OAK: Enriching Document Representations using Auxiliary Knowledge for Extreme Classification

Shikhar Mohan* [1] **Deepak Saini*** [2] **Anshul Mittal** [1] **Sayak Ray Chowdhury** [3] **Bhawna Paliwal** [3] **Jian Jiao** [2] **Manish Gupta** [1] **Manik Varma** [3]

Abstract

The objective in eXtreme Classification (XC) is to find relevant labels for a document from an exceptionally large label space. Most XC application scenarios have rich auxiliary data associated with the input documents, e.g., frequently clicked webpages for search queries in sponsored search. Unfortunately, most of the existing XC methods do not use any auxiliary data. In this paper, we propose a novel framework, Online Auxiliary Knowledge (OAK), which harnesses auxiliary information linked to the document to improve XC accuracy. OAK stores information learnt from the auxiliary data in a knowledge bank and during a forward pass, retrieves relevant auxiliary knowledge embeddings for a given document. An enriched embedding is obtained by fusing these auxiliary knowledge embeddings with the document's embedding, thereby enabling much more precise candidate label selection and final classification. OAK training involves three stages. (1) Training a linker module to link documents to relevant auxiliary data points. (2) Learning an embedding for documents enriched using linked auxiliary information. (3) Using the enriched document embeddings to learn the final classifiers. OAK outperforms current state-of-the-art XC methods by up to $\sim 5\%$ on academic datasets, and by $\sim 3\%$ on an auxiliary data-augmented variant of LF-ORCAS-800K dataset in Precision@1. OAK also demonstrates statistically significant improvements in sponsored search metrics when deployed on a large scale search engine.

*Equal contribution [1]Microsoft, India [2]Microsoft, USA [3]Microsoft Research, India. Correspondence to: Shikhar Mohan <shikharmohan@microsoft.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Document	Plinth Assemblage
Auxiliary Knowledge (AK)	Mount Meager massif, Geologic formations of British Columbia, Volcanism of British Columbia, Pleistocene volcanism, Beaufort Group, Paleontology in South Africa

NGAME (Ignores AK)	Job Assemblage, Assemblage, Pylon Assemblage, Capricorn Assemblage, Assemblage (art)
OAK (Uses AK)	Pylon Assemblage, Capricorn Assemblage, Job Assemblage, List of Cascade volcanoes, List of volcanoes in Canada

Table 1. OAK leverages auxiliary knowledge pieces (AKPs) for accurate XC. Consider an example from the WikiSeeAlsoTitles dataset. Given a Wikipedia page title, auxiliary knowledge indicates relevant categories. The related Wikipedia pages predicted by OAK are more accurate than the state-of-the-art XC algorithm, NGAME. OAK is able to accurately predict labels related to volcanoes thanks to the information from auxiliary knowledge. Legend: Black (correct), Red (incorrect), Green (correct; unique to OAK)

1. Introduction

eXtreme Classification (XC) is the problem of predicting the most relevant subset of labels for a data point from an extremely large set of labels. XC methods have proved to be effective for several applications like product recommendation (Dahiya et al., 2021b; Medini et al., 2019; Mittal et al., 2021a), document tagging (Babbar & Schölkopf, 2017; Chang et al., 2020; You et al., 2019), search and advertisement (Dahiya et al., 2021b; Jain et al., 2016; Prabhu et al., 2018b), and query recommendation (Chang et al., 2020; Jain et al., 2019). While earlier XC methods used sparse linear models (Babbar & Schölkopf, 2017; Prabhu et al., 2018b) recent ones have been deep-learning based (You et al., 2019; Dahiya et al., 2021b; Mittal et al., 2021a; Dahiya et al., 2021a; Kharbanda et al., 2022; Dahiya et al., 2023a; Jain et al., 2023; Dahiya et al., 2023b; Kharbanda et al., 2023).

For full-text XC datasets such as LF-Wikipedia-500K (Bhatia et al., 2016a), documents are represented using a more detailed description. However, short-text tasks abound in ranking and recommendation applications where data points are user queries or products/webpages represented using only their titles. In such cases, e.g., in the LF-WikiTitles-500K dataset, documents are represented by a 3-5 word textual description such as the name of a product or title of a webpage. The sparse document representation adds to the complexity of XC tasks on such short-text datasets.

However, besides the document text, often, a variety of auxiliary information is available in many domains, e.g., frequently clicked webpages for search queries in sponsored search, previously searched queries for web search query auto-completion, etc. Auxiliary information available from disparate but related tasks often have relevant diverse information that the input document does not, which can be leveraged to provide better predictions. Surprisingly, none of the previous XC methods have leveraged this rich auxiliary information. In this paper, our goal is to check how much accuracy improvements can be obtained for XC tasks by harnessing rich auxiliary information. Table 1 shows an example where usage of auxiliary information improves accuracy for the Wikipedia "See Also" prediction XC task.

Given a document x and a set of K related auxiliary knowledge pieces $\{a_k\}[1..K]$, an obvious approach is to concatenate the document with each of the knowledge pieces and compute embedding of this concatenated sequence as the document representation. If the auxiliary knowledge pieces (AKP) are not known for a document, popular methods from retrieval augmented language modeling (Guu et al., 2020) can be used to first discover relevant auxiliary knowledge pieces from a large pool. However, such a method has a few drawbacks: (1) accuracy may suffer if some AKPs are noisy, and (2) inference latency may increase due to increased length of the concatenated sequence. Another approach to use graph neural network methods like GraphFormers (Yang et al., 2021) and GraphSAGE (Hamilton et al., 2018) to encode the document-AKPs linkage information. However, they involve high storage and computational costs, and cannot leverage the auxiliary data sourced from disparate tasks effectively as observed in our experiments.

In this paper, we propose the Online Auxiliary Knowledge (OAK) classifier to enrich document representations using auxiliary information. Fig. 1 shows the detailed OAK architecture. Training for OAK involves three stages: (1) Linker training (2) OAK pretraining (which involves training the augmentation block as shown in the figure) (3) OAK finetuning. In the first stage, the linker module in OAK is trained to link documents with relevant AKPs from a large pool using an existing XC method. In the second stage, we combine the document embeddings with relevant AKP embeddings via attention-based pooling to get an enriched document representation. The second stage then trains the augmentation block (detailed later) in a Siamese fashion. The goal is to attain higher similarity between enriched document embeddings and relevant label embeddings as against non-relevant label embeddings. The third fine-tuning stage freezes the augmentation block parameters and learns a per-label refinement vector to fine-tune the label embeddings to obtain the final label classifiers.

OAK uses *trainable embeddings* that are completely free to move around to represent each AKP. These AKP representations are learnt *jointly* with the document encoder to optimize for the target XC task.

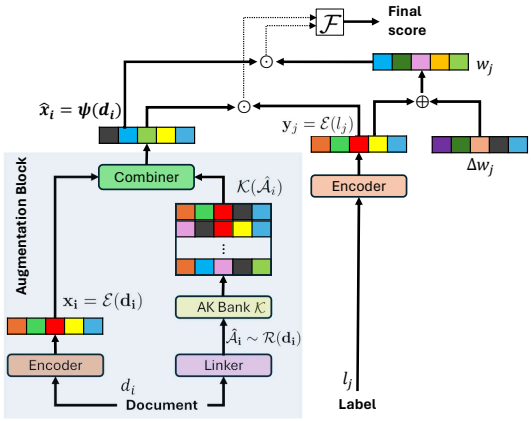

Figure 1. OAK model architecture. For a given document $\mathbf{d}_i$, document representations $\mathbf{x}_i$ are obtained using the encoder $\mathcal{E}$ with parameters θ_E. Linker R returns the indices corresponding to relevant AKPs, $\hat{\mathcal{A}}_i$, which correspond to the embeddings in AK Bank. These embeddings are concatenated and passed through the Combiner module $\mathcal{C}$ to obtain the auxiliary data enriched document representation $\hat{\mathbf{x}}_i$. We refer to the collection of the three modules (Encoder $\mathcal{E}$, AK Bank $\mathcal{K}$ and Combiner module $\mathcal{C}$ with parameters θ_E, θ_K and θ_C resp) as the Augmentation Block, ψ. Same encoder $\mathcal{E}$ is also used to obtain label representations $\mathbf{y}_j$ for a label $\mathbf{l}_j$. Augmentation block parameters are frozen when learning a per-label refinement vector Δw_j to obtain final classifier w_j for label $\mathbf{l}_j$.

This ensures that the OAK augmentation block can use signals from AKPs to get an enriched representation of a document while still adhering to the semantics imposed by the XC task. Further, to avoid large changes in encoder embeddings in stage 2, we introduce encoder regularization. Lastly, motivated by Direct Preference Optimization (DPO) (Rafailov et al., 2023), we add an novel calibration regularization term to the loss function to learn accurate enriched document representations.

On public benchmark datasets for Wikipedia document tagging, suggesting relevant Wikipedia titles and webpage prediction, OAK provides a gain of up to $\sim$ 5%, $\sim$ 1% and $\sim$ 3% respectively in Precision@1 metric over strong XC baselines. On a sponsored search task of matching user queries to advertiser keywords, OAK outperforms current state-of-the-art dense retrievers by 5% in retrieval recall@100. OAK adds minimal computational overhead during training. OAK's inference is efficient leading to an inference time of $<$10 milliseconds, enabling it to be deployed for a sponsored search task to get enhanced representation of a user query in *real-time for online serving*.

