IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

AI Resume Analyzer Using RoBERTa for Semantic Candidate-Job Matching

Mr.Vijay Dilip Wagh[1], Prof. Manoj V. Nikum[2], Prof. Kaustubh C. Bhave[3]

Research scholar MCA, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[1]

HOD MCA & Assistant Professor, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[2]

Assistant Professor, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India [3]

Abstract: Screening resumes by hand is slow, and the judgement applied differs from one recruiter to another. Applicant Tracking Systems that depend on keyword overlap reduce the workload but overlook candidates whose resumes describe the same skill in different words. This paper presents an AI Resume Analyzer that uses RoBERTa, a robustly optimised BERT-style transformer, to read resumes in context and match them with job requirements. Natural language processing techniques extract skills, qualifications, experience, education, certifications and projects from uploaded resumes, and a fine-tuned RoBERTa model estimates how closely each resume fits a given job. The system returns a relevance score, a suitability category and lists of matched and missing skills, which are shown to the recruiter on a web dashboard. The architecture separates a Java Spring Boot service, which manages authentication, storage and the interface, from a Python engine built on Hugging Face Transformers, PyTorch and spaCy. An evaluation plan against a TF-IDF keyword baseline, using accuracy, precision, recall, F1-score, Precision@k and NDCG, is also described. The proposed system aims to make first-round screening faster, more consistent and easier to explain.

Keywords: AI Resume Analyzer, RoBERTa, Natural Language Processing, Transformer, Candidate-Job Matching, Applicant Tracking System, Skill Extraction, Explainable Screening

I. INTRODUCTION

Every job opening that is advertised online tends to bring in far more resumes than a hiring team can read carefully. A resume is also not a form. Candidates write in different layouts, use different section names, mix tables with free text and describe the same experience in very different words. Most of the early screening is therefore done by a recruiter who skims each document, and the result depends on how tired or hurried that person is on the day. Two recruiters reading the same resume against the same vacancy can easily reach different conclusions.

To reduce this load, many organisations use an Applicant Tracking System (ATS). In practice most of these systems filter by keywords, so a resume that does not contain the exact words of the job description is ranked low or dropped. This works poorly for technical roles, where the vocabulary changes quickly and one skill has many names. A candidate who writes “built and deployed a classification model behind a REST service” may be rejected for a role asking for “machine learning model deployment” simply because the phrases do not overlap. The opposite also happens, where a resume stuffed with the right keywords passes without any real evidence of the skill.

Transformer language models offer a way around this problem because they represent a word according to the sentence it appears in. RoBERTa [1] is one such model. It keeps the architecture of BERT [2] but was pre-trained for longer, on more data and with dynamic masking, and it drops the next-sentence prediction objective. On several language understanding benchmarks it performed better than the original BERT, which makes it a reasonable base for a task where the meaning of a sentence, not the presence of a word, decides the outcome.

This paper describes an AI Resume Analyzer built around a fine-tuned RoBERTa model. The work addresses three problems seen in current screening practice: manual screening is slow and inconsistent, keyword-based filtering ignores context, and recruiters are rarely told why a candidate was or was not matched. The objectives are (i) to build a web-based system for contextual candidate-job matching, (ii) to extract structured information from resumes and compute a job-relevance score with a fine-tuned RoBERTa model, and (iii) to compare the transformer-based approach with keyword filtering using classification and ranking metrics.

The remaining sections are organised as follows. Section II reviews related work on resume parsing, keyword retrieval and neural person-job fit. Section III explains the architecture, the extraction pipeline, the matching model and the evaluation plan. Section IV discusses the evaluation set-up, an illustrative example and the limitations of the approach. Section V concludes the paper and lists directions for further work.

 This work is licensed under a Creative Commons Attribution 4.0 International License

II. LITERATURE REVIEW

A. Keyword-Based Screening and Term Weighting

Classical resume search follows the information retrieval tradition, where documents and queries are turned into sparse term vectors and compared with a similarity measure. TF-IDF weighting [3] gives more importance to terms that are frequent in a document but rare in the collection, and cosine similarity between the two vectors is then used for ranking [4]. The method is fast, needs no training data and is easy to explain, which is why it still sits behind many ATS products. Its main weakness is that it matches strings. Synonyms, abbreviations and paraphrases lie outside its vocabulary, and the order and context of words are lost.

