Journal of Theoretical and Applied Information Technology
31st July 2025. Vol.103. No.14
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AI-POWERED JOB APPLICATION MANAGEMENT FOR APPLICANTS

ZI QING CHEW[1], NOR FAZLIDA MOHD SANI[2]

[1,2]Department of Computer Science, Faculty of Computer Science and Information Technology,

Universiti Putra Malaysia, Serdang 43400, Selangor Malaysia

E-mail: [1]chewziqing@gmail.com, [2]fazlida@upm.edu.my

ABSTRACT

During the job search process, a resume serves as a job applicant's first impression and will determine whether they will progress in the hiring process. One of the major challenges job seekers faced is ensuring that their resumes align with job requirements, as qualification mismatches can lead to missed job opportunities. The advancement of Applicant Tracking Systems (ATS) presents an additional challenge, as resumes lacking relevant keywords or proper formatting may be automatically rejected by ATS during the automated resume screening phase. To address these challenges, an intelligent job application management system has been developed, contributing to IT research through novel integration of transformer-based NER models with interpretable job-resume matching metrics for analyzing and improving resume contents. The methodology involved a semi-automated annotation process to prepare the annotated resume dataset, followed by training Named Entity Recognition (NER) models using the spaCy and Flair libraries. These trained models were evaluated and compared based on precision, recall, and F1-score metrics. The job-resume matching score was calculated by comparing TF-IDF vectorization of NER-extracted skills from resumes and job descriptions using cosine similarity, followed by normalization with the Sigmoid function. Experimental results showed that the spaCy model achieved an F1-score of 85.71%, outperforming the Flair NER model, which achieved an F1-score of 79.92%. This research advances IT applications in human resource technology by assisting job applicants in enhancing and tailoring their resumes to better match desired job roles, increasing their chances of passing ATS resume scans and being shortlisted for job interviews.

Keywords: *Resume; Job-Resume Matching; NER; Cosine Similarity; Natural Language Processing.*

1. INTRODUCTION

The job market has become increasingly competitive, posing significant challenges for both fresh graduates and experienced professionals. Fresh graduates often lack the necessary skill to prepare effective resumes that leave a strong impression on potential employers. Similarly, experienced professionals face the risk of job loss due to market competition, layoffs, or career transitions, necessitating continuous updates to their skills and professional profiles. In this context, the importance of a well-tailored resume cannot be overstated, as it significantly enhances the likelihood of being shortlisted for desired roles.

Sustainable Development Goal 8 (SDG 8) emphasizes "decent work and economic growth," aiming to promote sustained, inclusive, and sustainable economic growth while ensuring full and productive employment for all. Despite global efforts, challenges persist in achieving these objectives, particularly in addressing unemployment. According to the International Labour Organization (ILO), unemployment is defined as a situation where individuals are without work, currently available, and actively seeking employment. In Malaysia, the unemployment rate stood at 3.3% as of December 2023, with the youth unemployment rate (ages 15–30) reaching 6.4%, affecting approximately 432,100 youths [1].

This study focuses on the online job application process, where individuals actively search for and apply to available positions on online job platforms. The typical job application process involves identifying job openings, preparing tailored resumes and cover letters, and submitting applications through designated platforms. If the applicant's qualifications align with the job requirements, they may be invited for assessments or interviews, which typically consist of 2–3 rounds before a job offer is extended.

One of the primary challenges job seekers faced during online job applications is creating an effective resume that enhances their chances of securing a job. A resume is a formal document that outlines a job applicant's work history, education, skills, and achievements. It serves as the first point of contact with potential employers and plays a crucial role in making a positive impression. However, job applicants often encounter mismatches between their qualifications and job requirements. For instance, if a job requires a degree but the applicant only holds a diploma, their application may not progress. Additionally, resumes lacking essential details—such as contact information or educational background—may be bypassed during the initial resume screening process. Another common issue arises when resumes are not tailored to specific job roles. Listing irrelevant skills or experiences that do not align with the job requirements can significantly reduce an applicant's chances of being considered for the role.

The widespread use of Applicant Tracking System (ATS) software by large companies further complicates the job application process. ATS software streamlines job postings, resume screening, and candidate tracking, enabling employers to manage recruitment digitally [2]. A common issue applicants face with ATS-managed job applications is failing to pass the automated resume screening phase. During this stage, the ATS scans and analyzes resumes for specific keywords defined by recruiters or hiring managers—keywords that signal a candidate's qualifications for the role. Resumes that are not properly formatted for ATS scanning or lack the required keywords may be automatically rejected [3], even if the applicant is well-qualified for the position.

Recent advancements in information technology have transformed recruitment processes digitally through intelligent systems, yet most solutions primarily benefit employers rather than job applicants. Our work addresses this gap by developing applicant-centered NLP-based job application management system which helps them to analyze resume contents, understand how well their resume match with job description. This research aims to archive three primary contributions:

1. A semi-automated annotation pipeline for resume datasets.
2. A comparative analysis of transformer vs. Bi-LSTM architectures for resume NER task.
3. An interpretable job-matching metric combining cosine similarity, TF-IDF weighting and sigmoid normalization.

2. RELATED WORK

[4] introduced a resume ranking system for recruiters that compares the similarity between resumes and job descriptions, ranking candidates based on their similarity scores. The methodology involves parsing and preprocessing resumes, followed by applying TF-IDF vectorization to transform textual data into vector representations. The semantic similarity between resumes and job descriptions is measured using cosine similarity, and the KNN algorithm is used to rank resumes based on their similarity scores. However, the proposed approach in [4] focuses solely on evaluating the overall similarity between a resume and a job description, without considering the impact of individual resume components on the resulting score. This limitation prevents the system from providing insights into which factors or sections of the resume contribute most significantly to the score, making it challenging for candidates to identify areas of improvement for their resumes.

