An Intelligent Decision Support System For Recruitment: Resumes Screening and Applicants Ranking

Arwa Najjar
Information Technology College, Hebron University, Hebron, Palestine
E-mail: ar1993wa@gmail.com

Belal Amro
Information Technology College, Hebron University, Hebron, Palestine
E-mail: Bilala@hebron.edu

Mário Macedo
Sciences and Technologies of Information and Communication College, Atlântica University, Lisbon, Portugal
E-mail: mariojcmacedo@gmail.com

Keywords: recruitment, Decision Support System (DSS), Intelligent Decision Support System (IDSS), Artificial Intelligence (AI), Machine Learning (ML), Natural Language Processing (NLP).

Received: November 7, 2020

The task of finding the best job candidates among a set of applicants is both time and resource-consuming, especially when there are lots of applications. In this concern, the development of a decision support system represents a promising solution to support recruiters and facilitate their job. In this paper, we present an intelligent decision support system named I-Recruiter, that ranks applicants according to the semantic similarity between their resumes and job descriptions; the ranking process is based on machine learning and natural language processing techniques. I-Recruiter is composed of three sequentially connected blocks namely 1) Training block: which is responsible for training the model from a set of resumes, 2) Matching block: that is responsible for matching the resumes to the corresponding job description, and 3) Extracting block: that is responsible for extracting the top n ranked candidates. Experimental results for accuracy and performance showed that I-recruiter is capable of doing the job with high confidence and excellent performance.

Povzetek: Predlagan je inteligentni sistem za podporo odločanju (IDSS) za pregledovanje in razvrščanje življenjepisov prosilcev na podlagi strojnega učenja in obdelave naravnih jezikov..

1 Introduction

Organizations always seek to hire employees who perfectly suit the job. Improper selection decisions for a new employee often have costly impacts on the work. Hence, Persons who stand behind the selection decision face an arduous task of selecting the most appropriate person from several applicants.

Recruitment is the process of searching, attracting, and hiring qualified applicants for employment in an organization [1]. Figure 1 presents an overview of the key steps of the recruitment process.

A recruitment process starts with the advertising of an available job position. This is carried out using diverse advertising channels such as websites, newspapers, and others. Job seekers who are interested in that job will apply for the job by creating their profiles using a designated online form or uploading their resumes through the organization's website. Received applications are then screened to find out the suitable candidates to interview.

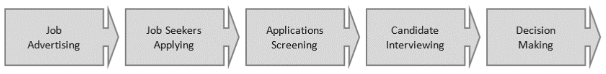


Figure 1: Overview of the recruitment process.

Screeners firstly should understand the requirement for the job. After that, they look through each of the submitted applications and reject applicants who do not meet the requirements. Finally, they find the best applicant who matches the job by comparing resumes with the job profile. The top few candidates listed during the screening stage will go along advanced stages in the process of evaluation, like interviews, written tests, and group discussions. The feedback received from the evaluation processes is used to make the final hiring decision. The candidate who passes the interview stage will be offered the position [2].

Human resources (HR) staff need to spend a significant amount of time going through applications in order to identify the few candidates who are truly qualified for the position. Automated systems can scan resumes for job compatibility, reducing efficiently HR's time spent analyzing documents and performing the job with high accuracy as well [3].

A Decision Support System (DSS) is an interactive computer system that helps decision-makers to use data, models, and knowledge in solving structured, semi-structured, or unstructured problems. Any DSS employs Artificial Intelligence (AI) techniques to generate decision alternatives called Intelligent Decision Support System (IDSS) [4]. AI is the development of machines that work and react as though they were intelligent [5]. The development of Decision Support Systems (DSS) is a promising solution for the candidate selection process for a job position in terms of time and effort. As organizations today receive a large number of resumes each time they advertise a job. The needed time and effort for screening is directly proportional to the number of applicants.

This paper presents an IDSS called I-Recruiter for applicants resumes screening to find the best job match ones in the Information Technology sector (IT). I-Recruiter ranks applicants according to the semantic similarity between the resume and job description. Then, it presents the top-ranked candidates' details to go ahead with the recruitment stages. System functions build on the basis of machine learning and natural language processing.