B. Resume Parsing and Information Extraction

Before a resume can be compared with a job, its content has to be located and labelled. Early parsers relied on handwritten rules and gazetteers of skills, institutions and job titles. Statistical named entity recognition improved coverage, and libraries such as spaCy [5] now provide trained pipelines together with rule-based matchers that can be combined in one workflow. Bhatia et al. [6] used BERT-based models for resume parsing and for finding candidates for a job description, and reported that contextual representations helped both the extraction and the matching steps. Their work is close to the direction followed here, but it treats parsing and matching mainly as a research pipeline and does not describe a recruiter-facing application with authentication and stored results.

C. Neural Person-Job Fit

Person-job fit asks whether a candidate suits a position. Zhu et al. [7] learned joint representations of job postings and resumes with convolutional networks and used them to predict whether a candidate fits a vacancy. Qin et al. [8] argued that a job requirement is really a set of abilities and proposed an ability-aware network with attention over requirements and over the candidate's experiences. Both studies show that learned representations capture relations that term overlap cannot. They were trained on data from specific recruitment platforms, so the models are not easy to reuse in a small organisation that has its own roles and its own vocabulary.

D. Transformers and RoBERTa

The Transformer [9] replaced recurrence with self-attention, which lets every token in a sequence attend to every other token. BERT [2] used this encoder for bidirectional pre-training and showed that one pre-trained model could be fine-tuned for many tasks with a small output layer. RoBERTa [1] revisited the pre-training recipe and obtained better results with the same architecture. It tokenises text into byte-level byte-pair-encoding units [10], so rare technical terms are split into known pieces instead of being treated as unknown. Sentence-BERT [11] showed how transformer encoders can be used to compare the meaning of two texts efficiently, an idea that is useful for the skill matching step in this work. The Hugging Face Transformers library [12] and PyTorch [13] make fine-tuning of such models practical on modest hardware.

E. Fairness and Explainability in Automated Hiring

Automated screening has also drawn criticism. A widely reported case was an experimental recruiting tool that learned to downgrade resumes associated with women because it had been trained on past hiring decisions [14]. Surveys of bias in machine learning [15] explain how such behaviour can enter through the data, the labels or the way a model is used. For a screening tool, two practical consequences follow. The system should not use attributes such as name, gender or photograph as inputs, and it should show the evidence behind a score so that a person can check it.

Taken together, the literature shows that keyword matching is simple but shallow, that transformer-based models handle context better, and that few published systems combine extraction, matching, explanation and a usable interface for recruiters. The proposed analyzer is aimed at this gap.

III. METHODOLOGY

A. System Overview

The system follows a two-layer design, shown in Fig. 1. The first layer is an enterprise backend written in Java with Spring Boot. It handles resume uploads, recruiter authentication with JSON Web Tokens (JWT) through Spring Security, job requirement configuration, candidate profiles and the storage of matching results in PostgreSQL. The second layer is an NLP and transformer engine written in Python. It extracts information from resume text, loads the fine-tuned RoBERTa model and returns scores and skill lists to the backend over a REST interface. A React.js front end styled with Tailwind CSS lets the recruiter upload resumes, define a job and inspect the results. Keeping the model in its own service allows it to be retrained or replaced without touching the web application.

