JobXMLC: EXtreme Multi-Label Classification of Job Skills with Graph Neural Networks

Nidhi Goyal
IIIT-Delhi

Jushaan Singh Kalra
DTU, Delhi

Charu Sharma
IIIT-Hyderabad

Raghava Mutharaju
IIIT-Delhi

Niharika Sachdeva
InfoEdge India Limited

Ponnurangam Kumaraguru
IIIT-Hyderabad

Abstract

Writing a good job description is an important step in the online recruitment process to hire the best candidates. Most recruiters forget to include some relevant skills in the job description. These missing skills affect the performance of recruitment tasks such as job suggestions, job search, candidate recommendations, etc. Existing approaches are limited to contextual modelling, do not exploit inter-relational structures like job-job and job-skill relationships, and are not scalable. In this paper, we exploit these structural relationships using a graph-based approach. We propose a novel skill prediction framework called JobXMLC, which uses graph neural networks with skill attention to predict missing skills using job descriptions. JobXMLC enables joint learning over a job-skill graph consisting of 22.8K entities (jobs and skills) and 650K relationships. We experiment with real-world recruitment datasets to evaluate our proposed approach. We train JobXMLC on $20,298$ jobs and $2,548$ skills within 30 minutes on a single GPU machine. JobXMLC outperforms the state-of-the-art approaches by 6% on precision and 3% on recall. JobXMLC is 18X faster for training tasks and up to 634X faster in skill prediction on benchmark datasets enabling JobXMLC to scale up on larger datasets. We have made our code and dataset public at https://precog.iiit.ac.in/resources.html.

1 Introduction

Online recruitment platforms such as LinkedIn and Glassdoor are extensively used to post jobs, find relevant candidates, and match resumes to the jobs posted. Recruiters create job positions mentioning skills, roles, and responsibilities to reach potential candidates. Among all these required fields, skills are crucial parameters to determine whether or not a candidate is suitable for a job position (Mehta et al., 2013). Recruiters often miss adding relevant and crucial skills required for the job due to a communication gap between the domain experts and recruiters. According to statistics, 65% of the job descriptions (JDs) do not include relevant and popular skills, and 40% of JDs miss listing 20% or more explicitly-stated skills in the prose description (Bhola et al., 2020). It reduces the number of relevant applications for the job posting and affects the performance of major recruitment tasks such as job-to-resume matching. Therefore, it is imperative to recommend such missing skills to improve the quality of job postings. Figure 1 shows the sample (fictitious) job posted over the recruitment platform where some skills are missing from the textual JD. Prior works (Bhola et al., 2020; VERMEER et al., 2020) explored missing skill recommendation task using large-scale pre-trained language models. Document embedding and Graph-based systems (Gugnani and Misra, 2020; Kivimäki et al., 2013) are used for skill extraction and recommendations. However, these approaches have a few shortcomings- (a) they do not exploit the structural relationships between jobs and skills across the whole dataset. For example, assume jobs j_1 and j_2 share a common skill s_1. If there is another skill s_2 relevant to j_2 and other similar jobs, we can infer that s_2 might also be relevant to j_1. Such transitive cues can be extremely useful for identifying missing skills. Current deep extreme classifiers (You et al., 2019; Prabhu and Varma, 2014) find it hard to model such implicit relationships unless the training set explicitly contains a pair (j_1, s_2), (b) they give equal importance to every skill corresponding to the job. However, each skill in the skill label set has different weights based on the frequency of their occurrence in job descriptions, (c) language models bring high computational costs at massive scales as a task not only involves predicting multiple missing skills but also requires to precisely organize the most relevant skills specific to the job posting. Graphs are naturally suitable to make the relationships explicit such as job-skill networks.

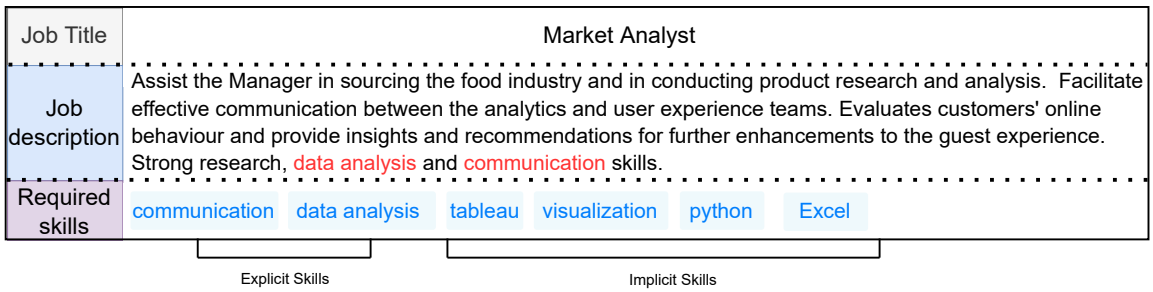

Figure 1: An example of a fictitious job posted over a recruitment platform. The job description does not include implicit and job-specific skills such as *'tableau'*, *'visualization'*, *'python'*, and *'Excel'*.