The rest of the paper is organized as follows. In section 2, we will present the related works. The system overview is explained in section 3. I-Recruiter implementation and results are discussed in section 4. In the end, a conclusion and future work is provided in section 5.

2 Related works

This work is related to two disciplines of Artificial Intelligence (AI). First, Machine Learning (ML) which is the discipline of giving programs the ability to learn and adapt. Here data represent the experiences where ML models derived. These models help in capturing complicated hidden patterns of new data [6]. The second discipline in AI is Natural Language Processing (NLP) which is the discipline of processing spoken or written forms of free text used by humans with the use of computational methods [7]

For the ML discipline, there are different approaches used approaches namely 1) supervised learning, 2) unsupervised learning, 3) reinforcement learning, and 4) deep learning [8]. In supervised learning, patterns are found from data with labeled features that define their meaning. It is used for weather forecasting. Unsupervised learning is more suitable for unlabeled data. As the data that comes from social media applications. Unlike previous approaches, reinforcement learning depends on trial and error and not on a set of data for training. This kind of learning can be used for training robots. The deep learning main concept relies on the base of incorporating neural networks in consecutive layers for learning. This approach is most suitable for the training of unlabeled and unstructured data in the cases of image recognition, speech, and computer vision [8].

NLP tasks include 1) Sentence Boundary Detection, 2) Tokenization, 3) Part-Of-Speech Assignment To Individual Words (POS Tagging), 4) Morphological Decomposition Of Compound Words, 5) Shallow Parsing (Chunking), 6) Problem-Specific Segmentation, Spelling/Grammatical Error Identification And Recovery, 7) Named Entity Recognition (NER), 8) Word Sense Dis-ambiguation (WSD), 9) Negation And Uncertainty Identification, 10) Relationship Extraction, 11) Temporal Inferences/Relationship Extraction, And 12) Information Extraction (IE) [9].

Over the last few years, deep learning-based NLP attained outstanding results on various NLP tasks. This was achieved via the success of word embeddings and deep learning methods [10]. Word Embedding (WE) is a numerical representation of words, usually as vectors [11]. WE training can be done in a variety of ways, including Word2vec, FastText, and BERT [12]. Word2Vec is the most used form of WEs. The Word2Vec takes text corpus as input and produces word vectors as output. The generation of WE with Word2Vec can be based on two types of models 1) the Continuous Bag Of Words (CBOW) model and 2) the Skip-Gram model [13], [14]. In the CBOW model, a word is predicted based on surrounding words. While in the Skip-Gram model, surrounding words are predicted based on a given word [13]. The architecture of these models is shown in Figure 2.

The main usefulness of WEs is detecting the similarity between words [10]. Measuring similarity between vectors is possible, using term-based similarity measures such as Block Distance, Cosine Similarity, Dice's Coefficient, Euclidean Distance, Jaccard Similarity, Matching Coefficient, and Overlap Coefficient [15].

[16] proposed a resume ranking and recommendation system named Smart Applicant Ranker, which is designed for IT companies to guide them in their recruiting process. This system used Ontology to find and classify the implicit and explicit linkages between the candidate models and the job requirement model. Smart Applicant Ranker architecture is composed of 3 modules: 1) information extraction, 2) candidate search, and 3) candidate ranking algorithms. The information extraction module is responsible for reading the resume information,

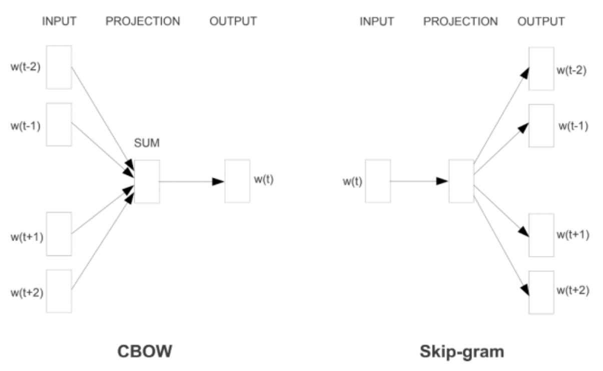


Figure 2: Word2Vec models architecture [13].

constructing an ontology model for a resume, and finally saving the information in the database. Then, the candidate search module provides the resumes ranked by their relative score after calculating the similarity value between the selected resume models and the supplied job requirements. In the last module, the nominated candidates will be evaluated using two fundamental algorithms that will assign rank points based on their 1) educational credentials and 2) skills and work experience.

