TF-IDF vs. LLMs for Bidirectional Resume and Job Matching

Shang Andrews II
DePaul University

Sue Yang
DePaul University

***Abstract*—This study evaluates two prominent information retrieval techniques, TF-IDF and semantic embeddings, for matching job descriptions to resumes and vice versa. Specifically, our system supports bidirectional retrieval: using resumes to find relevant jobs and using job descriptions to identify suitable resumes. By computing cosine similarity scores between job postings and resumes, the effectiveness of each model is analyzed through average similarity metrics and retrieval overlap. The retrieval accuracy is further tested through manual evaluation. The results reveal key differences in their performance and offer insights into the relative strengths of lexical and semantic matching approaches.**

***Keywords*—Information Retrieval, Cosine Similarity, Semantic Embedding, TF-IDF, LLM, Job Resume Match**

I. Introduction

In this study, the objective is to create an effective information retrieval system that supports two retrieval directions: finding relevant job postings given a resume, and retrieving suitable resumes for a given job description. The system should be beneficial for job seekers and recruitment professionals. The project aims to explore LLMs' capability in generating vector representations and compare this new approach with the traditional approach that focuses on lexical overlap. Term Frequency-Inverse Document Frequency (TF-IDF) can capture the important terms in documents. LLM-based semantic embeddings can capture contextual meaning. Those two methods are implemented in this retrieval system. This bidirectional approach allows us to evaluate both candidate- and recruiter-facing use cases.

The following sections include the research background, methodology, implementation of the experiment, data sets, evaluation results and conclusions. We compare two approaches in terms of their retrieval overlap and accuracy, which is manually evaluated based on selected resumes. The result prefers the semantic embedding approach for similarity matching between job postings and resumes. Finally, we explore the limitations of the project and provide further study directions.

II. Background

Matching job descriptions with relevant resumes is a critical task for both job seekers and recruitment professionals. Automated systems that can effectively rank job postings based on a resume help reduce manual job searching, and the automation of ranking highly matched resumes to job postings improves hiring efficiency. Traditional searching and retrieving techniques in this domain focus on keyword matching. However, with the increasing popularity of Language Models (LLMs), many studies have shifted their focus to semantical approaches. For example, the Sentence Transformers models provide faster inference and have better semantic representations. The resulting vectors can be compared using cosine similarity, clustered, or indexed for retrieval [1].

In addition to generating vector representations, researchers have experienced leveraging LLMs for tasks such as resume pre-processing and resume personalization. [3]. The generative capabilities of LLMs can assist in query expansion, enhancing information retrieval. However, it is not without limitations. LLMs have been associated with high computational cost and inability to be domain-specific without extensive fine-tuning. Researchers have attempted in resolving this issue by combining LLMs with vector databases. [2]

III. Methodology

This project mainly compares two text retrieval techniques:
TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used statistical technique in information retrieval and natural language processing that quantifies the importance of a word within a specific document relative to a corpus. The method is based on two key components: term frequency (TF), which measures how frequently a term appears in a document, and inverse document frequency (IDF), which downscales the weight of terms that appear frequently across many documents. By combining these two components, TF-IDF assigns higher weights to terms that are distinctive to a particular document. In practice, each document is transformed into a high-dimensional, sparse vector where each dimension corresponds to a vocabulary term, and the vector entries represent the TF-IDF weight of those terms. In the context of job and resume matching, both resumes and job descriptions can act as queries or documents depending on the retrieval direction. Each text is tokenized and converted into a high-dimensional sparse vector. The TF-IDF score reflects how important a term (e.g., ”Python”, ”Machine Learning”) is within a specific document while reducing the weight of common, non-discriminative terms that appear frequently across all

documents (e.g., ”work”, ”team”). This representation enables efficient lexical comparison between resumes and job postings based on the weighted frequency of shared terms.

Semantic embeddings, in contrast, provide dense vector representations of text that capture contextual and semantic information. These embeddings are typically generated using pre-trained transformer-based language models. In this project, we adopt the sentence transformers model `all-MiniLM-L6-v2`. Unlike TF-IDF, which relies on raw term frequency statistics, semantic embeddings encode richer linguistic features by leveraging large-scale language modeling objectives. As a result, they are capable of capturing relationships between words and phrases that do not share explicit lexical overlap, allowing for more nuanced similarity assessments. For instance, a resume stating ”developed data pipelines using Python and Spark” and a job posting mentioning ”experience with ETL processes” may still yield high similarity even with limited exact term matches, given their shared semantic content.

Both TF-IDF and semantic embedding approaches have their merits. TF-IDF excels at identifying exact term matches critical for keyword-based filtering, while semantic embeddings improve recall by capturing contextual and conceptual similarities essential for deeper resume-job relevance assessment.

For each retrieval direction, we embed the query document (either resume or job description) and compare it against the opposing corpus. This allows us to evaluate both job-seeking and hiring use cases using the same matching techniques. To measure the similarity between a job posting and a resume—both represented as vectors in either the sparse TF-IDF space or the dense semantic embedding space, we utilize distance metrics.

