GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification

Deepak Saini*
Microsoft Research
Bengaluru, India
desaini@microsoft.com

Arnav Kumar Jain*†
MILA
Montreal, Canada
arnavkj95@gmail.com

Kushal Dave*
Microsoft
Redmond, USA
kudave@microsoft.com

Jian Jiao
Microsoft
Redmond, USA
jiajia@microsoft.com

Amit Singh
Microsoft
Bengaluru, India
siamit@microsoft.com

Ruofei Zhang
Microsoft
Redmond, USA
bzhang@microsoft.com

Manik Varma
Microsoft Research
Bengaluru, India
IIT Delhi
Delhi, India
manik@microsoft.com

ABSTRACT

This paper develops the GalaXC algorithm for Extreme Classification, where the task is to annotate a document with the most relevant subset of labels from an extremely large label set. Extreme classification has been successfully applied to several real world web-scale applications such as web search, product recommendation, query rewriting, etc. GalaXC identifies two critical deficiencies in leading extreme classification algorithms. First, existing approaches generally assume that documents and labels reside in disjoint sets, even though in several applications, labels and documents cohabit the same space. Second, several approaches, albeit scalable, do not utilize various forms of metadata offered by applications, such as label text and label correlations. To remedy these, GalaXC presents a framework that enables collaborative learning over joint document-label graphs at massive scales, in a way that naturally allows various auxiliary sources of information, including label metadata, to be incorporated. GalaXC also introduces a novel label-wise attention mechanism to meld high-capacity extreme classifiers with its framework. An efficient end-to-end implementation of GalaXC is presented that could be trained on a dataset with 50M labels and 97M training documents in less than 100 hours on 4×V100 GPUs. This allowed GalaXC to not only scale to applications with several millions of labels, but also be up to 18% more accurate than leading deep extreme classifiers, while being upto 2-50× faster to train and 10× faster to predict on benchmark datasets. GalaXC is particularly well-suited to warm-start scenarios where predictions need to be made on data points with partially revealed label sets, and was found to be up to 25% more accurate than extreme classification algorithms specifically designed for warm start settings. In A/B tests conducted on the Bing search engine, GalaXC could improve the Click Yield (CY) and coverage by 1.52% and 1.11% respectively. Code for GalaXC is available at https://github.com/Extreme-classification/GalaXC

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; *Supervised learning by classification.*

KEYWORDS

Extreme classification, warm start, large output spaces, web-scale recommendation, sponsored search

ACM Reference Format:
Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. 2021. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449937

1 INTRODUCTION

Overview Extreme Classification (XC) refers to the task of tagging a document with its most relevant subset of labels from an extremely large label space. Several problems such as search advertising [42], related product recommendation [35], related query recommendation [22], etc have been formulated as XC problems leading to

*All authors contributed equally to this research.
†Work done while the author was employed at Microsoft.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449937

significant gains. In particular, XC algorithms can be effectively applied to sponsored search applications where retrieving the most relevant advertisement for a user query is crucial to improving user experience as well as business goals. Recently, deep XC techniques have gained popularity where diverse architectures such as multi-layer perceptron [35], CNN [31], attention [57], and transformers [9] have been utilized. These works have demonstrated that learning dense application-specific document representations can lead to better predictions than using application-agnostic features such as the traditional bag-of-words features.

Short Text Applications: This paper will focus on short-text applications that have generated significant interest in recent years [9, 10, 35, 38]. These describe situations where documents are presented with only short textual descriptions, with typically only 5-10 tokens. Examples include applications such as predicting related webpages or related products using only the title of a given webpage/product and predicting relevant ads/keywords/searches for user queries, among others. Short-text applications pose additional challenges to XC routines owing to the paucity of descriptive data available for each document.

Label Features and Tail Labels: The core utility of extreme classification comes from being able to accurately predict tail labels. These are labels that are rarely seen during training. Table 2 shows that in multiple datasets, the average label can be present in as few as 5 training documents. It becomes challenging to learn accurate label classifiers with such paucity of training data. However, XC applications often make available label metadata in various forms such as label text, label correlations or label hierarchies. Textual representations for labels can have highly relevant information imbued in them like product name, brand, or attributes (like color, size, etc.). Other examples of label text include the textual content of advertiser bid phrases in sponsored search, categories of articles on Wikipedia, and titles of products on e-commerce platforms such as Amazon, etc. These textual features can help capture semantic correlations between documents and labels promoting more accurate predictions, especially for tail labels. Furthermore, in sponsored search applications, tail queries can be longer in terms of the number of tokens and also contain more granular information in them. This makes it more advantageous to include label text in order to better service tail labels for which sufficient training data is otherwise not available [4, 23].

XC with Label Features: A few contemporary XC algorithms have explored utilizing label features. SwiftXML [41] incorporates label features based on pre-trained Word2Vec [36] vectors over the label text. However, the method does not learn task-specific label features. X-Transformers [9] obtained its label features from various sources such as pre-trained XLNet [53], Word2Vec [37], and TF-IDF embeddings. These label features were used to cluster the label set in order to fine-tune the embedding models to learn efficient document embeddings, and mine hard negatives for training the ranker. However, label features do not directly influence the learning of document features in XTransformer. The recently proposed method DECAF [38] does make direct use of label text while learning its classifier. This method will be of specific interest to GalaXC due to its use of label metadata although it does not incorporate label or document correlations in any form unlike GalaXC which learns over a joint document-label graph.

Warm-start and Auxiliary Sources of Data: In recommendation and multi-label applications, warm start [14, 30, 39, 41] refers to scenarios where a (small) subset of relevant labels for a (test) document is revealed before requiring further predictions to be made by the algorithm. For example, consider the Q2K application of matching a user query to the most relevant subset of keywords bid by advertisers. This can be formulated as an extreme classification task where the queries become the documents and the set of most monetizable keywords (from a pool of 50M available keywords) are the labels. For a lot of queries, there might already be a set of keywords available corresponding to which ad-clicks by users were previously observed [41]. These then become the warm-start labels for the Q2K problem. Although standard XC algorithms can also benefit from partially revealed ground truth, algorithms such as SwiftXML [41] have been designed specifically for warm start scenarios. It is possible to mine additional data from other sources which, although not directly related to the application of interest, may nevertheless help augment the primary sources of data. For example, for the Q2K application, information about correlations between queries can be obtained from related searches which consist of user preferences for other queries similar to the input query. Such connections obtained from supplementary sources can then help increase both the quality and quantity of data, particularly helping data scarce tail labels/documents. However, not many existing XC algorithms make effective use of such auxiliary signals.

Graph Neural Networks for XC: From another perspective, extreme classification can be seen as a link prediction problem in a bipartite graph given by $\mathbb{G} = (\mathbb{D} \cup \mathbb{L}, \mathbb{E})$ where $\mathbb{D}$ and $\mathbb{L}$ denote sets of nodes for documents and labels respectively, and $\mathbb{E}$ denotes the edges encoding relevance links i.e. there is an edge $(d, l) \in \mathbb{E}$ between document $d \in \mathbb{D}$ and label $l \in \mathbb{L}$ iff l is relevant to d i.e. l is a positive label for d. This interpretation allows rich correlation structures to be inferred. For instance, suppose documents d_1, d_2 share a common label l_1. If there is another label l_2 relevant to d_2, it can be inferred that l_2 might be relevant to d_1 as well. Such transitive inferences can be extremely valuable in XC settings where missing labels abound and training documents are seldom tagged exhaustively with all labels relevant to them (due to the inability of human annotators to offer comprehensive tag sets, impression biases, etc). However, current extreme classifiers find it difficult to model such implicit transitive relations unless the pair (d_1, l_2) explicitly presents itself in the training set [59]. Recent advancements in Graph Neural Networks (GNNs) [19, 51, 55] allow using node neighborhoods to collaboratively learn more discriminative features. However, existing works mostly use document-document graphs and not joint document-label graphs at extreme scales.