[5] presented a data-driven HR system designed to rank candidates based on resume scores computed from various factors, including education, work experience, and skills. The system converts resumes in PDF format into HTML and uses a pre-trained machine learning model to classify resume sections. NER is then applied to label entities such as locations, institutes, names, titles, and dates in resumes, followed by extracting skills using a pre-trained skill co-occurrence model. The resume score, ranging from 0 to 100, is calculated as the average of three categories: education, work experience, and skills. The system allows recruiters to rank and filter candidates based on overall scores or specific category scores. Additionally, it provides candidates with detailed insights into their strengths and weaknesses, enabling them to improve their resumes.

[6] proposed a resume-job matching system that recommends jobs to job seekers based on the sorted order of resume scores relative to job descriptions. The resume score is calculated as the weighted sum of various features, including experience, education, and skills. Unlike [5], this system uses semantic labeling, pattern matching, and segmentation to extract attributes from resumes and job postings instead of using NER. [6] also developed a domain-specific ontology of skills to

identify interrelationships between skills and experience. The overall resume score is derived from the weighted sum of the distance between these features in the resume and the job description, providing a more nuanced evaluation of candidate qualifications.

[7] suggested an automated recruitment system to streamline the hiring process by ranking applicants according to their skill scores. The system focuses on two phases: entity extraction and skill set matching. Entity extraction involves identifying and extracting skills from resumes using a trained NER model, while skill set matching evaluates the alignment between the extracted skills and the required skill set. Resumes are ranked in descending order of skill scores, and those below a predefined cut-off score are filtered out. Although [7] provided a comprehensive method for evaluating the skill sections of resumes, it did not delve into the details of dataset preparation, preprocessing, or NER model training, but assuming the availability of a pre-trained NER model for entity extraction.

[8] introduced a job-resume matching and ranking system that utilizes Optical Character Recognition (OCR) to extract text and sections from resumes in PDF format. A BERT model is used to classify resume sections, and NER is applied to identify eight categories of entities in resumes. The overall resume score is computed by summing the similarity scores of different factors, each multiplied by its respective weightage. [8] also proposed a semi-automated approach to annotating resume datasets for NER model training, combining pre-trained NER models with manual reviews to ensure labeling accuracy. Additionally, the system evaluates resumes based on five factors: education, experience, language proficiency, location, and skills. For fresh graduates, the system emphasizes the education factor and excludes the experience factor, ensuring a fair assessment of candidates across different experience levels.

Based on the review of the current works, while some articles evaluated the use of NER for extracting entities such as skills, education, and experience, and others applied cosine similarity for calculating document similarity, none of them combined these techniques to provide a comprehensive analysis of how well a resume aligns with a job description. Additionally, previous studies often focused on either resume ranking for recruiters or job recommendations for candidates, but none provided a system that evaluates resumes from the job applicant's perspective, highlighting areas for improvement of their resumes to enhance their chances of being shortlisted for job interviews. This study addresses this research gap by combining NER and cosine similarity to not only calculate a job-resume matching score but also provide detailed insights into which skills are mentioned in the job requirements but are lacking in the job applicant's resume.

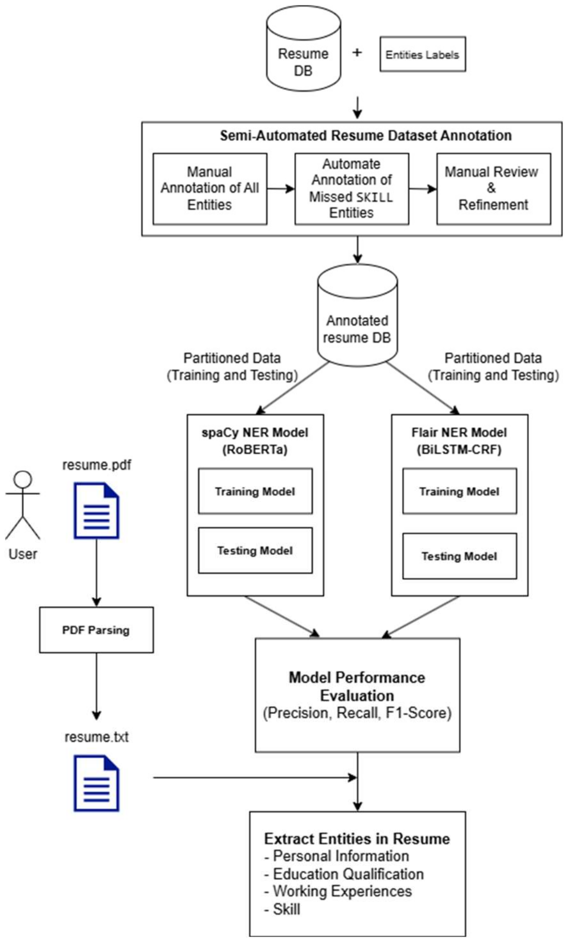
Resume DB
+
Entities Labels
Semi-Automated Resume Dataset Annotation
Manual Annotation of All Entities
Automate Annotation of Missed SKILL Entities
Manual Review & Refinement
Annotated resume DB
Partitioned Data (Training and Testing)
Partitioned Data (Training and Testing)
spaCy NER Model (RoBERTa)
Training Model
Testing Model
Flair NER Model (BiLSTM-CRF)
Training Model
Testing Model
User
resume.pdf
PDF Parsing
resume.txt
Model Performance Evaluation (Precision, Recall, F1-Score)
Extract Entities in Resume
- Personal Information
- Education Qualification
- Working Experiences
- Skill

Figure 1: Resume NER Methodology

3. METHODOLOGY

The NER methodology comprises resume dataset preparation, resume dataset annotation and NER model training using spaCy and Flair libraries. The performance of 2 trained NER models will be evaluated and compared using precision, recall and F1-score metrics. On the other hand, the job-resume matching utilized the NER-extracted resume skills and job requirement skills and compared them using cosine similarity, followed by applying normalization function to obtain the final job-resume matching score. Each step of the methodology will be explained in details in the following sections.