In [17], the authors proposed a recommendation system that is based on the employer's inquiry to recommend relevant resumes. The system was mainly based on the Vector Space Model (VSM). Where two databases are used to store the terms to be retrieved from the documents (skill DB and candidate DB). The vocabulary is then constructed using the corpus's unique terms. The documents are then represented as vectors using the Term Frequency (TF) approach, and cosine similarity is used to calculate document similarity. Finally, the document that has the highest similarity value is recommended.

[18] proposed a resume classifier application in the IT sector. The application classifies a candidate profile to the best match domain based on the information included in the resume and allocation of a project for the candidate in a particular domain. This application employs the ensemble learning based voting classifier, which consists of five individual classifiers Naïve Bayes, Multinomial Naïve Bayes, Linear SVC (Support Vector Classifier), Bernoulli Naïve Bayes, and Logistic Regression. The architecture is composed of 2 modules: 1) natural language processing pipeline and 2) classification module. The natural language processing pipeline is responsible for removing extraneous information from resumes and providing only the relevant data in the form of tokens. While the classification module analyzes the list of tokens to classify the resume into the appropriate domain.

[19] proposed a resume matching system. The parser system is composed of 4 main phases: 1) text segmentation, 2) named entity recognition, 3) text normalization, and 4) Co-reference merging and conflict resolution. In the first phase, the extracted text is separated into segments of similar information based on attributes such as Name, Phone, and so on. Next, the texts are categorized into named entities. The text normalization process guarantees that specified entities are transmuted to make them consistent and reliable and that abbreviations are enlarged with the help of a reference library. In the last phase, Co-reference resolution, a sort of textual or syntactic-semantic connection in which two or more nominal groups name the same object, is applied to the parsed resume. The outputs of the parser system then passed through a weighting task and matching process based on the firefly algorithm.

[20] proposed a system for resumes classification and matching to a specific job position. As an output of this system, the top ten candidates are selected from a set of applicants. This system first classifies resumes into different categories using Random Forest (RF), Multinomial Naive Bayes (NB), Logistic Regression (LR), and Linear Support Vector Classifier (Linear SVM) models. Then, applies the content-based recommendation using cosine similarity and k-Nearest Neighbors (k-NN) algorithm for ranking resumes.

Article	Basic Methods	Accuracy	Limitations
[16]	Ontology	83%	The proposed solution works only with structured data.
[17]	NLP Vector space model	Not mentioned	The model only considers the skills characteristic and neglects other factors such as education and work experience, besides, accuracy was not reported so we can not decide whether it is usable or not.
[18]	NLP Ensemble learning	91%	The model merely matches a candidate's profile to an appropriate domain, not to a specific employment position.
[19]	NLP Firefly algorithm	94% "only for a part of the whole system (parser)"	The model concentrates only on the job seeker's educational qualifications and skills.
[20]	NLP ML	79%	The main issue faced with this system is that some important data is lost because of text summarization, also it does not support some real-world resumes format such as PDF.
[21]	NLP Vector space model	Not mentioned	We can't say whether it's usable or not because the accuracy hasn't been reported.
[22]	Ontology	Not mentioned	The proposed system focused only on the CyberSecurity field. Also, other factors such as experience, education, and so on are not taken into account. As well accuracy hasn't been reported.

Table 1: Comparison of similar solutions.

[21] proposed an automated resumes screening system, which works in two phases. First, NLP is used to extract all relevant candidate information such as skills, work experience, education, certifications, and so on from the unstructured text in resumes. Then, resumes were ranked according to how well their content matched the job description using the Vector Space Model, where the documents are represented as vectors, and then similarity measurements such as cosine similarity are used to determine which group of resumes is the best fit for the job. Finally, a ranked list of applicants is generated.

