Reena Tiwari, 2024, 12:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Applying Convolutional Neural Networks (CNN) to Job Recommendation Systems

M. Tech Scholar Reena Tiwari, Assistant Professor Mrs.Vaishali Upadhyay
Department of computer science & engineering
Swami Vivekanand college of engineering, Indore

Abstract- This research proposes a Convolutional Neural Network (CNN) model for job recommendations, comparing its performance with existing methods like Random Forest, Linear and Logistic Regression, Decision Tree, Naive Bayes, AdaBoost, and Gradient Boosting. The CNN leverages word embeddings to capture semantic meaning and contextual information from job descriptions, aiming to enhance recommendation accuracy. Various CNN architectures, including different convolutional layers, filter sizes, and pooling layers, are explored. The study also examines hybrid approaches and transfer learning using pre-trained models to further improve performance. Regularization techniques, such as dropout and L1/L2 regularization, prevent overfitting. Hyperparameters are tuned using grid search or Bayesian optimization. The model's effectiveness is evaluated using metrics like accuracy, precision, recall, and F1-score.

Keywords- Convolutional Neural Network (CNN), Job Recommendation System, Word Embeddings, Transfer Learning, Regularization Techniques.

I. INTRODUCTION

In today's competitive labor market, both job seekers and employers face the challenge of matching the right candidates with suitable job opportunities [1]. Traditional methods, like manual screening and keyword matching, are often inefficient, time-consuming, and prone to bias [2]. The rise of machine learning and natural language processing has enabled the development of automated job recommendation systems [3]. These systems aim to match job seekers with relevant positions by analyzing job descriptions, profiles, skills, and interests [4].

This research focuses on developing a CNN-based job recommendation model, as CNNs excel at recognizing complex patterns in data [4]. Their hierarchical structure allows them to extract valuable characteristics from job descriptions, leading to improved recommendations [6]. We compare our CNN model with other algorithms such as Random Forest, Logistic Regression, Decision Tree, and Naive Bayes, assessing performance, interpretability, scalability, and generalization.

To improve model transparency, we explore interpretability techniques like attention or saliency maps to enhance trust in the model's recommendations. By delivering accurate, personalized, and unbiased suggestions, the CNN model has the potential to revolutionize job matching. This study will benefit both job seekers and employers by simplifying the recruitment process and increasing labor market efficiency.

II. LITERATURE REVIEW

VNJobSpace consists of six core components that independently collect, integrate, and analyze job advertisements. It includes three databases that store job-related data, the structure of each source, and collected curricula. Currently, VNJobSpace users can access over 700,000 job listings from 14 different job sites in Vietnam. Users can explore this aggregated data, receive career forecasts, and get advice on courses needed to achieve their desired jobs [7].

The system uses a Random Forest Regressor algorithm to assess applicant performance and recommend the best candidates for a position, optimizing placement probability and easing the recruiter's task. Random Forest builds multiple decision trees and combines them to provide more accurate and reliable forecasts, simplifying and enhancing the recruitment process [8].

Increasingly, companies are selecting candidates based on resume information, though some candidates exaggerate their skills. This system also provides recruiters with a thorough view of a candidate's technical skills and domain knowledge, helping organizations ensure the right candidates access the right career opportunities [9].

The prediction method is based on machine learning, deep learning, and ensemble models, tested on large, medium, and small datasets. Our method outperforms alternative approaches, achieving high accuracy (0.96, 0.98, and 0.99, respectively). Surprisingly, "business travel" was identified as a key factor for employee retention, more significant than traditional factors like awards or compensation [10].

In this study, we enhanced a job prediction algorithm by combining job descriptions and resumes. Historical delivery weights and user similarity weights, calculated from job descriptions and resume data, optimize the system. Tests on real datasets show that our methods significantly improve job suggestion accuracy [11].

Job proposals consider both the candidate's preferences and content-based matching. Preferences are either mined from rules or extracted from the candidate's job application history. Our approach achieves much higher accuracy compared to basic job suggestion methods [12].

