test} instances and L labels, the macro-average computes the scores on individual category first (F1_l), and then take an average over all the categories ($\text{F1} = \frac{1}{|\mathbb{L}|}\sum_{i\in\mathbb{L}} \text{F1}_l$), which reflects label level performance of the methods.

For micro-averaging P@k, we choose $k = 1, 3, 5$ the same as in other works. For macro-averaging F1@k, we choose $k = 19$ for Wiki10-31K because it has an average of 18.64 labels and $k = 5$ for the rest datasets.

Baselines Our method is compared with the state-of-the-art baselines including both the sparse and dense classifiers. Specifically, DisMEC (Babbar and Schölkopf, 2017), PfastreXML (Jain et al., 2016), Parabel (Prabhu et al., 2018), Bonsai (Khandagale et al., 2019) belong to the sparse classifiers, and X-Transformer (Chang et al., 2020), APLC-XLNet (Ye et al., 2020) and LightXML (Jiang et al., 2021), AttentionXML (You et al., 2018) belong to the dense

classifier. X-Transformer, LightXML, and APLC-XLNet employ pre-trained Transformers to encode a document into a fixed embedding. We report the single model performance (chosen from their papers) with BERT-large for X-Transformer, BERT-base for LightXML and XLNet-base for APLC-XLNet. The AttentionXML utitizes label-word attention to generate label-aware word embedding instead of a fix document representation. We provide an additional linear SVM model with our extracted tf-idf features as a sparse baseline, and BERT-base classifier as a dense baseline.

6.2 Experimental Results and Discussions

The performance of model evaluated on the micro-averaging P@5 metric is reported in table 3.

Sparse and dense classifiers: The sparse classifiers (first panel) generally underperform the dense models (second panel) under the micro-averaging evaluation metric. Our implementation of Linear SVM model and BERT model achieve comparable result with the sparse and dense classifiers respectively, except that the AttentionXML and X-Transformer achieve better result in Amazoncat-13K. The reason could be that the Amazoncat-13K product categorization relies more on the lexicon matching features, which gives the two methods performance gains.

KDE: Our KDE model is fine-tuned on top of the BERT model with the pseudo label descriptions generated from the SVM model. As shown in the result, KDE has additional gains on both of the classifiers, indicating KDE can: 1) leverage the keyword semantic from sparse model for better generalization, 2) alleviate the difficult of optimization of neural model with scarce data by providing label side information. The KDE model outperforms the baseline models on the Amazoncat-13K dataset, and the KDE+SVM (average of KDE and sparse classifier scores) performs the best on the other two datasets.

6.3 Tail Label Evaluation

Previously, we observe that the pre-trained Transformer-based models underperform the sparse classifier (Linear SVM baseline) on tail label prediction. In this section, we want to test the ability of KDE to generalize over the sparse classifier with the generated pseudo label description for tail label prediction. The experimental results are shown in figure 3, where we report the relative performance of the models under investigation with respect to

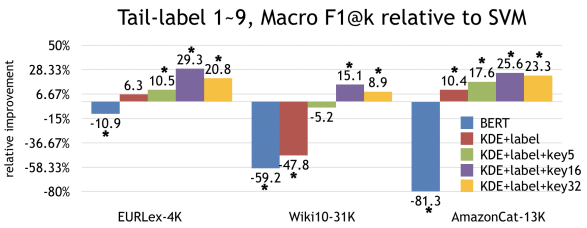

Figure 3: The evaluation of tail label prediction for a dense classifier base (BERT) and our proposed KDE with different settings. The metric is macro-averaging F1@k, where $k = 19$ for Wiki10-31K and $k = 5$ for the other datasets. The figure shows the relative improvement to the SVM baseline with $*$ indicating the significance for $p < 0.01$.

the sparse classifier. Specifically, we include the dense classifier (BERT) without any label side information as a baseline, which underperforms the SVM on all of the datasets.

Only Provided Label Text We fine-tune KDE with only the provided label text, denoted as KDE +label. The model outperforms the sparse classifier on the EURLex-4K and Amazoncat-13K dataset, with the latter one being significant. This is probably because these two datasets has higher quality label text compared with Wiki10-31K, where the label space is both large and noisy.

Pseudo Label Description We fine-tune KDE with pseudo label descriptions, denoted as KDE +label+key$k$, where k is the length of the pseudo label description after BERT tokenization. We observe that with keyword length 16, the model performs the best, which achieves significant improvement on the tail label prediction for all datasets. With larger $k = 32$ in the default setting, the model includes additional "less important" keywords, which may introduce noises and thus lower the performance.