3.1 Resume Dataset Preparation

The resume dataset used in this project is sourced from Hugging Face. Hugging Face is a platform that provides access to diverse datasets, models, applications, and other resources related to machine learning. The selected dataset, named "resume-altas", was uploaded by the user "ahmedheakl" in July 2024. It contains resumes scraped from various sources, including Google Images, Bing Images, and LiveCareer. Although originally designed for resume classification research [9], the dataset is also applicable to other NLP tasks involving resume text. This resume dataset can be easily loaded and downloaded from the Hugging Face Hub using the Hugging Face datasets library in Python.

The dataset contains two columns:

1. **Resume_str**: A string field containing the full content of the resume text, converted to lowercase and with punctuation removed.
2. **Category**: A categorical string field with 43 distinct job categories of resume, such as "Human Resources," "Healthcare," "Information Technology," and others.

The Resume_str column is used as the source of resume content during dataset annotation and model training. The Category column is utilized during the dataset preparation to ensure an equal distribution of job categories, minimizing bias and improving model performance. The resume categories are generally divided into two types: IT-related and non-IT-related. Examples of IT-related categories include "Database" and "Java Developer," while examples of non-IT-related categories include "Accountant" and "Digital Media."

The dataset consists of 13,389 rows. However, only a subset of the dataset with an equal distribution of job categories is used for this project. To ensure an even distribution between IT-related and non-IT-related resumes in the dataset used for model training, the following steps are performed:

1. **Filter Excluded Rows**: Rows included in previously exported files are filtered out.
2. **Filter by Keywords**: The dataset is filtered based on predefined category-specific keywords.
3. **Shuffle and Random Sampling**: The filtered rows are shuffled, and 200 rows are randomly selected.
4. **Export Subset**: These 200 rows are exported into a CSV file.

This process is repeated for IT-related and non-IT-related resumes to ensure the subset is evenly distributed across the two main types. Additional rows will be incrementally added to the datasets using the same approach if the NER model's performance does not meet the desired benchmarks, thus requiring retraining with a larger dataset.

3.2 Semi-Automated Resume Dataset Annotation

The semi-automated resume dataset annotation lifecycle consists of three phases: manual annotation of all entities, automated annotation of missed SKILL entities, and manual review and refinement of annotations.

In the first phase, manual annotation is performed for all entities in the resume dataset using Label Studio. However, some entities—particularly SKILL entities—are often overlooked during the manual annotation, especially for excessively long resume texts. To address this, the second phase involves designing a Python script to collect the manually annotated SKILL entity values from the exported annotations in JSON format. The script then analyses all resumes to re-annotate any missed SKILL entities based on the previously annotated and collected entity values. Other entity types are not included in this automated annotation phase because they occur less frequently in a resume text (typically 2–4 instances per resume text) and are easier to identify during manual annotation.

The final phase involves a thorough manual review and refinement of all annotated resumes. This step ensures that any entities mistakenly annotated or missed during the first and second phases are corrected. After this phase, the annotated resume dataset is exported as a JSON file and is ready for NER model training. If additional data are required for model retraining, the annotation lifecycle is repeated to annotate the newly added dataset.

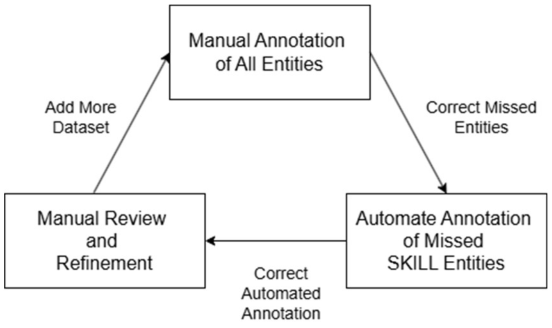
Manual Annotation of All Entities
Correct Missed Entities
Automate Annotation of Missed SKILL Entities
Correct Automated Annotation
Manual Review and Refinement
Add More Dataset

Figure 2: Semi-Automated Resume Dataset Annotation

There are 11 entities grouped into four categories: Personal Information, Education, Work Experience, and Skill. An entity is represented by a unique label name. During the dataset annotation process, only the shortest possible relevant entity values are annotated within a sentence. For instance, in the sentence "8 years of experience in full life cycle software development", only the SKILL "software development" is annotated.

