DGL4C: a Deep Semi-supervised Graph Representation Learning Model for Resume Classification

Wissem Inoubli[1,2,*,†], Armelle Brun[2]

[1]Keep In Touch, Strasbourg, France
[2]University of Lorraine, CNRS, LORIA, France

Abstract
The main goal of job seekers is to identify job offers that match their profile. The same stands for human resource departments that aim to identify candidates, through their resumes, that match the recruiter's expectations. However, the number of job seekers and job offers is so important that none of human resource employees nor job seekers is able to go through all the resumes and offers manually.
Recommender systems have emerged these last years with the goal to recommend job seekers and human resource departments, job offers and resumes respectively. One of the approaches adopted by the literature relies on the identification of content elements in the offers and resumes that contribute to perform matching. We propose to represent data under the form of graphs and approach this problem as a classification problem. We present DGL4C, a semi-supervised graph deep learning model, that learns the adequate representation from a graph and trains a classifier on this latent representation.
Experiments are carried out on an open dataset of anonymous resumes. Results show that DGL4C significantly improves precision and accuracy of a traditional deep learning models, such as sBERT and confirm the pertinence of relying on a graph structure for the classification task in HR domain.

Keywords
graph representation learning, deep learning, semi-supervised learning, resume classification

1. Introduction

Recruitment is the process of matching job offers with resumes. It is performed by both job seekers and human resource (HR) departments, through the use of Applicant Tracking Systems (ATS), e.g. JobSCAN [1].

Due to the huge amount of resumes and offers, matching becomes highly time consuming and cannot be performed manually anymore. Information retrieval (IR) algorithms have been traditionally used to perform this task. They combine features extraction techniques and a retrieval model (e.g. the standard boolean model). For example, in [1] the goal is to find relevant resumes in a corpus of resumes based on a recruiter needs. Recommender systems are now becoming a common tool designed to recommend either HR the resumes that match a given job offer or a job seeker the relevant offers for his/her profile [2].

Although job offers can be easily collected to form a dataset, resume collection is a more tricky task, due to privacy issues. Such datasets are generally collected by private firms and are not freely available. Worse, very few datasets contain the ground-truth matching between offers and resumes. Thus, the content-based recommendation approach, that identifies resumes and job offers similar in terms of content, is the most adequate approach to perform this matching.

Besides, due to this lack of ground-truth, the evaluation of recommendation models remains a challenging task. To cope with this limit, we propose to perform this matching by using higher level information about resumes and offers. Concretely, we believe that the profile of a candidate, e.g. *computer scientist senior*, and the occupation of a job offer, e.g. *computer scientist*, can be used to perform this matching. We propose to view the identification of this higher-level information as a classification problem, as proposed by [3]. The challenge here is thus to learn a classifier dedicated to resumes or job offers, i.e. to unstructured plain texts.

The text classification literature traditionally takes place in two steps. First, the texts are pre-processed to extract features. For example, TF-IDF (term- Frequency-inverse Document frequency), LDA (Latent Dirichlet Allocation) [4], and Word2vec [5] are traditional models for feature extraction and text representation. Second, classification is performed by supervised machine learning algorithms that exploit such representations. The literature has shown that the performance of classifiers

RecSys in HR'22: The 2nd Workshop on Recommender Systems for Human Resources, in conjunction with the 16th ACM Conference on Recommender Systems, September 18–23, 2022, Seattle, USA.
*Corresponding author.
†These authors contributed equally.
wissem.inoubli@loria.fr (W. Inoubli); armelle.brun@loria.fr (A. Brun)
https://members.loria.fr/WInoubli/home/ (W. Inoubli); https://members.loria.fr/ABrun/ (A. Brun)
0000-0001-5121-9043 (W. Inoubli); 0000-0002-9876-6906 (A. Brun)

[1]https://www.jobscan.co/applicant-tracking-systems

is highly dependent of the quality of the representation [6].