Overall, the main contributions of this work are as follows. (1) We propose the usage of semantically rich auxiliary

information associated with documents for diverse and accurate XC. (2) We propose the modular OAK architecture to leverage the AKPs. OAK can execute the joint training of AKP representations on top of any encoder, and exploit the signals from any auxiliary source. We also propose a 3-stage training scheme for OAK which involves linker training, OAK pre-training and OAK finetuning. (3) OAK's novel insight to efficiently increase model capacity by using trainable embeddings to represent AKPs ensures optimal document representation for the target XC task. At the same time, the novel mutual information calibration loss term ensures that the potentially orders of magnitude more parameters can be learnt tractably. (4) OAK leads to state-of-the-art results on several XC tasks including advertiser keyword prediction for user queries, Wikipedia categories and "See Also" prediction, and Webpage prediction by leveraging auxiliary information efficiently. The code will be released publicly upon acceptance of this paper.

2. Related Work

XC: XC is a key paradigm in several areas such as ranking and recommendation. The literature on XC methods is vast (You et al., 2019; Guo et al., 2019; Dahiya et al., 2021b; Mittal et al., 2021a; Saini et al., 2021; Gupta et al., 2023). Early XC methods used fixed (bag-of-words) (Babbar & Schölkopf, 2017; Prabhu et al., 2018b) or pre-trained (Jain et al., 2019) features and focused on learning only a classifier architecture. Recent advances have demonstrated significant gains by using task-specific features obtained from a variety of deep encoders (You et al., 2019; Jiang et al., 2021; Dahiya et al., 2023a). Training is scaled to millions of labels and training points (Dahiya et al., 2021b) by performing encoder pre-training followed by classifier training. A data point is trained only on its relevant labels and a select few irrelevant labels deemed most informative using *negative mining* (Dahiya et al., 2023a). However, none of the existing XC methods make use of any auxiliary information associated with documents, except PINA (Chien et al., 2023) which uses instance correlation signals to learn neighborhood aggregated representations. Our experiments show that OAK outperforms PINA by large margins.

Retrieval Augmented Language Models: REALM (Guu et al., 2020) leverages external knowledge sources to enhance accuracy using Transformer encoders. Based on the input text, first, a retriever selects relevant documents or passages from a large corpus. Then, an encoder concatenates the input text and relevant documents and computes an embedding for the sequence. Retrieval augmentation has also been extended to generation (RAG (Lewis et al., 2020)), but we do not discuss RAG in detail since XC is a classification problem.

Graph Neural Networks in Related Areas: A sizeable body of work exists on using graph neural networks such as graph convolutional networks (GCN) for recommendation. GCN-based methods such as GraphSAGE (Hamilton et al., 2018) and GraphFormers (Yang et al., 2021) learn node representations as functions of node metadata e.g. textual descriptions. This allows the methods to work in zero-shot settings but they still incur the high storage and computational cost of GCNs. Moreover, diminishing returns are observed with an increasing number of layers of the GCN (Chiang et al., 2019; Mittal et al., 2021b). As a baseline method, in this work, we experiment with enriching document representations using document-AKP graphs. Our experiments show that the OAK method offers a far more scalable alternative to GCNs and other popular graph-based architectures in XC settings, significantly reducing the overheads of graph-based learning, yet offering sustained and significant performance boosts in prediction accuracies.

3. Preliminaries/Background

Notation. Consider a dataset containing a set of documents $\mathcal{D}$, Auxiliary Knowledge Pieces (AKPs) $\mathcal{A}$ and labels $\mathcal{L}$. Let $\mathbf{d}_i, \mathbf{l}_j \in \mathcal{X}$ be the textual descriptions of the document i and label j respectively. Thus, the training dataset can be expressed as $T = \{d_i, l_i\}_{i=1}^{N}$. For each document $\mathbf{d}_i \in \mathcal{D}$, there exists a positive label set $\mathcal{L}^+_{\mathbf{d}_i} \subset \mathcal{L}$ such that for every $\mathbf{l}_j \in \mathcal{L}^+_{\mathbf{d}_i}$, $\mathbf{d}_i$ and $\mathbf{l}_j$ are relevant. Similarly, the positive AKP set exists $\mathcal{A}^+_{\mathbf{d}_i}$ for every document $\mathbf{d}_i$ as well. Note that AKP and label sets are the same across training and testing, but document-AKP links are not available at test time.

Task. Before discussing the details of our proposed OAK framework, we recap the fundamentals of the XC task at hand. The objective of an XC is, given a document $\mathbf{d}_i$, retrieve a set of relevant labels $\mathcal{L}^+_{\mathbf{d}_i}$ from a label set $\mathcal{L}$ where $|\mathcal{L}^+_{\mathbf{d}_i}| \ll |\mathcal{L}|$.

Siamese Networks. Several existing XC methods follow a Siamese architecture. Here, a (BERT-based) encoder, denoted by $\mathcal{E} : \mathcal{X} \to \mathcal{S}^{D-1}$, with trainable parameters θ, is used to embed both data point and label text onto the D-dimensional unit sphere $\mathcal{S}^{D-1}$, *i.e.,* the encoder provides unit norm embeddings. During training, a contrastive objective (triplet loss in our case) is used to pull the D-dimensional encoder representations of related documents and labels close as well as push unrelated documents and labels away. Formally, if we have $\mathbf{x}_i = \mathcal{E}(\mathbf{d}_i)$ and $\mathbf{y}_j = \mathcal{E}(\mathbf{l}_j)$, Triplet loss is defined as below.

$$\mathcal{L}_{\text{Trip}}(\theta) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq i} \left[\mathbf{x}_i \cdot \mathbf{y}_j^\top - \mathbf{x}_i \cdot \mathbf{y}_i^\top + \gamma \right]_+ \quad (1)$$

where γ is the margin. Typically training is done in a modular fashion in 2 stages where encoder $\mathcal{E}$ is trained in the first

stage and frozen. The second stage initializes $|\mathcal{L}|$ 1-vs-all classifiers, each corresponding to a label, with the label text encoder representations, and refines them using the triplet loss objective described above. For more details on Siamese architectures and training, please refer to (Dahiya et al., 2023a).

Interestingly, there is an equivalence between Siamese training and the maximization of mutual information between document and label encoder representations as described in the theorem below. Please refer to Appendix C.1 for the proof.

Theorem 1. *Siamese training via triplet loss serves to maximize mutual information between the distributions cast by the document and label encoder representations.*

4. The OAK Approach for XC

OAK Architecture: As depicted in Fig. 1, OAK consists of four components as follows. (1) An encoder $\mathcal{E} : \mathcal{X} \to \mathcal{S}^{D-1}$ with trainable parameters θ_E embeds documents and labels into $\mathcal{S}^{D-1}$ using their textual descriptions. OAK uses a DistilBERT-base (Sanh et al., 2019) encoder as $\mathcal{E}$. (2) A linker module $\mathbf{R} : \mathcal{X} \to P(\mathcal{A})$, where $P(\mathcal{A})$ is the power set of $\mathcal{A}$, has parameters θ_L. It predicts K relevant AKPs $\hat{\mathcal{A}}_i$ for a document $\mathbf{d}_i$. (3) An auxiliary knowledge bank $\mathcal{K}$ consists of $|\mathcal{A}|$ trainable embeddings with parameters θ_K. Formally, $\mathcal{K} \in \mathbb{R}^{M \times D}$, where $\mathcal{K}_j \in \mathbb{R}^D$ corresponds to row j in $\mathcal{K}$ for the j-th AKP. The AK bank stores information learnt from the auxiliary data. It also helps increase the model capacity via additional learnable parameters for AKP representations. (4) A combiner module $\mathcal{C} : \mathbb{R}^{(K+1)\times D} \to \mathcal{S}^{D-1}$ with trainable parameters θ_C fuses the encoder's document representation $\mathbf{x}_i = \mathcal{E}(\mathbf{d}_i)$ and AKP embeddings $\{\mathcal{K}(a_1), \mathcal{K}(a_1), ..., \mathcal{K}(a_K)\} \subset \mathcal{K}$ using cross attention. Here, K is a model hyperparameter that corresponds to the maximum number of AKPs the module can enrich the document representation with. The module is expected to learn higher weights for relevant AKPs and lower for noisy ones (see Figure 2). Overall, OAK training involves learning the set of parameters $\theta = \{\theta_E, \theta_K, \theta_C\}$.

Auxiliary Data Enriched Document Representation: The encoder, AK bank and combiner modules are jointly referred to as the Augmentation Block $\psi : \mathcal{X} \to \mathcal{S}^{D-1}$, which augments the document representation using AKP embeddings to enrich it. Augmented representations are obtained from the combiner module which fuses the trainable embeddings obtained from the AK Bank with the encoder's document representation. Since for document $\mathbf{d}_i$ we have $\mathbf{x}_i = \mathcal{E}(\mathbf{d}_i)$ and $\mathcal{K}(\hat{\mathcal{A}}_i) = \{\mathcal{K}(a_1), \mathcal{K}(a_1), ..., \mathcal{K}(a_K)\}$, we obtain the auxiliary data enriched document representation as

$$\psi(d_i) \stackrel{\text{def}}{=} \mathcal{C}\left(\mathbf{x}_i, \{\mathcal{K}(a_j)\}_{j=1}^{j=K}\right) . \tag{2}$$

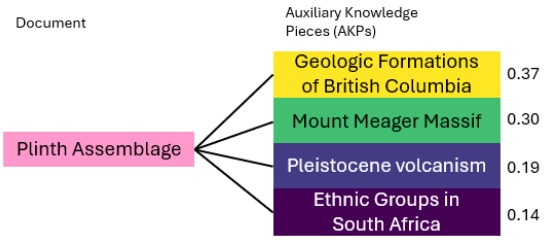

Figure 2. Cross Attention score visualization for the document "Plinth Assemblage" (same as in Table 1). In Table 1, we showed ground truth AKPs, while here we show 3 most relevant AKPs (all of which match with the ground truth) as predicted by the linker module. To demonstrate the combiner's capacity to filter out noise, we also add an incorrect AKP, "ethnic groups in south africa", and it is scored the lowest out of all AKPs.