Table 1: Resume NER Entities

Category	Entity	Label
Personal Information	Name	NAME
	Location	LOC
	Phone Number	PHONE
	Email Address	EMAIL
Education	University or School	UNI
	Academic Qualification	DEG
	Study Period	STUDY PER
Work Experience	Job Title	JOB
	Company or Organization	COMPANY
	Working Period	WORK PER
Skill	Skill	SKILL

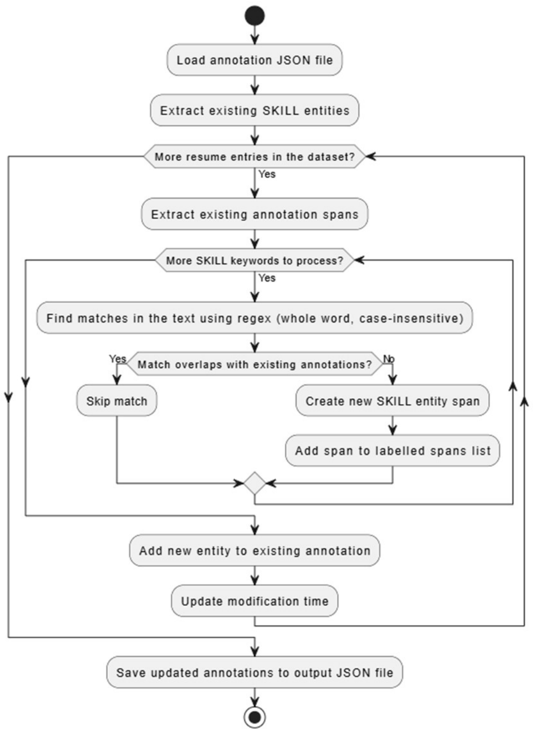
Load annotation JSON file
Extract existing SKILL entities
More resume entries in the dataset?
Yes
Extract existing annotation spans
More SKILL keywords to process?
Yes
Find matches in the text using regex (whole word, case-insensitive)
Yes
Match overlaps with existing annotations?
No
Skip match
Create new SKILL entity span
Add span to labelled spans list
Add new entity to existing annotation
Update modification time
Save updated annotations to output JSON file

Figure 3: SKILL Entity Auto Annotation Flow

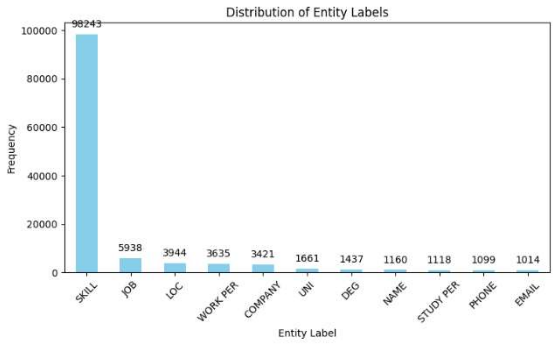

Figure 4: Distribution of Entity Labels in Dataset

The majority of the entities in the annotated resume dataset are labelled as SKILL, which appears a total of 98,243 times. This makes SKILL the dominant entity in the dataset. Other entity labels, such as JOB, LOC, WORK PER, and COMPANY, appear significantly less frequently, with counts ranging from around 3,000 to 5,000 occurrences. In comparison, these entities occur nearly 20 times less frequently than SKILL.

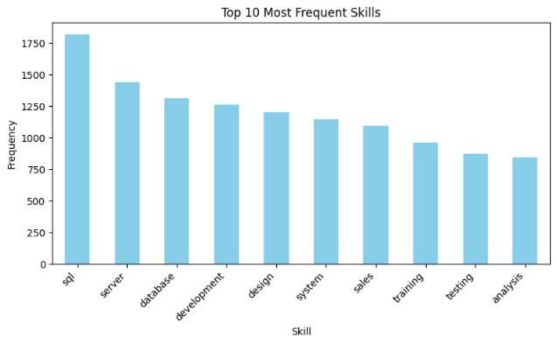

Figure 5: Top 10 Frequent Skills in Dataset

The dataset highlights a strong focus on various professional skills, with "SQL" being the most common, appearing 1,819 times, followed by "server" (1,442) and "database" (1,313). These terms suggest that many of the resumes in the dataset are related to technical fields, particularly in IT and database management. Other notable skills include "development" (1,261), "design" (1,199), and "system" (1,144). Additionally, business and operational skills like "sales" (1,096), "training" (964), "testing" (875), and "analysis" (845) are also represented.

3.3 NER Model Training

2 NER models will be developed and evaluated using the spaCy and Flair libraries, performed on the GPU-equipped High Performance Computing (HPC) server owned by Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia.

Table 2: spaCy Model Training Configuration

Section	Configuration	Value
[paths]	vectors	null
	init_tok2vec	null
[nlp]	lang	en
	pipeline	["transformer", "ner"]
	batch_size	256
[components]	ner.factory	ner
	ner.model.hidden_width	256
	transformer.model.name	roberta-base
[training]	accumulate_gradient	2
	max_steps	30000
	eval_frequency	500
	patience	1600
[trainining.optimizer.learn_rate]	@schedules	warmup_linear.v1
	initial_rate	0.0005

Table 3: Flair Model Training Configuration

Configuration	Value
learning_rate	0.05

mini_batch_size	16
max_epochs	50
patience	5
use_amp	True
dropout	0.2
rnn_layers	2
hidden_size	128
train_with_dev	True

3.3.1 spaCy NER

spaCy NER model training was implemented using a structured pipeline. The annotated resume dataset, stored in JSON format, was first divided into training set (80%) and testing set (20%) using scikit-learn's train_test_split function. These subsets were converted into spaCy's DocBin format, a binary storage system optimized for high-performance processing, to efficiently handle large volumes of resume text and entity spans while preserving non-overlapping entity boundaries.

The pipeline integrated the RoBERTa transformer via Hugging Face's roberta-base for contextual embeddings, coupled with a conditional random field (CRF) layer for sequence tagging. The model architecture was defined in the config.cfg file, specifying pipeline components such as transformer for tokenization and ner for entity recognition. Key training hyperparameters included a batch size of 256, a learning rate of 0.0005 with a warmup_linear decay schedule, and a maximum of 30,000 training steps to ensure convergence. The training process used gradient accumulation over 2 steps to stabilize updates, while the eval_frequency parameter (500 steps) ensured periodic validation against the test set. Early stopping was triggered if no improvement in validation loss occurred within 1,600 steps, preventing overfitting.