Deep learning, that has the characteristics of performing feature extraction and classification in a unique step, has been also studied. Many neural network variants were studied like long short-term memory (LSTM) [7], based on a recurrent neural network (RNN) architecture [8], or a convolution neural network (CNN) [9]. Deep learning has proven to perform better than classical machine learning approaches. However, in the context of HR, deep learning still suffers from high error rates and low classification accuracy, especially for resume classification [1]. This can be probably explained by the limited size of the training datasets used [10].

Besides, graph structures have been traditionally adopted to manage rich data. Recently, deep graph learning models [3], that allow to learn a non-euclidean space of data, have emerged. Surprisingly, they have not been studied in the HR context, especially for resume classification.

Thus, in this work, we propose DGL4C, that stands for Deep Graph Representation Learning for Classification, a new classification model based on deep graph learning. DGL4C is a semi-supervised model, designed for resume classification, that manages both labeled (resumes) and unlabeled (elements of resumes) data. Concretely, we propose two variants of DGL4C. DGL4C-GCN, Deep Graph Representation Learning for Classification with a Graph Convolution Network, is an end-to-end model, that learns all the stages between the initial input phase and the final output result (resume classification). DGL4C-GRL, made up of two stages: (i) text (resume) representation made by a GCN architecture, and (ii) a machine learning-based classifier.

The remainder of this paper is organized as follows. Section 2 introduces the literature related to resume classification. In Section 3, we introduce the two variants of DGL4C: DGL4C-GCN and DGL4C-GRL. Then, in Section 4, experimental results are described and analysed. Last, in Section 5, we conclude and propose perspectives.

2. Related Work

In this section, we briefly review some works related to resume classification in the area of human resources.
The literature has proposed several approaches, that can be divided into three categories: (i) ontology-based models, (ii) machine learning models and (iii) deep learning models.

Let us first consider the ontology-based models. After a feature extraction step, ontology-based models use ontologies to perform classification. An ontology is a conceptual meta-model that represents a domain knowledge [11]. Few international and national knowledge bases in HR are released. The most famous ones are DISCO[2], ISCO[3] and ESCO[4] [12]. Those knowledge bases represent occupational groups at various granularity levels. An occupation is defined as *a set of jobs whose main tasks and duties are characterized by a high degree of similarity* [12], and a skill is defined as *the ability to apply knowledge and know-how to complete tasks and solve problems.* For example, the ESCO standard contains 13,485 skills related to 2,942 occupations, being sorted on four granularity levels. In [12, 13], the authors use TextKernal Extract parser, an industrial tool[5] to extract skills from resumes. The ESCO base is then used to build a classifier based on the matching of the skills defined by ESCO with the skills extracted in the resumes.

Regarding machine learning models, which are widely used, they rely on training data, and also require a preprocessing step dedicated to feature extraction. Machine learning models, such as Random Forests, Decision Trees, Support Vector Machines, etc. have shown high efficiency and performance on the resume classification task [14]. In such models, the accuracy of the feature extraction step strongly impacts the classification performance.

At the opposite of ontology-based and machine learning models, deep learning models consider both feature extraction and classification in a unique step, which reduces the possible loss of information in the feature extraction step. Deep learning models are highly popular, and have shown a significant improvement of performance. Several works have been proposed for job classification [9, 2, 13, 15] where both a 1-D convolution neural network (CNN) and a recurrent neural network (RNN) architectures were adapted to the HR context.

Graph representation learning techniques have recently emerged and are used in many applications. Graphs are a traditional way to represent data, but models that rely on such a representation suffer from data sparsity and robustness to noise, which decreases the performance of predictive models. To overcome those limits, graph representation learning has been designed to represent data in a low dimension space, and has shown its efficiency for unstructured data such as images, texts and graphs. Graph representation learning can be categorized into three families: matrix factorization based models [16, 17], random-walk based models [18, 19] and neural network based models [20, 21]. The latter are neural networks that are used to learn node embeddings by aggregating information from neighboring node through edges. Neighborhood aggregation consists in forwarding and receiving back data between nodes, throughout their neighborhood. In GNN, a node has an unlimited number of direct neighbors, whereas in the other neural network