Two nodes (jobs) are likely to have common neighbors (skills) if the jobs have overlapping skills. To model these structural relationships between nodes (jobs and skills), a Graph neural network (GNN) is a well-known architecture for representing the knowledge and additional information (Wu et al., 2020). Missing skills applications also face extreme skill label sparsity; using label co-occurrence alone without graphs yields fractured correlations. To this end, we propose a framework called JobXMLC, which uses a GNN with label attention to jointly model the jobs and skills in the same space. Through the use of ubiquitous job description data, we aim to predict the missing skills using collaborative learning of jobs and skills. Therefore, we model our problem as an Extreme Multi-label Classification (XMLC) task as there is no existing dataset with manually-annotated labels from textual JDs, making it sub-optimal to train a sequence labeling task. The contributions of this work are summarized as follows:

- We construct a novel job-skill graph consisting of $22,844$ (jobs and skills) and 650K relationships that allow flexible integration of textual features and various pre-trained language representation models.
- We cast our problem as an XMLC using job-skill graph and propose JobXMLC comprising of graph neural networks with skill attention to learn multi-resolution graph neighborhoods with the sampling method.
- We also provide the performance comparison of JobXMLC, which outperforms by a margin of 6% from the best baselines.
- JobXMLC is lightweight, up to 18X faster in training and 634X in predicting than existing deep learning-based extreme classifiers to scale up to thousands of labels.

2 Background and Related Work

Recently, several works (Bhola et al., 2020; Jiechieu and Tsopze, 2021) have been done for skill prediction using Extreme Multi-label Classification. XMLC refers to the classification of text where the number of the set of labels is large, i.e., thousands or millions. One-vs-All (OVA) is a well-known method for text classification tasks with high accuracy (Khandagale et al., 2020). The OVA approach is computationally efficient for the XMLC task for modest-sized label sets (up to a few thousand labels).

These methods are broadly classified as (a) Deep learning-based models, (b) Graph-based, and (c) Domain-specific methods.

Deep learning-based methods. Deep learning models that use powerful text representation capabilities have also been explored for the XMLC problems (You et al., 2019; Chang et al., 2019). XML-CNN (Liu et al., 2017) applies a dynamic max-pooling scheme and a family of CNN models to learn text representations. AttentionXML (You et al., 2019) uses the attentional bidirectional long short-term memory (BiLSTM) networks to extract embeddings from raw text inputs. However, the CNN-based models cannot capture the most relevant parts of the information on each label. The RNN-based methods fail to model long-term dependencies due to vanishing gradients. Research (Bhola et al., 2020) explores the language models such as ELMo, Transformer and BERT (Devlin et al., 2019), and X-BERT (Chang et al., 2019) for XMLC task. These approaches model input language's syntactic and semantic structure to predict tokens based on the available contextual information. However, such models are computationally expensive and require a predefined meaning of la-

bels. In addition, the difficulty of scaling to the extreme label space remains in deep learning methods as the output layer scales linearly with the product of label size and feature dimension. The research gaps with deep-extreme classifiers motivate us to explore alternative approaches and techniques that do not require explicit label representation or predefined semantic meaning of labels and are scalable for extensive datasets.

Graph-based approaches. The recent proliferation of graph neural networks (Wu et al., 2020) allows using node neighborhoods to learn more discriminative features collaboratively. GraphSAGE (Hamilton et al., 2017) proposed the computation of node representations inductively by recursively aggregating over fixed-sized neighborhoods. Authors (Xu et al., 2018a) proposed the Graph Isomorphism Network (GIN) with discriminative power equal to that of the WL test. GraphSAINT (Zeng et al., 2020), a graph sampling-based inductive learning method, compute node representations based on the local graph structure and node attributes. However, job-skill graph-based collaborative learning at extreme scales is underexplored for missing skill prediction task.

Domain-specific methods. Research work identifies the skill bases used for analyzing the job market, the type of extracted skills (Khaouja et al., 2021), the skill identification methods, the studied sector and their granularity. Literature (Xu et al., 2018b) build a job-skill network to measure the popularity of the skills by exploring a large corpus of job postings. Research (Bhola et al., 2020) employs an Extreme Multi-label Classification method that utilizes the Transformer model to predict the required skills from a textual job descriptions.

However, these approaches are computationally expensive and either predict frequent skills or miss rare crucial skills for recruiters. To address all the existing challenges and limitations, we propose JobXMLC that uses graph neural networks with skill attention to learn multi-hop job-skill network. To the best of our knowledge, this work is the first to exploit GNNs for the job-skill prediction task.

