SKILLBERT: "SKILLING" THE BERT TO CLASSIFY SKILLS!

Anonymous authors
Paper under double-blind review

ABSTRACT

In the age of digital recruitment, job posts can attract a large number of applications, and screening them manually can become a very tedious task. We propose a BERT-based model, SkillBERT, the embeddings of which are used as features for classifying skills into groups referred to as "competency groups". A competency group is a group of similar skills and it is used as matching criteria (instead of matching on skills) for finding the overlap of skills between the candidates and the jobs. This proxy match takes advantage of the BERT's capability of deriving meaning from the structure of competency groups present in the skill dataset. The problem that we are trying to solve is a multi-label classification problem, as a single skill can belong to multiple competency groups. To solve multi-label competency group classification using binary classifier, we have paired each skill with each competency group and tried to predict the probability of that skill belonging to that particular competency group. SkillBERT, which is trained from scratch on the skills present in job requisitions, is shown to be better performing than the pre-trained BERT (Devlin et al., 2019) and the Word2Vec (Mikolov et al., 2013). We have also explored K-means clustering (Lloyd, 1982) and spectral clustering (Chung, 1997) on SkillBERT embeddings to generate cluster-based features. Both algorithms provide similar performance benefits. Last, we have experimented with different classification models like Random Forest (Breiman, 2001), XGBoost (Chen & Guestrin, 2016), and a deep learning algorithm Bi-LSTM (Schuster & Paliwal, 1997; Hochreiter & Schmidhuber, 1997) for the tagging of competency groups to skill. We did not observe a significant performance difference among the algorithms, although XGBoost and Bi-LSTM perform slightly better than Random Forest. The features created using SkillBERT are most predictive in the classification task, which demonstrates that the SkillBERT is able to capture the information about the skills' ontology from the data. We have made the source code, the trained models and the dataset (Electronic Recruitment Records, referred to as ERRs)[1] of our experiments publicly available. ERRs are stored in the form of tables in our recruitment database.

1 INTRODUCTION

Competency group can be thought of as a group of similar skills required for success in a job. For example, skills such as *Apache Hadoop*, *Hive* represent competency in Big Data analysis while *HTML*, *CSS* are part of Front-end competency. Classification of skills into the right competency groups can help in gauging candidate's job interest and automation of the recruitment process.

Recently, there has been a surge in online recruitment activity. The researchers are using the data available through these online channels to find patterns in the skills of candidates and jobs. Several semantic approaches are also being used to minimise the manual labour required in the recruitment industry.

Bian et al. (2019) proposed a system to match the sentences from job posting and candidate resume using a deep global match network. They proposed a system which consists of finding the sentence-level representation and then used it for the sentence-level match and global match.

[1] https://www.dropbox.com/s/wcg8kbq5btl4gm0/code_data_pickle_files.zip?dl=0

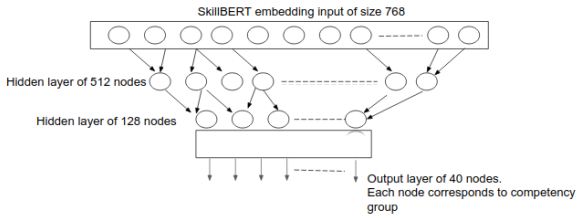


Figure 1: Classifier architecture

Xu et al. (2018) in their work measured the popularity of the job skills in the recruitment market using a multi-criteria approach and developed a novel Skill Popularity based Topic Model (SPTM).

Qin et al. (2018) developed an Ability-aware Person-Job Fit Neural Network (APJFNN) model on historical job application data using Recurrent Neural Network (RNN). They implemented four hierarchical ability-aware attention strategies to learn the job requirements based on the semantics.

Alabdulkareem et al. (2018) explained the dynamics such as the transition between occupations by workers, the comparative advantage of certain cities in new skills using skill topology. They also used clustering to show that two clusters are formed where one represents the social-cognitive skills and the second represents sensory-physical skills.

For learning features from the text data, several contextual word embedding models have been explored on various domain-specific datasets but no work has been done on exploring those models on job-skill specific datasets.

Fields like medical and law have already explored these models in their respective domains. Lee et al. (2019) in their BioBERT model trained the BERT model on a large biomedical corpus. They found that without changing the architecture too much across tasks, BioBERT beats BERT in several biomedical text mining tasks by a large difference.

Similarly, Elwany et al. (2019) in their work has shown the improvement in results on fine-tuning the BERT model on legal domain-specific corpora. They concluded that fine-tuning BERT gives the best performance and reduces the need for a more sophisticated architecture and/or features.