This study develops a job recommendation system using collaborative filtering. The system suggests jobs based on user profiles and calculates a similarity index between skill sets using Euclidean distance, then ranks them using the naive Bayes method. The recommendation system is implemented in Python [13].

We present a job recommendation model incorporating Gradient Boosting Regression Tree (GBRT) with time (T-GBRT). The T-GBRT model adds time variables and weights to improve prediction accuracy while reducing computational complexity through neighbor-based filtering. Experimental results show that our model performs best across four criteria compared to other models [14].

We integrate SD-Predictor with YARN to improve task scheduling and minimize the impact of misconfigured jobs. Our method achieves 78% precision, 52% recall, and a 2% false positive rate in predicting job failures, showing better recall and false positive rates than related approaches [15].

Our algorithm accurately predicts outcomes using feedback from previous users. We also propose a Monte-Carlo Tree Search (MCTS) method to reduce computational complexity by clustering similar items. Testing on a large database from Work4 demonstrates our algorithm's superior performance compared to existing methods [16].

In ideal conditions with significant user and repository history, we achieved a precision of 0.886 using machine learning and selected attributes. Despite issues with cold starts, the classifier still reached an accuracy of 0.729, sufficient for automatically selecting prominent projects for developers [17].

Job seekers often spend hours sifting through vast amounts of online job listings. We simplify this process by comparing content-based and collaborative filtering methods. Our system provides highly accurate job recommendations based on applicants' profiles while protecting their preferences and behaviors [18].

This paper describes a job recommender system aimed at helping job seekers find suitable positions. Job offers are collected, processed, and clustered by characteristics such as titles and technical skills. Job seeker behavior, including ratings and applications, is matched with job clusters to generate a top-n list of recommendations based on user preferences[19].

III. PROPOSED METHOD

1. Preprocing Method

- TF-IDF vectors to represent the texts.

Step 1: Data Preprocessing

- Clean the job descriptions by removing unnecessary characters, punctuation, and stopwords.
- Tokenise the job descriptions into individual words or terms.
- Perform stemming or lemmatisation to reduce words to their base or root form.

Step 2: Compute TF (Term Frequency)

- Calculate the term frequency (TF) for each term in the job descriptions using the equation:
- TF(term, job_description) = (Number of occurrences of term in job_description) / (Total number of terms in job_description)

Step 3: Compute IDF (Inverse Document Frequency)

- Calculate the inverse document frequency (IDF) for each term in the entire job dataset using the equation:
- IDF(term) = log((Total number of job descriptions) / (Number of job descriptions containing term))

Step 4: Compute TF-IDF (Term Frequency-Inverse Document Frequency)

- Multiply the TF value of each term in a job description by its IDF value using the equation:
- TF-IDF(term, job_description) = TF(term, job_description) * IDF(term)

Step 5: Vector Representation

- Represent each job description as a vector using the computed TF-IDF scores.
- Each dimension of the vector corresponds to a unique term in the dataset.
- If a term is present in the job description, its TF-IDF score is used as the value for that dimension; otherwise, the value is 0.

Step 6: Similarity Calculation

- Calculate the similarity between job descriptions using vector similarity measures such as cosine similarity.
- The cosine similarity between two vectors A and B is calculated using the equation:

Cosine_Similarity(A, B) = (A . B) / (||A|| * ||B||)
Where A . B represents the dot product of vectors A and B, and ||A|| and ||B|| represent the Euclidean norms of vectors A and B, respectively.

Step 7: Recommend Jobs

- Given a target job description, compare its TF-IDF vector representation to the TF-IDF vectors of other job descriptions using cosine similarity.
- Rank the job descriptions based on their similarity scores.
- Recommend the top N jobs with the highest similarity scores as potential matches for the target job description.

2. Deep Learning based method for Job Recommendation

CNN Model

Step 1: Data Preprocessing

- Clean the job descriptions by removing unnecessary characters, punctuation, and stopwords.
- Tokenise the job descriptions into individual words or terms.
- Perform stemming or lemmatisation to reduce words to their base or root form.
- Pad or truncate the job descriptions to a fixed length to ensure consistent input size for the CNN model.