GalaXC: GalaXC employs a joint graph over documents and labels to learn superior node representations by performing graph convolutions [26] over neighborhoods of various orders. As a result, GalaXC encourages collaborative learning of document and label embeddings that enable it to better utilize partially revealed labels and additional data sources such as label metadata. However, in extreme settings with millions of labels, the label set can exhibit significant variations in terms of label semantics and each label can have a different view of the graph, some preferring a more local view whereas others benefiting from broader aggregations. To address this, GalaXC proposes a novel label attention scheme

wherein several representations are learnt for each node, using convolutions of increasing orders. A label is then allowed to attend to these multi-resolution representations while applying a label classifier. In terms of standard XC metrics, GalaXC can be up to 18% and 25% more accurate than leading extreme classifiers in cold and warm start scenarios, respectively. These improvements are demonstrated on standard benchmark datasets, on a query-to-bid sponsored search dataset, as well as on live A/B traffic on the Bing search engine. Upon deployment on the Bing search engine for Q2K applications, GalaXC offered boosts in the CY (click Yield) of 1.52%. Further, GalaXC can be 2-50× faster to train and upto 10× faster at prediction time than competing methods, allowing it to train on a dataset with 50M labels in less than 100 hours, thereby making it suitable for high query-volume time-critical applications.

Contributions: This paper makes the following contributions.

(1) It establishes the utility of using a joint document-label graph to augment document features in short-text applications where document representations may contain very few tokens.
(2) It proposes a novel per-label attention mechanism that allows each label to attend to multi-resolution graph neighbourhoods.
(3) These insights are combined to develop the GalaXC algorithm which, to best of our knowledge, is the first method to utilize a joint label-document graph to learn features in extreme settings.
(4) An efficient implementation of GalaXC is proposed that allows it to be trained much faster than existing deep extreme classifiers and scale to tens of millions of labels.
(5) It presents a case study deploying GalaXC on live traffic on the Bing search engine to show gains over an ensemble of state-of-the art generative, XC, IR and two-tower models.

2 RELATED WORK

Extreme Classification: XC algorithms can be broadly categorized into two categories: 1) methods using pre-computed features such as bag-of-words (BoW) and FastText[24], and 2) deep extreme classifiers that learn task-specific embeddings jointly with the classifiers. Method using pre-trained features can be further divided into 3 sub-categories. a) 1-vs-All classifiers methods like DISMEC [3], PPDSparse [54], ProXML [4] that achieve state-of-the-art accuracies but require $\Omega(L)$ time at prediction and $\Omega(NL)$ time to train which becomes infeasible at extreme scales. b) 1-vs-Some classifiers that employ negative sampling strategies to speed up training, e.g., Parabel [42], Bonsai [25] learn a hierarchy over labels using BoW features and use it to sample the most confusing negatives, Slice [22] uses a small world Approximate Nearest Neighbor Search (ANNS) graph to sample the hardest negatives. c) Tree-based classifiers such as MLRF [2], FastXML [43], PfastreXML [23] that learn an ensemble of trees to partition the label space, where each node in the tree is split to optimize an objective such as the Gini index or nDCG. Unfortunately, the negative sampling data structures used by 1-vs-Some methods are not suitable when feature representations are jointly learnt with the classifiers. Tree-based algorithms can have large training times and model sizes. The above methods also do not make use of label features. On the other hand, deep XC techniques like X-Transformer [9], Astec [10], XML-CNN [31], MACH [35], DECAF [38], and AttentionXML [57] have shown better performance by learning task-specific representations. Of these, X-Transformer and DECAF employ label features as well. However, high capacity architectures such as CNN, RNN or attention employed by these methods can be inaccurate on short-text documents where limited training data is available. Moreover, the model ensembles learnt by MACH, X-Transformer and the label attention employed by AttentionXML do not scale well to extreme settings.

Warm Start: XC methods have also been developed for warm start scenarios [14, 30, 39, 41]. Notable is SwiftXML [41] which is a tree-based classifier using sparse BoW features for documents and pre-computed dense text embeddings [37] for features. SwiftXML partitions tree nodes using two hyper-planes learnt jointly in the label and document feature spaces.

Neural Networks over Graphs: Earlier methods [17, 45, 48] relied on RNNs that used neighborhood information to iterate over node embeddings until convergence. More recent algorithms applying convolutions over the graph can be divided into two categories – spectral and non-spectral. Spectral approaches [8, 11, 26] are based on an eigen-decomposition of the graph Laplacian. A major limitation with these methods is that they require retraining the spectral kernels even with small perturbations in the graph structure. Non-spectral methods [19, 49, 51] define convolutions on spatially proximal neighborhoods of a node. GraphSAGE [19] proposed the computation of node representations in an inductive way by recursively aggregating over fixed-sized neighborhoods. [51] provided theoretical foundations for GNNs based on the Weisfeiler Lehman (WL) test and proposed the Graph Isomorphism Network (GIN) with discriminative power equal to that of the WL test.

Attention Mechanisms: Using attention to focus on the most relevant parts of the input while making decisions has become a standard choice in many tasks [5, 15]. In extreme classification, AttentionXML [57] introduced the idea of per-label attention where for each label, the network attends to the most relevant parts of the document embedding. However, the complexity of the method increases with the number of labels making it unsuitable for large scale applications. More recently, there has been a growing interest towards applying attention in GNNs. Graph Attention Networks (GAT) [49] exploits the fact that node representation have different level of similarity with their neighbors and uses multi-head attention for feature aggregation with increased model capacity.

3 PROBLEM SETTING

We recall the interpretation of extreme classification as a link prediction problem on a vast bipartite graph as discussed in Section 1. This section formalizes this problem and identifies issues in directly applying existing GNN techniques to this task.

Notation Let L be the number of labels, V be the dictionary size, and N be the number of training documents. For each document $i \in [N]$, $\mathbf{y}_i \in \{-1, +1\}^L$ denotes its ground truth label vector with $y_{il} = +1$ if label $l \in [L]$ is relevant to the i^{th} document and $y_{il} = -1$ otherwise. The documents and labels are arranged in a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with $\mathbb{V} = \mathbb{D} \cup \mathbb{L}$ where $\mathbb{D}, \mathbb{L}$ are respectively the set of documents and labels[1]. $\mathbb{E} \subset \mathbb{D} \times \mathbb{L}$ is the set of edges denoting ground truth relation between documents and labels i.e.

[1] In some applications, such as related webpage prediction or related product prediction, some train documents may act as labels for other train documents. In such cases, the sets $\mathbb{D}, \mathbb{L}$ are not disjoint and the graph has less than $N + L$ nodes.

Num. of hops (K)	P@1	P@5	PSP@1	PSP@5
1	17.63	09.07	15.77	17.79
2	15.30	07.72	13.75	15.22
3	11.85	05.94	10.21	11.05

Table 1: Performance of GraphSage deteriorates with increasing number of hops on LF-WikiSeeAlsoTitles-320K. A fanout of 10 was used in each of the hops.

for document $i \in [N]$ and label $l \in [L]$, we have $(i, l) \in \mathbb{E}$ iff $y_{il} = +1$. For each document $i \in [N]$ and label $l \in [L]$, a D-dimensional initial representation of their textual features, respectively $\hat{\mathbf{x}}_i^0$ and $\hat{\mathbf{z}}_l^0$, is available (for instance using pre-trained FastText [7], BERT [12, 32], or CDSSM [47] architectures). To simplify the notation, for a node $n \in \mathbb{V}$, its D-dimensional initial representations will be referred to as $\hat{\mathbf{v}}_n^0$ (with $\hat{\mathbf{v}}_n^0 \equiv \hat{\mathbf{x}}_i^0$ if the node n is the document $i \in [N]$ and $\hat{\mathbf{v}}_n^0 \equiv \hat{\mathbf{z}}_l^0$ if node n is the label $l \in [L]$).

Graph Neural Networks: For tasks such as node classification [19], link prediction [58], etc, GNNs refine a node's initial embedding by recursively aggregating embeddings of nodes in its neighbourhood. Given a notion of neighborhood $\mathcal{N} : \mathbb{V} \to 2^{\mathbb{V}}$, GNNs repeatedly apply *convolution* $\mathcal{C}$ and *transformation* $\mathcal{T}$ operations to obtain refined node embeddings. Let $\hat{\mathbf{v}}_n^k$ denote the embedding of node $n \in \mathbb{V}$ after k such operations, or k *hops*, then

$$\hat{\mathbf{a}}_n^k = \mathcal{C}_k(\{\hat{\mathbf{v}}_m^{k-1}, \hat{\mathbf{a}}_m^{k-1} : m \in \mathcal{N}(n)\})$$
$$\hat{\mathbf{v}}_n^k = \mathcal{T}_k(\{\hat{\mathbf{v}}_n^{k-1}, \hat{\mathbf{a}}_n^{k-1}\}),$$

where for $k = 0$, we use $\hat{\mathbf{a}}_n^0 = \hat{\mathbf{v}}_n^0$ as a default. For sake of scalability, it is common to use subsampled neighborhoods, e.g. use only $|\mathcal{N}(v)| = \text{FO}[k]$ neighbors of node v during the convolution operation at the k^{th} hop. The fanout FO$[k]$ is tuned for hop lengths. GNNs variants differ in their convolution [19, 27, 49, 56] and transformation [16, 19, 26, 51, 52] operations, among other factors.