In this paper, we are proposing a competency group classifier, which primarily leverages: SkillBERT, which uses BERT architecture and is trained on the job-skill data from scratch to generate embeddings for skills. These embeddings are used to create several similarity-based features to capture the association between skills and group. We have also engineered features through clustering algorithms like spectral clustering on embeddings to attach cluster labels to skills. All these features along with SkillBERT embeddings are used in the final classifier to achieve the best possible classification accuracy.

2 METHODOLOGY

As no prior benchmark related to job-skill classification is available, we manually assigned each skill in our dataset to one or more competency groups with the help of the respective domain experts. We experimented with three different models: pre-trained BERT, Word2vec, and SkillBERT to generate skill embeddings. Word2vec and SkillBERT were trained from scratch on our skill dataset. We created similarity-based and cluster-based features on top of these embeddings. A detailed explanation of all the steps is mentioned in the next sections.

2.1 TRAINING DATA CREATION

Our approach uses a multi-label classification model to predict competency groups for a skill. However, as no prior competency group tagging was available for existing skills, we manually assigned labels for training data creation. For this task, the skill dataset is taken from our organization's database which contains 700,000 job requisitions and 2,997 unique skills. The 40 competency groups were created in consultation with domain experts of all major sectors.

Once training data is created, we trained a classification model using the features explained in the next section to classify a skill into 40 competency groups. As some skills can belong to more than one category hence our problem becomes a multi-label classification problem and therefore we prepared our final training data at *skill X competency group* level. In the training dataset, there is a total of 95,904 records while the testing dataset has 23,976 records.

2.2 Feature Engineering

2.2.1 SkillBERT Embedding

BERT has shown performance improvement for many natural language processing tasks. However, it has been minimally explored on the skill database. Hence, we leveraged BERT architecture on skill data to train SkillBERT model. The details of it are given below.

Training: For training SkillBERT, we extracted skills from 700,000 job requisitions and constructed a single document where each sentence represents the skills present in one requisition. These sentences are used as input instances for training SkillBERT model. As the training of BERT is highly resource-intensive, we leveraged AWS cloud machine type:*ml.p2.xlarge* with GPU memory *12 GiB* and processor *1xK80* GPU and it took around 72 hours for training. Once the training was finished, we extracted the last hidden layer output of size 768 from SkillBERT and further reduced the embedding size to 128 to decrease the training time of the classifier model used on top of the SkillBERT embedding. To make sure information from all the 768 dimensions is leveraged, we trained a 2-layer neural network classifier using SkillBERT embeddings as an independent feature and competency group as a dependent variable. Out of the 2,997 skills, 80% were used for training, and 20% were used for the validation. This model generates the probability values of a skill belonging to each of the 40 competency groups and was used as a feature in the final model at skill and competency group combination level. We have referred to this feature as "bert-prob" in rest of the sections. Figure 1 represents the architecture of the model used for getting these probabilities.

2.2.2 Other features

Similarity-based features: Cosine similarity values between skills and group embeddings were used as the features in the model.

Cluster-based feature: We experimented with *K-means* and *spectral clustering* on SkillBERT embeddings and generated 45 clusters using K-means and, 35 clusters using spectral clustering.

Skill TFIDF feature: TFIDF (Salton & McGill, 1986; Ramos, 1999) is used to find skills that are unique to a competency group and is used as a feature in the classification model.

3 Experiments

SkillBERT vs Word2vec vs Pre-trained BERT: As the first experiment, we compared SkillBERT, pre-trained BERT, and Word2vec models. For pre-trained BERT, we used "bert-base-uncased" model which produces embeddings of size 768. Similar to SkillBERT, we reduced embedding size to 128 and generated "bert-prob" feature. For Word2vec, we used embeddings of size 30. All features except cluster labels discussed in the feature engineering section were used. To better analyze the embeddings, we projected embeddings of skills present in competency groups in 2-D using t-SNE(van der Maaten & Hinton, 2008). Appendix Figure 6 and Figure 7 shows the difference in the segregation of groups generated using different embedding models.

K-means vs spectral clustering: In this experiment, we tried to see the effect of adding cluster-based features generated using K-means and spectral clustering on SkillBERT embedding. For the comparison, we applied XGBoost on the cluster labels and the features used in previous experiment.

Random Forest vs Bi-LSTM vs XGBoost: As part of this experiment, we applied Bi-LSTM, Random forest, XGBoost, and spectral clustering based features on SkillBERT to compare the performance. The pairs that were compared and had a statistically significant difference in performance are highlighted with a star in Table 1.