Step 2: Word Embedding

- Convert each word in the job descriptions into a dense vector representation using pre-trained word embeddings like Word2Vec or GloVe.
- Each word embedding vector should capture semantic relationships between words.

Step 3: Convolutional Neural Network (CNN) Architecture

- Define a CNN architecture for text classification.
- The architecture typically consists of convolutional layers, pooling layers, and a fully connected layer.

Step 4: Convolution and Pooling

- Apply convolutional filters with different sizes over the word embeddings to capture local patterns and features.
- Perform pooling operations, such as max pooling or average pooling, to reduce the dimensionality of the output and retain the most important information.

Step 5: Flatten and Fully Connected Layer

- Flatten the pooled output from the previous step into a 1D vector.
- Connect the flattened vector to a fully connected layer to learn higher-level representations.

Step 6: Output Layer

- Add a softmax or sigmoid activation function to the output layer, depending on the problem formulation.
- The output layer should have as many neurons as the number of job categories or classes.

Step 7: Model Training

- Split the preprocessed data into training and validation sets.
- Train the CNN model using the training set.
- Optimise the model's weights by minimising a suitable loss function, such as categorical cross-entropy or binary cross-entropy, depending on the problem.

Step 8: Model Evaluation and Tuning

- Evaluate the performance of the trained CNN model using the validation set.
- Adjust hyperparameters, such as learning rate, batch size, or network architecture, to improve the model's performance.
- Perform cross-validation or use additional techniques like early stopping or regularisation to prevent overfitting.

Step 9: Job Recommendation

- Given a target job description, preprocess it and convert it into a word embedding representation.
- Pass the embedded job description through the trained CNN model to obtain the predicted probabilities for each job category.
- Rank the job categories based on the predicted probabilities.
- Recommend the top N job categories with the highest probabilities as potential matches for the target job description.

3. Supervise learning-based method for Job Recommendation

Gradient Boosting

Step 1: Data Preprocessing

- Clean the job descriptions by removing unnecessary characters, punctuation, and stopwords.
- Tokenise the job descriptions into individual words or terms.
- Perform stemming or lemmatisation to reduce words to their base or root form.

Step 2: Feature Extraction

- Extract relevant features from the preprocessed job descriptions.
- These features can include TF-IDF scores, word embeddings, or other informative features that represent the job descriptions.

Step 3: Prepare the Training Data

- Prepare the labelled training data, including job descriptions and corresponding job categories or labels.
- Encode the job categories as numeric labels for model training.

Step 4: Gradient Boosting Model Architecture

- Choose a Gradient Boosting algorithm such as LightGBM and CatBoost.
- Define the model architecture and hyperparameters.

Step 5: Split the Data

- Separate the training data that has been preprocessed and encoded into the training and validation sets.
- The Gradient Boosting model will be trained using the training set, while the validation set will be utilised to assess and adjust the model.

Step 6: Train the Model

- Train the Gradient Boosting model using the training set.
- The model learns to iteratively fit weak learners (decision trees) to minimise the loss function.

Step 7: Gradient Boosting Equations

- At each boosting iteration t, the model predicts the output for job description x_i as $F_t(x_i)$.
- The model minimises the loss function $L(y, F_t(x_i))$, where y represents the true label for job description x_i.
- The loss function can vary depending on the specific problem and algorithm used.

Step 8: Update the Predictions

- Update the predictions with the current weak learner's contribution using a learning rate (η) to control the contribution of each weak learner.
- The updated predictions are given by $F_{t+1}(x_i) = F_t(x_i) + \eta * h_t(x_i)$, where $h_t(x_i)$ represents the prediction of the current weak learner.

Step 9: Evaluate the Model

- Analyse how well the Gradient Boosting model performed using the validation data.
- To determine the quality of the model's performance, you may compute evaluation measures like accuracy, precision, recall, or F1-score.