[2]European Dictionary of Skills and Competences
[3]International Standard Classification of Occupations
[4]European Skills, Competences, Qualifications and Occupations
[5]https://www.textkernel.com/

architectures, the number of direct neighbors is limited, (e.g. two for RNN architectures and eight in the case of 2D-CNN architectures). This unlimited number of direct neighbors has shown its ability to encode both structural and semantic (node features) information, which makes it successful. This neighborhood information leads to a neural network that learns better than other architectures, which is confirmed by the work of Ding Yao et al. [22] in the case of hyper spectral image classification. Graph Neural Network architectures (GNN) are receiving a growing attention [20] and many models are proposed in the literature. Starting from the initial Graph Convolution Networks (GCN), GraphSage [23] was then proposed to overcome the scalabilty issue of GCN, by changing the convolution method. In the same context, an attention mechanism was proposed by the model called GAT [24].

To the best of our knowledge, the GNN architecture has not been studied for modeling resumes or job offers, and we assume that they could be of interest.

3. DGL4C: a GNN-based Architecture for Classification

Considering that data representation is a core step of classification models, we propose a new approach for resume representation, based on both a graph structure and semantic information. Concretely, we propose a semi-supervised graph representation learning model, based on a GNN architecture. This representation is used in the classification DGL4C model, that we design.

As previously mentioned, the main motivation for the choice of a graph-based representation learning, and specially a GNN architecture comes from the neighborhood information (neighboring resumes) taken into consideration during the training step.

As deep learning requires a lot of training data to be effective, and since resume data are generally small, we decide to adopt a semi-supervised learning algorithm, which takes both labeled and unlabeled examples; therefore the training process runs on both example types: labeled and unlabeled [21] data. DGL4C aims to form a high quality latent representation of resumes, further used in the classification phase. In the following subsections, we present the way we propose to construct a graph of resumes, then the core idea of graph representation learning, and the way we develop DGL4C, designed to encode a dataset of resumes into a latent vector space.

3.1. Graph Construction

Before learning the graph representation, a first phase consists in constructing the graph of resumes. We propose to is inspire from the recent work conducted in [3]. Let $D = (R, L)$ denote a dataset. R is the set of resumes, $R_i \in R$ is a resume with $R_i = \{wd_1, wd_2, wd_j\}$ is the set of words of the resume. L is the set of labels with $L_i \in L$ is the label of resume R_i and $|L|$ is the number of labels (classes). $\mathbb{W}\mathbb{d} = \bigcup_i R_i$ is the vocabulary of D, i.e the set of distinct words in R.

Definition of a R-graph. Let G be an heterogeneous, attributed and unweighted graph built from R, the set of resumes. $G = (V, E)$ with V and E represent nodes and edges of G respectively. The set of nodes $V = \mathbb{R} \cup \mathbb{W}\mathbb{d}$ is made up of the union of the set of resumes and the set unique words of the resume dataset. As a consequence, V is made up of two types of nodes: resume nodes and word nodes.

The set of edges E is also divided into two types of edges, word-to-resume edges and word-to-word edges.

An edge between two nodes exists if the similarity between those nodes is positive, similarly to [3]. Recall that the edges are unweighted.

The way this similarity is evaluated depends on the type of edge. Word-to-resume edges are evaluated by the well-known term frequency-inverse document frequency (TF-IDF), which is computed as follows:

$$\text{TF-IDF(wd,r,R)} = tf(wd, r) \times idf(wd, R) \tag{1}$$

where $tf(wd, r)$ denotes the number of times the word wd appears in a resume r; $idf(wd, R)$ denotes the number of resumes that contain the word wd. Word-to-word edges are evaluated by the Point-wise Mutual Information (PMI), as in [3]. A positive PMI value means a high semantic correlation of the pair of words in the corpus. A negative value indicates that both words are not semantically close. Therefore, only the edges that are associated with a positive PMI value exist in the graph, under an unweighted form. The PMI value is computed as follows:

$$PMI(i, j) = log\frac{p(i, j)}{p(i)p(j)} \tag{2}$$

$$p(i, j) = \frac{\sharp W(i, j)}{\sharp W} \tag{3}$$

$$p(i) = \frac{\sharp W(i)}{\sharp W} \tag{4}$$

Where $\sharp W(i)$ is the number of resumes that contain the word wd_i, $\sharp W(i, j)$ is number of resumes that contain both words wd_i and wd_j and $\sharp W$ is the number of resumes in the corpus.