Table 1: Evaluation of results on different embedding models and feature sets

Model	Precision		Recall		F1-score	
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
*XGBoost + pre-trained BERT	98.83%	51.54%	95.85%	74.26%	97.21%	60.84%
*XGBoost + Word2vec	98.06%	68.34%	97.36%	65.21%	96.53%	66.73%
XGBoost + SkillBERT	99.32%	96.65%	99.47%	84.82%	99.39%	90.35%
XGBoost + SkillBERT + K-means	99.27%	96.92%	99.54%	85.24%	99.40%	90.70%
Random forest + SkillBERT + spectral clustering	99.28%	95.15 %	99.50%	83.48%	99.39%	88.93%
XGBoost + SkillBERT + spectral clustering	99.35%	97.23%	99.48%	85.09%	**99.41**%	**90.76**%
*Bi-LSTM + SkillBERT + spectral clustering	99.26%	95.86%	99.57%	86.43%	**99.42%**	**90.90%**

Impact evaluation: By matching competency groups instead of skills, we are broadening the spectrum for matching the skills between candidates and jobs. For instance, if a candidate has mentioned skills like 'AngularJS' and 'CSS', then there is a high probability that the candidate knows 'HTML' too as all of them belong to 'Web development' competency group. Matching using competency groups takes care of similar cases by providing an aggregation to the skills required for matching the skills between candidates and jobs. While screening the candidate resumes hiring managers come across many skills which are unknown to them. By normalizing the skills to the competency groups using SkillBERT, we are reducing the time taken by the hiring managers to find the domain of the skills. The difference in time is because the SkillBERT not only matches the skills to their domains (groups) but also shows constituent skills in each group, thereby providing more context. As of now, there is no automated way of tracking the resume screening rate on our platform. However, post introduction of SkillBERT, a 150% increase has been observed in the number of average resumes screened per day. The above metric does not account for the confounders like the hiring manager's experience and performance among other covariates.

4 RESULTS

Results shown in Table 1 conclude that SkillBERT improved the performance of the classification model over Word2vec and pre-trained BERT. Use of XGBoost with SkillBERT based features give F1-score of 90.35% for class 1 as compared to 60.83% and 66.73% of pre-trained BERT and Word2vec based features. Use of different machine learning (XGBoost and Random forest), deep learning (Bi-LSTM) algorithms, and clustering-based features (K-means and spectral clustering) on top of SkillBERT are not making a statistically significant difference and the results are very similar. The difference between the validation dataset and test dataset F1 scores was less than 0.65 and 0.5 percentage points and the variance of validation data F1 scores for different hyperparameter trials was 1.20 and 1.05 percentage points for XGBoost+SkillBERT+spectral clustering and Bi-LSTM+SkillBERT+spectral clustering models respectively. We computed feature importance using the XGBoost model and "bert-prob" explained in section 2.2.1 created using SkillBERT was the top feature in the list. TFIDF and similarity-based features were also highly predictive. All the reported results are statistically significant at $p<0.05$.

5 CONCLUSION

In this paper, we addressed the problem of recruiters manually scanning thousands of applications to find a suitable applicant for the posted job. To reduce the manual intervention, a multi-label skill classification model is developed to classify skills into competency groups and hence, helps in quick mapping of relevant applications to a job. The comparison among classification results of different machine learning models is also performed. Additionally, features like TFIDF, clustering labels, and similarity-based features are explored for better classification of skills. We trained BERT on a domain-specific dataset and a significant improvement is noticed while comparing the results with pre-trained BERT and Word2vec.

REFERENCES

Ahmad Alabdulkareem, Morgan R. Frank, Lijun Sun, Bedoor AlShebli, César Hidalgo, and Iyad Rahwan. Unpacking the polarization of workplace skills. *Science Advances*, 4(7), 2018. doi: 10.1126/sciadv.aao6030. URL https://advances.sciencemag.org/content/4/7/eaao6030.

Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. Domain adaptation for person-job fit with transferable deep global match network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4810–4820, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1487. URL https://www.aclweb.org/anthology/D19-1487.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A%3A1010933404324.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Emad Elwany, Dave Moore, and Gaurav Oberoi. {BERT} goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019. URL https://openreview.net/forum?id=rkeRMT9cLH.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz682. URL https://doi.org/10.1093/bioinformatics/btz682.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '18, pp. 25–34, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210025. URL https://doi.org/10.1145/3209978.3210025.