3 Problem Formulation

Consider the set of jobs $\mathcal{J} = \{j_1, j_2, \cdots\cdots, j_i\}$. A job $j_i \in \mathcal{J}$ corresponds to its textual description and $\mathcal{S}_i$ is the set of skill labels for the i^{th} job. The skill set is represented as, $\mathcal{S}_i = \{s_i^1, s_i^2, s_i^3, \cdots\cdots, s_i^k\}\ \forall\ 1 \leq k \leq n$, where n refers to total skills that vary differently for each dataset. The task of JobXMLC is to learn a function $f : \mathcal{J} \rightarrow 2^{\mathbb{S}}$ that maps a job $j_i \in \mathcal{J}$ to its target skill set $S_i \in \mathbb{S}$, where $\mathbb{S} = \{\mathcal{S}_1 \cup \mathcal{S}_2 \cup ... \cup \mathcal{S}_{|\mathcal{J}|}\}$.

4 JobXMLC: EXtreme Multi-label Classification of Job Skills

In this section, we introduce JobXMLC as shown in Figure 2. The architecture is inspired by the models proposed in (Saini et al., 2021).

The architecture comprises of three major components: 1) Job-skill graph 2) Graph Neural Network that learns multi-hop embeddings with neighbourhood selection approach on the job-skill graph 3) a scalable mechanism of extreme classifiers to predict skill labels in cold and warm-start scenarios.

4.1 Module I: Job-skill graph

We first formally define a job-skill graph, which is usually represented as a tuple $\mathbb{G}$ = ($\mathbb{V}$, $\mathbb{E}$), where $\mathbb{V}$ and $\mathbb{E}$ are the set of nodes and edges respectively. Here $\mathbb{V}$ consists of jobs belonging to $\mathcal{J}$ and skill set $\mathbb{S}$ (See Section 3). We construct an edge $e \in \mathbb{E}$ between j_i and s_i^k where $\mathbb{E} \subset \mathcal{J} \times \mathbb{S}$ iff s_i^k is relevant to j_i i.e., s_i^k is a positive label for j_i. Each node $v \in \mathbb{V}$, is initialized as a d-dimensional vector based on its textual features. We obtain initial embeddings by fine-tuning the fastText skip-gram model (Bojanowski et al., 2017) in an unsupervised manner. fastText is a lightweight embedding model that is well-suited when the document misses predicate-argument structure dependencies (Arora et al., 2020). We also leverage word-level information, including POS tagging (Kumawat and Jain, 2015) and word importance (TF-IDF), to parse the long document according to relevancy and structure. Since all the tokens present in job descriptions are not informative, we apply POS tagging to filter out verbs, adjectives, and adverbs, which are not indicators of skills in the job description. The underlying assumption is that skill labels would be mostly nouns such as *'java'*, *'python'*, etc. We found out that there are 60% nouns present in job descriptions. Further, we use an averaging technique to get the representation for every job. For each node v, its initial representation is $\hat{f}_v^0$ (with $\hat{f}_v^0 \equiv \hat{f}_j^0$ if the node v is the job $j \in \mathcal{J}$ and $\hat{f}_v^0 \equiv \hat{f}_s^0$ if node v is the skill label $s \in \mathbb{S}$).

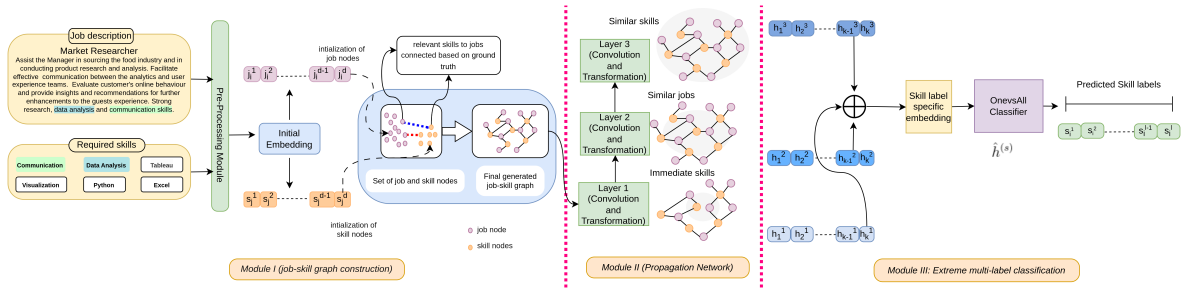

Figure 2: **JobXMLC** consists of three components: Module I consists of a mechanism to construct a job-skill graph, and Module II consists of a graph neural network-based architecture that learns embeddings using multi-hop neighborhoods using a job-skill graph effectively. Module III uses a scalable mechanism of extreme classifiers to predict missing skills.

4.2 Module II: Multi-hop job-skill graph neural network

To learn a job-skill graph, the module introduces the propagation network and neighborhood selection approach.