Shortcomings of GNNs for XC To study the applicability of existing GNN methods to XC problems, a GraphSAGE [19] model was trained over the document-label graph on a link prediction task with a total of K hops. At prediction time, labels were ranked using an Approximate Neighbor Neighbor Search (ANNS) [34] structure over the label embeddings $\hat{\mathbf{z}}_l^K : l \in [L]$ obtained after the last hop. The performance of the model (as measured by precision at 1,3,5 – see Table 1) deteriorated with increasing K. This corroborates other works [1, 60] that also observed that GNNs are unable to make optimal use of higher-order neighborhoods.

4 GalaXC: Graph neurAl networks with Labelwise Attention for eXtreme Classification

Summary: GalaXC consists of three components 1) a light-weight GNN architecture that effectively utilizes multi-hop neighborhoods at extreme scales to offer multi-resolution embeddings for documents, 2) high-capacity extreme classifiers and a label-wise attention mechanism over the multi-resolution embeddings, and 3) a scalable mechanism to incorporate test documents into the graph to make predictions in both cold- and warm-start settings.

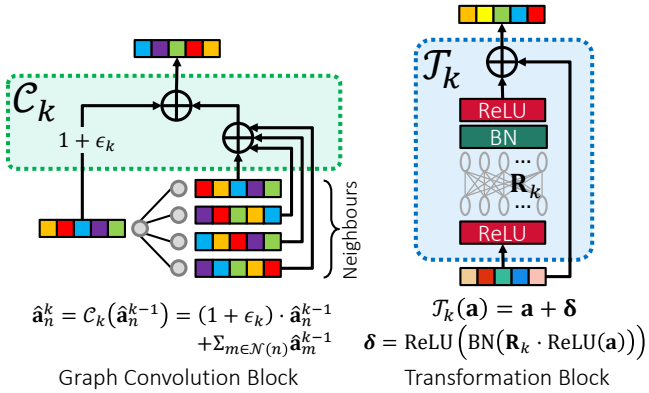

Figure 1: GalaXC uses light-weight yet effective components to build its multi-layer GNN (see Fig 2) with each layer k using a fresh instance $\mathcal{C}_k, \mathcal{T}_k$ of the components. The graph convolution block (Left) is parameterized by a scalar ϵ_k whereas the transformation block (Right) implements a residual layer parameterized by a matrix $\mathbf{R}_k \in \mathbb{R}^{D\times D}$, followed by batch-normalization and ReLU layers.

A Multihop-friendly GNN: Given initial embeddings $\hat{\mathbf{v}}_n^0, n \in \mathbb{V}$ for the nodes of the graph (recall that these are simply the initial embeddings for the documents $\hat{\mathbf{x}}_i^0, i \in [N]$ and labels $\hat{\mathbf{z}}_l^0, l \in [L]$) GalaXC uses light-weight convolution and transformation operators to learn embeddings for documents and labels in a way that benefits from multi-hop convolutions with several advantages:

(1) By enabling collaborative learning and the use of transitive relations such as those discussed in Section 1, this benefits tail labels for which scarce training data is available.
(2) This also benefits short-text applications in general where textual descriptions of documents and labels may not be very expressive and contain only 5-10 tokens.
(3) It allows additional sources of information e.g. warm starts, to be incorporated efficiently at prediction time.

The convolution operator $\mathcal{C}_k$ used by GalaXC is described in Fig 1. The weighted-sum aggregator [51], parameterized by a scalar ϵ_k, was used to implement convolutions.

$$\hat{\mathbf{a}}_n^k = (1+\epsilon_k)\cdot\hat{\mathbf{a}}_n^{k-1} + \sum_{m\in\mathcal{N}(n)} \hat{\mathbf{a}}_m^{k-1}$$

For every node $n \in \mathbb{V}$, a random set of FO$[k]$ neighbors was sampled for every node (see Table 3 for hyperparameters) to construct $\mathcal{N}(n)$. As a default, $\hat{\mathbf{a}}_n^0$ was set to the initial embeddings $\hat{\mathbf{v}}_n^0$ for every node. We note that the above architecture does not introduce any non-linearity between successive convolutions. Instead, the transformation block is used to convert convolved embeddings $\hat{\mathbf{a}}_n^k$ at a certain layer into final embeddings $\hat{\mathbf{v}}_n^k$ i.e.

$$\hat{\mathbf{v}}_n^k = \mathcal{T}_k(\hat{\mathbf{a}}_n^k),$$

where the transformation block $\mathcal{T}_k$ (see Fig 1) implements a residual layer parameterized by a matrix $\mathbf{R}_k \in \mathbb{R}^{D\times D}$ and uses ReLU [28] and batch-normalization (BN) [21] layers and a skip connection [20]. Similar residual and skip connection-based architectures have been

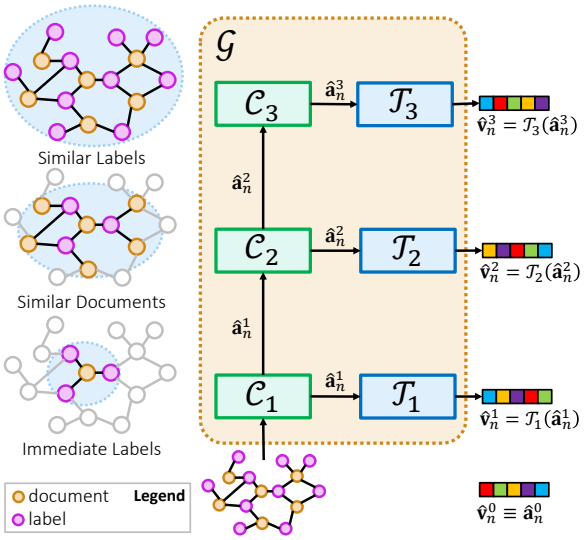

Figure 2: GalaXC's multi-hop GNN architecture allows a document to, in successive hops, aggregate information from its own labels, then related documents with which it shares labels, and finally other labels present in related documents.

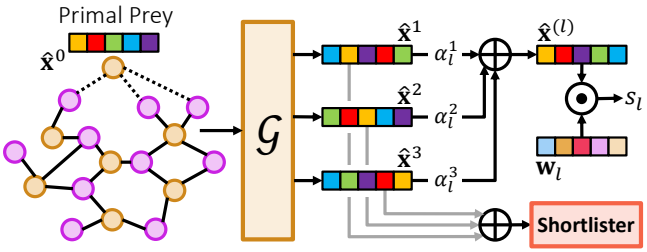

Figure 3: GalaXC's frugal prediction pipeline scales to millions of labels. Given a document, its initial embedding $\hat{\mathbf{x}}^0$ is used to introduce it into the graph. The GNN architecture is then used to obtain multi-resolution embeddings that are used to both shortlist the most likely labels, as well as make final predictions using the label-wise attention mechanism.

widely adopted in many applications [20, 29]. To implement multiple hops, each layer of GalaXC's GNN uses a fresh instance of these components. Fig 2 shows GalaXC's overall GNN architecture.