Juan Ramos. Using tf-idf to determine word relevance in document queries, 1999.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2572–2579. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16215.

6 APPENDIX

6.1 SPECTRAL CLUSTERING

Spectral clustering is a widely used unsupervised learning method for clustering. In spectral clustering, the data points are treated as nodes of a graph and these nodes are then mapped to a low-dimensional space using eigenvectors of graph laplacian that can be easily segregated to form clusters. Spectral clustering utilizes three matrices, details of those are given below.

1. Similarity graph (Affinity matrix): A similarity graph is a pair G = (V, A), where V=$\{v_1,....,v_m\}$ is a set of nodes or vertices. Different skills are forming different nodes as shown in Figure 2. A is a symmetric matrix called the affinity matrix, such that $ba_{ij} \geq 0$ for all i,j $\in \{1,.....,m\}$, and ba_{ii} = 0 for i = 1,.....,m. We say that a set$\{v_i,v_j\}$ is an edge if $ba_{ij} > 0$. Where ba_{ij} is bert affinity between nodes i and j computed using cosine similarity between SkillBERT embeddings of the corresponding skills. The corresponding (undirected) graph (V,E) with E = $\{\{v_i,v_j\} \mid ba_{ij} > 0\}$, is called the underlying graph of G. An example of similarity graph structure as affinity matrix is shown in Figure 2.

2. Degree matrix(D): If A is an m×m symmetric matrix with zero diagonal entries and with the other entries $ba_{ij} \in$ R arbitrary, for any node $v_i \in$V, the degree of v_i is defined as

$$d = d(v_i) = \sum_{j=1}^{m} |ba_{ij}| \tag{1}$$

and degree matrix D as

$$D = diag(d(v_1), ..., d(v_m)) \tag{2}$$

3. Graph laplacian (L): If D is a diagonal matrix and A is affinity matrix then we can compute L as follows :-

$$L = D - A \tag{3}$$

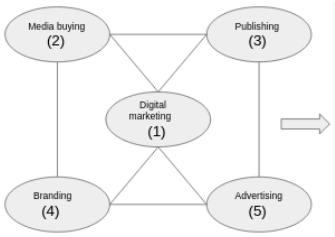
The Laplacian's diagonal is the degree of our nodes, and the off-diagonal is the negative edge weights (similarity between nodes). For clustering the data in more than two clusters, we have to modify our laplacian to normalize it.

$$L_{norm} = D^{-1/2} L D^{-1/2} \tag{4}$$

We know that

$$L_{norm} X = \lambda X \tag{5}$$

Where X is the eigenvector of L_{norm} corresponding to eigenvalue λ. Graph Laplacian is a semi-positive definite matrix and therefore, all its eigenvalues are greater than or equals to 0. Thus, we get eigenvalues $\{\lambda_1 , \lambda_2 , ... , \lambda_n\}$ where 0 = $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$



	Digital marketing	Media buying	Publishing	Branding	Advertising
Digital marketing	0	ba_{12}	ba_{13}	ba_{14}	ba_{15}
Media buying	ba_{12}	0	ba_{23}	ba_{24}	0
Publishing	ba_{13}	ba_{23}	0	0	ba_{35}
Branding	ba_{14}	ba_{24}	0	0	ba_{45}
Advertising	ba_{15}	0	ba_{35}	ba_{45}	0

Figure 2: Adjacency matrix representation of Graph

$d_1 = ba_{12} + ba_{13} + ba_{14} + ba_{15}$

$d_2 = ba_{12} + ba_{23} + ba_{24}$

$d_3 = ba_{13} + ba_{23} + ba_{35}$

$d_4 = ba_{14} + ba_{24} + ba_{45}$

$d_5 = ba_{15} + ba_{35} + ba_{45}$

	Digital marketing	Media buying	Publishing	Branding	Advertising
Digital marketing	d_1	$-ba_{12}$	$-ba_{13}$	$-ba_{14}$	$-ba_{15}$
Media buying	$-ba_{12}$	d_2	$-ba_{23}$	$-ba_{24}$	0
Publishing	$-ba_{13}$	$-ba_{23}$	d_3	0	$-ba_{35}$
Branding	$-ba_{14}$	$-ba_{24}$	0	d_4	$-ba_{45}$
Advertising	$-ba_{15}$	0	$-ba_{35}$	$-ba_{45}$	d_4

Figure 3: Graph laplacian for example in Figure 1

and eigenvectors X_1,X_2,...,X_n. An example of a sample laplacian matrix is given in Figure 3. Once we calculate the eigenvalues of L_{norm} and eigenvectors corresponding to smallest k eigenvalues where k is number of clusters, we create a matrix of these eigenvectors stacking them vertically so that every node is represented by the corresponding row of this matrix and use K-means clustering to cluster these new node representations into k clusters. For our experiment, we chose the first 35 eigenvectors to create 35 clusters and used them as features for model training. The number 35 was decided using the criteria of difference between two consecutive eigenvalues. As shown in Figure 4, the difference between eigenvalue 35 and 36 is significantly bigger.