In [22], an ontology-based recommender system was presented for analyzing and assessing information while taking into consideration the changing demands of the firm and the talents of the job applicant. It is composed of two main parts including ontologies construction and matching process. Where the construction of ontologies is done through three phases 1) Job requirements are represented as ontologies, 2) the system collects all of the information from job seekers' profiles and creates ontology models for them, and 3) two different ontologies for skills, IT skills, and Cybersecurity skills, are created. The matching process then uses a matching engine to compute matching scores using pre-generated ontologies and a set of matching rules, taking as input a job description and a number of job seeker's profiles to be matched.

A summary of the related work and comparison among several strategies is provided in Table 1. As seen from the table, [16] [17], [19], [21], and [22] used ML in its solution architecture which is different from what was used by I-recruiter. Besides, I-recruiter intersects with [17], [18], [19] and [20] in its support of unstructured resumes. In general, I-Recruiter uses different techniques from other proposed works such as word embeddings and supports continual learning to adapt to data changing. It also enables users to specify the number of ranks which makes it more flexible as well as its supports to real-world resumes format such as PDF.

3 System overview

All the time, companies have been spent a lot of time and effort on traditional recruiting, especially in cases with a large number of received resumes. With the support of technology power, recruitment can be more efficient with fewer resources needed. As an attempt to support and facilitate the recruitment process, an IDSS called I-Recruiter was designed to speed up the screening step of recruitment. I-Recruiter automatically finds the top-ranked candidates based on the degree of semantic similarity between the job description and applicants' resumes. The architecture of the I-Recruiter is shown in Figure 2. The system consists of three main building blocks namely 1) training, 2) matching, 3) and extracting.

In the next sub-sections, we will explain how does each block of the I-Recruiter work.

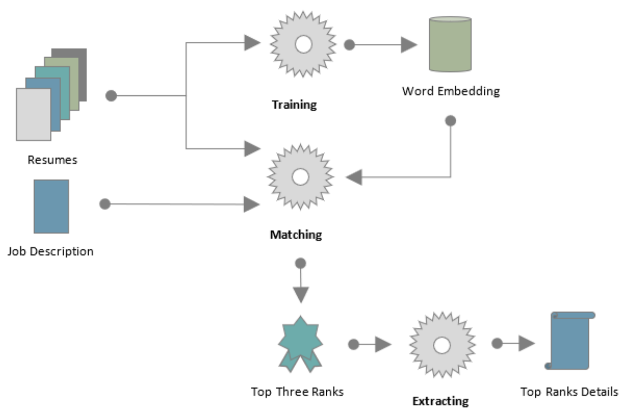


Figure 3: I-Recruiter architecture.

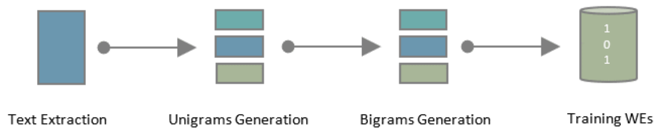
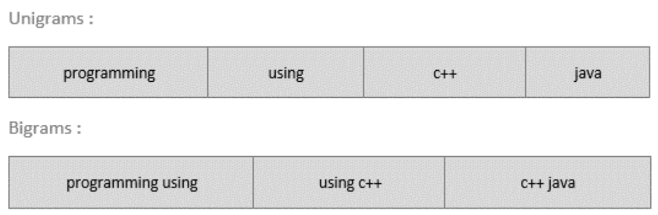


Figure 4: Levels of the training process.



Unigrams :

programming	using	c++	java

Bigrams :

programming using	using c++	c++ java

Figure 5: Unigram and bigrams of the cleaned version of the sentence "Programming using C++ and Java".

3.1 Training block

This block is responsible for training the domain WE from a set of resumes. In this subsystem, the Skip-Gram model is used. According to [13] skip-gram model is more efficient than CBOW as rare words or phrases are well presented. Also, this subsystem is responsible for updating the basic WE with any new inputted resume, this assists the system to continuously develop knowledge. The generated WE models from this block will be used later in the matching block. Both training and updating processes are done on four levels as shown in Figure 4.