7 Conclusion

In this paper, we analyze the difficulty of optimization of classification systems for XMTC with skewed label distribution. We alleviate the issue with the a trained sparse classifier to generate pseudo label descriptions and propose a keyword-selected dual encoder framework to leverage the input text document with label descriptions. We show the relation of performance between our model and the sparse classifier in the theoretical analysis and demonstrate the effectiveness of our model empirically on the benchmark datasets.

References

Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729.

Rohit Babbar and Bernhard Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351.

Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. 2002. Limitations of learning via embeddings in euclidean half spaces. *Journal of Machine Learning Research*, 3(Nov):441–461.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3163–3171.

Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International Conference on Machine Learning*, pages 2330–2340. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540*.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. *arXiv preprint arXiv:2104.07186*.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, BV Durme, and Jamie Callan. 2004. Complementing lexical retrieval with semantic residual embedding (2020). *URL: http://arxiv. org/abs*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944.

Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. 2021. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *arXiv preprint arXiv:2101.03305*.

William B Johnson and Joram Lindenstrauss. 1984. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai – diverse and shallow trees for extreme multi-label classification.

Qiaoyan Kuang and Xiaoming Xu. 2010. Improvement and application of tf• idf method based on text classification. In *2010 International Conference on Internet Technology and Applications*, pages 1–4. IEEE.

Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. 2019. Contextualized sparse representations for real-time open-domain question answering. *arXiv preprint arXiv:1911.02896*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

Eneldo Loza Mencía and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. page 165–172. Association for Computing Machinery.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 993–1002.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian Davison. 2020. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In *International Conference on Machine Learning*, pages 10809–10819. PMLR.

Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *arXiv preprint arXiv:1811.01727*.

Arkaitz Zubiaga. 2012. Enhancing navigation on wikipedia with social tags.

A Proof of Theorem 3

Notations: Let $\phi_t(\boldsymbol{x})$ be the normalized tf-idf feature vector of text, s.t. $\|\phi_t(\boldsymbol{x})\|_2 = 1$. Let the learned sparse label embeddings be $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L\}$ with $\|\boldsymbol{w}_i\|_2 \leq 1$ and $w_{ij} \geq 0, i \in \{1, \ldots, L\}, j \in \{1, \ldots, V\}$. In fact, since we use a ranking metric, we can always normalize the label embeddings by $\frac{\boldsymbol{w}_l - \min(\{w_{ij}\})}{\max(\{\|\boldsymbol{w}_i - \min(\{w_{ij}\})\|_2\})}$, without changing the prediction rank. Let selected keywords be $\boldsymbol{z}_l$, and $\boldsymbol{v}_l$ be the keyword-selected label embedding with $v_{li} = w_{li}$ if i is keyword and 0 otherwise. Let $\phi_n(\boldsymbol{x}) \in \mathbb{R}^d$ be the dense neural embedding.

Assumptions: Similar to Luan et al. (2020), we treat neural embedding as fixed dense vector $\boldsymbol{E} \in \mathbb{R}^{d \times v}$ with each entry sampled from a random Gaussian $N(0, d^{-1/2})$, which provides a very loose bound, if not a lower bound, for neural model performance. Then, $\phi_n(\boldsymbol{a}) = \boldsymbol{E}\boldsymbol{a}$ is weighted average of word embeddings whose weights are determined by a sparse vector $\boldsymbol{a}$. According to the famous the *Johnson-Lindenstrauss Lemma* (Johnson and Lindenstrauss, 1984; Ben-David et al., 2002), even if the entries of $\boldsymbol{E}$ are sampled from a random normal distribution, with large probability, $\langle \phi_t(\boldsymbol{x}), \boldsymbol{v} \rangle$ and $\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}\boldsymbol{v} \rangle$ are close.