4.1. Training OAK

Training of OAK is divided into three stages: (1) training the Linker Module $\mathcal{R}$, (2) training the Augmentation Block, and (3) training the label classifiers.

Training the Linker Module. Treating the AKPs as labels, predicting relevant AKPs for documents at test time can be modeled as an XC task. Hence, we use a performant XC method as the linker module, trained on the document to AKP linkage training data.

Training the Augmentation Block using MI Maximization. The Augmentation Block ψ consists of encoder $\mathcal{E}$, AK Bank $\mathcal{K}$, combiner module $\mathcal{C}$. To generate enriched document representations optimized to maximize XC task accuracy, we need to ensure that both $\mathcal{E}$ and ψ provide semantically rich representations, predictive of the labels. This implies that we need to maximize mutual information (1) between enriched document representations from ψ and label representations using $\mathcal{E}$, and (2) between document representations from $\mathcal{E}$ and label representations using $\mathcal{E}$. Formally, we have random variables $X \sim \mathbb{P}[\mathcal{E}(\mathbf{d})]$, $X_A \sim \mathbb{P}[\psi(\mathbf{d})]$, $Y \sim \mathbb{P}[\mathcal{E}(\mathbf{l})]$ and we maximize $I(X_A; Y) + I(X; Y)$. As elaborated in Theorem 1, Triplet loss serves to maximize mutual information between the distributions cast by the two towers of the Siamese network. Hence, we leverage triplet loss to define the MI maximization requirement.

For a document $\mathbf{d}_i$, we sample $\mathbf{k}_p \sim \mathcal{L}_{d_i}^+$. We define $\mathcal{L}_{d_i}^-$ as the set of negative labels $L_{d_i}^-$ obtained using in-batch sampling from the set of labels which are relevant for other documents in the current batch but irrelevant to document $\mathbf{d}_i$. The triplet loss formulation to maximize $I(X_A; Y)$ for a batch of N samples then naturally follows.

$$\mathcal{L}_{\text{Trip},\psi}(\theta) = \sum_{i=1}^{N} \sum_{\mathbf{l}_n \in \mathcal{L}_{d_i}^-} \left[\hat{\mathbf{x}}_i \cdot \mathbf{y}_n^\top - \hat{\mathbf{x}}_i \cdot \mathbf{y}_p^\top + \gamma\right]_+ \tag{3}$$

where for clarity, we denote $\psi(\mathbf{d}_i)$ using $\hat{\mathbf{x}}_i$, and $\mathcal{E}(\mathbf{l}_j)$ us-

ing $\mathbf{y}_j$. Similarly, to maximize $I(X;Y)$, the triplet loss formulation can be written as follows.

$$\mathcal{L}_{\text{Trip},\mathcal{E}}(\theta_E) = \sum_{i=1}^{N} \sum_{\mathbf{l}_n \in \mathcal{L}^-_{d_i}} \left[\mathbf{x}_i \cdot \mathbf{y}_n^\top - \mathbf{x}_i \cdot \mathbf{y}_p^\top + \gamma\right]_+ \quad (4)$$

where $\mathbf{x}_i = \mathcal{E}(\mathbf{d}_i)$. In addition to optimizing $I(X;Y)$, $\mathcal{L}_{\text{Trip},\mathcal{E}}(\theta_E)$ also ensures that the Knowledge Bank parameters $\mathcal{K}$ don't overfit the objective and the encoder doesn't lose quality during joint training. Please refer to section 5.4 for ablations on this.

Combining Equations 3 and 4, to achieve MI maximization $I(X_A;Y) + I(X;Y)$, the loss is as follows.

$$\mathcal{L}_{\text{MI-Max}}(\theta) = \mathcal{L}_{\text{Trip},\,\psi}(\theta) + \mathcal{L}_{\text{Trip},\,\mathcal{E}}(\theta_E) \quad (5)$$

Improving the Augmentation Block training using MI Calibration. Since ψ represents AK-enriched document representation, it is expected to have more information that can predict accurate labels compared to $\mathcal{E}$. That is, we want to ensure $I(X_A;Y) \geq I(X;Y)$. To achieve this, we define a calibration loss $\mathcal{L}_{\text{Calib}}$. Inspired by (Ma et al., 2023), we define the calibration loss as expected confidence difference $\text{CD}_{ij} = \delta_{ij}(\mathcal{E}(\mathbf{d}_i) \cdot \mathcal{E}(\mathbf{l}_j)^\top - \psi(\mathbf{d}_i) \cdot \mathcal{E}(\mathbf{l}_j)^\top)$ where δ_{ij} is $+1$ if $\mathbf{d}_i$ and $\mathbf{l}_j$ are related, -1 otherwise. This gives

$$\mathcal{L}_{\text{MI-Calib}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\text{CD}_{ij} + \gamma\right]_+ , \quad (6)$$

where γ is a margin term and N is the batch size. The intuition is that the model is penalized for being less confident when scoring a relevant pair with more (auxiliary) information at input, thereby ensuring $I(X_A;Y) \geq I(X;Y)$. It also ensures that the model is penalized for being more confident when scoring a irrelevant pair with more (auxiliary) information at input. Please refer to Section 5.4 for ablations on the contribution and design of this loss.

In the following Theorem 2, we show why using $\mathcal{L}_{\text{MI-Calib}}$ is meaningful for OAK training. Please refer to Appendix C.2 for the proof.

Theorem 2. *$\mathcal{L}_{Calib}$ gradients specifically update the Knowledge Bank parameters to ensure a higher likelihood of positive labels and a lower likelihood of negative labels.*

Overall, Theorem 2 shows that whenever a set of parameters ψ encourages negative labels getting sampled, the calibration loss encourages the next update to move away from those parameters. It is worth noting that recently popular preference-based learning algorithms for conditional language generation (see, e.g., Rafailov et al. (2023)) employ a similar calibration strategy to train a language model policy. However, they work with binary cross entropy loss in contrast to the triplet or hinge loss that we consider here.

Algorithm 1 Augmentation Module Training

Input: Init trainable θ parameters for $\mathcal{E}$, $\mathcal{K}$ and $\mathcal{C}$ and a trained $\mathcal{R}$. Additionally, batch size B.
for $i = 0$ **to** $|\mathcal{D}|$ **step** B **do**
  Obtain $\mathbf{d}_{i:i+B}$ and $\mathbf{l}_{i:i+B}$
  $\mathbf{x} = \mathcal{E}(\mathbf{d}_{i:i+B})$, $\hat{\mathbf{x}} = \psi(\mathbf{d}_{i:i+B})$, $\mathbf{y} = \mathcal{E}(\mathbf{l}_{i:i+B})$
  $\mathcal{L} = \mathcal{L}_{\text{Trip}}(\hat{\mathbf{x}}, \mathbf{y}) + \mathcal{L}_{\text{Trip}}(\mathbf{x}, \mathbf{y}) + \lambda\mathcal{L}_{\text{MI-Calib}}(\hat{\mathbf{x}}, \mathbf{x}, \mathbf{y})$
  Update parameters θ using $\mathcal{L}$.
end for

The final loss for augmentation block training can be written by combining Equations 5 and 6 as follows.

$$\mathcal{L}_{\text{OAK}}(\theta) = \mathcal{L}_{\text{MI-Max}}(\theta) + \lambda\mathcal{L}_{\text{MI-Calib}}(\theta) \quad (7)$$

Please refer to algorithm 1 for a summary of the overall training method.

Training the Label Classifiers After the augmentation block is trained, we freeze its parameters θ. Then per-label refinement vectors Δw_j are learned to fine-tune the label embeddings and obtain the final label classifiers w_j for each label l_j. For this stage, we follow the same procedure as module M2 training of NGAME (Dahiya et al., 2023a).

4.2. Inference using OAK

Serving label predictions for a given document $\mathbf{d}_i$ is a four step procedure. Firstly, the encoder network $\mathcal{E}$ generates the base embeddings $\mathbf{x} = \mathcal{E}(\mathbf{d}_i)$. In parallel, the linker R is used to predict the top-k relevant AKPs for $\mathbf{d}_i$ and the retrieved indices are used to obtain the AKP embeddings from $\mathcal{K}$. Thirdly, the combiner module $\mathcal{C}$ takes as input the encoder block embedding x and AKP embeddings $\{\mathcal{K}(a_1), \mathcal{K}(a_1), ..., \mathcal{K}(a_K)\}$ obtained from the previous step and generates the enriched representation $\hat{\mathbf{x}}$ as shown in Equation 2. Finally, $\hat{\mathbf{x}}$ is used to query ANNS (Approximate Nearest Neighbor Search) data structure over the one-vs-all classifiers w_j. This helps to retrieve set of labels with high scores from the ANNS lookup. The final score $\mathcal{F}$ for each label l_j is computed as a multiplication of similarity between $\hat{\mathbf{x}}$ and y_j and ANNS lookup score. Labels with a high final score are returned as relevant labels.

For OAK, the augmentation adds <2ms per document during serving. This enables leveraging rich auxiliary information with marginal impact on latency, enabling the AK enriched document representation being generated in average ∼7ms and 99th percentile ∼11ms end-to-end latency.

5. Experiments and Results

In this section, we evaluate the proposed OAK method for the Auxiliary Data enhanced XC task in three ways. Firstly, through comparisons with other leading methods which employ different ways to leverage auxiliary data we demon-

Table 2. Dataset statistics summary for benchmark datasets.