- Level 1 — text extraction: resume files are being read and text is extracted from them.
- Level 2 — unigrams generation: a list of one word generated from resumes text in two steps. First, the text is pre-processed with the removal of unnecessary parts of the text, such as links, symbols, and stop words. Also, by dividing it into tokens with NLP tokenization and returning words to their basic form with NLP lemmatization. Second, words for training are extracted from the text using POS tagging. In which unannotated words in natural language are labeled with Parts-Of-Speech labels such as verbs, nouns, adverbs, adjectives, etc. [7]. Figure 5 shows an example of unigrams.
- Level 3 — bigrams generation: a list of word pairs is generated from resumes text in two steps. First, bigrams

are generated and scored with the use of the Natural Language Toolkit (NLTK), which is a suite of libraries for language processing [23]. Second, top-scored bigrams are cleaned to remove any noisy pairs that do not harmonize together or are not related to the job. This cleaning process employs POS tagging and NER techniques for defining unwanted words. NER recognizes the named entities occurring in the text such as persons, organizations, locations, dates, etc. [7]. Cleaning is also based on some frequent words that appear in resumes and must be ignored like name, contact, and so on. Other IT-related words must not be ignored like C++, 3Ds, and so on. Figure 5 shows an example of bigrams.

- Level 4 — training word embeddings: models are trained at the last level from previously generated lists of bigrams and unigrams using the Word2Vec technique.

3.2 Matching block

This block is responsible for matching the text of the job description with the text of each applicant's resume. It depends on pre-trained WEs to create vectors for all in-putted resumes and the job description. Figure 6 shows a sample of the vector representation of words. After that, the average of generated vectors is computed as the vector's average estimation leads to a meaningful representation of longer pieces of text [24]. Lastly, the semantic similarity degree is calculated with the cosine similarity method. Cosine similarity refers to the measure of similarity between two vectors, where vectors represent the compared documents and the cosine degree between these vectors represents similarity degree [25]. The similarity is calculated as shown in the following equation where A and B are vectors and ∅ represent the angle between them:

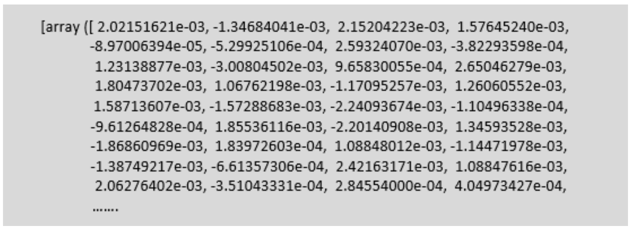
$$Similarity(\vec{A},\vec{B}) = \cos\emptyset = \frac{A\,.\,B}{\|A\|\,\|B\|}$$

$$= \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \times \sqrt{\sum_{i=1}^{n} {B_i}^2}} \qquad (1)\ [26]$$

The top ranks of the resumes are for the top highest similarity degrees. These top-ranked resumes will be transferred to the next block for data extraction.

3.3 Extracting block

This block is responsible for extracting the top-ranked candidates' basic information that including candidate name, phone number, and email. NLP gazetteer and pattern matching approaches are used in this stage. For extracting emails and phone numbers, the pattern matching approach is used. Where extraction patterns are defined using Regular Expressions (RE). The result is then matched with a given input text and the matched text will be extracted [7]. I-Recruiter is designed to detect any text that matches the email pattern for finding candidates' email addresses. Concerning getting candidates' phone numbers, the text detected whatever matches the Palestine dialing code. Regard extracting names, the gazetteer approach was used. Gazetteer or gazette is a pre-defined list of all possible values of a named entity [7]. Here the



```
[array ([ 2.02151621e-03, -1.34684041e-03,  2.15204223e-03,  1.57645240e-03,
         -8.97006394e-05, -5.29925106e-04,  2.59324070e-03, -3.82293598e-04,
          1.23138877e-03, -3.00804502e-03,  9.65830055e-04,  2.65046279e-03,
          1.80473702e-03,  1.06762198e-03, -1.17095257e-03,  1.26060552e-03,
          1.58713607e-03, -1.57288683e-03, -2.24093674e-03, -1.10496338e-04,
         -9.61264828e-04,  1.85536116e-03, -2.20140908e-03,  1.34593528e-03,
         -1.86860969e-03,  1.83972603e-04,  1.08848012e-03, -1.14471978e-03,
         -1.38749217e-03, -6.61357306e-04,  2.42163171e-03,  1.08847616e-03,
          2.06276402e-03, -3.51043331e-04,  2.84554000e-04,  4.04973427e-04,
         .......
```

Figure 6: Sample of words vector representation.