 This work is licensed under a Creative Commons Attribution 4.0 International License

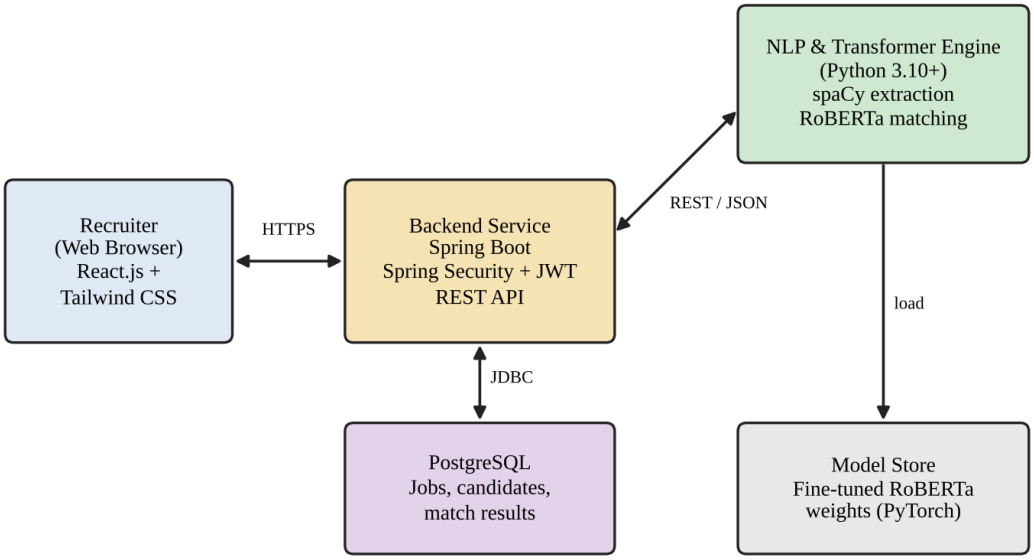


Fig. 1 Architecture of the proposed AI Resume Analyzer

B. Data Preparation and Text Preprocessing

The input to the system is a resume in PDF or DOCX format and a job description or a list of requirements entered by the recruiter. Text is first extracted from the file and cleaned. Cleaning removes page headers and footers, repeated whitespace and non-text symbols, and joins words broken at line ends. Resumes in which no text can be extracted, such as scanned images, are reported to the recruiter as unreadable instead of being scored on empty input.

For fine-tuning, the training set is made of resume and job description pairs, each labelled by a reviewer as suitable, partially suitable or not suitable. The pairs are divided into training, validation and test sets, and a resume that appears in one subset is kept out of the others so that the test results are not inflated by repeated content. Personal identifiers such as names, phone numbers, e-mail addresses and photographs are removed from the text that is passed to the model.

C. Resume Information Extraction

Extraction is done in three steps. In the first step the cleaned text is split into sections (education, skills, experience, projects and certifications) using heading patterns and layout cues. In the second step spaCy [5] pipelines and rule-based matchers pick out entities from each section: skills, degrees, institutions, job titles, organisations, durations and certification names. In the third step the extracted skills are normalised against a skill dictionary, so that variants such as "ML", "machine-learning" and "Machine Learning" point to one canonical entry. The output is a structured candidate profile that is saved in the database and also passed on to the matching stage. Fig. 2 shows the full flow from upload to result.