Dataset	# Train Docs	# Labels (L)	# Test Docs	Avg. Docs/label	Avg. labels/Doc	AK Types	# AKPs (M)	Avg. AKPs/Doc
LF-WikiSeeAlsoTitles-320K	693K	312K	177K	2.11	4.67	category	656K	4.89
LF-WikiSeeAlso-320K	693K	312K	177K	2.11	4.67	category	656K	4.89
LF-WikiTitles-500K	1.8M	501K	783K	4.74	17.15	hyper-link	2.1M	15.95
LF-Wikipedia-500K	1.8M	501K	783K	4.74	17.15	hyper-link	2.1M	15.95
LF-ORCAS-800K	7.4M	797K	2.5M	1.75	16.13	related-query	4.9M	15.76

strate the superiority of OAK's design choices. Secondly, via ablations we detail how each component of our architecture is crucial to OAK's performance. Thirdly, we analyse our method's performance on tail data - rare documents and rare labels.

5.1. Datasets

There exist numerous datasets for XC benchmarking, but very few of them offer ground truth auxiliary data. To fix this, we attach ground truth auxiliary data from the original dumps to existing XC datasets. Table 2 shows summary of dataset statistics.

Wikipedia Datasets. The Wikipedia datasets are created from publicly available Wikipedia dumps[1]. The task in the LF-WikiSeeAlsoTitles-320K and LF-WikiSeeAlso-320K (full text version of the former) datasets is to, given a Wikipedia article/page, predict the other Wikipedia articles to be recommended in the 'See Also' section. The wikipedia categories these articles are tagged with are used as auxiliary data in this case. Similarly, LF-WikiTitles-500K and LF-Wikipedia-500K are datasets where the task is to, given a Wikipedia article/page, predict the Wikipedia categories the article should be tagged with. Other Wikipedia article titles connected to the original page via hyperlinks in the article are used as auxiliary data in this case.

ORCAS Dataset. An XC benchmarking dataset LF-ORCAS-800K is created from the ORCAS dump (Dahiya et al., 2023b), which encapsulates the search query to web URL prediction task. However, this dataset doesn't have ground truth auxiliary data attached. Hence, we use GPT-4 (Achiam et al., 2023) to generate "related user queries" given a particular search query, which forms the auxiliary data for this dataset. This related queries dataset used to train OAK's linker will be released publicly upon acceptance of the paper.

Proprietary Dataset. Experiments are also conducted on a proprietary large-scale SponsoredSearch-150M dataset created by mining the logs of a popular search engine. The central task in this dataset is to predict advertiser keywords relevant for a user query. The auxiliary data is obtained by mining the organic search webpages titles clicked in response to the query on the search engine. User-typed queries and the bid keyword corresponding to surfaced advertisements yielded query-keyword training pairs. These pairs were then passed through basic sanity filters based on click-through rate (CTR), clicks, and impressions to create the training dataset. Further, the same search engine logs were mined for webpages clicked by users in response to a query to obtain the query-webpage AK links. The dataset obtained has around 500M, 30M, and 150M queries, webpages, and advertiser keywords respectively.

[1] https://dumps.wikimedia.org/enwiki/20220520/

5.2. Metrics, Baselines and Experimental Setup

In evaluating all methods, we compare on the basis of Precision@K, nDCG@K and Propensity Scored Precision@K (to compare performance on *tail* labels). Detailed explanations of these metrics are provided in Appendix D.

We compare the performance of OAK against existing competitive methods – ANCE (Xiong et al., 2021), NGAME (Dahiya et al., 2023a) and PINA (Chien et al., 2023), graph convolution based methods – GraphFormers (Yang et al., 2021), which employs GNN-Nested transformers layers to aggregate related AKP information within transformer layers and GraphSAGE, where a GIN (Hamilton et al., 2018) convolution performs node-aggregation over documents and related AKP embeddings. To prove the effectiveness of our architecture, we show significant accuracy gains over NGAME, the current *state-of-the-art* Siamese method for XC.

5.3. Results

We clearly see that in Table 3, OAK outperforms every other method. We see strong improvements of $\sim$2% in P@1 over NGAME, which is the closest competitor. This categorically shows OAK's benefits in improving the representation of the documents via augmenting with auxiliary information.

Compared to GraphFormer, OAK demonstrates substantial gains in accuracy, over 15-20% and compared to GraphSAGE, 5-7% higher P@1 across all relevant datasets. This validates that directly aggregating neighboring AKP features leads to intent dilution in other methods. However, OAK's jointly trained Augmentation Block with trainable AKP embeddings mitigates this issue since our the AK Bank increases the model capacity, allowing it to learn how to avoid diluting the document's intent during training to deliver the best performance. Note that experiments with

Table 3. Results on public benchmark datasets. OAK offers ∼5% higher P@1 on standard XC benchmark datasets. Metrics for PINA couldn't be reproduced, only values for those reported in the original paper are shown.

Method	P@1	P@5	N@5	PSP@1	PSP@5	P@1	P@5	N@5	PSP@1	PSP@5	P@1	P@5	N@5	PSP@1	PSP@5	P@1	P@5	N@5	PSP@1	PSP@5
	LF-WikiSeeAlsoTitles-320K					LF-WikiTitles-500K					LF-WikiSeeAlso-320K					LF-Wikipedia-500K				
OAK (Ours)	**33.71**	**17.12**	**34.35**	**25.83**	**30.83**	44.82	**17.67**	**33.72**	**25.79**	**24.90**	**48.57**	**23.28**	**49.16**	**33.92**	**40.44**	**85.23**	**50.79**	**77.26**	45.28	**60.80**
DEXA	32.91	16.77	33.62	24.63	29.55	**47.41**	17.62	33.64	25.27	24.03	47.11	22.71	47.62	31.81	38.78	84.92	50.51	76.8	42.59	58.33
NGAME	32.64	16.60	33.21	24.41	29.87	39.04	16.08	30.75	23.12	23.03	46.40	18.05	46.64	28.18	33.33	84.01	64.69	75.97	41.25	57.04
ANCE	30.79	15.36	31.45	25.14	28.73	29.68	12.51	25.10	23.18	21.18	45.64	17.32	45.43	29.60	32.83	77.92	40.95	68.70	**50.99**	57.33
GraphFormers	21.94	11.79	24.02	19.24	22.70	24.53	11.33	20.35	22.04	19.53	18.14	8.81	20.81	16.85	20.98	31.10	14.00	24.87	25.16	21.83
GraphSAGE	23.13	8.26	25.12	17.84	18.73	21.14	18.79	22.61	21.32	11.82	19.30	10.82	22.67	17.56	23.50	32.53	15.50	25.33	22.34	19.14
PINA	-	-	-	-	-	-	-	-	-	-	44.54	22.92	-	-	-	82.83	50.11	-	-	-

Table 4. OAK offers 5% higher P@1 on LF-ORCAS-800K.

Method	P@1	P@5	N@5	PSP@1	PSP@5
OAK (Ours)	**75.25**	**28.18**	**80.26**	**59.12**	**80.30**
ANCE	72.47	26.60	76.60	58.70	76.68

Table 5. Ablation for AKP representation.

Method	P@1	P@5	N@5	PSP@1	PSP@5
LF-WikiSeeAlsoTitles-320K					
OAK	32.75	16.64	33.65	25.67	30.51
OAK-AK Bank	31.88	15.97	32.45	25.38	29.38
LF-WikiTitles-500K					
OAK	43.80	17.23	32.94	26.00	24.42
OAK-AK Bank	42.64	16.25	31.60	26.39	23.59

GraphFormers and GraphSAGE are provided the ground truth document-AKP linkages at train as well test time, whereas our method uses predictions from the Linker Module trained on the document-AKP linkages only in the train set, without assuming they're available at test time. This gives the former methods an unfair advantage, which is why OAK's performance gains over them conveys an even stronger argument for our design choices. The consistent gains over various auxiliary data-based methods like PINA prove OAK's capability of effectively incorporating AKPs to enrich document representations for improved XC.

Similarly, results in Table 4 shows that OAK offers 5% higher P@1 on LF-ORCAS-800K dataset.

5.4. Ablations

We show ablations on LF-WikiTitles-500K and LF-WikiSeeAlsoTitles-320K. Models for ablation experiments on LF-WikiTitles-500K are trained for only 100 epochs (to save compute) resulting in slightly lower accuracies than shown in Table 3, but remaining hyperparameters are the same as before. We run 300 epochs on the other dataset. Furthermore, ablations are done using label embeddings from Stage 2.

AK Bank. We remove the AK Bank module and obtain AKP embeddings using the encoder $\mathcal{E}$. This not only results in deteriorated performances, as can be seen in Table 5, but also requires upto $\sim 10\times$ more GPU memory and takes $\sim 4\times$ as long to train.

Regularization. Our choice of loss function has three components, $\mathcal{L}_{\text{Trip},\,\psi}$, $\mathcal{L}_{\text{Trip},\,\mathcal{E}}$ and $\mathcal{L}_{\text{MI-Calib}}$. Table 6 ablates over this to demonstrate the importance of each loss component.

Table 6. Ablation for regularisation loss.