1-vs-All Label Classifiers and Label-wise Attention: GalaXC uses high capacity 1-vs-All (OvA) classifiers $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L] \in \mathbb{R}^{D \times L}$ that outperform tree- and embedding-based classifiers [3, 4, 9, 22, 42, 54]. However, GalaXC also exploits the multi-resolution embeddings for documents $\hat{\mathbf{x}}_i^k, k \in [K]$ offered by the GNN architecture, to allow each label to weigh these resolutions differently. For example, tail labels do not have sufficient context because of their low connectivity in the graph, and might benefit from embeddings that cover a larger neighborhood, which is offered by larger k whereas data-rich head labels might prefer focussed embeddings offered by small k. This phenomenon is confirmed in experiments (see Fig 6). To effect label-wise attention, real scalars $e_{lk} \in \mathbb{R}$ are learnt for every label $l \in [L]$ and layer $k \in [K]$ in the GNN. Attention weights are obtained by using a softmax operation i.e. $\alpha_{lk} = \exp(e_{lk}) / \sum_{k' \in [K]} \exp e_{lk'}$. Given multi-resolution embeddings $\hat{\mathbf{x}}^k, k \in [K]$ for a document, when calculating the score for a label $l \in [L]$, first a label-specific embedding is calculated as

$$\hat{\mathbf{x}}^{(l)} = \sum_{k \in K} \alpha_{lk} \cdot \hat{\mathbf{x}}^k,$$

and then the 1-vs-all classifier is applied to obtain the score for this label as $s_l = \left\langle \mathbf{w}_l, \hat{\mathbf{x}}^{(l)} \right\rangle$. GalaXC's label-wise attention mechanism requires learning $O(LK)$ additional parameters where $K = 3$ which increases training time by around 10%. However, it is far more cost-effective than attention mechanisms employed by methods such as AttentionXML [57] which are more expensive yet less accurate.

Cold- and Warm-start Prediction Pipeline: For the standard cold start setting where test documents are not part of the document-label graph used for training, new documents need to be first introduced into the graph. This is done by constructing an ANNS graph using the initial embeddings of all training labels and documents i.e. $\hat{\mathbf{x}}_i^0, i \in [N]$ and $\hat{\mathbf{z}}_l^0, l \in [L]$. Given the initial embedding of a test document $\hat{\mathbf{x}}^0$, edges are introduced to its 4 nearest neighbors. Note that these neighbors may be labels or even train documents. In warm start settings, edges are introduced to the warm-start labels instead. Having done this, the GNN architecture is used to obtain multi-resolution embeddings $\hat{\mathbf{x}}^k, k \in [K]$ for the test document.

Shortlisting and Prediction: XC applications demand predictions in milliseconds which necessitates a prediction time complexity no worse than $O(D \log L)$ whereas evaluating label scores for all labels takes $\Omega(DL)$ time. To remedy this, a shortlister is created to return, for any document, a set $\mathcal{S}$ of $O(\log L)$ labels that seem most relevant to it. For a test document, label-wise embeddings $\hat{\mathbf{x}}^{(l)}$ are created only with respect to labels $l \in S$. To create the shortlister, multi-resolution embedings of labels are averaged to obtain $\bar{\mathbf{z}}_l = \frac{1}{K} \sum_{k \in [K]} \hat{\mathbf{z}}_l^k$ and a second ANNS graph is created over these averaged embeddings. Given a test document, its multi-resolution embeddings (obtained as shown above) are similarly averaged to obtain $\bar{\mathbf{x}} = \frac{1}{K} \sum_{k \in [K]} \hat{\mathbf{x}}^k$ and the top 500 neighbors of $\bar{\mathbf{x}}$ are shortlisted to form the set $\mathcal{S}$ of potential labels for which label-wise embeddings are then calculated to rank them. Fig 3 shows the frugal prediction pipeline adopted by GalaXC that performs prediction in $O(D \log L)$ time. GalaXC's shortlister was found to offer much better recall than if using the initial embeddings to create shortlists. For example, on the LF-AmazonTitles-131K dataset, the recall for the top 100 labels shortlisted using the initial and convolved embeddings were around 49% and 64% respectively.

4.1 Efficient Training: the DeepXML Pipeline

Summary: GalaXC adopts the scalable DeepXML pipeline [10] that splits training into 4 *modules*. However, unlike previous works such as Astec [10] and DECAF [38] which also use this pipeline, GalaXC executes a single training run, with the first few epochs executing Module I and the rest of the epochs executing Module IV. In summary, Module I jointly learns the GNN architecture, extreme

classifiers, and attention model weights using online and random negative mining. Module II retrieves label shortlists for all data points to make further training more efficient using hard negatives. Module III simply restarts training with model parameters set to their values at the end of Module I. Module IV then continues training but with shortlisted negative labels instead.

Initial Embeddings: Although GalaXC is agnostic to the initial embeddings used for labels and documents, for experiments, pre-trained tokens from Astec [10] or CDSSM [47] were chosen to be combined to obtain initial embeddings (see Fig 4). Both documents and labels were presented as sparse bag-of-tokens vectors i.e. document i is presented as $\mathbf{x}_i \in \mathbb{R}^V$ where x_{it} is the TF-IDF weight of token $t \in [V]$ in the i^{th} document. Similarly, label l is presented as $\mathbf{z}_l \in \mathbb{R}^V$. Given token embeddings for all V tokens in the vocabularly i.e. $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_V] \in \mathbb{R}^{D \times V}$, initial embeddings were obtained as $\hat{\mathbf{x}}_i^0 = \mathbf{E}\mathbf{x}_i, \hat{\mathbf{z}}_l^0 = \mathbf{E}\mathbf{z}_l$. This light-weight embedding architecture has been shown to be well-suited for short-text applications [10, 38]. To be sure, GalaXC does not fine-tune the token embeddings further and consequently, initial embeddings remain frozen throughout training.

Loss Function, Regularization and Initialization: The binary cross-entropy loss was used with the Adam optimizer. ϵ_k values in the convolution blocks C_k were initialized to uniformly random values in $[0, 1)$. A dropout layer with rate 0.5 was used after every ReLU layer. The residual matrices $\mathbf{R}_k$ in the transformation blocks were initialized as identity and batch normalization parameters were initialized to $\gamma = 1$ and $\beta = 0$. The pre-attention weights i.e. e_{lk} were initialized using the normal distribution $\mathcal{N}(0, 1)$. The extreme 1-vs-All classifiers were initialized by drawing each of their coordinates from the uniform distribution $\mathcal{U}(-a, a), a = 1/\sqrt{(D)}$ where D is dimensionality of the learnt embeddings. We refer to Section 5 for details of other hyperparameters.

Negative Mining: Learning 1-vs-All classifiers at extreme scales poses computational challenges as doing so naively requires $\Omega(NDL)$ time where D is the dimensionality of the embeddings being learnt. This is infeasible when working with millions of labels. A common remedy is to instead train a document with all its positive labels (of which there are typically only $O(\log L)$) but only a $O(\log L)$-sized subset of its negative labels. Choosing "hard" negatives, which are at most risk of getting mispredicted as positives, makes training more efficient. GalaXC uses online and random negative mining for Module I and does hard negative mining in Module II.

Module I: The GNN parameters, i.e., $\epsilon_k, \mathbf{R}_k, k \in [K]$ and batch-normalization parameters are jointly trained with the pre-attention weights e_{lk} and the extreme 1-vs-All classifiers $\mathbf{w}_l, l \in [L]$. Given a mini-batch B of training documents, a random set R of labels is sampled (see Tab 3 for hyperparameter values). For every document $i \in B$ in this mini-batch, a set $\hat{N}_i$ of negative labels is obtained by combining labels negative for document i among positive labels of other documents in the mini-batch and those among the sampled label set R i.e. $\hat{N}_i := \{l : y_{il} = -1, y_{jl} = +1, j \in B\} \cup \{l : y_{il} = -1, l \in R\}$. Binary cross entropy loss terms are computed only with respect to labels in the set $P_i \cup \hat{N}_i$ where $P_i := \{l : y_{il} = +1\}$ is the set of positive labels of document i. Since typically $|P_i| \leq O(\log L)$ and $|R|$ is restricted, we have $|P_i \cup \hat{N}_i| \ll L$ which allows efficient training.

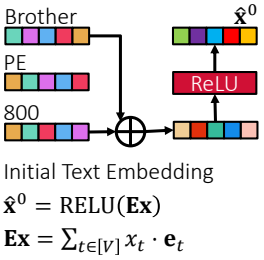

Figure 4: GalaXC uses pre-trained token embeddings to obtain initial embeddings for documents and labels. However, a variety of other architectures, e.g. BERT, could be used to obtain initial embeddings.

Module II: In this module, a shortlister similar to the one used during prediction based on averaged embeddings of labels i.e. $\bar{\mathbf{z}}_l = \frac{1}{K} \sum_{k \in [K]} \hat{\mathbf{z}}_l^k$, is used to create an ANNS structure using which label shortlists are drawn for each training document $i \in [N]$. Excluding the positive labels of this document from this shortlist yields a set S_i of negative labels for this document. These are arguably hard negatives for this document as the shortlister found it challenging to immediately distinguish them from the positive labels.

