

AWS-Based Automated Applicant Tracking System for Job Matching and Recommendation

Bammidi Lokesh

M.Tech – Computer Networks and Information Security (CNIS), Andhra University, Visakhapatnam, India

Email: [lokeshbammidi954\[at\]gmail.com](mailto:lokeshbammidi954[at]gmail.com)

Abstract: *The rapid growth of online recruitment platforms has created a large volume of job postings, making it difficult for candidates to identify positions that closely match their skills and professional experience. This paper presents an AWS-based automated Applicant Tracking System (ATS) for job collection, processing, candidate-job matching, recommendation, and notification. The proposed system collects job postings from multiple sources and stores raw data in Amazon Simple Storage Service (Amazon S3). AWS Lambda is used for serverless ingestion and orchestration, while AWS Glue performs extraction, transformation, normalization, and conversion of job data into a query-efficient processed format. Amazon Athena provides SQL-based access to the processed data. Two matching approaches are implemented: a traditional ATS approach using textual similarity, skills, job title, and experience information, and a machine-learning-based approach combining TF-IDF, BM25, Sentence-BERT semantic similarity, skill matching, and experience matching. Selected recommendations can be delivered through Amazon Simple Email Service (Amazon SES). The architecture is designed to be scalable, modular, and suitable for automated job recommendation workflows. Experimental evaluation includes threshold sensitivity and matching-score analysis to study the behavior of the recommendation pipeline.*

Keywords: Applicant Tracking System; AWS; BM25; Job Recommendation; Machine Learning; Sentence-BERT

1. Introduction

Recruitment systems generate and process a continuously changing collection of job advertisements containing job titles, descriptions, skills, locations, employment types, salary information, and experience requirements. A candidate who manually examines a large number of postings may spend considerable time identifying relevant opportunities. Automated matching can reduce this effort by comparing candidate information with job requirements and ranking relevant positions.

Traditional ATS implementations commonly depend on keyword or textual matching. Such methods are useful for identifying explicit skill overlap, but they may not adequately capture semantic relationships between a resume and a job description. For example, related technical terms can express similar concepts even when the exact words differ. This motivates the use of complementary information-retrieval and semantic techniques.

This work proposes an AWS-based pipeline that combines serverless cloud services with text-processing and machine-learning techniques. The system separates data ingestion, ETL processing, query, matching, evaluation, and notification into modular components. This design supports repeatable processing and allows the matching methodology to be improved without redesigning the complete data-ingestion layer.

2. Objectives

- Automate collection of job postings from multiple sources.
- Store raw and processed job information in a cloud-based data lake.
- Normalize job titles, descriptions, skills, and experience-related fields.

- Provide SQL-based analysis of processed jobs using Amazon Athena.
- Calculate candidate-job compatibility using traditional and ML-based matching.
- Include candidate experience and job-required experience in recommendation analysis.
- Generate job recommendations and configurable email alerts for high-scoring matches.
- Evaluate the effect of ATS thresholds on recommendation and notification behavior.

3. Related Work

Earlier job recommendation and resume-screening systems have commonly used keyword matching, vector-space similarity, information-retrieval techniques, and machine-learning models. TF-IDF represents documents using term importance and provides a straightforward baseline for textual similarity. BM25 improves information retrieval by considering term frequency, document length, and inverse document frequency. More recent transformer-based sentence-embedding methods provide semantic representations that can identify relationships beyond exact keyword overlap.

Cloud computing provides an additional opportunity to operationalize these techniques as scalable services. Serverless functions can handle ingestion and orchestration without requiring continuously running servers, object storage can maintain large volumes of raw and processed data, and SQL query engines can provide analytical access. The proposed system combines these capabilities into an end-to-end recruitment analytics pipeline.

4. Proposed System

The proposed architecture consists of six logical stages: job ingestion, raw data storage, ETL processing, analytical

querying, candidate-job matching, and recommendation/notification. Job records from external sources are collected through an ingestion layer and stored in Amazon S3. AWS Glue processes and normalizes the data before storing the processed representation. Amazon Athena provides a query layer over the processed dataset.

The matching layer reads candidate and job information and produces a compatibility score. The traditional ATS branch emphasizes textual and explicit skill similarity. The ML branch combines lexical, information-retrieval, semantic, skill, and experience features. The resulting recommendations can be stored in Amazon S3 and high-scoring results can be sent through Amazon SES. Fig. 1 shows the overall architecture.