Propagation Network. This network captures higher-order job-skill graph structure using multiple layers of aggregation, each layer aggregating information from the previous layer's node representations. Therefore, we utilize Graph Isomorphism Network (GIN) (Xu et al., 2018a) encoder for representations considering its outstanding expressive capacity and model simplicity. It consists of a convolution to aggregate information from a node's neighbors and a transformation operation to update the node representation based on the convolved embeddings ($f_v^{(k)}$). To avoid over-smoothing, we utilize the skip connection operation that gathers information from historical representations of nodes. We also learn multi-hop representations with k-hop (fanouts) neighbors of each node, where k is a hyperparameter. For instance, if k=1, the encoder would only consider the immediate neighbors (skills) of each node (job) in the graph. If k=2, it would consider the neighbors (jobs) of the neighbors (skills) and so on. Equation 1 shows the graph neural network layer that updates the node representation using a weighted sum of neighboring node features:

$$f_v^{(k)} = (1+\lambda_k) f_v^{(k-1)} + \sum_{j \in \mathbb{N}_v, j \neq v} f_j^{(k-1)} \quad (1)$$

where $\mathcal{N}_v$ be the set of neighboring nodes of an i^{th} node; $f_v^{(k)}$ be the representation of the v^{th} node after layer k, and λ is a fixed scalar for layer k.

Equation 2 shows the final embeddings after transformation:

$$h_v^{(k)} = f_v^{(k)} + g(\delta(R_k * g(f_v^{(k)}))) \quad (2)$$

where $g(.)$ is ReLU activation, $\delta(.)$ is batch normalization and R_k is a parameter matrix for the residual layer.

Neighborhood selection. Instead of considering all k-hop neighbors of each node, we sampled a subset of the neighbors at each layer of the network. The goal of selection is to reduce the computational cost of the network and ensure that our model is scalable in dense settings. Therefore, we select the top l neighbors based on their frequency. Formally, for every node $v \in |\mathbb{V}|$, we accomplish frequency-based sorting where a set of fanouts $[k]$ neighbors are sampled for every node to construct $\mathbb{V}(n)$.

4.3 Module III: Extreme multi-label classification

In this module, we discuss skill attention and prediction pipeline for Extreme multi-label classification task.

Skill attention. We incorporate label-wise attention for every skill $s_i \in [\mathcal{S}]$ and layer $k \in [\mathcal{K}]$ in the propagation network. We obtain attention weights α_k using a softmax operation. $\alpha_{sk} = exp(e_{sk}) / \sum_{k' \in [K]} exp(e_{sk'})$. Given multi-resolution embeddings $\hat{h}_v^k$, $k \in [\mathcal{K}]$ for a job description, when calculating the score for a label $s \in [S]$, first a label-specific embedding is calculated as given in Equation 3 and then One-vs-all classifier depicted as $C = [c_1, c_2, ..., c_{\mathbb{S}}] \in \mathbb{R}^{\mathcal{J} \times \mathbb{S}}$ is used to obtain a score for the skill label as $score_s = \left\langle c_s, \hat{h}^{(s)} \right\rangle$.

$$\hat{h}^{(s)} = \sum_{k \in K} \alpha_{sk} . \hat{h}^{(s)} \quad (3)$$

Element	mycareersfuture.sg	StackOverflow Jobs
No. of job posts	20,298	20,320
# of distinct skills	2,548	275
# of skills with 20 or more mentions	1,209	50
Average skill tags per job post	19.98	2.8
Average token count per job post	162.27	200.8
Maximum token count in a job post	1,127	800

Table 1: Dataset statistics for mycareersfuture.sg and StackOverflow Jobs.

Prediction pipeline. Initial representations are used to construct an Approximate Nearest Neighbors graph (ANNS). Suppose a test job appears at runtime, First, the relationships in the graph are introduced to its *prediction-introduce-edges* (See Table 8) nearest neighbors. These neighbors can be a part of $\mathcal{J}or\ \mathcal{S}$. New job nodes are first introduced into the graph for the standard cold start setting, where jobs are not part of the job-skill graph used for training. Relationships are introduced to the partially revealed skill labels in warm start settings. This setting is possible when the recruiter enters some skills before writing the job description. Then, JobXMLC is used to obtain multi-resolution embeddings $\hat{h}^k \in [\mathcal{K}]$ for the test job.

Shortlisting. Since evaluating skill scores for all skill labels would take $\Omega(\mathcal{JS})$ time. To predict in milliseconds, the prediction time complexity should not be worse than $\Theta(\mathcal{J}log(\mathcal{S}))$. Therefore, we utilize a shortlisted where a set of $\mathcal{O}(log\mathcal{S})$ skill labels are shortlisted that seem most relevant to it. For a test job description, the label-wise embeddings $\hat{h}^{(s)}$ are created with respect to skill labels $s \in \mathcal{S}$. To create the shortlister, multi-resolution representations of skill labels are averaged and a second ANNS graph is created over these averaged embeddings. We rank the top *num_shortlist* (See Table 8) neighbors, based on their cosine similarity, for shortlisting to form the set $\mathcal{S}$ of potential labels for which label-wise embeddings are calculated.

5 Experimental Setup

5.1 Datasets

We utilize two real-world recruitment datasets, namely mycareersfuture.sg (Bhola et al., 2020) and StackOverflow Jobs[1] collected from popular recruitment platforms. These datasets consist of over 20,000 richly-structured job posts with 23 informative fields about the advertisement details and current status. Table 1 reports the statistics for recruitment domain datasets. Small-scale datasets vary from 100 to 300, whereas large-scale ranges from 300 to millions of labels. Similar scales for the XMLC task are demonstrated in (Jain et al., 2019; Liu et al., 2017).