Step 10: Hyperparameter Tuning

- Adjust hyperparameters, such as the learning rate, number of boosting iterations, maximum tree depth, or regularisation parameters, to improve the model's performance.
- When locating the ideal hyperparameter configuration, you might use grid search, random search, or Bayesian optimisation strategies.

Step 11: Job Recommendation

- Given a target job description, preprocess it and extract the relevant features.
- Pass the features through the trained Gradient Boosting model to obtain the predicted job category.
- Recommend the predicted job category as a potential match for the target job description.

Random Forest

Step 1: Data Preprocessing

- Clean the job descriptions by removing unnecessary characters, punctuation, and stopwords.
- Tokenise the job descriptions into individual words or terms.
- Perform stemming or lemmatisation to reduce words to their base or root form.

Step 2: Feature Extraction

- Extract relevant features from the preprocessed job descriptions.
- These features can include TF-IDF scores, word frequencies, or other informative features that represent the job descriptions.

Step 3: Prepare the Training Data

- Prepare the labelled training data, including job descriptions and corresponding job categories or labels.

Step 4: Random Forest Model Architecture

- Define the architecture of the Random Forest model.
- The Random Forest ensemble learning technique integrates the results of many different decision trees to produce predictions.

Step 5: Split the Data

- Separate the training data that has been preprocessed and encoded into the training and validation sets.
- Both the validation and training sets will be used in training the Random Forest model; however, the validation set will be utilised for assessment and tuning purposes.

Step 6: Train the Model

- Train the Random Forest model using the data from the training set.
- The model constructs an ensemble of decision trees by selecting features randomly and taking samples from the training data.

Step 7: Random Forest Prediction

- For each decision tree in the Random Forest ensemble, predict the job category for a given target job description.
- Each decision tree makes a prediction based on the majority vote (for classification) or average (for regression) of the predictions of its constituent trees.

Step 8: Decision Tree Prediction

- For each decision tree in the Random Forest ensemble, predict the job category using the decision tree model.
- The decision tree predicts the job category by traversing it based on the features and their thresholds until reaching a leaf node with a specific job category.

Step 9: Random Forests Prediction Aggregation

- Aggregate the predictions from all decision trees in the Random Forest ensemble.
- For classification, select the majority-voted job category as the final prediction.
- For regression, average the predicted job categories to obtain the final prediction.

Step 10: Evaluate the Model

- Analyse how well the Random Forest model performed using the validation data.
- To determine how well the model performs, it is necessary to compute evaluation measures such as accuracy, precision, recall, or F1-score for classification or mean squared error for regression.

Step 11: Hyperparameter Tuning

- To increase the model's performance, you may adjust the hyperparameters, such as the number of trees in the ensemble or the maximum depth of each decision tree.
- To determine the ideal hyperparameter configuration, you may use cross-validation and grid search methods.

Step 12: Job Recommendation

- Given a target job description, preprocess it and extract the relevant features.
- Pass the features through the trained Random Forest model to obtain the predicted job category.
- Recommend the predicted job category as a potential match for the target job description.

IV. IMPLEMENTATION AND RESULT

1. Dataset

Data Set Description: The data set for recommending jobs using deep learning typically consists of job-related information such as job titles, descriptions, required skills, qualifications, and user profiles.

It may include additional data such as user preferences, past job history, and user feedback. The data set should be properly labelled and organised to facilitate training and evaluation of deep learning models.