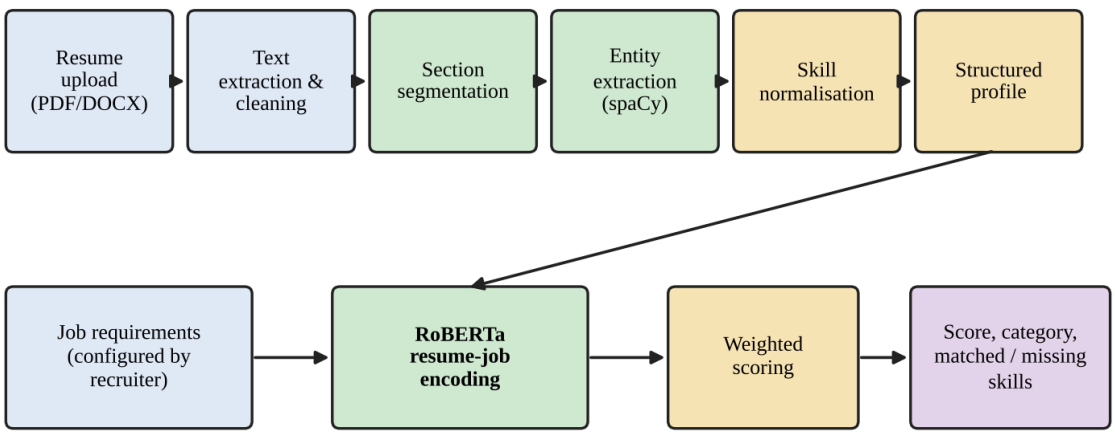


Fig. 2 Processing pipeline from resume upload to candidate-job relevance result

D. RoBERTa-Based Candidate-Job Matching

The matching model is a RoBERTa-base encoder (12 layers, 768 hidden units, 12 attention heads) with a classification layer on top of the first token. A resume segment and the corresponding job requirement are joined into one input sequence separated by RoBERTa's special tokens, so that self-attention can compare the two texts directly. Because a full resume often exceeds the 512-token limit of the model, the resume is not fed in as one block. Instead, each section (skills, experience, education, projects) is paired with the job text separately and the outputs are combined at the scoring stage.

The model is fine-tuned with cross-entropy loss over the three suitability classes. The planned set-up follows the usual recommendations for RoBERTa-base [1]: the AdamW optimiser [16] with a learning rate of about 2e-5, linear warm-up and decay, a batch size of 16 and three to four epochs, with the checkpoint chosen by validation F1-score. For a pair (resume segment, job text) the model returns class probabilities, and the semantic score is computed as

$$S_{sem} = P(\text{suitable}) + 0.5 \times P(\text{partial}) \tag{1}$$

E. Scoring, Categorisation and Skill Gap Analysis

The overall relevance score combines the semantic score with three structured components taken from the extracted profile:

$$S = w_1 \cdot S_{sem} + w_2 \cdot S_{skill} + w_3 \cdot S_{exp} + w_4 \cdot S_{edu}\ , \quad \Sigma\, w_i = 1 \tag{2}$$

Here S_{skill} is the share of required skills that are covered by the candidate, S_{exp} compares the candidate's years of relevant experience with the requirement, and S_{edu} checks the qualification level. A required skill r is counted as covered when its best match among the candidate's skills reaches a similarity threshold τ, where the similarity between two skill embeddings a and b is the cosine

$$\text{sim}(a, b) = (a \cdot b) / (|a| \cdot |b|) \tag{3}$$

This is what allows "REST API development" on a resume to count towards a requirement written as "web services", even though no word is shared. The weights w_1 to w_4 and the threshold τ are configurable by the recruiter for each job, with the default favouring the semantic and skill components. Candidates are then placed in one of three categories: *Highly Suitable* when S is at least 0.75, *Moderately Suitable* between 0.50 and 0.75, and *Low Match* below 0.50. These cut-off values are defaults and can be changed. Along with the score, the system lists the skills that were matched and the required skills that were not found, so that the recruiter can see the reason behind the category.

F. Backend, Database and Dashboard

Recruiters register and log in through Spring Security, and every API call carries a JWT. The backend stores recruiters, jobs, candidates, uploaded resumes, extracted entities and match results in PostgreSQL, with a match result linked to exactly one candidate and one job. When a resume is uploaded, the backend sends the text to the Python service, receives the profile and scores, and saves them. The React dashboard shows the extracted profile of each candidate, a ranked list for a job, the category and score, the matched and missing skills, and a chart view that lets the recruiter compare candidates side by side. Access to resumes is limited to the recruiter who uploaded them.

G. Baseline and Evaluation Plan

The keyword baseline represents a conventional ATS. Resumes and job descriptions are converted to TF-IDF vectors [3] and ranked by cosine similarity, and a resume is labelled suitable when its score passes a threshold tuned on the validation set. Both approaches are run on the same held-out test pairs. Classification quality is measured with accuracy, precision, recall and F1-score, where

$$F1 = 2PR / (P + R) \tag{4}$$

and P and R are precision and recall. Because screening is about ranking candidates for a job, ranking metrics are also used: Precision@k, Mean Reciprocal Rank and Normalised Discounted Cumulative Gain [17],

$$\text{NDCG@k} = \text{DCG@k} / \text{IDCG@k}\ , \quad \text{DCG@k} = \Sigma\, rel_i / \log_2(i + 1),\ i = 1 \ldots k \tag{5}$$

where rel_i is the relevance label of the candidate at rank i. Extraction quality is measured separately as entity-level precision, recall and F1 against manually annotated resumes, and processing time per resume is recorded. Differences

between the two methods on the same test pairs are checked with a paired significance test, for example McNemar's test for the classification outcomes.