6.2 MISCELLANEOUS

This section contains the results of experiments done for hyperparameter selection and some figures referenced in the paper. Figure 5 Shows the elbow method graph for deciding the number of clusters in K-means. Table 2 shows the effect of different SkillBERT embedding size on the results of XGBoost classifier. Table 3 contains the range of hyperparameters tested during hyperparameter tuning of the classifier models (Word2vec, XGBoost, Bi-LSTM) and the best hyperparameter values. Table 4 contains the training time required for each classifier model. To better analyze the quality of embeddings produced using Word2vec, pre-trained BERT, and SkillBERT, we projected high dimensional embeddings of skills present in competency groups in 2-D using t-SNE. From the visualization shown in Figure 6 and Figure 7, it is clear that SkillBERT embeddings reduced the overlapping gap between groups and gave well-defined cluster boundaries as compared to word2vec and pre-trained BERT.

6.3 SKILLBERT TRAINING

The dataset used for training the SkillBERT model can be downloaded from here. It contains the list of skills present in job requisitions. Table 5 contains examples of some candidate and job profiles. We leveraged Bert-Base architecture on the job-skill data to generate embeddings of size 768, details of it can be found here. Finally, the embeddings generated using the SkillBERT model can be downloaded here.

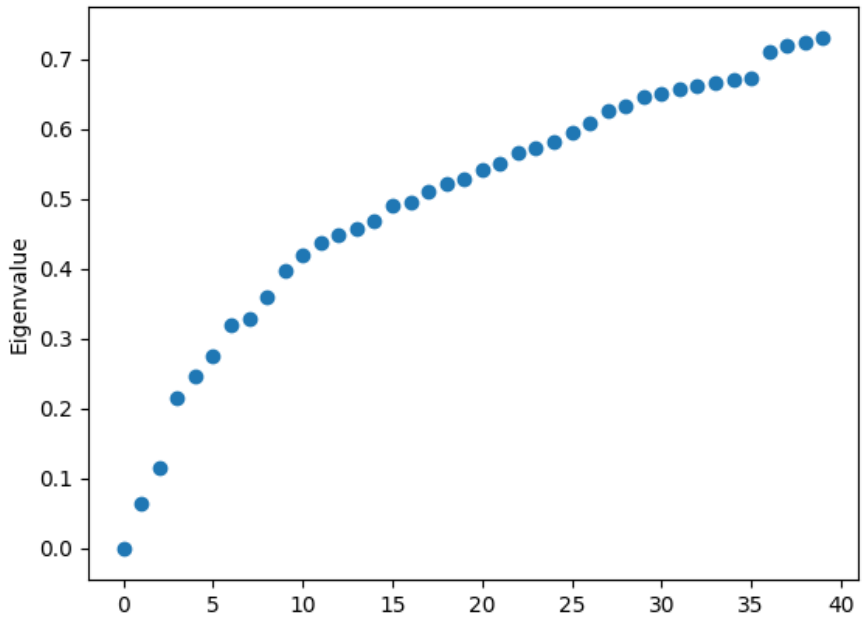


Figure 4: Scatter plot of eigenvalues to determine number of eigenvectors and clusters in spectral clustering

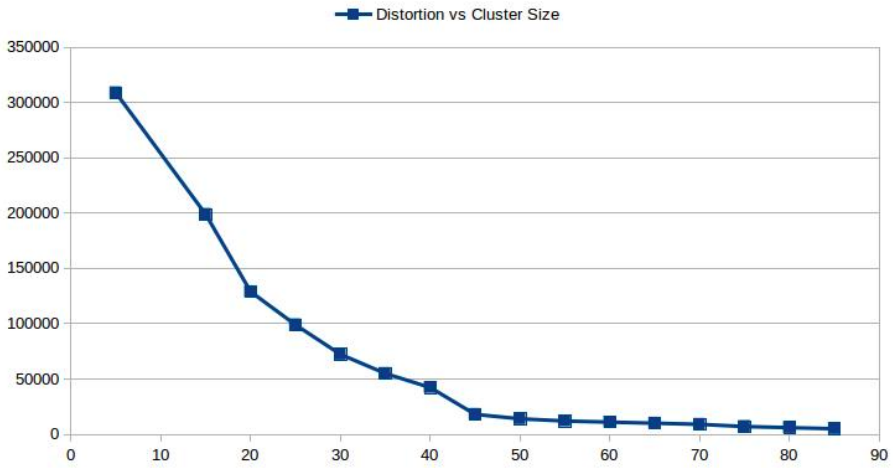


Figure 5: Elbow method graph to determine the number of clusters in K-means clustering