Modules III and IV: Parameters of convolution and transformation layers are fixed and training of extreme classifiers and attention weights is now resumed but this time, for a document i, binary cross entropy loss terms are computed only with respect to labels in the set $P_i \cup \hat{N}_i$ where $\hat{N}_i$ is now defined as the set of negative labels for document i among the positive and shortlisted hard negative labels for other documents in the mini batch i.e. $\hat{N}_i := \{l : y_{il} = -1, y_{jl} = +1, j \in B\} \cup \{l : y_{il} = -1, l \in S_j, j \in B\}$ where S_j is the set of hard negatives for document j obtained in Module II. We note that on some of the datasets, GalaXC did not require hard negative mining and the online + random negative mining strategy used in Module I sufficed. For such datasets, Modules II, III, IV are never used.

5 EXPERIMENTS

Datasets and features GalaXC's performance is benchmarked on three publicly available datasets namely LF-AmazonTitles-131K, LF-WikiSeeAlsoTitles-320K and LF-AmazonTitles-1.3M. The applications range from predicting frequently co-bought Amazon products (LF-AmazonTitles-131K and LF-AmazonTitles-1.3M) to recommending related articles on Wikipedia web page (LF-WikiSeeAlsoTitles-320K). Experiments were also conducted on two proprietary datasets (Q2K-450K and Q2K-50M) mined from the click logs of the Bing search engine. These required a user query to be mapped to the most relevant subset of advertiser bid phrases. To conduct offline experiments, a dataset was created by mining Bing search logs for a specific time period. Each user query was considered a document and the clicked bid phrases of the corresponding surfaced advertisements became its labels. The pairs obtained were passed through a basic sanity filter based on click-through rate (CTR), clicks and impressions. The final dataset, named Q2K-450K, had around 450K labels and 541K training points (Refer to Tab 2 for more details). GalaXC gave state-of-the-art results in warm-start scenario

Table 2: Dataset Statistics. A ‡ sign denotes information that was redacted for proprietary datasets. The first three rows are public datasets and the last two rows are proprietary datasets.

Dataset	Train Documents N	Labels L	Test Instances N'	Average Labels per Document	Average Points per label
Short text dataset statistics					
LF-AmazonTitles-131K	294,805	131,073	134,835	2.29	5.15
LF-WikiSeeAlsoTitles-320K	693,082	312,330	177,515	2.11	4.68
LF-AmazonTitles-1.3M	2,248,619	1,305,265	970,237	22.20	38.24
Proprietary dataset					
Q2K-450K	541,197	443,146	135,250	‡	‡
Q2K-50M	97,205,641	50,000,000	24,301,410	‡	‡

as well. The LF-AmazonTitles-131K and LF-WikiSeeAlsoTitles-320K datasets were modified for this purpose by: 1) Retaining only those test documents which have at at least two labels. 2) For the retained documents, revealing half the labels and using the other half to evaluate the model. 3) Making available the test documents and their revealed labels to all algorithms at training time.

Baseline algorithms GalaXC was compared to leading deep extreme classifiers including DECAF [38], DeepXML [10], MACH [35], XTransformer [9], and AttentionXML [57], as well as extreme classification algorithms based on fixed features such as Bonsai [25], Parabel [42], DiSMEC [3] and Slice [22]. For warm-start settings, results on SwiftXML [41] are also presented. Results are only presented for methods where the available implementation could scale to a dataset on a single GPU within a week. Implementations of all aforementioned algorithms were provided by their respective authors. The hyper-parameters of all baseline algorithms were set as suggested by their authors wherever applicable and by fine-grained validation otherwise. For all the baselines which require an ANNS index to make predictions, HNSW [34] with hyper parameters $M = 100, efC = 300, efS = 100$ was used. Also, most baseline algorithms build an ensemble of models. For such baselines, the prescribed ensemble was used for comparison. However, it is notable that GalaXC uses a single model that decreases maintenance costs in production as well as training and prediction times.

Evaluation Performance was evaluated on traditional extreme classification metrics, like Precision@k (P@k), Normalized Discounted Cumulative Gain@k(N@k) and their propensity scored variants (PSP@k, PSN@k) [23] at different truncation levels k=1,3,5. For evaluation, guidelines provided on the XML repository [6] were followed. In the cold start setting, all prediction points were presented to all algorithms simultaneously in a batch. This is similar to several real world scenarios where predictions are to be made in an offline fashion for all test documents and don't require streaming predictions. GalaXC could utilize this to add connections among test points in addition to training documents and labels. In warm start settings, labels which were partially revealed were filtered out from both predictions as well as the ground truth at the time of evaluation. Training time in hours (TT), per document prediction time in milliseconds (PT), and the model size in GigaBytes (MS) are also reported. All metrics on public datasets were benchmarked on a 24-core Intel Xeon 2.6 GHz machine with a single V100 GPU. While reporting model sizes, node embeddings were included along with the model parameters as these are required for prediction.

Hyper-parameters GalaXC's tuneable hyper-parameters include the batch size, number of random negatives to sample per-batch in Module I, the number of epochs in Module I and Module IV, and the learning rate. Tab 3 reports the hyper-parameters used in the experiments. For single GPU training, the number of random negatives per mini-batch in Module I was kept as $0.1L$ subject to a minimum of 30K and maximum of 100K. For larger datasets where a parallel implementation was done by splitting the classifiers across multiple GPUs, 100K labels were independently sampled per GPU. For Module IV training, 500 hard negatives were sampled in Module II per training document. Additional hyper-parameters for the GNN include the number of hops(K), fanouts for different hops, and initial embeddings for labels and documents. $K = 3$ was used for all datasets. For all other parameters, please refer to Tab 3. For smaller datasets, Module IV training was not required as no significant performance improvement was observed. For the LF-AmazonTitles-1.3M dataset, which has high connectivity between nodes, considering only the top 5 neighbors for head labels for graph construction (based on ANNS with $\hat{\mathbf{x}}^0$) gave slightly better results. At prediction time, a test document was introduced in the graph by connecting it to closest 4 nodes (based on ANNS with $\hat{\mathbf{x}}^0$ or CTR in case of the Aux-Test setting – see Tab 10).

Results on benchmark datasets: Tab 5 presents the results of GalaXC and baseline algorithms in the standard cold start setting. Compared to leading deep extreme classifiers like Astec, MACH and AttentionXML which don't use label metadata, GalaXC can be up to 18% more accurate, while being 2-50× faster to train on a single GPU. Specifically, GalaXC is at least 4% better than AttentionXML (which also employs label attention) in P@5 which helps demonstrate the efficacy of the attention mechanism used by GalaXC. Further, GalaXC is 1-3% better than Astec in P@5 indicating the benefit of collaborative feature learning though the joint graph. Compared to DECAF and X-Transformer, both of which utilizes label features for training, GalaXC is upto 3% and 11% more accurate in terms of PSP@1, respectively. Furthermore, GalaXC is around 7× faster to train than DECAF. GalaXC is up to 15% better in P@1 compared to extreme classifiers like Slice, Parabel, Bonsai that use BoW or fixed features , indicating that prediction quality can be improved by learning task-specific features. Further, GalaXC leads

Table 3: Hyper-parameters for different datasets for GalaXC. Astec refers to 300-dimensional embeddings obtained from Astec's surrogate task. CDSSM refers to proprietary 64-dimensional CDSSM embeddings, fine tuned on queries and advertiser keywords. For all datasets, a starting learning rate of 0.001 was used. The learning rate was reduced by the DLR factor after each epoch as per the LR schedule. A dropout with probability 0.5 was used consistently.

Dataset	Init Embeddings	Fanouts	Num Batch random	Num Epochs Module I, IV	Batch Size	LR schedule	DLR factor
LF-AmazonTitles-131K	Astec	4, 3, 2	30000	30, 0	256	5, 10, 15, 20, 25, 28	0.5
LF-WikiSeeAlsoTitles-320K	Astec	4, 3, 2	32000	30, 0	256	5, 10, 15, 20, 25, 28	0.5
LF-AmazonTitles-1.3M	Astec	3, 3, 3	100000	24, 15	512	4,8,12,16,18,20,22	0.5
Q2K-450K	Astec	4, 3, 2	45000	30, 0	512	5, 10, 15, 20, 25, 28	0.5
Q2K-50M	CDSSM	3, 3, 3	100000 per GPU split	4, 0	1024	0, 1, 2, 3	0.5

Table 4: Recall@100 when predictions are made by querying test document embeddings on an ANNS index built on label embeddings offered by various methods.