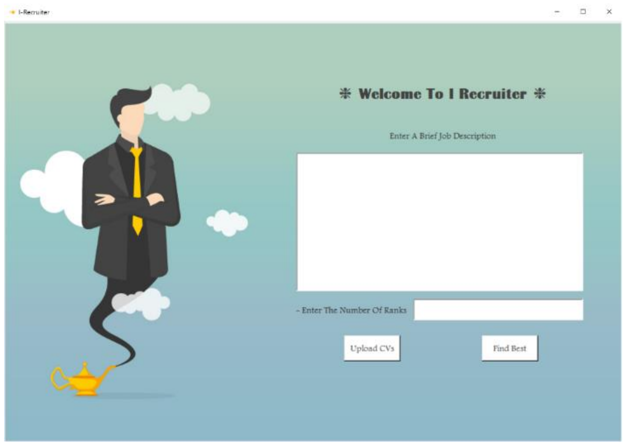

I-Recruiter
✻ Welcome To I Recruiter ✻
Enter A Brief Job Description
- Enter The Number Of Ranks
Upload CVs
Find Best

Figure 7: I-Recruiter interface.

system detects entities named ' PERSON ' in the process of finding candidates' names.

4 Implementation and discussion

In this section, we will explain the implementation of the I-Recruiter prototype, data collection, and I-Recruiter testing as well as a discussion of the results.

4.1 Prototype development

A prototype of I-Recruiter has been developed as a desktop application with the use of Python programming language version 3.8 escorted by python artificial intelligence libraries such as Gensim, spaCy, and Natural Language Toolkit (NLTK). Figure 7 presents the I-Recruiter interface.

4.2 Data collection

A dataset of 101 unstructured resumes in the domain of information technology (IT) were collected from different resources. All resumes within this dataset are in PDF format.

4.3 Test and results

To test I-Recruiter, we designed two experiments, the first aimed to test the performance (elapsed time) where the second aimed to test the accuracy of I-Recruiter results.

4.3.1 Performance of I-Recruiter.

The time required to train models using 101 resume files was calculated. The calculated time has been carried out in one trial. Also, the time required to match 101 resumes to a job position and the time required to extract information from the top three ranked candidates' resumes were calculated. Four trials with different job descriptions for each trial were used for calculating the required time of both matching and extracting. Table 2 shows the results of these tests.

As shown in Table 1, I-Recruiter took approximately 327.23 seconds for train 101 resumes, an average of 36.34 seconds for matching, and an average of 0.73 seconds for extracting.

There were no reports about total execution time for other related work, however, an average of 37.07 seconds execution time is acceptable and effectively applicable in comparison to the lengthy manual method. According to this, I-Recruiter can work efficiently and will do save time for the human resources department.

4.3.2 Accuracy of I-Recruiter.

Following [16] and [18], accuracy was measured by comparing system results to individual ones. In our experiment, I-Recruiter was configured to find out the best 3 candidates among 10 applications for 4 job positions. The same was performed manually by an IT specialist and both results were compared together in Table 3.

I-Recruiter has proved that it can be reliable with an overall accuracy of 100% in candidate selection and 92% in rank order. These degrees of accuracy were calculated by finding the average of the results of all testing trials. The 8% error percentage appears because of the existence of some noisy bigrams that were not removed in the cleaning process. To overcome this issue of wrong ordering, users can specify more than needed ranks. Then, reorder them manually.

In comparison with other proposed solutions and models, I-Recruiter showed excellent performance in terms of execution as well as accuracy. The average accuracy of I-Recruiter is 96% while for [16], [18], and [20] it is 83%, 91%, and 79% respectively. Hence, it is obvious that I-Recruiter outperforms the other models in terms of accuracy and hence is reliable to be used for the purpose it was proposed.