Method	P@1	P@5	N@5	PSP@1	PSP@5
LF-WikiSeeAlsoTitles-320K					
$\mathcal{L}_{\text{Trip},\,\psi} + \mathcal{L}_{\text{Trip},\,\mathcal{E}} + \lambda\mathcal{L}_{\text{MI-Calib}}$	32.75	16.64	33.65	25.67	30.51
$\mathcal{L}_{\text{Trip},\,\psi} + \mathcal{L}_{\text{Trip},\,\mathcal{E}}$	31.91	16.16	32.70	24.83	29.48
$\mathcal{L}_{\text{Trip},\,\psi}$	30.51	15.85	31.70	23.40	28.55
LF-WikiTitles-500K					
$\mathcal{L}_{\text{Trip},\,\psi} + \mathcal{L}_{\text{Trip},\,\mathcal{E}} + \lambda\mathcal{L}_{\text{MI-Calib}}$	43.80	17.23	32.94	26.00	24.42
$\mathcal{L}_{\text{Trip},\,\psi} + \mathcal{L}_{\text{Trip},\,\mathcal{E}}$	42.16	16.93	32.32	25.90	24.34
$\mathcal{L}_{\text{Trip},\,\psi}$	40.57	16.27	30.81	23.14	22.49

Combiner Module The Combiner module $\mathcal{C}$ uses cross-attention to decide how much weight to give individual AKP embeddings when fusing with the encoder's document representation to create an enriched representation. In Table 7, we compare this combiner design against a simple mean pooling strategy, where every AKP embedding is given the same weight.

Table 7. Ablation for combiner architecture.

Method	P@1	P@5	N@5	PSP@1	PSP@5
LF-WikiSeeAlsoTitles-320K					
Cross Attention	32.75	16.64	33.65	25.67	30.51
Mean Pooling	27.09	14.49	28.82	19.12	25.49
LF-WikiTitles-500K					
Cross Attention	43.80	17.23	32.94	26.00	24.42
Mean Pooling	34.91	14.51	27.34	20.97	20.71

Early concatenation vs late attentive fusion. OAK broadly does late attentive fusion of the input document and related auxiliary information. Table 8 shows results on the LF-WikiSeeAlsoTitles-320K dataset. In the "Early concatenation" approach, similar to REALM, we simply concatenate the AKP text directly to the document text. The table shows that OAK provides significantly better results across all metrics because of careful fusion of auxiliary information that helps it ignore noisy AKPs.

Table 8. Ablation for Early concatenation vs late attentive fusion (LF-WikiSeeAlsoTitles-320K dataset).

Method	P@1	P@5	N@5	PSP@1	PSP@5
OAK	33.71	17.12	34.35	25.83	30.83
Early concatenation (similar to REALM)	28.49	14.52	29.46	22.26	26.52

5.5. Further Analysis

Performance on tail. To check OAK's predictions for rare documents as well as rare labels, we compare OAK with best Siamese baseline (NGAME) and best node aggregation method (GraphFormers) to see their performances across five document and label frequency quantiles. Firstly, we take a look at tail documents, where in Figure 3 (left) we can clearly see that OAK outperforms NGAME and GraphFormers across all quantiles consistently. This is possible because rare documents have tokens that are uncommon during training. OAK can effectively leverage associated auxiliary data to make accurate predictions in such cases, which NGAME and GraphFormers cannot. Secondly, we take a look at tail labels as well. Figure 3 (right) shows that with the help of auxiliary data, OAK has a fuller understanding of the undiluted document intent which enables it to predict rare labels as well.

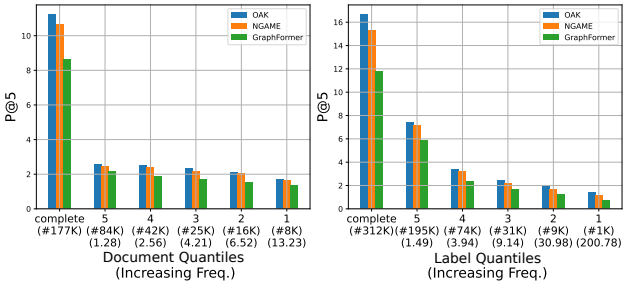

Figure 3. Comparison of OAK, NGAME and GraphFormer P@5 across five document (left)/label (right) frequency quantiles at tail on LF-WikiSeeAlsoTitles-320K. The leftmost column compares overall P@5 and subsequent columns compare P@5 at increasing document/label frequency quantiles. Along the x-axis, first row of brackets denotes how many documents/labels fall into the given quantile, and the second row denotes average number of labels/documents they are linked to. OAK consistently outperforms NGAME and GraphFormer, and is able to predict precisely for rare documents as well as rare labels.

Generalized OAK Framework. Similar to how we built OAK on top of NGAME, it is evident that incorporating the OAK framework to a new Siamese XC method involves simply swapping out the encoder and using the method's training methodology, alongside loss functions specific to OAK. In Table 9 we show how OAK improves upon DPR (Karpukhin et al., 2020b) as well, a powerful Siamese dense retrieval method.

Oracle Linker. Table 10 shows results when we use ground truth AKPs rather than those predicted by the linker. For short text datasets, as expected, results are better when using ground truth AKPs. Surprisingly, for full text datasets, where trained linker quality is better than for short-text datasets, note that the results with ground truth AKPs are indeed lower compared to ones with predicted AKPs. We observe that this is because of missing AKPs in the ground truth that the linker is able to retrieve.

Table 9. OAK's performance over DPR on LF-WikiSeeAlsoTitles-320K.

Method	P@1	P@5	N@5	PSP@1	PSP@5
OAK+DPR	28.91	14.81	30.32	23.21	28.64
DPR	26.62	13.73	28.21	22.00	25.51

Table 10. Results using Oracle Linker AKPs.

	P@1	P@5	N@5	PSP@1	PSP@5
LF-WikiSeeAlsoTitles-320K	38.92	19.35	40.35	29.69	34.85
LF-WikiTitles-500K	52.53	24.25	41.98	30.62	32.26
LF-WikiSeeAlso-320K	47.73	23.04	48.62	33.62	40.06
LF-Wikipedia-500K	83.96	50.12	76.22	46.45	60.86

Training and Inference Time. We observe that OAK brings a non-trivial but acceptable increment in inference latency and required training time/compute. In comparison to a simple NGAME-trained encoder, on a 2xA100 setup we observe that OAK causes a $\sim 1.5\times$ increment in training time, and during inference on production infrastructure, it results in $\sim$ 2ms average increment in latency.

5.6. Application to Sponsored search

A key challenge in sponsored search is to accurately match user queries to billions of bid keywords submitted by advertisers to ensure semantic relevance. Furthermore, sponsored search has strict matching criteria since advertisers bid varying amounts depending on the relevance and the semantic relationship of their keyword to the user query[2,3]. OAK learns AKP representation as learnable embeddings optimized for the target query to advertiser keyword matching application, where auxiliary data is obtained from the seemingly disparate application of organic webpages clicked in response to a user query. This allows OAK to avoid query intent dilution but still be able to reformulate its representation with additional contextual information to enhance retrieval effectiveness.

Offline Results. We compare OAK against several state-of-the-art dense retrieval encoders currently deployed in the production system. OAK is trained on the SponsoredSearch-150M dataset. For a fair evaluation, we sample 1M user

[2] https://support.google.com/google-ads/answer/7478529?hl=en

[3] https://help.ads.microsoft.com/#apex/ads/en/50822/1

queries at random from the search engine logs from a time period distinct from the one during which the train data was collected. Since a vast majority of queries seen on the search engine are tail queries, this random sampling ensures that the test queries in this experiment are tail in nature. The production dense retrieval baseline algorithms are anonymized due to intellectual property restrictions. Results in Table 11 demonstrate that OAK outperforms the best production system by at least 5% in terms of Recall@100 on this challenging large-scale setup. Performance is measured in terms of Recall along with Precision because the retrieval algorithm can be followed by a separate re-ranker to optimize for precision in some pipelines.

Table 11. Results on offline evaluation for Sponsored Search. We see that OAK categorically performs better than proprietary variations of leading dense retrieval algorithms deployed in production.

Method	P@1	P@5	R@20	R@50	R@100
OAK	**36.27**	**20.74**	**46.28**	**58.74**	**68.80**
M1	34.26	19.39	42.07	52.62	60.98
M2	23.09	12.32	25.76	31.96	36.88
M3	25.15	14.57	37.61	51.21	63.16

Online Results. We conduct extensive online A/B testing of OAK on live traffic sampled from Bing. OAK is thus not only compared to leading dense retrieval algorithms, but also leading XC, graph-based, and generative language models. The key metrics tracked are: (1) Click-Through Rate (CTR): Ratio of clicks to query impressions, indicates ad relevance. (2) Impression Yield (IY) and Click Yield (CY): Average number of ad impressions and clicks per user query search. (3) Keyword Density (KD): Fraction of predicted keywords passing relevance filters, assesses prediction quality. All metrics are reported after reaching statistically significant p-value of < 0.001. OAK led to a 0.84% increase in CTR demonstrating OAK's ability to retrieve more relevant ads, thereby improving user experience. The KD improved by 2.7% with OAK, validating the predictive power of AKP-enriched query representations. Additionally, Fig. 4 shows the IY and CY improvements in different query quantiles, where it can be observed that OAK shows disproportionately more gains for tail user queries. OAK could effectively use webpages as AKPs to retrieve keywords like "genealogist services" for the user query "ancestry com", which none of the previous ensemble of algorithms including state-of-the-art dense retrievers could. Table 12 shows more such examples obtained by infusing the query with auxiliary knowledge in OAK. In labeling by expert judges, OAK was found to increase the percentage of excellent predictions to 21.6% as against 19.7% for the leading in-production denser retriever (M1 in Table 11).

The gains on these key metrics clearly highlight OAK's benefits in a real-world production setting. The large-scale live A/B test provides strong statistically significant results, with metrics computed over millions of user searches and ad impressions. Overall, OAK substantially improves sponsored search ad retrieval through enhanced query representations.