Data Pre-processing. We filtered out *job descriptions*, *job title*, *required skills* corresponding to every job posting. From mycareersfuture.sg dataset, we consider concatenation of *'roles & responsibilities'* and *'job requirements'* fields as the *'job description'*, and *'required skills'* as the set of target discrete labels. Similarly, for StackOverflow Jobs dataset, we consider the *'job description'* and *'required skills'* sections. We filtered out the jobs with either empty or single words in the textual content. We also perform lower-casing, stopwords removal, and removal of less important strings such as *'available'*, *'requirements'*, which are present in most JDs. StackOverflow Jobs dataset consists of 6M words with 298,729 unique words. We split the dataset into training, validation and testing datasets with an 80:10:10 proportion. Similar splits has been utilized by competitive methods (Bhola et al., 2020).

5.2 Implementation and Competing Methods

This section will discuss the training details and baselines.

Training details: We utilize binary cross-entropy loss and Adam optimizer. We use the drop out layer after every ReLU layer. We conducted our experiments using the list of hyperparameters reported in Table 8 and Table 9 (See appendix B) for details.

Baselines: We show the effectiveness of different aspects of JobXMLC and evaluate our model performance against competitive transformer-based baselines. These constitute CNN (Kim, 2014), LSTM (Rocktäschel et al., 2015), BiLSTM (Sun

[1] https://stackoverflow.com/

Model	R@5	R@10	R@30	P@5	P@10	P@30
CNN	14.17	23.58	45.34	56.67	47.17	30.23
LSTM†	11.67	18.44	35.02	46.67	36.89	23.34
Bi-LSTM†	13.02	21.37	41.54	52.07	42.75	27.70
Bi-GRU†	13.98	23.43	44.41	55.94	46.87	29.61
BERT +XMLC	15.27	25.96	51.18	61.06	51.92	39.32
RoBERTa+XMLC	16.15	26.52	51.99	60.08	53.85	39.87
BERT +XMLC+CAB	16.72	29.45	58.98	66.87	58.90	41.21
GalaXC	16.31	28.34	54.16	65.25	56.7	36.11
JobXMLC (GraphSaint)	16.23	27.79	53.32	64.93	55.59	35.55
JobXMLC (GraphSAGE)	16.84	29.18	56.89	67.36	58.36	37.93
JobXMLC	**18.29**	**32.33**	**63.18**	**73.20**	**64.66**	**42.22**

Table 2: Results of JobXMLC along with state-of-the-art approaches on mycareersfuture.sg dataset. For RNN-based models (†), we have limited all model architectures to two layers.

et al., 2017), BiGRU (Halder et al., 2018), BERT-XMLC (Bhola et al., 2020), RoBERT-XMLC (VERMEER et al., 2020), GalaXC (Saini et al., 2021), JobXMLC (GraphSaint) (Hamilton et al., 2017), and JobXMLC (GraphSAGE) (Hamilton et al., 2017). We discuss transformer-based approaches, BERT-XMLC (Bhola et al., 2020) encodes the words of the job descriptions using a pre-trained BERT model. The encoding of the [CLS] token is then used as representation of the job description. The job representation is passed to a bottleneck layer (i.e., an added linear layer before the output layer). The last layer treats every skill as a binary classification problem, so for each skill it calculates the probability that the skill is associated with JD.

State-of-the-art models such as CNN, LSTM, Bi-GRU, and Bi-LSTM are self-explanatory. We utilize two neural network layers for all RNN-based models. GalaXC (Saini et al., 2021) describes a novel framework for extreme classification using graph neural networks (GNNs). GraphSAGE (Hamilton et al., 2017) and GraphSaint (Hamilton et al., 2017) encodes the node information and useful for graphs that have rich node attribute information for extreme multi-label classification.

5.3 Evaluation metrics

We utilize Precision@k (P@k), Recall@k (R@k), Normalized Discounted NDCG@k (N@k), Mean Reciprocal Rank (MRR), EIM, REIM, RIIM (Bhola et al., 2020) as evaluation metrics for the skill prediction task.

Precision@k: includes the proportion of skills in the top-k skill prediction list that are relevant.

Recall@k: includes the proportion of relevant skills found in the top-k skill prediction list.

NDCG@k: discounts the true positives that occur later in the prediction rankings.

MRR: indicates the position (reciprocal) of the first true positive in the predicted set of skills.

EIM (Explicit Inference Measure): the micro, instance-based measure of explicit skills predicted by the model, compared against gold-standard explicit skills mentioned, for instance.

RIIM (Relative Implicit Inference Measure): macro, the recall-based measure of implicit skills predicted by the model, relative to the entire set of implicit skills.

REIM (Relative Explicit Inference Measure): macro, recall-based measure of explicit skills predicted by the model compared to the entire set of explicit skills.