Equation (5) represents the adjacency matrix A of the graph G.

$$A_{i,j} = \begin{cases} 1 & i, j \text{ are words, and } PMI(i, j) > 0 \\ 1 & i \text{ is a document and } j \text{ is a word, and } TF\text{-}IDF_{i,j} > 0 \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

3.2. Graph Representation Learning

After building the graph G from the set of resumes R, the graph representation learning is the second step of the proposed model. Most of neural networks have the same universal architecture, namely a set of multi-layer perceptron neural networks connected that operate on the input data. GCN [21] is a convolutional neural network on graphs that performs similar operations than CNN, except that it applies convolution over a graph instead of convolution on a 2-D array as input. GCN learns a latent representation by propagating information from direct neighbors in the graph and applies a linear transformation. The information propagation procedure consists in aggregating information from direct and n-hops neighborhood. Next, as perceptrons, GCN applies linear transformation followed by pointwise non-linearity. By stacking k GCN layers, each node aggregates information from nodes k-hops away. The GCN [21] propagation rule is defined as follows:

$$H^l = \sigma(\hat{A}H^{l-1}W^l), \ with \ H^0 = X \tag{6}$$

Where $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and H^l and H^{l+1} are the previous and the new hidden state matrix respectively, W^l is a trainable weights matrix for layer l, and σ denotes any non-linear activation function (e.g., ReLU). The convolution step in GCN is based on message passing that is divided into sub-steps (i) message gather and (ii) aggregation. Message gather consists of getting messages from n-hops neighbors, and the aggregation consist of normalization of all messages in order to get an embedding of a node v. The message passing form of equation (6) can be written as follows:

$$\begin{gathered} m_u^l = \sum_{v \in \mathcal{N}_v} \frac{1}{\sqrt{|\mathcal{N}_v|}\sqrt{|\mathcal{N}_u|}} h_v^{l-1} \\ h_u^l = \sigma(m_u^l W^l + b^l) \end{gathered} \tag{7}$$

In [3] the author used the initial nodes features as an identity matrix $X = I_{|V|}$.

In DGL4C, the node features matrix X is the feature vector for each node of G. The well-known pre-trained model sBERT [25] is used to encode both words and resumes to construct X. The final layer of the GCN represents the latent space Z that encodes resumes as a numerical matrix, and it is normalised by the softmax function. The loss function is defined as the cross-entropy error over only on the labeled nodes:

$$L = -\sum_{r \in Y_r} \sum_{f}^{F} Y_{rf} \ln Z_{rf} \tag{8}$$

where Y_r is the set of resume indices that have labels and F is the dimension of the last layer of the GCN, which is the number of profiles. As it is used by the loss function, only the resumes nodes are used where the graph contains resumes and elements of resumes (words) that show the semi-supervised training presented by DGL4C.

In addition, experiments that we conducted showed that the graphSage aggregation method performs better than GCN [21] and GAT [24] aggregation methods, thus the graphSage aggregation (mean aggregation) is the one kept for DGL4C.

We propose two variants of DGL4C, that differ in the number of steps they are made up of. DGL4C-GCN is an end-to-end model with a unique step that includes both representation and classification. Regarding DGL4C-GRL, it is made up of two stages: (i) text (resume) representation, and (ii) a machine learning classifier.

4. Experiments

In this section, we aim at evaluating the performance of both DGL4C-GCN and DGL4C-GRL.