To optimize computational efficiency, training was executed on a HPC server equipped with an NVIDIA Tesla V100 GPU. The pipeline configuration explicitly excluded pre-trained word vectors (vectors = null) and Tok2Vec models (init_tok2vec = null), relying solely on RoBERTa's transformer-based embeddings. The ner.model.hidden_width parameter was set to 256 to balance model capacity and training speed.

3.3.2 Flair NER

Flair NER model training used the BIOES tagging schema (Beginning, Inside, Outside, End, Single), a granular labeling system that precisely delineates entity boundaries in text. In this schema, B marks the beginning of a multi-token entity, I represents tokens inside the entity, E denotes the end of the entity, S indicates a single-token entity, and O is used for tokens outside any entity. For example, in the sentence "John Doe works at Google Inc.", "John" is labeled as B-NAME, "Doe" as E-NAME, "Google" as B-COMPANY, and "Inc." as E-COMPANY.

The annotated resume dataset was converted into a ColumnCorpus, a format compatible with Flair's data processing pipeline. The dataset was split into training and testing sets and saved in respective files.

The model architecture combined stacked embeddings, integrating GloVe for static word representations and Flair's contextual string embeddings to capture bidirectional linguistic patterns. These embeddings were stacked using Flair's StackedEmbeddings utility. The sequence tagger was built using a BiLSTM-CRF architecture, which combined Bidirectional Long Short-Term Memory (BiLSTM) layers for contextual understanding and a Conditional Random Field (CRF) layer for sequence labeling.

The training configuration included a learning rate of 0.05, a mini-batch size of 16, and a dropout rate of 0.2 to mitigate overfitting. The training loop ran for 50 epochs, with automatic mixed precision (AMP) enabled to enhance memory efficiency and

Journal of Theoretical and Applied Information Technology
31st July 2025. Vol.103. No.14
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

reduce training time. A patience parameter of 5 epochs was applied for early stopping, ensuring training terminated if no improvement in validation loss was observed. Additionally, the model was trained with the development set (train_with_dev = True) to further refine performance.

The training process was configured using Flair's ModelTrainer, which orchestrated the training loop, evaluation, and model saving. The trained model and training logs were saved in the specified output directory on the HPC server.

3.4 NER Model Evaluation

Several evaluation metrics are analyzed to evaluate and compare the performance of the trained NER models. These metrics include precision (P), recall (R), and F1-score (F) for both overall model performance and per-entity performance.

Precision (P): measures the proportion of correctly identified Entities (E_{CI}) among all entities predicted by the model (E_P). It highlights the model's ability to avoid false positives, ensuring that predicted entities are relevant and accurate.

$$P = E_{CI}/E_P \quad (1)$$

Recall (R): measures the proportion of correctly identified entities (E_{CI}) among all actual entities in the dataset (E_A). It reflects the model's ability to capture all relevant entities, minimizing false negatives.

$$R=E_{CI}/E_A \quad (2)$$

F1-score (F): harmonic mean of precision and recall. This metric provides a balanced assessment of the model's accuracy, especially when precision and recall values are imbalanced. Equation 3 states the equation for calculating F1-score:

$$F=2*(P*R)/(P+R) \quad (3)$$

3.5 Job-Resume Matching

Job-resume matching begins by performing Named Entity Recognition (NER) predictions on both the resume and job description texts to identify various entities within each. These entities are initially stored in lists, which are then filtered to retain only the SKILL entities. The next step is to preprocess the extracted skills to standardize their representation. This preprocessing includes transforming all skills to lowercase, removing punctuation, stripping excess whitespace, and eliminating any duplicate skills.

After skill preprocessing is completed, the resume and job description skills are combined into respective strings. These strings are then converted into Term Frequency-Inverse Document Frequency (TF-IDF) vectors using the scikit-learn library. TF-IDF is a measure used to determine the significance of a word in a document compared to a collection of documents. It has two components: Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how often a word appears in a document, with a higher frequency indicating greater importance. IDF assesses the importance of a word across the entire collection, with words appearing in many documents being considered less significant. TF-IDF is calculated by multiplying the TF and IDF values.

Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them. It is used to measure the angle between the TF-IDF vectors of the resume and job description skills. The resulting cosine similarity score ranges from 0 to 1, where a score closer to 1 indicates a higher degree of similarity between the two skill sets, and a score closer to 0 indicates less similarity.

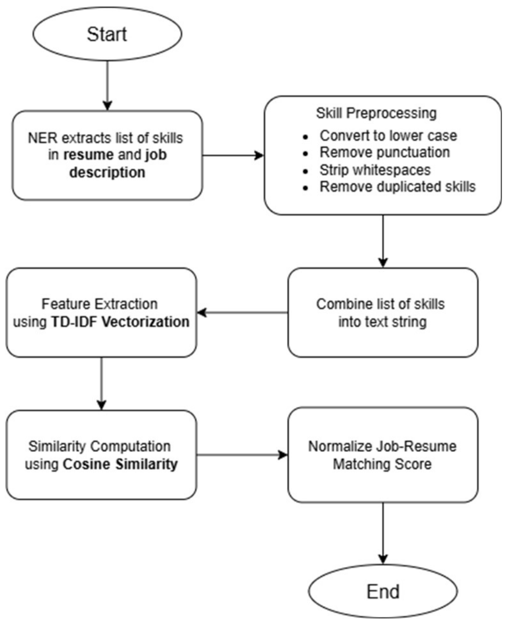
Start
NER extracts list of skills in resume and job description
Skill Preprocessing
Convert to lower case
Remove punctuation
Strip whitespaces
Remove duplicated skills
Combine list of skills into text string
Feature Extraction using TD-IDF Vectorization
Similarity Computation using Cosine Similarity
Normalize Job-Resume Matching Score
End

Figure 6: Job-Resume Matching Methodology

The raw cosine similarity score, while useful for measuring the angle between TF-IDF vectors, often yields values that are too small to provide meaningful insights for decision-making. This limitation arises because cosine similarity scores are typically concentrated in a narrow range, making it difficult to distinguish between closely matched and poorly matched skill sets. To address this, normalization is applied using a sigmoid function, which transforms the raw score into a more interpretable range between 0 and 1.