6 Results and Analysis

Table 2 and Table 3 reports Recall@k and Precision@k for all state-of-the-art approaches and JobXMLC on both datasets. Compared to leading deep extreme classifiers, BERT-XMLC and RoBERTa-XMLC, JobXMLC is up to 18X faster to train on a single GPU. Compared to other baselines, JobXMLC is at least 3% better than Bi-LSTM (Sun et al., 2017) in R@5, which helps demonstrate the efficacy of modelling the sequence by JobXMLC. Further, fastText initialization in JobXMLC is 7-8% better than BERT-XMLC+CAB (Bhola et al., 2020) in R@5, indicating that the global relationships improve the model

Model	R@5	R@10	R@30	P@5	P@10	P@30
CNN	25.16	39.39	64.80	15.24	11.72	6.36
LSTM†	26.63	40.47	67.89	16.07	11.95	6.65
Bi-LSTM†	41.46	55.27	76.38	23.83	16.12	7.56
Bi-GRU†	46.15	59.01	78.61	26.68	17.23	7.79
BERT +XMLC	35.50	50.95	76.06	20.75	14.99	7.58
RoBERTa +XMLC	36.20	52.23	77.05	21.98	15.09	7.88
BERT +XMLC+CAB	37.20	51.24	**78.98**	22.18	15.02	8.03
GalaXC	43.27	51.47	67.50	24.23	14.53	6.50
JobXMLC (GraphSaint)	39.16	51.73	73.99	22.28	14.88	7.22
JobXMLC (GraphSAGE)	38.76	52.26	74.19	21.98	14.99	7.23
JobXMLC	**47.85**	**59.26**	74.53	**26.92**	**16.94**	**7.23**

Table 3: Results of JobXMLC along with state-of-the-art approaches on StackOverflow Jobs dataset. For RNN-based models (†), we have limited all model architectures to two layers.

better in addition to local connections through joint learning. Compared to BERT-XMLC (Bhola et al., 2020) and RoBERTa (VERMEER et al., 2020), both utilize transformer-based embeddings and skill correlation-based features for training, JobXMLC is 4% better in recall and precision metrics. JobXMLC outperforms Graph-based methods such as GalaXC (Saini et al., 2021) across all metrics. Table 4 reports NDCG and MRR values for mycareersfuture.sg dataset. We observe that JobXMLC outperforms all state-of-the-art approaches by significant margin of 8% from deep extreme classifiers.

Model	N@5	N@10	N@30	N@50	N@100	MRR
CNN	28.21	40.23	60.60	66.37	71.96	0.77
LSTM	29.27	40.66	59.43	69.61	71.53	0.70
Bi-LSTM	30.32	48.07	44.55	50.30	57.04	0.76
Bi-GRU	30.83	50.52	46.45	52.37	59.15	0.76
BERT-XMLC	28.05	38.81	57.62	64.68	71.28	0.83
BERT-XMLC+CAB	29.13	40.74	60.60	67.51	73.74	0.85
GalaXC	32.86	44.51	63.73	70.11	74.77	0.82
JobXMLC	**37.91**	**49.63**	**67.83**	**73.81**	**78.94**	**0.90**

Table 4: Normalized Discounted Cumulative Gain (NDCG) is represented by N and Mean Reciprocal Rank (MRR) comparison of JobXMLC along with State-of-the-art approaches on mycareersfuture.sg dataset.

Inference time: Table 5 presents the results of JobXMLC and leading deep extreme classifiers like BERT, ROBERTa which shows that JobXMLC is 18X faster than BERT+XMLC+CAB for mycareersfuture.sg dataset.

Analysis on Implicit and Explicit skills: Table 6 shows the Explicit and Implicit Metrics for the skill prediction task. We are interested in the relevance and implicitness of the retrieved implicit skills and false positives. We find the explicit and implicit skills underline the noisy nature of the skill labels. For example, *'machine learning'* and *'python'* are clear required (explicit) skills and *'communication'* comes across as a vast skill. In terms of false positives, we note that the job description explicitly mentions *'good knowledge of python'* as a required skill (for Data Scientist job). Most relevant skills are not very distinctive to the job role, causing the model to mispredict the skill.

7 Ablation Study

Initial embeddings: JobXMLC shortlisting criteria offered much better recall if we use the initial fastText embeddings to create shortlists. We observe that fastText worked best for our recruitment domain dataset in comparison to other recent pre-trained language representation models (See Appendix B) including BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), and Paraphrase-mini-LM-L6 (Reimers and Gurevych, 2019). For example, on the mycareersfuture.sg dataset, the recall for the top 100 labels shortlisted using the initial fastText and BERT embeddings are around 85.20% and 56.90%, respectively. Based on an average of 20.61 skills per job, about 4 skills were derived within the top 5 and 19 within the top 100 of derived skills.