4.1. Experimental Setup

4.1.1. Dataset

The dataset used is a corpus of 2,484 anonymous resumes[6]. Each resume is associated with one label, that represents the resume profile, and that we consider as being the class label. 24 profiles (classes) are available. Each resume is written in natural language and contains personal information, education, experience, etc.

In the experiments conducted, the aim is the evaluation of the performance of the proposed models, and compare them to several baseline models from the literature. In addition, we are interested in evaluating the impact of the number of classes (profiles) on the accuracy of the models. Thus, we form several datasets so that they fit a predefined number of classes. Statistics about each of the resulting datasets and associated graphs are presented in Table 1.

4.1.2. Evaluation Metric

We evaluate the models using the popular accuracy metric. Accuracy measures the ratio of predictions a model gets right. It is simply a ratio of correctly predicted (true positive+ true negative) observations to the total number of observations.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

where TP, TN, FP, FN denote the true positives, true negatives, false positives and false negatives, respectively.

[6]https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

Table 1
Statistics about the datasets

Dataset	# Resumes	# Profiles (# Classes)	Graph-based Representation # Nodes	# Labeled Nodes	# Edges
D1	500	5	6,705	500	3,887,179
D2	1,000	11	10,304	1,000	6,724,780
D3	1,500	16	11,076	1,500	7,917,124
D4	2,000	20	12,838	2,000	9,566,638
D5	2,484	24	14,183	2,484	10,976,856

4.2. Parameters Settings

DGL4C-GCN and DGL4C-GRL were implemented using the DGL framework [7] with two convolution layers of the GraphSage [23] architecture to allow message passing among nodes, and the mean aggregation. From an architectural point of view, we set the embedding size of the first convolution layer at 500, fixed from initial experiments. We tuned other parameters and set the learning rate as 0.001, dropout as 0.2. Both models have been trained over 200 epochs with a batch size of 32 training samples. For each dataset, we randomly use 80% of resumes for training and the remaining resumes for test and perform this selection 10 times. As a consequence, the accuracy evaluated and reported in the experiments is the mean test accuracy.

4.3. Experimental Results

To evaluate the effectiveness of DGL4C-GCN and DGL4C-GRL, we compare their performance with several models from the literature, that differ in either the text representation or the classifier step. Each model is a pair of text representation model and a classifier, except for the end-to-end model DGL4C-GCN. The popular text representation models, mentioned in the related work section are used. The list of models is presented in Table 2.

4.3.1. Impact of the Text Representation

We first focus on the evaluation of the impact of the text representation, by fixing the classifier. We choose to use the popular random forest (RF) algorithm. The models studied are listed in the two upper parts of Table 2. Table 3 presents the test accuracy of these models.

Let us first compare accuracy across models, on the complete dataset (D5). As expected, TF-IDF+RF is the less performing model (30.09 accuracy), TF-IDF being the historical representation and is a quite simple way to represent texts. Deep-learning based representations: Word2Vec+RF and sBERT+RF perform better, with an accuracy of 49.65 and 60.23 respectively. sBERT+RF performs better than Word2Vec+RF, which is in line with the literature, sBERT being the current best performing model in NLP, specifically on the semantic textual similarity task [25].

[7]https://www.dgl.ai/

Let us now consider the graph-based representation models. DGL4C-GCN, the end-to-end model we propose, performs slightly better than sBERT+RF, but this increase is not statistically significant. Regarding DGL4C-GRL+RF, it performs significantly better than sBERT+RF. We can conclude that graph-based representations are adequate for the resume classification task and that the use of neighborhood information in the representation learning, that combines both of semantic and structural information, is useful. Especially, this information can be viewed as a way to compensate the lack of data faced by deep learning models.