Dataset	fastText	Astec	DECAF	GalaXC
LF-AmazonTitles-131K	38.15	49.04	51.41	64.03
LF-WikiSeeAlsoTitles-320K	34.05	33.50	33.69	49.21

to disproportionately more gains in terms of propensity scored precision metrics, which reward predicting tail labels more, indicating that GalaXC can predict tail labels better. Specifically, compared to DECAF, GalaXC can be up to 3% better in PSP@5 metric. Also, as indicated in Fig 5, better predictions on tail labels are the major contributor to GalaXC's performance gain over methods like DECAF, Astec, MACH and AttentionXML. The document and label embeddings trained collaboratively using GalaXC can be of independent interest for extremely low latency applications which disallow the use of extreme classifiers and only allow ANNS calls at prediction time. Table 4 shows that GalaXC's embeddings offer 13% and 15% better recall@100 than DECAF and Astec embeddings on the LF-AmazonTitles-131K dataset.

Warm start setting Tab 6 compares GalaXC with various baselines on the warm start setting. Compared to SwiftXML which is specifically designed for the warm start scenario, GalaXC is up to 8% and 25% better in P@5 and PSP@5 metrics respectively. GalaXC also outperforms other XC methods like DECAF, Astec by upto 10%. This could be attributed to the novel graph based document encoding block of the GalaXC where multi-hop embeddings of the same document can efficiently encode text embeddings of partially revealed labels.

Ablations: GalaXC's accurate and scalable architecture is the result of carefully chosen design choices. Other possible choices for the various components of the GalaXC are discussed below.

(1) **Encoder architectures:** Experiments were conducted with different graph encoders like GraphSAGE [19] and GIN [51]. Tab 7 shows that GalaXC's per-label attention block can be upto 2% and 4% more accurate as compared to GIN and GraphSAGE, respectively. Experiments were also conducted by removing the per-label attention mechanism in GalaXC (GalaXC-NoAtt) and averaging the embeddings from various hops. GalaXC was found to be more accurate than GalaXC-NoAtt which justifies the use of per-label attention over the multi-resolution embeddings.

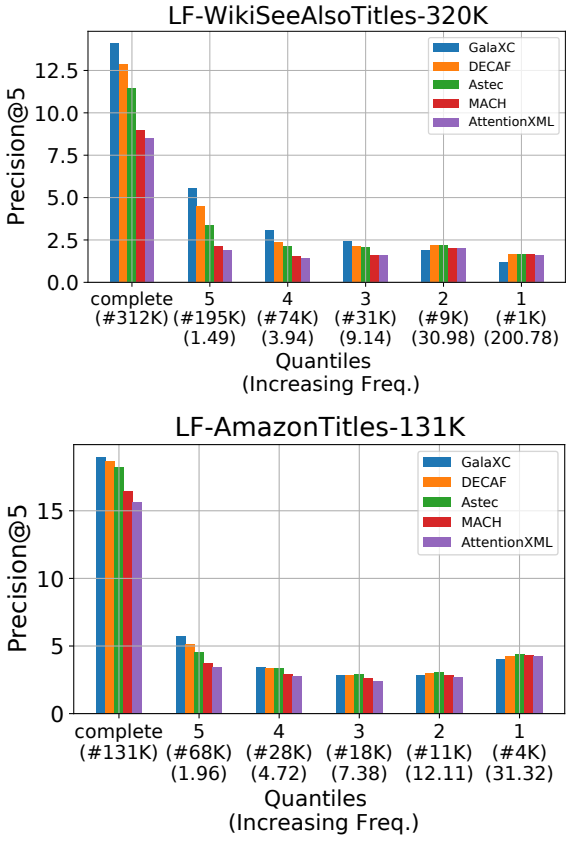

Figure 5: Gains of GalaXC over existing methods on different equi-voluminous quantiles based on label popularity.

(2) **Initial embedding**: As GalaXC is agnostic to the choice of initial node embeddings, experiments were conducted with pre-trained embeddings from Astec [10], DECAF [38], and FastText [7]. FastText embeddings were trained on the Wikipedia corpus [7]. whereas Astec [10] and DECAF [38] embeddings were extracted after training their respective surrogate tasks. GalaXC trained with Astec embeddings was found to work better than DECAF as well as FastText. Please refer to Tab 8 for more details.

Qualitative Analysis A plot of average attention weights for label bins with increasing label frequency (Fig 6) reveals that data

Table 5: Results on standard cold start setting. TT = train time, PT = prediction time, MS = model size. GalaXC can be 18% more accurate and up to 50× faster to train than leading deep extreme classifiers like DECAF, Astec, MACH, AttentionXML and XTransformer, and an order of magnitude faster at prediction time. Results are only provided for algorithms that could scale to a dataset.

Method	P@1	P@3	P@5	N@3	N@5	PSP@1	PSP@3	PSP@5	PSN@3	PSN@5	TT (hr)	PT (ms)	MS (GB)
LF-AmazonTitles-131K													
GalaXC	**39.17**	**26.85**	**19.49**	**40.82**	**43.06**	**32.50**	**38.79**	**43.95**	**36.86**	**39.37**	0.42	1.54	0.67
DECAF	38.4	25.84	18.65	39.43	41.46	30.85	36.44	41.42	34.69	37.13	2.16	0.1	0.81
XTransformer	29.95	18.73	13.07	28.75	29.6	21.72	24.42	27.09	23.18	24.39	-	15.38	-
Astec	37.12	25.2	18.24	38.17	40.16	29.22	34.64	39.49	32.73	35.03	1.83	2.34	3.24
MACH	33.49	22.71	16.45	34.36	36.16	24.97	30.23	34.72	28.41	30.54	3.3	0.23	2.35
AttentionXML	32.25	21.7	15.61	32.83	34.42	23.97	28.6	32.57	26.88	28.75	20.73	5.19	2.61
Bonsai	34.11	23.06	16.63	34.81	36.57	24.75	30.35	34.86	28.32	30.47	0.1	7.49	0.24
Slice	30.43	20.5	14.84	31.07	32.76	23.08	27.74	31.89	26.11	28.13	0.08	1.58	0.39
Parabel	32.6	21.8	15.61	32.96	34.47	23.27	28.21	32.14	26.36	28.21	0.03	0.69	0.34
DiSMEC	35.14	23.88	17.24	36.17	38.06	25.86	32.11	36.97	30.09	32.47	3.1	5.53	0.11
LF-WikiSeeAlsoTitles-320K													
GalaXC	**27.87**	**18.75**	**14.30**	**26.84**	**27.60**	**19.77**	**22.25**	**24.47**	**21.70**	**23.16**	1.08	1.65	1.56
DECAF	25.14	16.90	12.86	24.99	25.95	16.73	18.99	21.01	19.18	20.75	11.16	0.09	1.76
Astec	22.72	15.12	11.43	22.16	22.87	13.69	15.81	17.5	15.56	16.75	4.17	2.67	7.3
MACH	18.06	11.91	8.99	17.57	18.17	9.68	11.28	12.53	11.19	12.14	8.23	0.52	2.51
AttentionXML	17.56	11.34	8.52	16.58	17.07	9.45	10.63	11.73	10.45	11.24	56.12	7.08	6.02
Bonsai	19.31	12.71	9.55	18.74	19.32	10.69	12.44	13.79	12.29	13.29	0.37	14.82	0.37
Slice	18.55	12.62	9.68	18.29	19.07	11.24	13.45	15.2	13.03	14.23	0.2	1.85	0.94
Parabel	17.68	11.48	8.59	16.96	17.44	9.24	10.65	11.8	10.49	11.32	0.07	0.8	0.6
DiSMEC	19.12	12.93	9.87	18.93	19.71	10.56	13.01	14.82	12.7	14.02	15.56	11.02	0.19
LF-AmazonTitles-1.3M													
GalaXC	49.81	44.23	40.12	47.64	46.47	**25.22**	**29.12**	**31.44**	**27.81**	**29.36**	9.55	2.69	5.8
DECAF	**50.67**	**44.49**	**40.35**	**48.05**	**46.85**	22.07	26.54	29.3	25.06	26.85	74.47	0.16	9.62
Astec	48.82	42.62	38.44	46.11	44.8	21.47	25.41	27.86	24.08	25.66	18.54	2.61	26.66
MACH	35.68	31.22	28.35	33.42	32.27	9.32	11.65	13.26	10.79	11.65	60.39	2.09	7.68
AttentionXML	45.04	39.71	36.25	42.42	41.23	15.97	19.9	22.54	18.23	19.6	380.02	29.53	28.84
Bonsai	47.87	42.19	38.34	45.47	44.35	18.48	23.06	25.95	21.52	23.33	7.89	39.03	9.02
Slice	34.8	30.58	27.71	32.72	31.69	13.8	16.87	18.89	15.62	16.74	0.79	1.45	5.98
Parabel	46.79	41.36	37.65	44.39	43.25	16.94	21.31	24.13	19.7	21.34	1.5	0.89	11.75