5 Conclusion and future work

I-Recruiter is an intelligent decision support system for screening a set of applicants' resumes for a job position and find out the most appropriate candidates. This system is composed of three main building blocks training, matching, and extracting. Domain-trained word embeddings are generated from the training block. While the matching block finds the top candidates based on the semantic similarity between resumes and the job description. Basic information on top-ranked applicants was extracted in the last block. I-Recruiter showed very good results with a high degree of accuracy and a short time of work. We'll work on increasing the system's performance and accuracy in the future, as well as introducing more capabilities like personality analysis from personal pictures.

Testing Device	Processor: intel core i7-6700HQ RAM: 16 GB System Type: 64-Bit			
System Block	***Training***		***Matching***	***Extracting***
Time / Seconds	327.23	***Trial 1***	37.55	0.69
		Trial 2	35.74	0.86
		Trial 3	36.11	0.43
		Trial 4	35.95	0.94

Table 2: Performance of I-Recruiter.

Job Description	**Top Ranks**	**I-Recruiter Result**	**Human Result**	**Selections Correctness**	**Order Correctness**
JD 1	***Rank 1***	CV1	CV1	100%	67%
	Rank 2	CV8	CV5		
	Rank 3	CV5	CV8		
JD 2	***Rank 1***	CV2	CV2	100%	100%
	Rank 2	CV8	CV8		
	Rank 3	CV3	CV3		
JD 3	***Rank 1***	CV8	CV8	100%	100%
	Rank 2	CV6	CV6		
	Rank 3	CV5	CV5		
JD 4	***Rank 1***	No one matches the job	No one matches the job	100%	100%
	Rank 2				
	Rank 3				

Table 3: Accuracy of I-Recruiter.

References

[1] M. B. R. Devi and Dr. Mrs. P. V. Banu, “Introduction to Recruitment,” SSRG Int. J. Econ. Manag. Stud. SSRG-IJEMS, vol. 1, no. 2, p. 4, 2014. [Online]. Available: https://www.internationaljournalssrg.org/IJEMS/2014/Volume1-Issue2/IJEMS-V1I2P102.pdf

[2] A. Singh, C. Rose, K. Visweswariah, V. Chenthamarakshan, and N. Kambhatla, “PROSPECT: a system for screening candidates for recruitment,” in Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM ’10, Toronto, ON, Canada, 2010, p. 659. doi: https://doi.org/10.1145/1871437.1871523.

[3] M. L. Gusdorf, “Recruitment and Selection: Hiring the Right Person,” p. 19, 2008. [Online]. Available: https://www.shrm.org/certification/educators/Documents/Recruitment%20and%20Selection%20IM.pdf

[4] G. Phillips-Wren, “Intelligent Decision Support Systems,” in Multicriteria Decision Aid and Artificial Intelligence, M. Doumpos and E. Grigoroudis, Eds. Chichester, UK: John Wiley & Sons, Ltd, 2013, pp. 25–44. doi: https://doi.org/10.1002/9781118522516.ch2.

[5] W. Ertel, Introduction to Artificial Intelligence. Cham: Springer International Publishing, 2017. doi: https://doi.org/10.1007/978-3-319-58487-4.

[6] J. Zhang and P. S. Yu, “Machine Learning Overview,” in Broad Learning Through Fusions, Cham: Springer International Publishing, 2019, pp. 19–75. doi: https://doi.org/10.1007/978-3-030-12528-8_2.

[7] S. Singh, “Natural Language Processing for Information Extraction,” ArXiv180702383 Cs, Jul. 2018, Accessed: Sep. 11, 2020. [Online]. Available: http://arxiv.org/abs/1807.02383

[8] J. Hurwitz, Machine Learning For Dummies®, IBM Limited Edition. 2018. [Online]. Available: https://www.ibm.com/downloads/cas/GB8ZMQZ3

[9] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: an introduction,” J. Am. Med. Inform. Assoc., vol. 18, no. 5, pp. 544–551, Sep. 2011, doi: https://doi.org/10.1136/amiajnl-2011-000464.