Let us now focus on the impact of the number of classes on the performance of the models, by studying performance on D1 to D5 datasets, i.e. from 5 to 24 classes. As expected, we can see that the performance of each model is negatively impacted by the increase of the number of classes. For example, the accuracy of DGL4C-GRL+RF is 94.38 with 5 classes and decreases to 67.87 with 24 classes. However, this performance does not decrease linearly with the number of classes. Especially, the performance between 11 to 20 classes remain stable. A significant decrease occurs between 20 to 24 classes, from 75.76 to 67.87. A similar decrease also occurs for the other graph-based model DGL4C-GCN. However, this is not the case for deep-learning-based models, nor for TF-IDF.

We can conclude that graph representation based models perform better than traditional deep learning based models. However, they seem to be less robust as the number of classes grows. Additional experiments would deserve to be conducted to identify if the decrease in performance is due to the number of classes or to characteristics of the 4 additional classes of D5.

4.3.2. Impact of the Classifier

We now focus on the evaluation of the impact of the classifier on the performance of DGL4C-GRL. We evaluate several well-known classifiers, namely Support Vector Classification, Multi-layer Perceptron, Logistic Regression, that we compare to the previously studied Random

Table 2
List of models studied

Model Name	Text Representation	Classifier
TF-IDF+RF	TF-IDF [26]	Random Forest
Word2Vec+RF	Word2Vec [5]	Random Forest
sBERT+RF	sBERT [25]	Random Forest
DGL4C-GCN	DGL4C-GCN	DGL4C-GCN
DGL4C-GRL+RF	DGL4C-GRL	Random Forest
DGL4C-GRL+LR	DGL4C-GRL	Logistic Regression
DGL4C-GRL+SVC	DGL4C-GRL	Support Vector Classification
DGL4C-GRL+MLP	DGL4C-GRL	Multi-Layer Perceptron

Table 3
Mean accuracy of the models studied.

Model \ Dataset	D1	D2	D3	D4	D5
TF-IDF+RF	70.50	42.43	35.65	34.65	30.09
Word2Vec+RF	78.43	63.16	51.76	52.24	50.64
sBERT+RF	92.23	73.15	68.14	66.97	61.65
DGL4C-GCN	93.54	**75.23**	**74.65**	73.66	62.43
DGL4C-GRL+RF	**94.38**	73.65	73.76	**75.76**	**67.87**

Forest. The list models studied is presented in the two lower parts of Table 2. The mean accuracy of these models is presented in Table 4, which also recalls the performance of the best deep learning model sBERT+RF and the graph-based end-to-end DGL4C-GCN model.

First of all, we can see that whatever is the classifier used, DGL4C-GRL still performs better than sBERT+RF on most of the datasets versions. If we focus on the impact of the classifier on the performance of DGL4C-GRL, LR and SVC are the two best performing classifiers, that slightly outperform the performance of RF. However, this improvement is not statistically significant. We can thus conclude that the nature of the classifier does not significantly impact the performance of the model. On the contrary, the graph representation seems to be the most influential step for the performance, which confirms the findings of the literature.

Considering DGL4C-GCN, it is the best performing model for two of the five datasets (D2 and D3). However, DGL4C-GCN has a significantly lower performance on D5 (62.43 accuracy) compared to the best performing model DGL4C-GRL+SVC (69.04 precision). This can be explained by the fact that an end-to-end model has one optimization function that optimises the representation learning and the classification, whereas DGL4C-GRL has two optimization functions used separately, which makes the classifier more flexible.

5. Conclusion and Future Work

In this paper we have proposed DGL4C, a deep semi-supervised graph representation learning based model for resume classification. This model can be used to provide recommendations to ATSs, human resource departments, and professional online social networks (e.g. Linkedin, Viadeo, Meetup, JobCase, etc).

DGL4C relies on a deep learning approach and adapts the GNN architecture to textual data. The experiments conducted demonstrate the performance of the two variants of DGL4C: DGL4C-GCN and DGL4C-GRL. Especially, both variants perform better than machine learning-based and deep learning-based models from the literature, including sBERT that has shown good performance on close uses cases. Experiments thus confirm the relevance of relying on a graph-based representation in the HR context.