IV. RESULT AND DISCUSSION

A. Evaluation Set-up

The evaluation compares three configurations on the same test pairs: the TF-IDF keyword baseline, the RoBERTa model used alone, and the complete system in which the semantic score is combined with the skill, experience and education components. This separation shows how much of any improvement comes from the transformer and how much comes from the structured extraction around it. Table I summarises the measures and the question each one answers.

TABLE I
EVALUATION MEASURES USED IN THE STUDY

Measure	Level	What it shows
Accuracy, Precision, Recall, F1	Pair classification	How often the suitability class is correct, and how well suitable resumes are found without too many false alarms
Precision@k, MRR	Ranking	Whether the suitable candidates appear near the top of the list for a job
NDCG@k	Ranking	Quality of the whole ordering, giving more weight to the top positions
Entity-level F1	Extraction	Correctness of extracted skills, degrees, titles and durations
Time per resume	System	Whether the analysis is fast enough for routine recruiter use

B. Illustrative Example

The difference between the two approaches is easiest to see on a single case. Suppose a vacancy for a junior data engineer lists the requirements in the first column of Table II, and a resume describes the candidate's work in the second column. The example is constructed to explain the mechanism and is not a measured result.

TABLE II
ILLUSTRATIVE COMPARISON OF KEYWORD MATCHING AND SEMANTIC MATCHING

Job requirement	Resume statement	Keyword	Semantic
Machine learning model deployment	Built a classification model and served it through a REST API	Not matched	Matched
Relational database design	Designed PostgreSQL tables for an inventory application	Not matched	Matched
Spring Boot	Developed backend services using Spring Boot	Matched	Matched
Cloud platform experience	Academic projects with no cloud usage mentioned	Not matched	Missing skill

A keyword filter would credit only the third row, and the candidate would be ranked as a weak match. The semantic approach recognises that the first two statements satisfy the requirements even though the wording differs, and it correctly reports the cloud requirement as missing instead of inventing a match. The recruiter sees a moderate-to-high score together with one named gap, which is more useful than a single low number with no explanation.

C. Expected Behaviour Compared with Keyword Filtering

The measured values for the metrics in Table I depend on the annotated test split and will be reported once the experiments on it are complete. Based on the design of the two methods and on the behaviour reported for transformer models in related work [6], [7], the expected differences are summarised in Table III.

TABLE III
EXPECTED DIFFERENCES BETWEEN THE TWO APPROACHES

Aspect	TF-IDF keyword filtering	Proposed RoBERTa-based system
Handling of synonyms and paraphrases	Not handled; exact terms are required	Handled through contextual representations
Effect of keyword stuffing	Raises the score without evidence	Reduced, because context is considered
Explanation for the recruiter	Matched terms only	Score, category, matched and missing skills
Training data needed	None	Labelled resume-job pairs
Computation per resume	Very low	Higher; needs model inference
Consistency between runs	High	High, since the model is fixed after training

 This work is licensed under a Creative Commons Attribution 4.0 International License

Two points deserve attention. First, the transformer approach is more expensive, so the processing time should be checked against the volume of resumes a small or mid-sized recruiter actually handles. Second, the gain over keyword filtering is expected to be largest for resumes that describe skills in their own words, and smaller for resumes that already follow the wording of the job description.