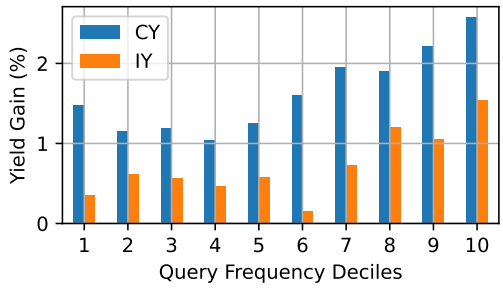

Figure 4. Plotting IY and CY *gains* versus query frequency deciles. Here we can clearly see that OAK brings diverse and accurate predictions to the ensemble of algorithms it is deployed with, thereby demonstrating gains in Impression Yields and Click Yields. These statistics are calculated against a performant ensemble of algorithms in an A/B testing scenario, after achieving $p < 0.001$ for statistical significance. OAK provides yield gains across all frequency deciles and notably, disproportionately larger gains for tail user queries.

Table 12. Advertiser keywords predicted for a user query by OAK which were missed by the production ensemble of leading dense retrieval, graph-based, XC, and generative language models. OAK can go beyond just text similarly by enriching the query representation with auxiliary knowledge made available from webpages clicked when the query is asked by a representative user on the search engine.

User Query	Advertiser Keyword
12 month online dnp programs	nursing practical program
zoho	crm software
godaddy	website builder
comcast internet	xfinity
euro to dollar	oanda
deflector mower replacement	lawn mower parts
internet explorer	ms browser

6. Conclusion

We introduced OAK, a novel framework for enriching document representations using relevant auxiliary information. OAK strategically expands model capacity by learning an AK Bank, while preserving the original document intent preservation through joint training with the main XC task. OAK's modular architecture enables seamlessly improving existing XC methods. The representations learned for AKPs are optimized directly for the target XC task, making OAK robust to noisy AKPs. The calibration loss ensures that the AK bank adds useful information to the enriched document representations. Through extensive experiments on a leading sponsored search engine and public datasets, we demonstrated OAK's effectiveness in improving XC accuracy, especially for rare documents and labels.

Impact Statement

Our usage of data and terms of providing service to people around the world has been approved by our legal and ethical boards. In terms of social relevance, our research is helping millions of people find the goods and services that they are looking for online with increased efficiency and a significantly improved user experience. This facilitates purchase and delivery without any physical contact which is important given today's social constraints. Furthermore, our research is increasing the revenue of many small and medium businesses including mom and pop stores while also helping them grow their market and reduce the cost of reaching new customers.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Babbar, R. and Schölkopf, B. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*, 2017.

Babbar, R. and Schölkopf, B. Data scarcity, robustness and extreme multi-label classification. *ML*, 2019.

Bai, J., Kong, S., and Gomes, C. P. Gaussian mixture variational autoencoder with contrastive learning for multi-label classification. In *International Conference on Machine Learning*, pp. 1383–1398. PMLR, 2022.

Barezi, E. J., W., I. D., Fung, P., and Rabiee, H. R. A Submodular Feature-Aware Framework for Label Subset Selection in Extreme Classification Problems. In *NAACL*, 2019.

Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu, Y., and Varma, M. The extreme classification repository: Multi-label datasets and code, 2016a. URL http://manikvarma.org/downloads/XC/XMLRepository.html.

Bhatia, K., Dahiya, K., Jain, H., Mittal, A., Prabhu, Y., and Varma, M. The Extreme Classification Repository: Multi-label Datasets & Code, 2016b. URL http://manikvarma.org/downloads/XC/XMLRepository.html.

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. Extreme Multi-Label Legal Text Classification: A case study in EU Legislation. In *ACL*, 2019.

Chang, W.-C., H.-F., Y., Zhong, K., Yang, Y., and Dhillon, I.-S. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *KDD*, 2020.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Chiang, W., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *KDD*, 2019.

Chien, E., Zhang, J., Hsieh, C., Jiang, J., Chang, W., Milenkovic, O., and Yu, H. PINA: Leveraging side information in eXtreme multi-label classification via predicted instance neighborhood aggregation. In *ICML*, 2023.

Dahiya, K., Agarwal, A., Saini, D., Gururaj, K., Jiao, J., Singh, A., Agarwal, S., Kar, P., and Varma, M. SiameseXML: Siamese Networks meet Extreme Classifiers with 100M Labels. In *ICML*, 2021a.

Dahiya, K., Saini, D., Mittal, A., Shaw, A., Dave, K., Soni, A., Jain, H., Agarwal, S., and Varma, M. DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents. In *WSDM*, 2021b.

Dahiya, K., Gupta, N., Saini, D., Soni, A., Wang, Y., Dave, K., Jiao, J., Gururaj, K., Dey, P., Singh, A., Hada, D., Jain, V., Paliwal, B., Mittal, A., Mehta, S., Ramjee, R., Agarwal, S., Kar, P., and Varma, M. Ngame: Negative mining-aware mini-batching for extreme classification. In *WSDM*, March 2023a.

Dahiya, K., Yadav, S., Sondhi, S., Saini, D., Mehta, S., Jiao, J., Agarwal, S., Kar, P., and Varma, M. Deep encoders with auxiliary parameters for extreme classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 358–367, 2023b.

Faghri, F., Fleet, D.-J., Kiros, J.-R., and Fidler, S. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *BMVC*, 2018.

Guo, C., Mousavi, A., Wu, X., Holtmann-Rice, D.-N., Kale, S., Reddi, S., and Kumar, S. Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *NeurIPS*, 2019.

Gupta, N., Khatri, D., Rawat, A. S., Bhojanapalli, S., Jain, P., and Dhillon, I. S. Efficacy of dual-encoders for extreme multi-label classification. In *ICLR*, 2023.

Gupta, V., Wadbude, R., Natarajan, N., Karnick, H., Jain, P., and Rai, P. Distributional Semantics Meets Multi-Label Learning. In *AAAI*, 2019.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive Representation Learning on Large Graphs, 2018.

He, K., Fan, H., W., Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020a.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pp. 639–648, 2020b.

Hofstätter, S., Lin, S.-C., Yang, J.-H., Lin, J., and Hanbury, A. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *SIGIR*, 2021.

Huang, W., Zhang, T., Rong, Y., and Huang, J. Adaptive Sampling Towards Fast Graph Representation Learning, 2018.

Jain, H., Prabhu, Y., and Varma, M. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *KDD*, August 2016.

Jain, H., Balasubramanian, V., Chunduri, B., and Varma, M. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *WSDM*, 2019.

Jain, V., Prakash, J., Saini, D., Jiao, J., Ramjee, R., and Varma, M. Renee: End-to-end training of extreme classification models. *Proceedings of Machine Learning and Systems*, 2023.

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hullermeier, E. Extreme F-measure Maximization using Sparse Probability Estimates. In *ICML*, 2016.

Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., and Zhuang, F. LightXML: Transformer with Dynamic Negative Sampling for High-Performance Extreme Multi-label Text Classification. In *AAAI*, 2021.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020a.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020b.

Khandagale, S., Xiao, H., and Babbar, R. Bonsai: diverse and shallow trees for extreme multi-label classification. *ML*, 2020.

Kharbanda, S., Banerjee, A., Schultheis, E., and Babbar, R. Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. In *NeurIPS*, 2022.

Kharbanda, S., Banerjee, A., Gupta, D., Palrecha, A., and Babbar, R. Inceptionxml: A lightweight framework with synchronized negative sampling for short text extreme classification. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 760–769, 2023.

Kingma, P. D. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

Lee, K., Chang, M.-W., and Toutanova, K. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Liu, J., Chang, W., Wu, Y., and Yang, Y. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*, 2017.

Luan, Y., Eisenstein, J., Toutanova, K., and Collins, M. Sparse, Dense, and Attentional Representations for Text Retrieval. In *TACL*, 2020.

Ma, H., Zhang, Q., Zhang, C., Wu, B., Fu, H., Zhou, J. T., and Hu, Q. Calibrating multimodal learning. In *International Conference on Machine Learning*, pp. 23429–23450. PMLR, 2023.

Medini, T. K. R., Huang, Q., Wang, Y., Mohan, V., and Shrivastava, A. Extreme Classification in Log Memory using Count-Min Sketch: A Case Study of Amazon Search with 50M Products. In *NeurIPS*, 2019.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*, 2013.

Mineiro, P. and Karampatziakis, N. Fast Label Embeddings via Randomized Linear Algebra. In *ECML/PKDD*, 2015.

Mittal, A., Dahiya, K., Agrawal, S., Saini, D., Agarwal, S., Kar, P., and Varma, M. DECAF: Deep Extreme Classification with Label Features. In *WSDM*, 2021a.

Mittal, A., Sachdeva, N., Agrawal, S., Agarwal, S., Kar, P., and Varma, M. ECLARE: Extreme Classification with Label Graph Correlations. In *WWW*, 2021b.

Prabhu, Y., Kag, A., Gopinath, S., Dahiya, K., Harsola, S., Agrawal, R., and Varma, M. Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*, 2018a.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*, 2018b.

Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W. X., Dong, D., Wu, H., and Wang, H. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Saini, D., Jain, A., Dave, K., Jiao, J., Singh, A., Zhang, R., and Varma, M. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification. In *WWW*, 2021.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, 2019.

Siblini, W., Kuntz, P., and Meyer, F. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In *ICML*, 2018.

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

Tagami, Y. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *KDD*, 2017.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, k., Vinyals, O., and Graves, A. Conditional image generation with pixelcnn decoders. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/b1301141feffabac455e1f90a7de2054-Paper.pdf.

Wei, T., Tu, W. W., and Li, Y. F. Learning for Tail Label Data: A Label-Specific Feature Approach. In *IJCAI*, 2019.

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., and Dembczynski, K. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NIPS*, 2018.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*, 2021.