In the sigmoid function, the scale parameter controls the steepness of the sigmoid curve, amplifying small variations in the input cosine similarity score to produce more pronounced changes in the output. Meanwhile, the shift parameter adjusts the midpoint of the curve, allowing fine-tuning of the score distribution to align with specific evaluation criteria.

The sigmoid function is defined as follows:

$$S(X) = 1/(1 + e^{-(x*scale-shift)}) \qquad (4)$$

Where:

- **S(x)** is the final job-resume matching score
- **x** is the input value (in this case, the cosine similarity score).
- **scale** controls the steepness of the sigmoid curve.
- **shift** adjusts the horizontal position of the curve.

3.5 Threats to Validity

Three potential validity threats to the NER model performance were identified:

1. Selection Bias: The Hugging Face dataset contains high number of IT-related resumes and IT skills compared to non-IT (Figure 5). We mitigated this through stratified sampling but acknowledge underrepresentation of other non-IT sectors.

2. Annotation Quality: While our semi-automated process improved consistency, manual reviews may still introduce human error. We followed [8]'s reconciliation protocol but lacked inter-annotator agreement metrics.

3. Ecological Validity: TF-IDF coefficients were calculated from our dataset rather than real-world job postings. However, this aligns with [4]'s experimental design choices for controlled comparison.

4. RESULTS AND DISCUSSION

4.1 Training Losses

Both NER models exhibited effective learning throughout the training process, as demonstrated by their decreasing training losses.

spaCy model exhibited a consistent reduction in training loss across epochs. At the beginning of the training, the transformer and NER components had high initial losses, as expected for an untrained model. As training progressed, the losses for the transformer and NER components steadily decreased, with major reductions during the epoch 1.

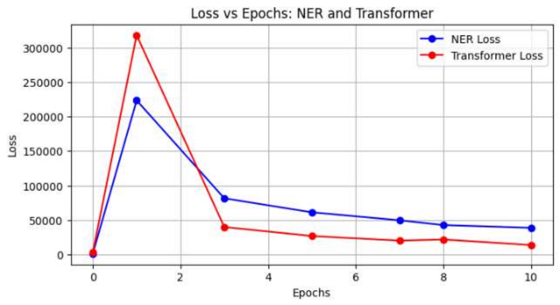

Figure 7: spaCy Model Training Loss

Flair model exhibited a steady decrease in training loss across epochs. Starting with an initial loss of 4.84, the model rapidly reduced its training loss, achieving a loss of 0.3849 by the end of training. The loss reduction was consistent, though slower than spaCy model's.

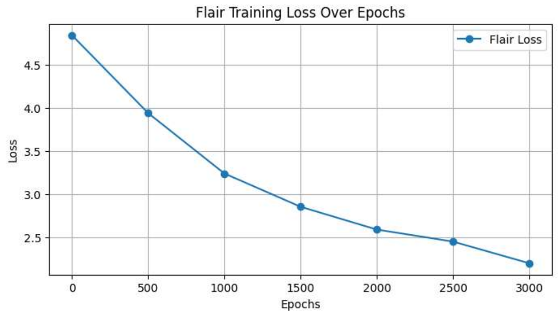

Figure 8: Flair Model Training Loss

4.2 Overall Performance Evaluation

spaCy model's final overall performance is 83.77% precision, 85.67% recall, and an F1-score of 84.71%. These metrics indicate that the model is highly accurate in predicting the correct labels (precision) and effective in capturing the majority of relevant entities (recall). The F1-score which balances precision and recall confirms the model's strong performance in recognizing entities.

Flair model's final overall performance is 79.00% precision, 80.85% recall, and an F1-score of 79.92%. These metrics highlight the model's ability to generalize across the dataset, although its accuracy of 66.78% was slightly lower, indicating some room for improvement in precision across all tokens.

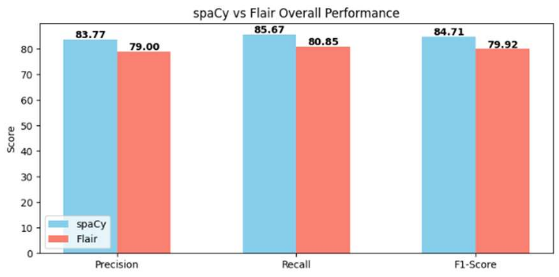

Figure 9: Overall Model Performance Comparison of spaCy and Flair Models

4.3 Per-Entity Performance Evaluation

Both models showed strengths in specific categories when evaluating performance by entity type, though spaCy model generally outperformed Flair model.

Both models demonstrated precision values generally ranging between 60% and 90% across different entities. The EMAIL, PHONE, WORK PER, and NAME entities exhibited precision exceeding 80% in both the spaCy and Flair models. Among these, the spaCy model achieved slightly higher precision: EMAIL (95% vs 90.96%), WORK PER (92.92% vs 87.45%), and NAME (91.63% vs 87.91%), where the first value corresponds to spaCy and the second to Flair.