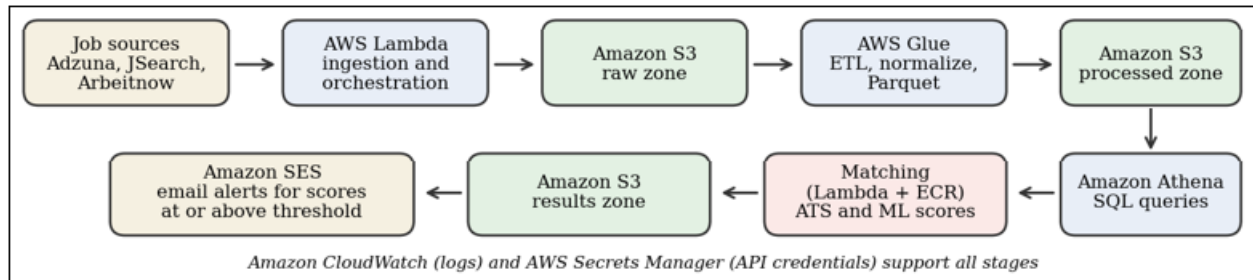


Figure 1: Architecture of the proposed AWS-based ATS pipeline.

4.1 AWS Components

- Amazon S3: Data lake storage for raw, processed, and result datasets.
- AWS Lambda: Serverless ingestion, orchestration, matching, and notification functions.
- AWS Glue: ETL processing, normalization, and Parquet generation.
- Amazon Athena: SQL querying and analytical access to processed job data.
- Amazon DynamoDB: Storage for evaluation or ground-truth records where required.
- Amazon ECR: Container image repository for the machine-learning matching workload.
- Amazon SES: Email delivery for selected job recommendations.
- AWS Secrets Manager: Secure storage of external API credentials.
- Amazon CloudWatch: Execution logs, monitoring, and troubleshooting.

4.2 Data Processing Pipeline

The data pipeline follows a raw-to-processed design. Incoming job records are first stored without changing the source information. During ETL, fields are standardized and text is normalized. Relevant attributes include job identifier, source, country, title, company, description, location, category, URL, salary range, currency, employment type, remote status, posted date, contract type, normalized title, normalized description, and ETL processing metadata. The processed data is stored in Parquet format to support efficient analytical queries.

5. Matching Methodology

5.1 Traditional ATS Matching

The baseline ATS model calculates compatibility from multiple components. TF-IDF similarity measures lexical

similarity between the candidate resume and job description. Skill matching measures the overlap between candidate skills and skills identified from the job. Job-title similarity contributes an additional signal when the candidate title and job title are closely related. Experience information is extracted from the candidate profile and job description so that recommendations can distinguish between textual similarity and practical experience requirements.

A configurable notification threshold is applied to the resulting ATS score. In the implemented workflow, a threshold of 70% is used for notification selection, while threshold analysis can be performed across a wider range of values.

5.2 Machine-Learning-Based Matching

The ML-based approach combines five signals. TF-IDF provides lexical similarity, BM25 provides information-retrieval relevance, Sentence-BERT provides semantic similarity, skill matching measures explicit skill overlap, and experience matching compares candidate experience with the job requirement. The implemented weighted formulation uses 15% TF-IDF, 20% BM25, 35% BERT, 20% skill matching, and 10% experience matching.

The resulting score is computed as a weighted combination of the component scores. Fig. 2 shows the component weights. This design allows the semantic component to contribute strongly while retaining explicit skills and experience as interpretable signals.

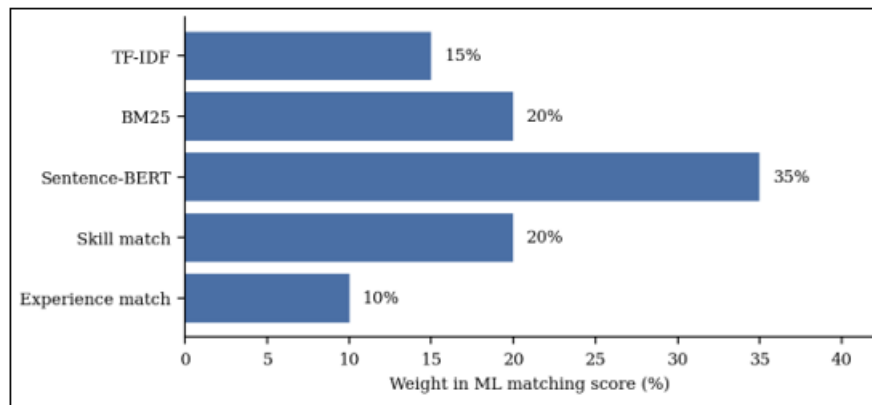


Figure 2: Component weights used in the ML-based matching score

5.3 Experience Matching

Experience matching is treated separately from general textual similarity. Candidate experience is extracted as a number of years where sufficient evidence is available. Job descriptions are processed to identify expressions such as required years of experience. The system reports both candidate experience and job-required experience in the recommendation output. This information provides an additional interpretation layer for the ATS score.