In future works, we plan to adopt an unsupervised graph representation learning instead of a semi-supervised learning, which will be associated to the possibility of collecting and evaluating on larger datasets.

References

[1] A. Zaroor, M. Maree, M. Sabha, Jrc: a job post and resume classification system for online recruitment, in: 29th ICTAI, IEEE, 2017, pp. 780–787.

[2] A. Giabelli, L. Malandri, F. Mercorio, M. Mezzan-

Table 4
Mean accuracy of DGL4C-GRL with different classifiers

Model \ Dataset	D1	D2	D3	D4	D5
DGL4C-GRL+LR	**95.67**	73.99	74.23	74.85	67.10
DGL4C-GRL+SVC	94.20	72.03	74.60	75.65	**69.04**
DGL4C-GRL+MLP	95.08	72.25	72.96	73.76	65.77
DGL4C-GRL+RF	94.38	73.65	73.76	**75.76**	67.87
DGL4C-GCN	93.54	**75.23**	**74.65**	73.66	62.43
sBERT+RF	92.23	73.15	68.14	66.97	61.65

zanica, A. Seveso, Skills2job: A recommender system that encodes job offer embeddings on graph databases, Applied Soft Computing 101 (2021) 107049.

[3] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: AAAI, volume 33, 2019, pp. 7370–7377.

[4] D. Kim, D. Seo, S. Cho, P. Kang, Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec, Information Sciences 477 (2019) 15–29.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, NIPS 26 (2013).

[6] F. Hasan, A. Roy, S. Pan, Integrating text embedding with traditional nlp features for clinical relation extraction, in: ICTAI, IEEE, 2020, pp. 418–425.

[7] Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, arXiv preprint arXiv:1508.01991 (2015).

[8] C. Zhou, C. Sun, Z. Liu, F. Lau, A c-lstm neural network for text classification, arXiv preprint arXiv:1511.08630 (2015).

[9] K. Jiechieu, N. Tsopze, Skills prediction based on multi-label resume classification using cnn with model predictions explanation, Neural Computing and Applications 33 (2021) 5069–5087.

[10] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Generation Computer Systems 74 (2017) 76–85.

[11] M. Fazel-Zarandi, M. S. Fox, Semantic matchmaking for job recruitment: an ontology-based hybrid approach, in: ISWC, volume 525, 2009, p. 2009.

[12] M. de Groot, J. Schutte, D. Graus, Job posting-enriched knowledge graph for skills-based matching, arXiv preprint arXiv:2109.02554 (2021).

[13] H. Sajid, J. Kanwal, S. U. R. Bhatti, S. A. Qureshi, A. Basharat, S. Hussain, K. U. Khan, Resume parsing framework for e-recruitment, in: IMCOM, IEEE, 2022, pp. 1–8.

[14] S. Fareri, N. Melluso, F. Chiarello, G. Fantoni, Skillner: Mining and mapping soft skills from any text, Expert Systems with Applications 184 (2021) 115544.

[15] E. Abdollahnejad, M. Kalman, B. H. Far, A deep learning bert-based approach to person-job fit in talent recruitment, in: CSCI, IEEE, 2021, pp. 98–104.

[16] X. Zhang, K. Xie, S. Wang, Z. Huang, Learning based proximity matrix factorization for node embedding, in: SIGKDD, 2021, pp. 2243–2253.

[17] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural computation 15 (2003) 1373–1396.

[18] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: SIGKDD, 2014, pp. 701–710.

[19] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: SIGKDD, 2016, pp. 855–864.

[20] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: IJCNN, volume 2, 2005, pp. 729–734.

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

[22] Y. Ding, X. Zhao, Z. Zhang, W. Cai, N. Yang, Graph sample and aggregate-attention network for hyperspectral image classification, Geoscience and Remote Sensing Letters 19 (2021) 1–5.

[23] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[25] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).

[26] H. C. Wu, R. W. P. Luk, K. F. Wong, K. L. Kwok, Interpreting tf-idf term weights as making relevance decisions, ACM TOIS 26 (2008) 1–37.