Yang, J., Liu, Z., Xiao, S., Li, C., Lian, D., Agrawal, S., Singh, A., Sun, G., and Xie, X. Graphformers: Gnn-nested transformers for representation learning on textual graph. *NeurIPS*, 34:28798–28810, 2021.

Yang, Y., Huang, C., Xia, L., and Li, C. Knowledge graph contrastive learning for recommendation. In *SIGIR Conference*, pp. 1434–1443, 2022. URL https://github.com/yuh-yang/KGCL-SIGIR22.

Ye, H., Chen, Z., Wang, D.-H., and Davison, B. D. Pretrained Generalized Autoregressive Model with Adaptive Probabilistic Label Clusters for Extreme Multi-label Text Classification. In *ICML*, 2020.

Yen, E. I., Huang, X., Dai, W., Ravikumar, P., Dhillon, I., and Xing, E. PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *KDD*, 2017.

You, R., Dai, S., Zhang, Z., Mamitsuka, H., and Zhu, S. AttentionXML: Extreme Multi-Label Text Classification with Multi-Label Attention Based Recurrent Neural Networks. In *NeurIPS*, 2019.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*, 2020.

Zhang, J., Chang, W.-c., Yu, H.-f., and Dhillon, I. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In *NeurIPS*, 2021.

Zhang, W., Wang, L., Yan, J., Wang, X., and Zha, H. Deep Extreme Multi-label Learning. *ICMR*, 2018.

Zhu, J., Cui, Y., Liu, Y., Sun, H., Li, X., Pelger, M., Yang, T., Zhang, L., Zhang, R., and Zhao, H. Textgnn: Improving text encoder via graph neural network in sponsored search. In *theWebConf*, pp. 2848–2857, 2021.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., and Gu, Q. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks, 2019.

A. Implementation details and Hyper-parameters

In this section we detail the training strategy used, important hyperparameters and experiment design for baselines. For our Encoder Module, we use a DistilBERT-base encoder and for the combiner module we use a single cross-attention layer with pooling. For the AK Bank $\mathcal{K}$, we use a 768 dimensional trainable embedding for every AKP's trainable embedding. We jointly train everything together using AdamW (Kingma & Ba, 2014) for dense parameters and SparseAdam[4] for the AK Bank. We train this model for 300 epochs on 2xNVidia A100-80GB GPUs for all datasets, with a batch size of 1024 and a linear LR scheduler with warmup. We use top 3 predicted AKPs from the linker during inference. At train time, we use all the ground truth AKPs. We use the same hyperparameters (wherever applicable) for training all other methods as well.

B. Detailed Related Work

XC: XC is a key paradigm in several areas such as ranking and recommendation. The literature on XC methods is vast (Dahiya et al., 2021b; Guo et al., 2019; Wydmuch et al., 2018; Zhang et al., 2018; Medini et al., 2019; Mittal et al., 2021a;b; Saini et al., 2021; Liu et al., 2017; You et al., 2019; Jiang et al., 2021; Chalkidis et al., 2019; Ye et al., 2020; Zhang et al., 2021; Mineiro & Karampatziakis, 2015; Babbar & Schölkopf, 2017; Jasinska et al., 2016; Khandagale et al., 2020; Jain et al., 2016; Prabhu et al., 2018b; Tagami, 2017; Yen et al., 2017; Wei et al., 2019; Siblini et al., 2018; Barezi et al., 2019; Jain et al., 2019; Gupta et al., 2019; 2023). Early XC methods used fixed (bag-of-words) (Mineiro & Karampatziakis, 2015; Babbar & Schölkopf, 2017; Jasinska et al., 2016; Khandagale et al., 2020; Jain et al., 2016; Prabhu et al., 2018b; Tagami, 2017; Yen et al., 2017; Wei et al., 2019; Siblini et al., 2018; Barezi et al., 2019) or pre-trained (Jain et al., 2019) features and focused on learning only a classifier architecture. Recent advances have demonstrated significant gains by using task-specific features obtained from a variety of deep encoders such as bag-of-embeddings (Dahiya et al., 2021b; 2023a), CNNs (Liu et al., 2017), LSTMs (You et al., 2019), and transformers (Jiang et al., 2021; Chalkidis et al., 2019; Ye et al., 2020; Zhang et al., 2021). Training is scaled to millions of labels and training points (Dahiya et al., 2021b) by performing encoder pre-training followed by classifier training. A data point is trained only on its relevant labels (that are usually few in number) and a select few irrelevant labels deemed most informative using *negative mining* (Mikolov et al., 2013; Dahiya et al., 2021a; Guo et al., 2019; Faghri et al., 2018; Chen et al., 2020; He et al., 2020a; Karpukhin et al., 2020a; Lee et al., 2019; Luan et al., 2020; Hofstätter et al., 2021; Xiong et al., 2021; Qu et al., 2021; Dahiya et al., 2023a). None of these XC methods make use of any auxiliary information associated with documents, except PINA (Chien et al., 2023) which uses instance correlation signals to learn neighborhood aggregated representations. Our experiments show that OAK outperforms PINA by large margins.

Retrieval Augmented Language Models: Retrieval Augmented Language Models are a class of models that leverage external knowledge sources to enhance accuracy using Transformer encoders (Guu et al., 2020). Such models typically consist of two main components: a retriever and a knowledge-augmented encoder. The retriever selects relevant documents or passages from a large corpus based on the input query. The encoder concatenates query and relevant documents and computes an embedding for the sequence. Retrieval augmentation has also been extended to generation (RAG (Lewis et al., 2020)), but we not discuss RAG in detail since XC is a classification problem.

Graph Neural Networks in Related Areas: A sizeable body of work exists on using graph neural networks such as graph convolutional networks (GCN) for recommendation (Hamilton et al., 2018; Chen et al., 2018; Zou et al., 2019; Huang et al., 2018; Chiang et al., 2019; Zeng et al., 2020; Yang et al., 2021; Zhu et al., 2021; He et al., 2020b; Yang et al., 2022). Certain methods e.g., FastGCN (Chen et al., 2018), KGCL (Yang et al., 2022), LightGCN (He et al., 2020b) learn label embeddings as (functions of) learnable parameter embeddings. This makes it difficult for these methods to ingest novel labels and their use is restricted to warm-start scenarios. Other GCN-based methods such as GraphSAGE (Hamilton et al., 2018) and GraphFormers (Yang et al., 2021) learn node representations as functions of node metadata e.g. textual descriptions. This allows the methods to work in zero-shot settings but they still incur the high storage and computational cost of GCNs. Moreover, diminishing returns are observed with increasing number of layers of the GCN (Chiang et al., 2019; Mittal et al., 2021b). As a baseline method, in this work, we experiment with enriching document representations using document-AKP graphs. Our experiments show that the OAK method offers a far more scalable alternative to GCNs and other popular graph-based architectures in XC settings, significantly reducing the overheads of graph-based learning, yet offering sustained and significant performance boosts in prediction accuracies.

[4] https://pytorch.org/docs/stable/generated/torch.optim.SparseAdam.html

C. Theoretical Analyses

C.1. Equivalence between Siamese training and Triplet Loss

Theorem 3. *Siamese training via triplet loss serves to maximize mutual information between the distributions cast by the document and label encoder representations.*

Proof. Formally, given random variables $X \sim \mathbb{P}[\mathcal{E}(\mathbf{d})]$ and $Y \sim \mathbb{P}[\mathcal{E}(\mathbf{l})]$, we observe that the Mutual Information between X and Y is intractable. However, the InfoNCE (van den Oord et al., 2016) objective is a tractable lower bound on it, i.e., $I(X;Y) \geq I_{\text{NCE}}$ (Tschannen et al., 2019). Here, for some scale parameter $\tau > 0$, the InfoNCE objective is defined as follows:

$$I_{\text{NCE}} \stackrel{\text{def}}{=} \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\mathbf{x}_i \cdot \mathbf{y}_i^\top/\tau)}{\frac{1}{N}\sum_{j=1}^{N}\exp(\mathbf{x}_i \cdot \mathbf{y}_j^\top/\tau)}\right],$$

where the expectation is over i.i.d. random draws of $(\mathbf{x}_i, \mathbf{y}_i)$ from the joint distribution $\mathbb{P}[\mathcal{E}(\mathbf{d})] \times \mathbb{P}[\mathcal{E}(\mathbf{l})]$.

Simple algebra shows that maximizing InfoNCE is equivalent to minimizing the expected multi-class N-pair loss (Sohn, 2016) $\mathcal{L}_{N\text{-pair-mc}}(\theta)$, given by

$$\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\sum_{j\neq i}\exp(\mathbf{x}_i \cdot \mathbf{y}_j^\top/\tau - \mathbf{x}_i \cdot \mathbf{y}_i^\top/\tau)\right).$$

Now, under the simple setting of only one negative label $\mathbf{y}_j$ per document $\mathbf{x}_i$ and also setting $\tau = 1/2$, the above N-pair multi class loss can be approximated as follows (Bai et al., 2022).