6. Implementation

The system was implemented using AWS cloud services and Python-based processing. The data lake uses an S3 bucket with separate raw, processed, and result prefixes. The processed job dataset is exposed through an Athena database and jobs table. The ETL stage creates a structured Parquet representation suitable for analytical querying.

The ingestion workflow supports multiple job sources, including Adzuna, JSearch, and Arbeitnow in the configured implementation. The combined ingestion dataset is processed before being queried by the matching layer. The architecture is extensible so that additional sources can be added without changing the core matching methodology.

The ML matching workload uses containerized dependencies where required. Results include job identity, company, location, score, skills, experience information, and application URL. Notification logic prevents repeated alerts for the same selected job-candidate combination.

7. Experimental Results and Evaluation

The implemented data-ingestion workflow produced a combined dataset containing hundreds of job records from multiple sources. In the project environment, the combined ingestion metadata recorded 550 job records for the evaluated ingestion run. The processed dataset was made available through Amazon Athena for SQL-based analysis.

The ML evaluation workflow was also tested on a larger candidate-job comparison set. The matching pipeline ranks jobs according to the combined score and retains the configured top recommendations subject to the selected

threshold. The evaluation workflow includes threshold sensitivity analysis over values ranging from 60% to 80%.

Because recommendation quality depends on the candidate profile, job-source coverage, parsing quality, and the selected threshold, the numerical results should be reported using the exact output generated by the final project run. The following table is therefore structured for insertion of the final measured values before submission.

Metric	Value	Description
Ingested Jobs	550*	Combined job records in the evaluated ingestion run
Notification Threshold	70%	Configured threshold for alert selection
ML TF-IDF Weight	15%	Lexical similarity contribution
ML BM25 Weight	20%	Information-retrieval contribution
ML BERT Weight	35%	Semantic similarity contribution
ML Skill Weight	20%	Explicit skill overlap contribution
ML Experience Weight	10%	Experience compatibility contribution
Evaluation Range	60–80%	Threshold sensitivity analysis range

**Replace this value with the final verified dataset count if the final submission run differs.*

8. Discussion

The proposed architecture demonstrates how recruitment matching can be separated into independent cloud-processing stages. The raw and processed zones provide traceability, while Athena supports analytical inspection without requiring a traditional relational database for the full data lake. The use of multiple matching signals provides a richer representation of candidate-job compatibility than a single keyword score.

The principal limitation is that job descriptions and resumes can contain incomplete, ambiguous, or inconsistently formatted information. Experience extraction based on textual patterns can also miss implicit requirements. In addition, an ATS score represents algorithmic similarity and should not be interpreted as a definitive measure of candidate suitability. These limitations motivate continued evaluation with a larger labeled dataset and more systematic ground-truth annotation.

9. Conclusion

This paper presented an AWS-based automated ATS for collecting, processing, matching, ranking, and notifying users about relevant job opportunities. The architecture integrates Amazon S3, AWS Lambda, AWS Glue, Amazon Athena, and supporting AWS services into a modular recruitment data pipeline. A traditional ATS model and an ML-based model were implemented to combine lexical similarity, information retrieval, semantic similarity, skills, and experience.

The project demonstrates the feasibility of using serverless cloud services and machine-learning techniques for automated job recommendation. Future work can improve the system through larger labeled datasets, more robust resume and job-description parsing, learning-to-rank models, multilingual embeddings, explainable recommendation features, and continuous model evaluation.

References

- [1] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," Proceedings of ECML, 1998.
- [2] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.
- [3] T. Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality," Advances in Neural Information Processing Systems, 2013.
- [4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proceedings of EMNLP-IJCNLP, 2019.
- [5] Amazon Web Services, "Amazon S3 User Guide," AWS Documentation. [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>
- [6] Amazon Web Services, "AWS Lambda Developer Guide," AWS Documentation. [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>
- [7] Amazon Web Services, "AWS Glue User Guide," AWS Documentation. [Online]. Available: <https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html>
- [8] Amazon Web Services, "Amazon Athena User Guide," AWS Documentation. [Online]. Available: <https://docs.aws.amazon.com/athena/latest/ug/what-is.html>
- [9] Amazon Web Services, "Amazon SES Developer Guide," AWS Documentation. [Online]. Available: <https://docs.aws.amazon.com/ses/latest/dg/Welcome.html>