D. Limitations and Ethical Considerations

Several limitations should be stated plainly. The 512-token input limit forces the section-wise design, and information that depends on several sections together may be weakened by it. Scanned or image-only resumes cannot be analysed without optical character recognition, which is not part of the present scope. The quality of the model depends on the labelled pairs, and if those labels reflect the personal preferences of the reviewers, the model will learn the same preferences [14], [15]. Skill dictionaries also need regular updating, because new tools appear all the time.

For these reasons the system is meant to assist the recruiter and not to replace the decision. Personal identifiers are removed before text reaches the model, scores come with visible evidence, and the final shortlisting remains a human step. Stored resumes contain personal data and should be kept only as long as needed for the recruitment process, with access restricted to authorised users.

V. CONCLUSION

This paper presented an AI Resume Analyzer that uses a fine-tuned RoBERTa model to match resumes with job requirements by meaning and not by shared keywords. The system extracts skills, qualifications, experience, education, certifications and projects from resumes, produces a relevance score and a suitability category, and lists matched and missing skills in a web dashboard. A hybrid architecture was chosen, with a Java Spring Boot backend for authentication, storage and the interface and a Python engine for NLP and transformer inference, so that each part can be maintained and improved independently.

The study also set out an evaluation plan that compares the proposed approach with TF-IDF keyword filtering using classification and ranking metrics, so that any improvement in accuracy and consistency is shown by evidence. In use, the system is expected to reduce the manual effort of first-round screening, give recruiters reasons they can check, and support more informed decisions at the early stage of hiring.

Several extensions are possible. Optical character recognition would allow scanned resumes to be processed. A larger and more varied set of labelled pairs, including several domains and non-technical roles, would test how well the model generalises. Bias audits across groups of candidates, multilingual resume support, and the use of explanation methods that point to the exact sentences driving a score would make the tool more dependable in real recruitment.

ACKNOWLEDGMENT

The authors thank the Department of MCA and the institute for the facilities and support provided during this project, and the project guide for the direction and feedback given throughout the work.

REFERENCES

[1]. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", *arXiv preprint arXiv:1907.11692*, 2019.

[2]. J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proc. NAACL-HLT*, pp. 4171-4186, 2019.

[3]. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval", *Information Processing & Management*, vol. 24, no. 5, pp. 513-523, 1988.

[4]. C. D. Manning, P. Raghavan and H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[5]. M. Honnibal, I. Montani, S. Van Landeghem and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python", Zenodo, 2020. doi: 10.5281/zenodo.1212303.

[6]. V. Bhatia, P. Rawat, A. Kumar and R. R. Shah, "End-to-End Resume Parsing and Finding Candidates for a Job Description Using BERT", *arXiv preprint arXiv:1910.03089*, 2019.

[7]. C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding and P. Li, "Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning", *ACM Transactions on Management Information Systems*, vol. 9, no. 3, pp. 1-17, 2018.

 This work is licensed under a Creative Commons Attribution 4.0 International License

[8]. C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen and H. Xiong, "Enhancing Person-Job Fit for Talent Recruitment: An Ability-aware Neural Network Approach", *Proc. 41st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 25-34, 2018.

[9]. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need", *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998-6008, 2017.

[10]. R. Sennrich, B. Haddow and A. Birch, "Neural Machine Translation of Rare Words with Subword Units", *Proc. 54th Annual Meeting of the ACL*, pp. 1715-1725, 2016.

[11]. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks", *Proc. EMNLP-IJCNLP*, pp. 3982-3992, 2019.

[12]. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing", *Proc. EMNLP 2020: System Demonstrations*, pp. 38-45, 2020.

[13]. A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024-8035, 2019.

[14]. J. Dastin, "Amazon scraps secret AI recruiting tool that showed bias against women", *Reuters*, Oct. 2018.

[15]. N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman and A. Galstyan, "A Survey on Bias and Fairness in Machine Learning", *ACM Computing Surveys*, vol. 54, no. 6, pp. 1-35, 2021.

[16]. I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization", *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019.

[17]. K. Jarvelin and J. Kekalainen, "Cumulated Gain-Based Evaluation of IR Techniques", *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422-446, 2002.

 This work is licensed under a Creative Commons Attribution 4.0 International License