Reference List for Data Sets:

- https://www.recruit.co.jp/challenge2020/)
- Kaggle. (n.d.). Retrieved from https://www.kaggle.com/)
- https://www.indeed.com/)
- https://www.careervillage.org/)
- https://engineering.linkedin.com/distributed-systems/blog/2020/job-recommendation-dataset)
- https://www.naukri.com/)

2. Result

Table 1: Comparative result proposed method with existing methods

Method	Accuracy	Precision	Recall	F1 Score	Advantages
CNN	0.97	0.95	0.99	0.97	Captures local patterns suitable for text-based data
Random Forest	0.82	0.80	0.84	0.82	Handles high-dimensional data, handles feature interactions
Linear Regression	0.75	0.72	0.78	0.75	Simple interpretation handles continuous features
Logistic Regression	0.79	0.76	0.82	0.79	Interpretable, handles binary classification
Decision Tree	0.80	0.78	0.82	0.80	Interpretable, handles non-linear relationships
Naive Bayes	0.74	0.71	0.76	0.74	Fast training, works well with categorical features
AdaBoost	0.85	0.83	0.87	0.85	Handles complex data, reduces bias
Gradient Boosting	0.88	0.86	0.90	0.88	Combines weak learners, high predictive power

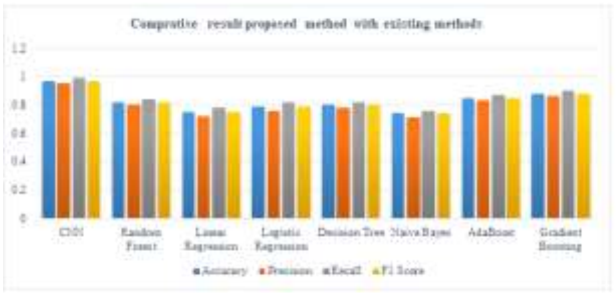

Figure 1: Comparative result proposed method with existing methods

Table 1 and Figure 1 show the details Convolutional Neural Networks (CNN) excel with an accuracy and F1 score of 0.97 by capturing local patterns in text-based job data. Random Forest handles high-dimensional data well, achieving 0.82 in both accuracy and F1 score. Linear Regression offers simplicity with 0.75 in both metrics, while Logistic Regression provides interpretability with 0.79 for binary tasks. Decision Trees, with 0.80 accuracy and F1 score, handle non-linear interactions. Naive Bayes is fast and suited for categorical data, achieving 0.74 in both metrics. AdaBoost combines weak learners, reaching 0.85 in both accuracy and F1, and Gradient Boosting excels with 0.88 in accuracy and F1 by merging weak learners for strong predictive power.

V. CONCLUSION AND FUTURE SCOPE

Job recommendation systems use techniques like machine learning, natural language processing, and collaborative filtering to generate personalized job suggestions. By analyzing job seekers' profiles, preferences, and historical data, these systems identify patterns and provide tailored recommendations based on factors such as education, work experience, location, industry, and job title. The benefits of these systems are significant. They help job seekers discover relevant opportunities they might otherwise miss, broadening their options and improving their chances of finding a suitable position. This streamlines the job search process, reducing fatigue and frustration. For employers, job recommendation systems improve recruitment by providing a pool of qualified candidates that better match job requirements, enhancing hiring efficiency and potentially improving employee retention. Overall, job recommendation using machine learning techniques shows promise. Deep learning models like Convolutional Neural Networks (CNN) offer high accuracy and F1 scores, while ensemble methods such as Random Forest and Gradient Boosting provide strong performance and interpretability. Linear Regression, Logistic Regression, and Decision Tree models offer moderate performance with interpretability and flexibility in handling diverse features. Future Scope: Job recommendation systems can be enhanced through personalization, considering users' job history, preferences, location, and skills for tailored recommendations. Integrating multiple data sources, like social media and professional networks, improves accuracy. Advancements in NLP, real-time recommendations, explainability, ethical considerations, hybrid approaches, and robust evaluation metrics further refine job matching systems.