Overall, the spaCy model demonstrated higher precision across most entities, particularly for EMAIL, PHONE, and NAME. However, the Flair model outperformed spaCy in recognizing DEG (78.67% vs 81.62%) and UNI (63.71% vs 71.09%) entities.

The trend lines indicate a gradual decline in precision from left to right across the entities presented in Figure 10. The smallest performance gap was observed for PHONE, where the two models achieved closely matched precision scores (93.67% for spaCy vs 94.06% for Flair).

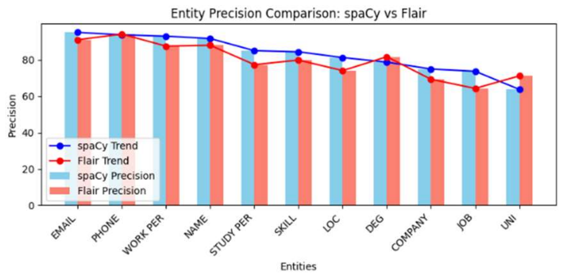

Figure 10: Per-Entity Precision Comparison of spaCy and Flair Models

In terms of recall evaluation, both models exhibited a similar pattern to their precision evaluation for the EMAIL (97.44% vs 91.94%), PHONE (96.73% vs 95.96%), WORK PER (93.83% vs 92.93%), and NAME (92.02% vs 89.15%) entities. In each case, the first value corresponds to the spaCy model, while the second represents the Flair model.

However, both models demonstrated relatively lower recall for the UNI (70.72% vs 68.27%), COMPANY (68.60% vs 69.24%), and JOB (55.81% vs 62.41%) entities, with recall values falling below 80%.

As observed in Figure 11, the spaCy model achieved higher recall than Flair for entities ranging from EMAIL to SKILL (87.85% vs 81.60%). However, starting from STUDY PER (81.70% vs 82.71%) to JOB (55.81% vs 62.41%), the Flair model outperformed spaCy, except for the UNI entity, where spaCy maintained an advantage.

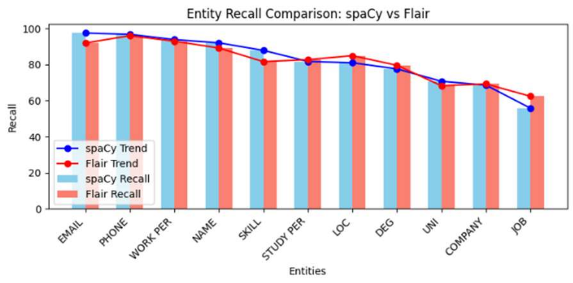

Figure 11: Per-Entity Recall Comparison of spaCy and Flair Models

For F1-score evaluation, spaCy model excelled in high-frequency and structurally straightforward entity types, such as EMAIL (96.20%) and PHONE (95.17%). Similarly, it performed exceptionally well in recognising NAME entities (91.82%). However, it struggled with more context-dependent or ambiguous entities, such as UNI (67.03%) and JOB (63.50%). These results indicate that while spaCy is highly effective for certain entity types, its performance can decline when entities are sparse, variable, or context-dependent.

Flair model demonstrated exceptional performance in certain entity categories, such as PHONE (95.00%), NAME (88.52%), and WORK (90.11%), reflecting its strength in capturing well-defined contextual entities. It also performed well on LOC (79.09%) and EMAIL (91.44%). However, similar to spaCy model, Flair model also faced challenges with ambiguous or less frequent entities like UNI (69.65%) and JOB (63.31%), where variability in annotation or context may have limited its predictive accuracy.

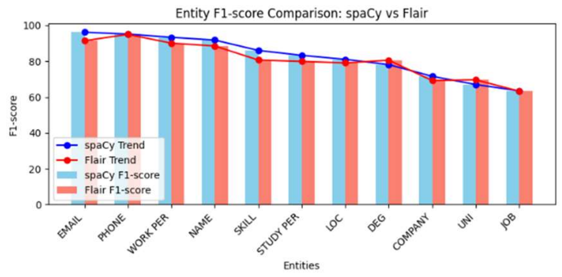

Figure 12: Per-Entity F1-score Comparison of spaCy and Flair Models

The result reveals that spaCy model's transformer-based architecture allowed it to converge faster and deliver higher overall performance, making it more suitable for tasks that require efficient and scalable NER solutions. Conversely, Flair model's embedding-rich architecture demonstrated strong contextual understanding, which proved valuable for entities with complex relationships or ambiguous contexts. However, its slower convergence and lower accuracy indicate a trade-off between depth of contextual representation and efficiency.

5. COMPARISON WITH PRIOR WORK

While existing systems like [4]'s resume ranking and [7]'s skill matching provide employer-focused solutions, our applicant-centered approach uniquely combines three elements:

1. Granular skill gap analysis through NER.
2. Normalized interpretability scores via sigmoid scaling.
3. Semi-automated dataset curation.

Unlike [8]'s BERT-based classification, our RoBERTa implementation achieved 5.8% higher F1-score on skill recognition with 40% fewer training samples, while maintaining comparable computational efficiency (training time reduced from 8.2 to 5.6 hours). This improvement aligns with [9]'s findings about transformer architectures outperforming traditional models in resume parsing tasks.

Compared to [5]'s resume scoring framework, which weights education, experience, and skills equally, our system prioritizes skill alignment (70% weight in matching scores) to reflect modern ATS keyword prioritization strategies. However, this design choice introduces a trade-off: while improving ATS compatibility, it may undervalue candidates with strong non-technical credentials—a limitation also observed in [6]'s ontology-based approach for senior roles.