Table 2: Result for different embedding size (In this experiment, XGBoost was used as a classifier and bert-prob was used along with emdeddings of different sizes as independent variable. No other feature apart from these was used)

SkillBERT embedding size	Precision		Recall		F1-score	
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
32	98.12%	91.65%	95.47%	80.12%	96.78%	85.50%
64	98.32%	91.80%	97.26%	81.10%	97.79%	86.12%
128	99.12%	92.65%	97.47%	83.80%	98.29%	88.00%
256	99.12%	92.56%	97.40%	83.79%	98.25%	87.96%

Table 3: Machine Learning model Hyperparameters

Machine Learning Models	Best Hyperparameters	Hyperperameter bound
XGBoost	N_estimators:800, Depth:5	N_estimators:400-1000, Depth:3-7
Random forest	N_estimators:700, Depth:4	N_estimators:400-1000, Depth:3-7
Bi-LSTM	Layers:2(Nodes: 128, 64), Optimizer:Adam, Dropout:0.2	Layers:2 - 4 , Nodes: 32 - 512, Dropout: 0.1 - 0.5

6.4 ANNOTATED DATA PREPARATION

We had domain experts related to each field for manual annotation of competency group to a skill for generating the training dataset. Following instructions were given to them:

1. A single skill can belong to multiple groups

2. If they don't know about a particular skill then they can google it and based on that information can annotate that skill

The mapping of competency groups and skills can be downloaded here.

6.5 FEATURES

The details of features used in the training of Bi-LSTM model, which gave us the best performance are given in Table 6 .

6.6 TRAINING DATA FORMAT

The data format used for generating bert_ prob feature and final model is shown in Figure 8 and Figure 9 respectively.

6.7 RUNNING THE EXPERIMENT

The code to run all the experiments mentioned in the paper can be downloaded here. This codebase uses python 3.6 and all the packages used for this experiment can be downloaded by installing requirements.txt. An overview of all the folders present in the code is given below:

Table 4: Machine Learning model training time

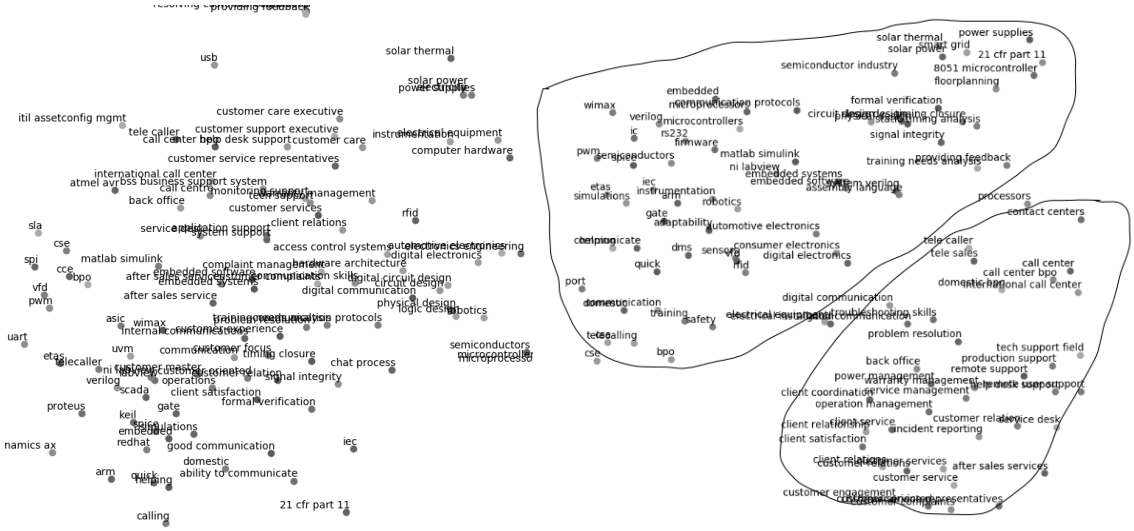
Machine Learning Models	Training Time (in seconds)
XGBoost	122
Random forest	87
Bi-LSTM	167