starved tail labels do prefer document embeddings convolved over broader neighborhoods as compared to head labels which tend to pay more attention to the local neighborhood, possibly to avoid topic drift. Tail labels pay comparatively more attention to the related documents (hop 2), while head labels pay more attention to the document's own relevant labels (hop 1) when computing the document's label-specific embedding i.e. $\hat{\mathbf{x}}^{(l)}$. Additionally, barring extreme tail, all quantiles consistently give lower weight to hop 3 embeddings. Nonetheless, eliminating the third hop altogether and using a variant of GalaXC with just $K = 2$ hops was found to be around 2% worse in P@1, indicating that GalaXC does offer benefit from higher order neighborhood information. To further understand the gains offered by GalaXC, Fig 5 divides the label set into 5 equi-voluminous label bins and provides the contribution of each bin to the total precision score of a method. GalaXC's primary gains are due to more accurate tail label predictions which is essential for real-world scenarios. Tab 9 shows an example where GalaXC offered qualitatively better predictions than leading deep extreme classifiers. Specifically, DECAF, which uses label text alone, retrieves the labels "Prey" and "Primal" wrongly based on token similarity with the document. However, in the joint document-label graph employed by GalaXC, the test document "Primal Prey" is connected to the training document "Carnivores" which in turn is connected to the nodes "Carnivores 2", "Carnivores: Ice Age", "Carnivores 2 (Jewel Case)", "Primal Prey", "Carnivores: Cityscape". GalaXC is thus able to make use of higher order graph correlations and retrieve the correct predictions.

Table 6: Results on warm start setting. GalaXC is upto 25% more accurate than SwiftXML specifically designed for warm start settings and also outperform deep extreme classifiers like DECAF, Astec, MACH, and AttentionXML by 17%.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
LF-AmazonTitles-131K						
GalaXC	**44.23**	**28.83**	**20.15**	**38.80**	**51.87**	**60.71**
DECAF	36.58	23.31	16.43	30.66	40.79	48.32
Astec	34.86	21.83	15.25	28.60	37.32	43.91
MACH	27.94	17.95	12.53	21.50	29.21	34.41
AttentionXML	29.56	18.24	12.71	22.30	29.28	34.54
SwiftXML	30.05	18.42	12.73	24.11	30.45	35.38
Bonsai	31.28	19.50	13.60	24.20	32.32	38.23
Slice	26.57	16.46	11.52	21.08	27.65	32.80
Parabel	25.83	15.77	10.93	19.83	25.63	30.02
LF-WikiSeeAlsoTitles-320K						
GalaXC	**28.89**	**19.34**	**14.27**	**22.05**	**26.48**	**30.36**
DECAF	21.94	15.46	11.86	16.48	20.83	24.90
Astec	16.95	11.66	8.78	11.67	14.80	17.50
MACH	11.39	7.78	5.85	7.23	9.10	10.75
AttentionXML	11.52	7.54	5.60	7.20	8.76	10.28
SwiftXML	13.44	8.30	5.95	8.53	9.49	10.65
Bonsai	13.22	9.03	6.87	8.62	11.06	13.27
Slice	10.98	7.41	5.77	7.87	8.72	10.14
Parabel	10.23	6.82	5.12	7.01	8.47	9.92

Table 7: Ablations using different graph encoders and a variant of GalaXC without the per label attention mechanism (GalaXC-NoAtt).

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
LF-AmazonTitles-131K						
GraphSAGE	35.28	24.10	17.53	28.66	34.36	39.24
GIN	37.94	26.46	19.37	30.34	37.53	43.21
GalaXC-NoAtt	38.39	25.93	18.70	32.26	38.01	42.71
GalaXC	**39.17**	**26.85**	**19.49**	**32.50**	**38.79**	**43.95**
LF-WikiSeeAlsoTitles-320K						
GraphSAGE	22.80	15.46	11.84	15.96	18.36	20.37
GIN	23.99	16.68	12.97	14.89	18.11	20.74
GalaXC-NoAtt	27.32	18.02	13.62	19.72	22.24	24.17
GalaXC	**27.87**	**18.75**	**14.30**	**19.77**	**22.25**	**24.47**

5.1 Sponsored Search: a Case Study

Significant efforts have been put into developing sponsored search retrieval systems to surface the most relevant advertisements for end users. This case study focuses on the Q2K application that is of vital importance in sponsored search systems. Apart from XC methods, leading algorithms to match queries to keywords can be divided into three categories: 1) *Generative Models* like GPT-2 [44, 50] which are sequence-to-sequence architectures that generate bid phrases for each input query. 2) *Siamese* architectures [33, 46] that train two towers, one each for query and keyword with the objective to bring similar pairs closer in the (shared) embedding manifold. These generally rely on ANNS at prediction time in order to retrieve the closest keywords for a particular query, and 3) *Graphical Models* [13, 18, 19] that aim to predict new links by learning over an existing click-through graph. Table 10 compares GalaXC with GPT-2, TwinBERT and P-Simrank. For GPT-2, 200 predictions were generated and only those present in the keyword set were kept. Though GPT-2 can be highly accurate in surfacing the most relevant keyword for a query, it took around 150× more time to generate predictions. At the same time, GPT-2 was found to be around 22% worse than GalaXC when top 5 predictions were compared. Furthermore, GalaXC was 3% better in PSP@5 metric than if retrieving keywords using ANNS over embeddings obtained from TwinBert. Finally, GalaXC beat the recent graph based method P-Simrank [13] by at least 10% in all metrics. GalaXC was also more accurate than leading deep extreme classification algorithms like DECAF, Astec and MACH. Specifically, GalaXC was found to be around 1% and 1.8% more accurate than DECAF in P@5 and PSP@5, respectively.

Auxillary Information Sources: In production settings, GalaXC could readily incorporate other forms of meta-data. For instance, 1) additional sources of information like query expansion using

Table 8: Ablation for using different initial embeddings

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
LF-AmazonTitles-131K						
GalaXC-Astec	**39.17**	**26.85**	**19.49**	**32.50**	**38.79**	**43.95**
GalaXC-DECAF	38.47	25.92	18.79	32.38	37.90	42.71
GalaXC-fastText	35.83	23.93	17.25	30.27	35.14	39.53
DECAF	38.40	25.84	18.65	30.85	36.44	41.42
Astec	37.12	25.20	18.24	29.22	34.64	39.49
LF-WikiSeeAlsoTitles-320K						
GalaXC-Astec	**27.87**	**18.75**	**14.30**	**19.77**	**22.25**	**24.47**
GalaXC-DECAF	22.36	15.20	11.66	15.54	17.96	19.98
GalaXC-fastText	25.64	17.13	13.06	19.57	21.69	23.73
DECAF	25.14	16.90	12.86	16.73	18.99	21.01
Astec	22.72	15.12	11.43	13.69	15.81	17.5

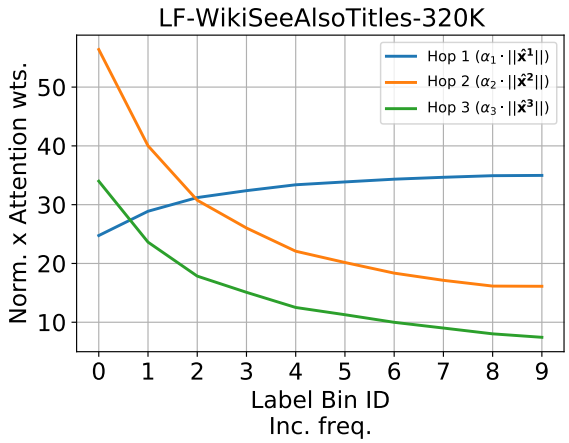

Figure 6: Plot for the softmax normalized attention weights scaled by the average norm of the corresponding hop embedding of train documents. The label quantiles are in order of increasing frequency.