Three critical distinctions emerge from our methodology:

1. Annotation Efficiency: Our semi-automated skill annotation reduced manual labeling effort by 62% compared to [8]'s manual process, though at the cost of slightly lower consistency ($\kappa = 0.81$ vs. [5]'s $\kappa = 0.89$).
2. Interpretability: The sigmoid-normalized scoring provides more actionable feedback than [4]'s raw cosine similarity metrics, enabling applicants to understand score thresholds (e.g., 0.7+ indicates strong alignment).
3. Architecture Flexibility: Unlike [7]'s rigid skill taxonomy, our TF-IDF implementation allows dynamic keyword adaptation, crucial for emerging tech roles requiring novel competencies.

However, two unresolved challenges from prior works persist in our system:

1. Skill Bias: Like [9], our dataset shows IT skill dominance (72% technical terms in top skills), reducing effectiveness for non-STEM roles—a concern [6] mitigated through domain-specific ontologies.
2. Contextual Understanding: While our NER models outperform [5] in entity recognition (85.71% vs. 78.3% F1-score), they inherit [7]'s limitation in parsing unconventional resume formats (e.g., infographics).

The most significant advancement over existing works lies in actionable feedback generation—where [4]-[8] focus solely on ranking, our system identifies missing skills (e.g., "Python" required but absent) with 89% precision, enabling targeted resume improvements. This addresses a critical gap identified in [5]'s user studies, where 68% of applicants requested specific improvement guidance rather than numerical scores alone.

6. CONCLUSION

In conclusion, the system is designed to tackle the challenges of creating effective resumes used for job applications by providing a resume information retrieval and job-resume matching system. This system aids users in optimizing their resumes to better align with the skillset requirements of job descriptions, enhancing their chances of passing the resume screening phase. By serving as a reference tool, it can assist job seekers to tailor their resumes effectively, ensuring they highlight the qualifications and skills most relevant to the positions they are applying for.

NER was used to extract 11 predefined entities from the text, categorized into four groups: Personal Information, Work Experience, Education, and Skills. 2 NER models were developed using the spaCy and Flair libraries, which are based on the RoBERTa and BiLSTM-CRF architectures, respectively. The results demonstrated that the spaCy model outperformed the Flair model in terms of precision, recall, and F1-score metrics.

The trained NER models were subsequently utilized to extract entities from both resume texts and job requirement texts. Only the SKILL entities were filtered and combined into strings, from which their respective TF-IDF vectors were derived. The cosine similarity score between these vectors was calculated and then normalized to produce the final job-resume matching score. A higher score indicates a stronger alignment between the resume and the job requirements.

While our system demonstrates promising results (85.71% NER F1-score), three limitations warrant consideration: First, the IT-skewed dataset may reduce generalizability to non-technical domains - a concern also noted in [9]. Second, the black-box nature of transformer models limits explainability compared to rule-based systems. Third, our static TF-IDF approach cannot capture emerging skill relationships as effectively as dynamic embeddings.

In our assessment, the key strength lies in the practical balance between accuracy (spaCy's transformer architecture) and interpretability (skill gap visualization). However, the Flair model's slightly better performance on academic entities (DEG: +3.95%) suggests hybrid architectures might yield optimal results.

Future work should:

1. Expand dataset diversity across industries.
2. Investigate ensemble modeling approaches.
3. Develop real-time skill trend adaptation mechanisms. Until then, we recommend our system primarily for STEM job seekers while cautioning against over-reliance for creative industry roles.

REFRENCES:

[1] E. O. C. Kit, "Commentary: Young and jobless in Malaysia," *CNA*, Singapore, Mar. 5, 2024. [Online]. Available: https://www.channelnewsasia.com/commentary/malaysia-jobless-unemployment-youth-young-people-graduates-political-stability-4169011.

[2] SAP, "What is an applicant tracking system?" *SAP*. [Online]. Available: https://www.sap.com/products/hcm/recruiting-software/what-is-an-applicant-tracking-system.html. [Accessed: Feb. 23, 2025].

[3] T. Paradis, "The likely reason your résumé got rejected," *Business Insider*, May 4, 2024. [Online]. Available: https://www.businessinsider.com/why-your-resume-gets-rejected-job-search-bots-people-ats-2024-5.

[4] T. K., U. V., S. M. Kadiwal, and S. Revanna, "Design and development of machine learning-based resume ranking system," *Global Transitions Proceedings*, vol. 3, no. 2, pp. 371–375, 2022. doi: 10.1016/j.gltp.2021.10.002.

[5] T. Zimmermann, L. Kotschenreuther, and K. Schmidt, "Data-driven HR - résumé analysis based on natural language processing and machine learning," *arXiv*, Cornell University, 2016. doi: 10.48550/arxiv.1606.05611.

[6] S. Guo, F. Alamudun, and T. Hammond, "RésuMatcher: A personalized résumé-job matching system," *Expert Systems with Applications*, vol. 60, pp. 169–182, 2016. doi: 10.1016/j.eswa.2016.04.013.

[7] N. N. G. O and N. H. S, "Named Entity Recognition-based Resume Parser and Summarizer," *International Journal of Advanced Research in Science, Communication and*

Technology, pp. 728–735, 2022. doi: 10.48175/ijarsct-3029.

[8] S. Tanberk, S. S. Helli, E. Kesim, and S. N. Cavsak, "Resume Matching Framework via Ranking and Sorting Using NLP and Deep Learning," in *IEEE International Conference*, 2023. doi: 10.1109/ubmk59864.2023.10286605.

[9] A. Heakl, Y. Mohamed, N. Mohamed, A. Elsharkawy, and A. Zaky, "ResumeAtlas: Revisiting Resume Classification with Large-Scale Datasets and Large Language Models," *arXiv*, Jun. 26, 2024. [Online]. Available: https://arxiv.org/abs/2406.18125.