Lemma 4. *Let $\boldsymbol{v}$ be the δ-bounded keyword-selected label embedding of $\boldsymbol{w}$. For two labels p, n, the error margins satisfy: $|\mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n) - \mu(\phi_t(\boldsymbol{x}), \boldsymbol{v}_p, \boldsymbol{v}_n)| \leq \delta$*

Proof. By the definition of δ-bounded keyword-selected embedding,

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle \tag{6}$$

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_n \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle \tag{7}$$

which is equivalent to

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle \tag{8}$$

$$-\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle \leq -\langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_n \rangle \leq -\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle + \delta \tag{9}$$

Adding equation 8 and equation 9, we obtain

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle + \delta \tag{10}$$

□

Lemma 5. *Let $\phi_t(\boldsymbol{x})$ and $\phi_n(\boldsymbol{x})$ be the sparse and dense (dimension d) feature, $\boldsymbol{w}_l$ be the label embedding and $\boldsymbol{z}_l$ be the δ-bounded keywords. Let p be a positive label and n be a negative label ranked below p be the sparse classifier. The error margin is $\epsilon = \mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n)$. An error event $\mathcal{E}$ occurs when $\mu(\phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_p), \phi_n(\boldsymbol{z}_n)) \leq 0$. The probability $P(\mathcal{E}) \leq 4\exp(-\frac{(\epsilon-\delta)^2 d}{50})$.*

Proof. We first state the **Johnson-Lindenstrauss Lemma** (JL Lemma) (Ben-David et al., 2002):
For vector product states that for any two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^v$, let $\boldsymbol{E} \in \mathbb{R}^{d \times v}$ be a random matrix such that the entries are sampled from a random Gaussian. Then for every constant $\gamma > 0$:

$$P\left(|\langle \boldsymbol{E}\boldsymbol{a}, \boldsymbol{E}\boldsymbol{b} \rangle - \langle \boldsymbol{a}, \boldsymbol{b} \rangle| \geq \frac{\gamma}{2}\left(\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2\right)\right) \leq 4\exp\left(-\frac{\gamma^2 d}{8}\right) \tag{11}$$

Let $\gamma = \frac{2}{5}(\epsilon - \delta)$, $\boldsymbol{a} = \phi_t(\boldsymbol{x})$ and $\boldsymbol{b} = \boldsymbol{v}_p - \boldsymbol{v}_n$. Since $\|\boldsymbol{a}\|_2 = 1$ and $\|\boldsymbol{b}\|_2 \leq (\|\boldsymbol{v}_p\|_2 + \|\boldsymbol{v}_n\|_2)^2 \leq 4$, the JL Lemma gives

$$P\left(|\mu(\phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_p), \phi_n(\boldsymbol{z}_n)) - \mu(\phi_t(\boldsymbol{x}), \phi_t(\boldsymbol{z}_p), \phi_t(\boldsymbol{z}_n))| \geq \epsilon - \delta\right) \tag{12}$$

$$= P\left(|\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n) \rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n \rangle| \geq \epsilon - \delta\right) \tag{13}$$

$$\leq 4\exp(-\frac{(\epsilon - \delta)^2 d}{50}) \tag{14}$$

To complete the proof, we need to show

$$P\left(\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n) \rangle \leq 0\right) \leq P\left(|\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n) \rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n \rangle| \geq \epsilon - \delta\right) \tag{15}$$

An error event occurs when

$$\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle \leq 0 \tag{16}$$

$$\implies \quad \langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle \leq -\epsilon \tag{17}$$

$$\implies \quad |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle| \geq \epsilon \tag{18}$$

$$\implies \quad |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \geq \epsilon - \delta \tag{19}$$

where the equation 19 is derived by Lemma 4:

$$|\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \tag{20}$$

$$= |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle + \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \tag{21}$$

$$\geq |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle| - |\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \tag{22}$$

$$\geq \epsilon - \delta \tag{23}$$

The above shows that the event by equation 19 contains the event by equation 16, which completes the proof.

Note: Luan et al. (2020) showed that there could be tighter bounds, but that doesn't affect our analysis. Our goal is not to derive a tight bound, but to get the relation between the defined terms from the theoretical analysis. □

Proof of **Theorem** 3

Proof. The Lemma 2 shows that

$$P(\mathcal{E}_i) \leq 4\exp(-\frac{(\epsilon_i - \delta)^2 d}{50}) \leq 4\exp(-\frac{(\epsilon - \delta)^2 d}{50}) \tag{24}$$

Apply an union bound on the error events $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{M_p}\}$,

$$P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq \sum_{i=1}^{M_p} 4\exp(-\frac{(\epsilon_i - \delta)^2 d}{50}) \tag{25}$$

$$= 4M_p \exp(-\frac{(\epsilon - \delta)^2 d}{50}) \tag{26}$$

□

When $(\epsilon-\delta)^2 \geq 10\sqrt{\frac{\log M_p}{d}}$, we have $\exp(-\frac{(\epsilon-\delta)^2 d}{50}) \leq \frac{1}{4{M_p}^2}$ and therefore $P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq \frac{1}{M_p}$.