Table 9: Predictions by GalaXC and other deep extreme classifiers on the test document "Primal Prey" from the LF-AmazonTitles-131K dataset. Predictions aligning with the ground truth are typeset normally in black font color whereas those disjoint from ground truth use light gray color. The document had 5 ground truth labels, the 4 correctly predicted by GalaXC, along with "Paraworld".

Method	Top 5 predictions
GalaXC	Carnivores: Cityscape, Carnivores 2 (Jewel Case), Carnivores: Ice Age, Carnivores 2, Hunter Prey (2009)
DECAF	Shadow of the Colossus, Primal, God of War, Prey, Perfect Dark Zero Limited Collector's Edition
Astec	Shadow (2009), Trail of a Serial Killer (2004), Perfect Dark Zero Limited Collector's Edition, The Darkness, The Devil in Gray
MACH	Shadow (2009), Mark of Kri, PlayStation 2 Memory Card (8MB), God of War, Blood Rayne 2
AttentionXML	Trail of a Serial Killer (2004), Sonic the Hedgehog 2, Street Fighter II, Mortal Kombat II, Super Mario World

Table 10: Offline results on the Q2K-450K dataset. In GalaXC-Aux-Test, related searches were used to introduce test documents in the graph. GalaXC-Aux-Train exploits related searches to densify the graph during training itself.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
Q2K-450K						
GPT-2	59.39	46.88	38.81	31.85	33.98	36.43
P-Simrank	48.78	42.07	35.13	26.67	29.91	33.01
TwinBERT	57.65	53.04	49.78	33.93	35.49	38.49
DECAF	72.49	64.88	59.53	33.40	38.22	40.12
Astec	71.95	64.51	59.31	34.26	37.73	39.16
MACH	66.77	59.18	54.66	32.53	35.90	38.17
Parabel	67.35	61.30	57.33	30.87	35.64	37.88
Slice+fastText	64.55	56.79	49.08	28.36	31.88	34.16
GalaXC	71.60	65.59	60.65	35.71	39.90	41.98
GalaXC-Aux-Test	72.57	66.43	61.29	36.44	40.68	42.71
GalaXC-Aux-Train	**74.49**	**68.59**	**63.01**	**37.85**	**41.45**	**43.21**

related searches and co-bidded clicked keywords logs could be used to introduce test documents in the graph(GalaXC-Aux-Test), and 2) a query-query graph could also be added for new links amongst existing nodes in the graph for training (GalaXC-Aux-Train). Table 10 shows the results of GalaXC with addition of other forms of meta-data. Going from GalaXC to GalaXC-Aux-Test and then to GalaXC-Aux-Train increased the P@1 by 1% and 2%, respectively. This demonstrates that introducing novel documents in the graph using user preferences revealed from auxiliary sources of data can lead to significantly better results.

A/B test on Bing: An efficient implementation of GalaXC in PyTorch [40], with mixed precision training to reduce GPU memory usage and speed up training allowed GalaXC to be trained on the Q2K-50M dataset with around 50 million labels and 97 million training points on 4× 32GB V100 GPUs within 4 days. GalaXC was deployed on the Bing search engine to perform A/B testing on live search engine traffic. Performance was measured in terms of Click-Yield (CY), Query Coverage (QC), and Click-Through Rate (CTR). Click Yield (CY) is defined as the number of clicks per unit search query. GalaXC was found to increase the CY and CTR by 1.52% and 0.74%, respectively. This indicates that ads surfaced using GalaXC were more relevant to the end user and hence translated into clicks. Additionally, GalaXC could increase the Query Coverage by 1.11%, demonstrating the ability of the algorithm to serve for queries where ads were not previously shown. Further, human labelling by expert judges established that GalaXC could increase the quality of predictions by 16% over the state-of-the-art in-production techniques. As an example, for the query "Brother PE 800", GalaXC could exploit its graph to learn that semantic concepts like "sewing" and "embroidery" are related to the query. GalaXC could thus bring new keyword matches like "Brother embroidery machine" and "Brother PE sewing", which were not captured previously by other techniques in production.

6 CONCLUSION

This paper developed the GalaXC algorithm to obtain better representation of documents by incorporating neighborhood information with the help of a joint graph over documents and labels. GalaXC also enables more effective extreme classifiers to be learnt by using a per-label attention mechanism that attends to multi-resolution embeddings obtained by a scalable GNN architecture. These contributions allowed GalaXC to be upto 18% more accurate than leading deep extreme classifiers, with GalaXC performing particularly well for tail labels. It was further demonstrated that GalaXC could efficiently utilize partially revealed labels in warm start scenarios and other auxiliary sources of data in general. GalaXC could be 2-50× faster to train compared to deep extreme classifiers and be efficiently scaled to datasets with tens of millions of labels. This allowed GalaXC to be deployed in the Bing Sponsored search stack with real world gains over state-of-the art generative, XC, IR and two tower models currently in production.

ACKNOWLEDGMENTS

Thanks are due to Qiang Zhang, Alex Samylkin, Xuihui Liu, and Atul Gupta for infrastructure support for large scale training, to Gururaj K, and Sakina Bohra for help in running A/B tests, and to Kunal Dahiya for helpful feedback. Special thanks are due to Purushottam Kar and Anshul Mittal for valuable feedback and help in better understanding and presenting the work.

REFERENCES

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. arXiv:1905.00067 [cs.LG]

[2] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*.

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

[4] R. Babbar and B. Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *ML* (2019).

[5] D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

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

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* (2017).

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

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

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

[11] M. Defferrard, X. Bresson, and P. Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[12] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL* (2019).

[13] P. Dey, K. Goel, and R. Agrawal. 2020. P-Simrank: Extending Simrank to Scale-Free Bipartite Networks. In *Proceedings of The Web Conference 2020*. 3084–3090.

[14] Z. Gantner, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. 2012. Personalized ranking for non-uniformly sampled items. In *Proceedings of KDD Cup 2011*. 231–247.

[15] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344* (2016).

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212* (2017).

[17] M. Gori, G. Monfardini, and F. Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2. IEEE, 729–734.

[18] A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[19] W. Hamilton, Z. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[20] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[21] S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

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

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

[24] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*.

[25] S. Khandagale, H. Xiao, and R. Babbar. 2019. Bonsai - Diverse and Shallow Trees for Extreme Multi-label Classification. *CoRR* (2019).

[26] T. N. Kipf and M. Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[27] T. N. Kipf and M. Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[29] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

[30] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. 2014. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*. 85–96.

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

[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[33] W. Lu, J. Jiao, and R. Zhang. 2020. TwinBERT: Distilling knowledge to twin-structured BERT models for efficient retrieval. *arXiv preprint arXiv:2002.06275* (2020).

[34] Y. A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836.

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

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

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

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

[39] N. Natarajan and I. S Dhillon. 2014. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* 30, 12 (2014), i60–i68.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*. 8026–8037.

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

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

[43] Y. Prabhu and M. Varma. 2014. FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning. In *KDD*.

[44] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1, 61–80.

[46] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 101–110.

[47] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. WWW 2014. https://www.microsoft.com/en-us/research/publication/learning-semantic-representations-using-convolutional-neural-networks-for-web-search/

[48] A. Sperduti and A. Starita. 1997. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks* 8, 3 (1997), 714–735.

[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML]

[50] O. Vinyals, M. Fortunato, and N. Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*. 2692–2700.

[51] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[52] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536* (2018).

[53] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*. 5753–5763.

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

[55] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[56] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (July 2018). https://doi.org/10.1145/3219819.3219890

[57] R. You, S. Dai, Z. Zhang, H. Mamitsuka, and S. Zhu. 2019. AttentionXML: Extreme Multi-Label Text Classification with Multi-Label Attention Based Recurrent Neural Networks. (2019).

[58] M. Zhang and Y. Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*. 5165–5175.

[59] Y. Zhang, D. Wang, and Y. Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *The World Wide Web Conference*. 2390–2400.

[60] Z. Zhou and X. Li. 2017. Graph Convolution: A High-Order and Adaptive Approach. arXiv:1706.09916 [cs.LG]