Table 5: Examples of some candidate and job profiles

Candidate or Job	Skill set	Probable competency groups
Candidate1	Design, knockoutjs, corel draw	Tool design, mechanical design, front end, web development
Candidate2	Statistical modeling, statistical process control	Statistics, production operations
Job1	Analytical skills, project execution, accounting	Financial operations, business analytics, statistics, accounts
Job2	Digital marketing, cash management, ms office, ms excel, ms word, tally	Taxation, banking, statistics

Table 6: Feature Description

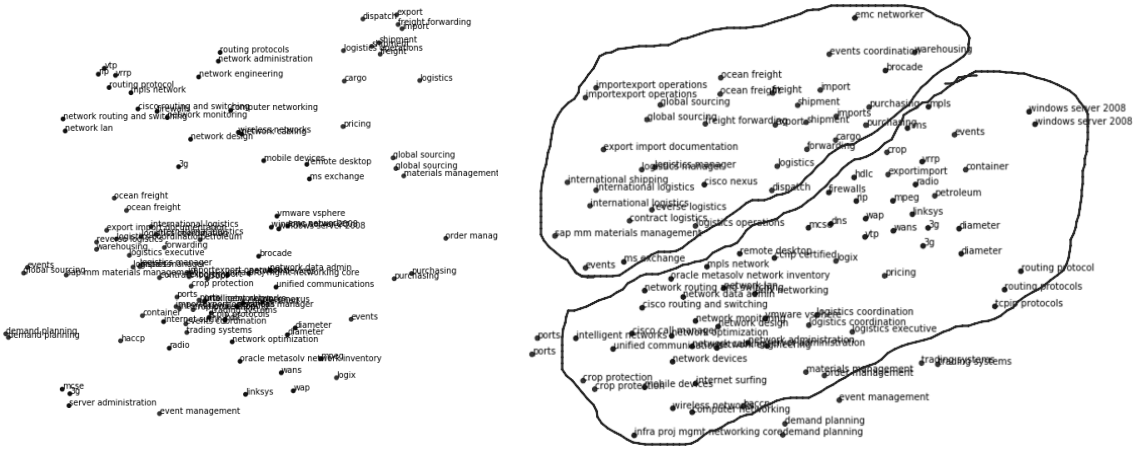
Feature Name	Feature Type	Dimensionality
bert_0 - bert_127	SkillBERT Embedding	128
bert-prob	SkillBERT Embedding	1
0-34	Spectral clustering label	35
value1-value3	skill-skill similarity	3
tf-idf	TFIDF	1
bert_grp_sim	skill-group similarity	1



(a) t-SNE plot of skills using pre-trained BERT

(b) t-SNE plot of skills using SkillBERT

Figure 6: t-SNE plot of embeddings of ”Customer Support” and ”Electronics” competency group. The left image shows the projection generated using pre-trained BERT embedding and the right image is the SkillBERT plot. The top cluster shown in SkillBERT t-SNE plot represents ”Electronics” competency group while the bottom cluster represents ”Customer Support”.



(a) t-SNE plot of skills using Word2vec

(b) t-SNE plot of skills using SkillBERT

Figure 7: t-SNE plot of embeddings of "Logistic" and "Network" competency group. The left image shows the projection generated using Word2vec embedding and the right image is the SkillBERT plot. The top cluster shown in SkillBERT t-SNE plot represents "Logistic" competency group while the bottom cluster represents "Network".