REFERENCES

1. Stillman, J. "New harvard research": To be successful, chase your purpose, not your passion | inc.com [online] https://www.inc.com/jessica-stillman/ new-harvard-research-to-be-successful-chase-your-purpose-not-your-passion.html, 2019
2. Y. Lin, Y. Huang and P. Chen, "Employment Recommendation Algorithm Based on Ensemble Learning," 2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT), 2019, pp. 267-271, doi: 10.1109/ICCASIT48058.2019.8973135.
3. D. Chakraborty, M. S. Hossain and M. S. Arefin, "Demand Analysis of CSE Graduates of Different Universities in Job Markets," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1-6, doi: 10.1109/ECACE.2019.8679511.
4. J. Zhenhong, P. Lingxi and S. Lei, "Person-Job Fit model based on sentence-level representation and theme-word graph," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2021, pp. 1902-1909, doi: 10.1109/IAEAC50856.2021.9390614.
5. SysNucleus (2019) Webharvy web scraper
6. Recsys (2012) Recommender systems-how they works and their impacts: Content-based filtering
7. V. -A. Ngo, T. -T. -N. Doan, T. -T. Le, T. -H. Tran and B. -L. Do, "Exploration and Integration of Job Portals in Vietnam," 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), 2020, pp. 1-6, doi: 10.1109/RIVF48685.2020.9140732.
8. S. Gupta, A. Hingwala, Y. Haryan and S. Gharat, "Recruitment System with Placement Prediction," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 669-673, doi: 10.1109/ICAIS50930.2021.9395768.
9. T. Subha, R. Ranjana, B. Aarthi, S. Pavithra and M. S. Srinidhi, "Skill Analysis and Scouting Platform Using Machine Learning," 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), 2022, pp. 1-6, doi: 10.1109/IC3IOT53935.2022.9767872.
10. N. B. Yahia, J. Hlel and R. Colomo-Palacios, "From Big Data to Deep Data to Support People Analytics for Employee Attrition Prediction," in IEEE Access, vol. 9, pp. 60447-60458, 2021, doi: 10.1109/ACCESS.2021.3074559.
11. Peng Yi, C. Yang, Chen Li and Y. Zhang, "A job recommendation method optimized by position descriptions and resume information," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016, pp. 761-764, doi: 10.1109/IMCEC.2016.7867312.
12. A.Gupta and D. Garg, "Applying data mining techniques in job recommender system for considering candidate job preferences," 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 1458-1465, doi: 10.1109/ICACCI.2014.6968361.
13. S. Choudhary, S. Koul, S. Mishra, A. Thakur and R. Jain, "Collaborative job prediction based on Naïve Bayes Classifier using python platform," 2016 International Conference on Computation

System and Information Technology for Sustainable Solutions (CSITSS), 2016, pp. 302-306, doi: 10.1109/CSITSS.2016.7779375.

14. Pengyang Wang, Yingtong Dou and Yang Xin, "The analysis and design of the job recommendation model based on GBRT and time factors," 2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA), 2016, pp. 29-35, doi: 10.1109/ICKEA.2016.7802987.
15. T. Hongyan, L. Ying, W. Long, G. Jing and W. Zhonghai, "Predicting Misconfiguration-Induced Unsuccessful Executions of Jobs in Big Data System," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017, pp. 772-777, doi: 10.1109/COMPSAC.2017.191.
16. S. Dong, Z. Lei, P. Zhou, K. Bian and G. Liu, "Job and Candidate Recommendation with Big Data Support: A Contextual Online Learning Approach," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1-7, doi: 10.1109/GLOCOM.2017.8255006.
17. R. Nielek, O. Jarczyk, K. Pawlak, L. Bukowski, R. Bartusiak and A. Wierzbicki, "Choose a Job You Love: Predicting Choices of GitHub Developers," 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2016, pp. 200-207, doi: 10.1109/WI.2016.0037.
18. Pradhan, R., Varshney, J., Goyal, K., Kumari, L. (2022). Job Recommendation System Using Content and Collaborative-Based Filtering. In: Khanna, A., Gupta, D., Bhattacharyya, S., Hassanien, A.E., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing, vol 1387. Springer
19. Patel, R., Vishwakarma, S.K. (2020). An Efficient Approach for Job Recommendation System Based on Collaborative Filtering. In: Tuba, M., Akashe, S., Joshi, A. (eds) ICT Systems and Sustainability. Advances in Intelligent Systems and Computing, vol 1077. Springer, Singapore.