[10] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent Trends in Deep Learning Based Natural Language Processing,” ArXiv170802709 Cs, Nov. 2018. [Online]. Available: https://arxiv.org/pdf/1708.02709.pdf

[11] A. Mandelbaum and A. Shalev, “Word Embeddings and Their Use In Sentence Classification Tasks,” ArXiv161008229 Cs, Oct. 2016, Accessed: Sep. 17, 2020. [Online]. Available: http://arxiv.org/abs/1610.08229

[12] C. Wang, P. Nulty, and D. Lillis, “A Comparative Study on Word Embeddings in Deep Learning for Text Classification,” in Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, Seoul Republic of Korea, Dec. 2020, pp. 37–46. doi: https://doi.org/10.1145/3443279.3443304.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” ArXiv13013781 Cs, Sep. 2013, Accessed: Sep. 11, 2020. [Online]. Available: http://arxiv.org/abs/1301.3781

[14] P. Sitikhu, K. Pahi, P. Thapa, and S. Shakya, “A Comparison of Semantic Similarity Methods for Maximum Human Interpretability,” ArXiv191009129 Cs, Oct. 2019, Accessed: Sep. 11, 2020. [Online]. Available: http://arxiv.org/abs/1910.09129

[15] V. M.K and K. K, “A Survey on Similarity Measures in Text Mining,” Mach. Learn. Appl. Int. J., vol. 3, no. 1, pp. 19–28, Mar. 2016, doi: https://doi.org/10.5121/mlaij.2016.3103.

[16] A. Mohamed, W. Bagawathinathan, U. Iqbal, S. Shamrath, and A. Jayakody, “Smart Talents Recruiter - Resume Ranking and Recommendation System,” in 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), Colombo, Sri Lanka, Dec. 2018, pp. 1–5. doi: https://doi.org/10.1109/ICIAFS.2018.8913392.

[17] S. N, S. V, A. S, and S. P, “Validating effective resume based on employer’s interest with recommendation system,” Int. J. Pure Appl. Math., vol. 119, 2018, [Online]. Available: https://hal.archives-ouvertes.fr/hal-01826687/document

[18] S. T. Gopalakrishna and V. Varadharajan, “Automated Tool for Resume Classification Using Sementic Analysis,” Int. J. Artif. Intell. Appl., vol. 10, no. 01, pp. 11–23, Jan. 2019, doi: https://doi.org/10.5121/ijaia.2019.10102.

[19] G. Deepak, V. Teja, and A. Santhanavijayan, “A novel firefly driven scheme for resume parsing and matching based on entity linking paradigm,” J. Discrete Math. Sci. Cryptogr., vol. 23, no. 1, pp. 157–165, Jan. 2020, doi: https://doi.org/10.1080/09720529.2020.1721879.

[20] P. K. Roy, S. S. Chowdhary, and R. Bhatia, “A Machine Learning approach for automation of Resume Recommendation system,” Procedia Comput. Sci., vol. 167, pp. 2318–2327, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.284.

[21] C. Daryani, G. S. Chhabra, H. Patel, I. K. Chhabra, and R. Patel, “An Automated Resume Screening System Using Natural Language Processing And Similarity,” In Ethics And Information Technology, Jan. 2020, pp. 99–103. doi: https://doi.org/10.26480/etit.02.2020.99.103.

[22] M. Maroun and A. Ivanova, “Ontology-based approach for cybersecurity recruitment,” Tomsk, Russia, 2021, p. 070014. doi: https://doi.org/10.1063/5.0042320.

[23] “Natural Language Toolkit,” NLTK 3.5 documentation, 2020. https://www.nltk.org/

[24] T. Kenter and M. de Rijke, “Short Text Similarity with Word Embeddings,” in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM ’15, Melbourne, Australia, 2015, pp. 1411–1420. doi: https://doi.org/10.1145/2806416.2806475.

[25] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, “Cosine similarity to determine similarity measure: Study case in online essay assessment,” in 2016 4th International Conference on Cyber and IT Service Management, Bandung, Indonesia, Apr. 2016, pp. 1–6. doi: https://doi.org/10.1109/CITSM.2016.7577578.

[26] F. Rahutomo, T. Kitasuka, and M. Aritsugi, “Semantic Cosine Similarity,” in The 7th International Student Conference on Advanced Science and Technology ICAST, 2012, p. 3. [Online]. Available: https://www.researchgate.net/profile/Faisal-Rahutomo/publication/262525676_Semantic_Cosine_Similarity/links/0a85e537ee3b675c1e000000/Semantic-Cosine-Similarity.pdf