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\exp(2\mathbf{x}_i \cdot \mathbf{y}_j^\top - 2\mathbf{x}_i \cdot \mathbf{y}_i^\top)\right)\\ &\approx \frac{1}{N}\sum_{i=1}^{N}\left(1+2\mathbf{x}_i \cdot \mathbf{y}_j^\top - 2\mathbf{x}_i \cdot \mathbf{y}_i^\top\right),\end{aligned}$$

It is easy to see that this is twice of the triplet loss (Eq. 1) where margin=1/2. Thus, under the above assumptions, one can show that a Siamese encoder θ minimizes the triplet loss for margin $\gamma = 1/2$ iff it minimizes the N-pair-mc loss for $N = 1$, i.e., the two loss functions are equivalent. □

C.2. Impact of $\mathcal{L}_{\text{Calib}}$ gradients on the Knowledge Bank parameters

Theorem 4. *$\mathcal{L}_{Calib}$ gradients specifically update the Knowledge Bank parameters to ensure a higher likelihood of positive labels and a lower likelihood of negative labels.*

Proof. We analyse a degenerate case of $\mathcal{L}_{\text{Calib}}$ for one positive and one negative label per document to prove this. Formally, we have for our calibration loss:

$$\begin{aligned}\mathcal{L}_{\text{Calib}} = \mathbb{E}_{\mathbf{d},\mathbf{l}_p,\mathbf{l}_n}\big[&\mathcal{E}(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_p)^\top - \psi(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_p)^\top\\ &+ \psi(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_n)^\top - \mathcal{E}(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_n)^\top\big]_+ .\end{aligned}$$

Under the softmax distribution on labels $\mathbf{l}$ given a document $\mathbf{d}$, we have for a fixed function $f : \mathcal{X} \to \mathcal{S}^{D-1}$:

$$\mathbb{P}_f[\mathbf{l}|\mathbf{d}] = \frac{\exp(f(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l})^\top)}{\sum_{\mathbf{l}' \in \mathcal{L}}\exp(f(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}')^\top)} .$$

For a positive label $\mathbf{l}_p$ and a negative label $\mathbf{l}_n$, it holds that

$$\log\frac{\mathbb{P}_f[\mathbf{l}_p|\mathbf{d}]}{\mathbb{P}_f[\mathbf{l}_n|\mathbf{d}]} = f(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_p)^\top - f(\mathbf{d}) \cdot \mathcal{E}(\mathbf{l}_n)^\top .$$

Then, the calibration loss can be rewritten as

$$\begin{aligned}\mathcal{L}_{\text{Calib}} &= \mathbb{E}_{\mathbf{d},\mathbf{l}_p,\mathbf{l}_n}\left[\log\frac{\mathbb{P}_{\mathcal{E}}[\mathbf{l}_p|\mathbf{d}]}{\mathbb{P}_{\mathcal{E}}[\mathbf{l}_n|\mathbf{d}]} - \log\frac{\mathbb{P}_{\psi}[\mathbf{l}_p|\mathbf{d}]}{\mathbb{P}_{\psi}[\mathbf{l}_n|\mathbf{d}]}\right]_+ \\ &= \mathbb{E}_{\mathbf{d},\mathbf{l}_p,\mathbf{l}_n}\left[\log\frac{\mathbb{P}_{\psi}[\mathbf{l}_n|\mathbf{d}]}{\mathbb{P}_{\mathcal{E}}[\mathbf{l}_n|\mathbf{d}]} - \log\frac{\mathbb{P}_{\psi}[\mathbf{l}_p|\mathbf{d}]}{\mathbb{P}_{\mathcal{E}}[\mathbf{l}_p|\mathbf{d}]}\right]_+ \\ &= \mathbb{E}_{\mathbf{d},\mathbf{l}_p,\mathbf{l}_n}\left[r_{\psi}(\mathbf{l}_n,\mathbf{d}) - r_{\psi}(\mathbf{l}_p,\mathbf{d})\right]_+ \ ,\end{aligned}$$

where $r_{\psi}(\mathbf{l},\mathbf{d}) = \log\frac{\mathbb{P}_{\psi}[\mathbf{l}|\mathbf{d}]}{\mathbb{P}_{\mathcal{E}}[\mathbf{l}|\mathbf{d}]}$ denotes the log-likelihood ratio between $\mathbb{P}_{\psi}$ and $\mathbb{P}_{\mathcal{E}}$ for a label $\mathbf{l}$ given a document $\mathbf{d}$. We omit dependence on the encoder $\mathcal{E}$ for brevity.

Now, we compute (sub)gradients of the calibration loss w.r.t. the knowledge bank paramteres (Augmentation Block) parameters ψ and keeping the encoder $\mathcal{E}$ fixed. The gradients take the form:

$$\nabla_{\theta_K}\mathcal{L}_{\text{Calib}} = \mathbb{E}\left[\mathbb{I}\{r_{\psi}(\mathbf{l}_n,\mathbf{d}) > r_{\psi}(\mathbf{l}_p,\mathbf{d})\}\nabla_{\theta_K}\log\frac{\mathbb{P}_{\psi}[\mathbf{l}_n|\mathbf{d}]}{\mathbb{P}_{\psi}[\mathbf{l}_p|\mathbf{d}]}\right] \ ,$$

where $\mathbb{I}$ denotes the indicator function. Thus, the knowledge bank parameter updates (using negative gradients of the calibration loss) increase likelihood of positive labels being sampled and decrease that of negative labels. Importantly, the updates take place only when the likelihood ratio of negative labels are higher than that of positive labels at current knowledge bank parameters. □

D. Evaluation metrics

Performance has been evaluated using propensity scored precision@k and nDCG@k, which are unbiased and more suitable metric in the XC setting (Jain et al., 2016; Babbar & Schölkopf, 2019; Prabhu et al., 2018a;b). The propensity model and values available on The Extreme Classification Repository (Bhatia et al., 2016b) were used. Performance has also been evaluated using vanilla precision@k and nDCG@k (with k = 1, 3 and 5) for extreme classification.

Let $\hat{\mathbf{y}} \in \mathbb{R}^L$ denote the predicted score vector and $\mathbf{y} \in \{0,1\}^L$ denote the ground truth vector (with $\{0,1\}$ entries this time instead of ± 1 entries, for sake of convenience). The notation $rank_k(\hat{\mathbf{y}}) \subset [L]$ denotes the set of k labels with highest scores in the prediction score vector $\hat{\mathbf{y}}$ and $\|\mathbf{y}\|_1$ denotes the number of relevant labels in the ground truth vector. Then we have:

$$P@k = \frac{1}{k}\sum_{l\in rank_k(\hat{\mathbf{y}})} y_l$$

$$PSP@k = \frac{1}{k}\sum_{l\in rank_k(\hat{\mathbf{y}})} \frac{y_l}{p_l}$$

$$DCG@k = \frac{1}{k}\sum_{l\in rank_k(\hat{\mathbf{y}})} \frac{y_l}{\log(l+1)}$$

$$PSDCG@k = \frac{1}{k}\sum_{l\in rank_k(\hat{\mathbf{y}})} \frac{y_l}{p_l\log(l+1)}$$

$$nDCG@k = \frac{DCG@k}{\sum_{l=1}^{\min(k,||\mathbf{y}||_0)}\frac{1}{\log(l+1)}}$$

$$PSnDCG@k = \frac{PSDCG@k}{\sum_{l=1}^{k}\frac{1}{\log l+1}}$$

$$FN@k = 1 - \frac{\sum_{l\in rank_k(\hat{\mathbf{y}})} y_l}{\|\mathbf{y}\|_1}$$

Here, p_l is propensity score of the label l calculated as described in Jain et al. (2016).

E. Qualitative Analysis

Table 13 shows a few sample predictions from OAK, NGAME and GraphFormer along with ground truth labels. OAK provides more accurate and diverse label predictions.

Table 13. Sample predictions from OAK, NGAME and GraphFormer along with ground truth labels. OAK provides more accurate and diverse label predictions. Legend: Black (ground truth), Red (incorrect), Green (correct)

Document	Predicted AKPs	Ground truth labels	OAK predictions	NGAME predictions	GraphFormer Predictions
Cerastes (genus)	Snake Genera, Viperinae, Reptiles of Western Sahara	Snakebite, List of viperine species and subspecies, Viperinae by common name, Viperinae by taxonomic synonyms	Viperinae by common name, Viperinae by taxonomic synonyms, List of viperine species and subspecies, Snakebite, List of crambid genera	List of prehistoric cartilaginous fish, List of Greek mythological figures, Snakebite, List of snake genera, List of prehistoric echinoid genera	Penaeus monodon, Chororapithecus, Heracleidae, Lituites, Pachyrhachis
Beachcomber 25	Keelboats, Sailing Yachts, Trailer Sailers	List of sailing boat types, US Yachts US 25, Catalina 250, Tanzer 25, MacGregor 25, Bayfield 25, Cal 25, Cal 2-25, Capri 25, Catalina 25, Dufour 1800, Hunter 25.5, O'Day 25, Merit 25, Northern 25	Bayfield 25, List of sailing boat types, O'Day 25, Cal 25, Cal 2-25	Mirage 25, Focke-Wulf A 17, MacGregor 25, Lock Crowther, Stout 2-AT Pullman	Sea Sprite 27, Shark 24, Bowman 48, San Juan 24, Searunner 34
West Nakdong River	Nakdong River, Rivers of North Gyeongsang, Rivers of Busan	List of Korea-related topics, Geography of South Korea, List of rivers of Asia, Nakdong River	Nakdong River, List of rivers of Asia, List of Korea-related topics, Geography of South Korea, Han River	Nakdong River, List of rivers of Asia, Stung Sen River, River systems of Thailand, Yellow Sea	Nakdong River, Rivers of Korea, Yangjaecheon, Seoraksan, List of rivers of Korea
List of rap rock bands	Lists of rock musicians by subgenre, List of bands, Lists of musicians by genre, Lists of heavy metal bands	List of nu metal bands, List of funk rock bands, List of alternative metal artists, Rap rock, Rap metal	Rap rock, Rap metal, List of funk rock bands, List of groove metal bands, List of nu metal bands	List of hip hop groups, List of alternative-rock bands, List of nu metal bands, List of funk rock bands, List of hip hop musicians	Rap rock, List of hip hop groups, Gangsta rap, Rap metal, List of hardcore punk bands