Warm and Cold Start Scenarios: Table 7 reports the results in warm-start and cold-start settings separately. JobXMLC is initialized with fine-tuned fastText embeddings which achieve P@k, R@k, and MRR of 72.86, 18.26, and 0.89 respectively in cold-start scenario. JobXMLC is initialized with

Datasets →	mycareersfuture.sg		StackOverflow Jobs	
Models ↓	**TT**	**PT**	**TT**	**PT**
BERT+XMLC	5.50	1200	1.63	350
RoBERTa+ XMLC	4.72	1200	1.24	350
BERT+XMLC+CAB	9.20	1200	4.86	350
JobXMLC	**0.51**	**1.89**	**0.31**	**1.71**

Table 5: Comparison of JobXMLC with stronger baselines. JobXMLC is faster to train than leading Deep Extreme Classifiers like BERT at training and prediction time. Here TT= Train Time (in hours), PT= Prediction Time (in ms).

Metrics	EIM	RIIM	REIM
BERT+XMLC +CAB	115.89	64.60	25.73
JobXMLC	**121.09**	**66.36**	**33.04**

Table 6: Comparison of EIM, RIIM, and REIM metrics on JobXMLC on mycareersfuture.sg dataset.

fine-tuned fastText embeddings which achieve P@k, R@k, and MRR of 75.76, 22.09, and 0.91 respectively in warm-start scenario. In both scenarios, the value of k=5. These results show that partially reveal achieved comparable precision and relatively higher recall than cold-start settings.

Model	P@5	R@5	MRR
JobXMLC (cold-start)	72.86	18.26	0.89
JobXMLC (warm-start)	75.76	22.09	0.91

Table 7: Effectiveness of JobXMLC in warm-start and cold-start scenarios on mycareersfuture.sg dataset.

Qualitative Analysis: We compared the JobXMLC and analyzed the skills predicted correctly and incorrectly as shown in Figure 3. JobXMLC captures the structural relationships between jobs and skills effectively. JobXMLC predicts *'Java'*, *'Software Development'*, *'XML'*, *'JavaScript'*, *'jQuery'*, etc. as required skills whereas BERT-XMLC with CAB predicts *'Java'*, *'C++'*, *'Linux'*, *'Python'* as skills where more relevant skills such as *'JavaScript'* and *'Web Applications'* are missed.

8 Discussion

We compare with existing graph-based methods such as GalaXC (Saini et al., 2021), which are more well-suited to handle short text inputs for product queries. JobXMLC leverages word-level components, including syntactic roles (POS tags) such as nouns and verbs present in each job and word importance (TF-IDF), which explains the long document from the perspective of text relevancy and structure. We believe that JobXMLC is generalizable across many other applications. Our raw dataset is relatively preprocessed, simple and misses predicate-argument structure dependencies. Therefore, we hypothesize that non-contextual embeddings such as fastText (having 98.69% of words from our dataset present in vocabulary) outperformed BERT as it understands word-level information. Similar observations are made by (Arora et al., 2020) for classic embeddings with competitive (or even slightly better) performance than contextual embeddings.

9 Conclusion

In this work, we propose a JobXMLC framework, which uses a graph neural network to incorporate neighborhood information with the help of a collaborative graph over jobs and skills. JobXMLC leverages skill attention mechanism for more effective extreme classifiers and attends to multi-resolution representations of jobs and skills. JobXMLC outperforms leading deep extreme classifiers on precision and recall metrics by 6% and 3%. JobXMLC also operate in warm and cold-start scenarios effectively. JobXMLC is 18X faster on training and 634X faster on predicting than deep extreme classifiers and can be scaled efficiently to real-world datasets with thousands of labels. We believe that JobXMLC can be deployed on large-scale recruitment platforms for predicting missing skills using job descriptions.

10 Limitations

We perform experiments on jobs sampled from a popular Singaporean government job portal and StackOverflow, which is limited to the English lan-

Figure 3: Shows the skills predicted by BERT–XMLC+CAB and JobXMLC where input is job description. Purple shows correct skill predictions by JobXMLC as compared with required skills (ground truth). Green shows the extra skills predicted by JobXMLC. Red skills are missed by BERT+XMLC+CAB model as compared with ground truth.

guage. Our approach can handle missing skills which are part of our skill vocabulary, but it cannot infer new emerging skills from job descriptions, i.e., out-of-vocabulary. We will consider domain knowledge and the popularity of job skills to generalize our approach for job-candidate mapping applications for future work. We wish to expand our work to other recruitment domain applications with resumes and candidate profiles.

11 Ethical Considerations

The paper investigates the missing skills problem with the help of a graph-based framework by incorporating word-based embeddings that can be insightful for other researchers in academia and industry. Any biases present in the dataset or embedding model can creep into the proposed approach.

Acknowledgements

The authors acknowledge the support of the PreCog Research Group and the Machine Learning Lab at IIIT-H, the Infosys Center for Artificial Intelligence (CAI) and the KRaCR Lab at IIIT-Delhi. We also thank anonymous reviewers and the area chairs for their detailed and helpful feedback. Special thanks to Prashant, Saurabh, Anmol Goel, Shivangi, Amrit, Dr. Kajal Kansal, and Dr. Siddharth Asthana for critically reviewing the manuscript and stimulating discussions.