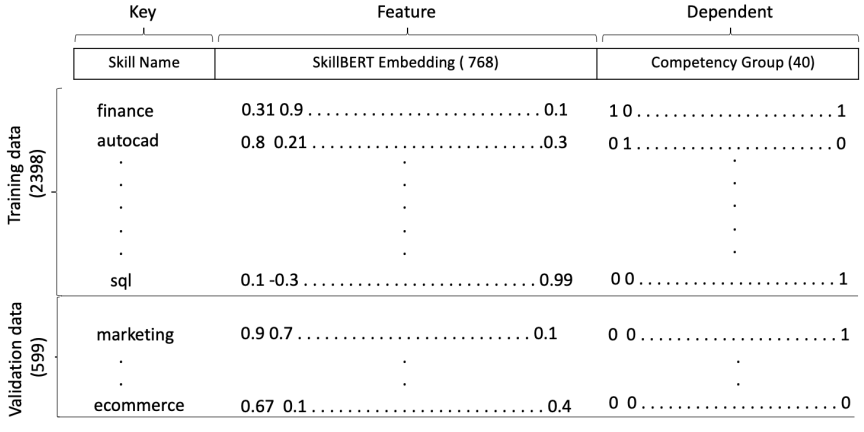
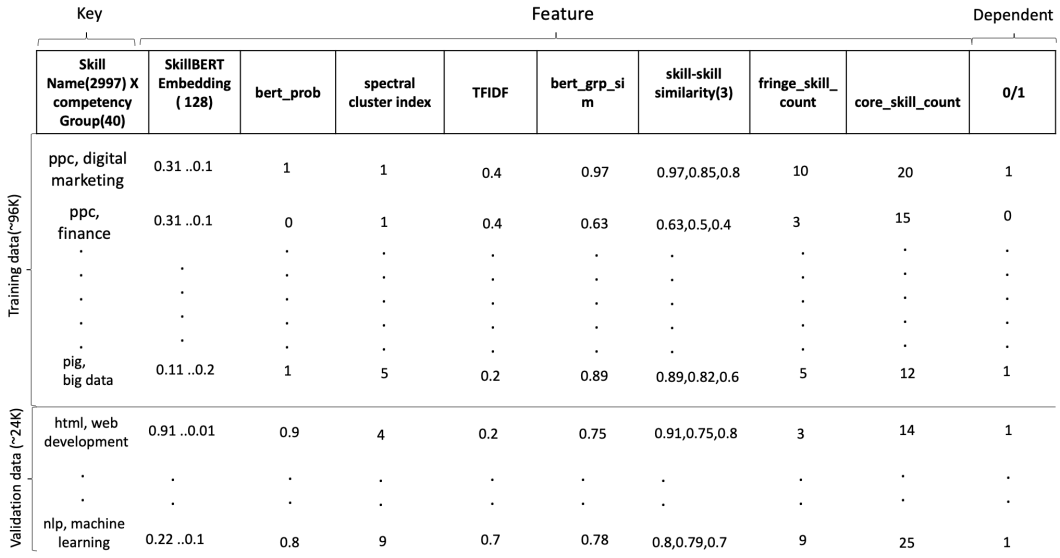


Figure 8: Data format used for creating bert_ prob feature



	Skill Name(2997) X competency Group(40)	SkillBERT Embedding (128)	bert_prob	spectral cluster index	TFIDF	bert_grp_sim	skill-skill similarity(3)	fringe_skill_count	core_skill_count	0/1
Training data(~96K)	ppc, digital marketing	0.31 ..0.1	1	1	0.4	0.97	0.97,0.85,0.8	10	20	1
	ppc, finance	0.31 ..0.1	0	1	0.4	0.63	0.63,0.5,0.4	3	15	0
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	pig, big data	0.11 ..0.2	1	5	0.2	0.89	0.89,0.82,0.6	5	12	1
Validation data (~24K)	html, web development	0.91 ..0.01	0.9	4	0.2	0.75	0.91,0.75,0.8	3	14	1
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	nlp, machine learning	0.22 ..0.1	0.8	9	0.7	0.78	0.8,0.79,0.7	9	25	1

Figure 9: Data format used for final model creation

1. training_codes: This folder contains the main python files used for running the experiments mentioned in the paper. Inside the main() method there are functions for data preparation, training, and testing. We have provided comments in each section for better understating of the modules. The code present in the file "skillbert_spectral_clustering.py" is used to train Bi-LSTM model on SkillBERT and spectral clustering related features which gave us the best performance. You can directly jump to this code if you don't want to run other intermediary experiments.

Apart from these if you want to run other experiments mentioned in the paper, you can do so by running "word2vec_only.py" for classifying skills using only Word2vec model, "skillbert.py" for classifying skills using only SkillBERT model, "bert_pretrain_only.py" for classifying skills using only pre-trained BERT model and "skillbert_and_kmeans.py" for classifying skills using SkillBERT and k-means on SkillBERT embedding.

2. feature_creation: This folder contains the code for creating features used for training the models. If you don't want to go through each code, features created using these code files are already available in the feature_data folder. Codes present in the training_code also uses these CSV files directly for the model training.

3. feature_data: As mentioned before, this folder contains CSV files of features generated using codes present in feature_creation folder.

4. model: This folder contains the final model trained using all the experiments mentioned in the paper. Folder "skill_bert_spectral_clustering" contains the Bi-LSTM model which has been used as the final model.

5. dataset: This folder contains the final training and testing data used for each experiment. You can use these files to directly test the corresponding model.