References

Simran Arora, Avner May, Jian Zhang, and Christopher Ré. 2020. Contextual embeddings: When are they worth it? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2663, Online. Association for Computational Linguistics.

Akshay Bhola, Kishaloy Halder, Animesh Prasad, and Min-Yen Kan. 2020. Retrieving skills from job descriptions: A language model based extreme multi-label classification framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5832–5842.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. 2019. X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers. *arXiv preprint arXiv:1905.02331*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Akshay Gugnani and Hemant Misra. 2020. Implicit skills extraction using document embedding and its use in job recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13286–13293.

Kishaloy Halder, Lahari Poddar, and Min-Yen Kan. 2018. Cold start thread recommendation as extreme multi-label classification. In *Companion Proceedings of the The Web Conference 2018*, pages 1911–1918.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

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

Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. 2021. Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*, 33:5069–5087.

Sujay Khandagale, Han Xiao, and Rohit Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109:2099–2119.

Imane Khaouja, Ismail Kassou, and Mounir Ghogho. 2021. A survey on skill identification from online job ads. *IEEE Access*, 9:118134–118153.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ilkka Kivimäki, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Hugues Bersini, and Marco Saerens. 2013. A graph-based approach to skill extraction from text. In *Proceedings of TextGraphs-8 graph-based methods for natural language processing*, pages 79–87.

Deepika Kumawat and Vinesh Jain. 2015. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6).

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124.

Sameep Mehta, Rakesh Pimplikar, Amit Singh, Lav R Varshney, and Karthik Visweswariah. 2013. Efficient multifaceted screening of job applicants. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 661–671.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

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*, pages 3733–3744.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Chengjie Sun, Yang Liu, Chang'e Jia, Bingquan Liu, and Lei Lin. 2017. Recognizing text entailment via bidirectional lstm model with inner-attention. In *International Conference on Intelligent Computing*, pages 448–457. Springer.

NINANDE VERMEER, VERA PROVATOROVA, DAVID GRAUS, THILINA RAJAPAKSE, and SEPIDEH MESBAH. 2020. Using robbert and extreme multi-label classification to extract implicit and explicit skills from dutch job descriptions. *acm*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018a. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. 2018b. Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*.

A Hyper-parameter Details

This section reports the set of hyperparameters used for experiments conducted in the paper.
1. **No. of Epochs**: refers to number of epochs for JobXMLC.
2. **num_HN_epochs**: number of hard negative epochs for JobXMLC.
3. **learning rate (lr)**: is the learning rate for JobXMLC.
4. **attention_lr**: is the learning rate used by skill attention.
5. **dlr_factor**: defines factor by which learning rate is decayed.
6. **batch_size**: refers to batch size used during training of JobXMLC.
7. **num_HN_shortlist**: refers to number of hard negative labels to be selected by sampling from other data points from the same batch.
8. **num_shortlist**: refers to number of skills sampled by shortlister.
9. **prediction_introduce_edges:** refers to total edges that should be introduced to graph at prediction time.
10. **fanouts:** refers to number of neighbors to sample for layer k.

Hyperparameter	Value
No. of epochs	20
num_HN_epochs	20
learning rate (lr)	0.0003
attention_lr	0.0003
dlr_factor	0.5
batch_size	256
fanouts	5, 5, 5
num_HN_shortlist	500
embedding_type	fastText
num_shortlist	1500
prediction_introduce_edges	3

Table 8: Hyper-parameters for mycareersfuture.sg dataset for JobXMLC. fastText refers to 300-dimensional embeddings obtained by fine-tuning fastText model on job descriptions.

B Evaluation Metrics for different initializations

This section reports the EIM, RIIM, REIM measures for Mini-LM (Reimers and Gurevych, 2019) model initialization. We observe that Mini-LM is a transformer-based model which captures context well. However, JobXMLC is more benefitted with global view rather than just local job description text (context-based) embeddings.

Hyper-parameter	Value
No. of epochs	30
num_HN_epochs	20
learning rate (lr)	0.0003
attention_lr	0.0003
dlr_factor	0.5
batch_size	256
fanouts	5, 5, 5
num_HN_shortlist	3
embedding_type	fastText
num_shortlist	275
prediction_introduce_edges	3

Table 9: Hyper-parameters for StackOverflow Jobs dataset for JobXMLC. As number of skill labels corresponding to job description are less in StackOverflow Jobs dataset, a lower fanout value gives better results.

Table 10: EIM, RIIM, REIM measures for JobXMLC and state-of-the-art approaches using Mini-LM, BERT and RoBERTa embedding initializations.

Metrics	EIM (Explicit inference measure)	RIIM (Relative implicit inference measure)	REIM (Relative explicit inference measure)
BERT+XMLC+CAB	115.89	64.60	25.73
JobXMLC(with Mini-LM)	86.45	27.28	17.76
JobXMLC(with BERT)	84.07	25.77	15.59
JobXMLC(with RoBERTa)	86.45	24.12	15.02