Cosine similarity is used to measure matches generated by TF-IDF and semantic embeding model. Cosine similarity computes the cosine of the angle between two vectors in an inner product space and outputs a score ranging from -1 to 1. A value of 1 indicates that the two vectors are perfectly aligned (i.e., highly similar), while a value close to 0 suggests orthogonality (i.e., low similarity), and -1 indicates diametrically opposed vectors. By focusing on the orientation of the vectors rather than their magnitude, cosine similarity ensures that documents of different lengths (e.g., a brief job description versus a detailed resume) can still be meaningfully compared.

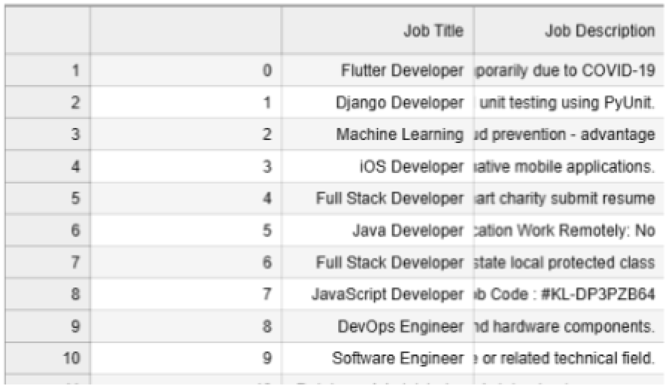
There are other popular distance measurements such as dot product, Euclidean, and Manhattan distance metrics. Dot product is often used to evaluate BERT-based models. Unlike cosine similarity, which normalizes vectors to compare angles, the dot product is a raw similarity score that takes into account both direction and magnitude. However, if the model is trained for cosine similarity, using the dot product for evaluation may loss performance. Euclidean and Manhattan distance measurements are not good options since distance is not a meaningful indicator of semantic similarity in hyper dimensional space. Given this reason, we chose cosine similarity for matching and retrieval.

IV. Data

The datasets used in this project include:

Job Title and Job Description Dataset. The dataset from Kaggle.com is sourced from reputable job listing platforms such as Glassdoor and Indeed. It mostly represents tech job roles such as developers and network administrators. The file is in CSV format with two columns, Job Title and Job Description. The Job Description outlines responsibilities and qualifications. There are 2276 records in this dataset after removing duplicates. The Job Title column is used as a label for pre-selecting relevant jobs. The Job Descriptive column is used for vectorization in this project.

Example of the Job Dataset:



		Job Title	Job Description
1	0	Flutter Developer	porarily due to COVID-19
2	1	Django Developer	unit testing using PyUnit.
3	2	Machine Learning	d prevention - advantage
4	3	iOS Developer	ative mobile applications.
5	4	Full Stack Developer	art charity submit resume
6	5	Java Developer	cation Work Remotely: No
7	6	Full Stack Developer	state local protected class
8	7	JavaScript Developer	b Code : #KL-DP3PZB64
9	8	DevOps Engineer	d hardware components.
10	9	Software Engineer	or related technical field.

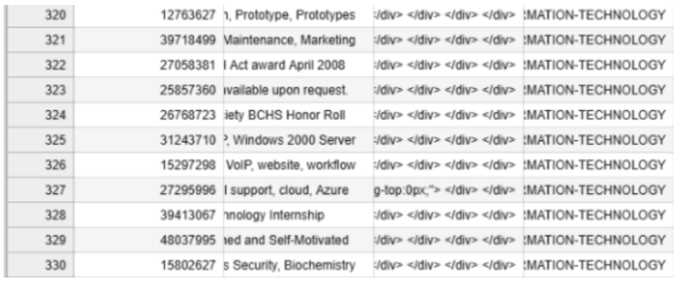
Fig. 1. Job Dataset

Resume Dataset. This dataset from Kaggle.com is a collection of Resume Examples taken from livecareer.com. It includes over 2400 resumes in varying categories. The CSV file contains 4 columns.

- ID: Unique identifier and file name for the respective pdf.
- Resume_str : Contains the resume text only in string format.
- Resume_html : Contains the resume data in html format as present while web scraping.
- Category : Category of the job the resume was used to apply.

We use ID and Resume_str column for the retrieval system. We filter the dataset by the category to only include the ”Information-Technology” label because our Job descriptions only pertain to technology-related roles. A total of 1200 records are included in the filtered dataset.

Example of resume Dataset:



320	12763627	ı, Prototype, Prototypes	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
321	39718499	Maintenance, Marketing	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
322	27058381	l Act award April 2008	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
323	25857360	ıvailable upon request.	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
324	26768723	iety BCHS Honor Roll	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
325	31243710	?, Windows 2000 Server	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
326	15297298	VoIP, website, workflow	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
327	27295996	l support, cloud, Azure	g-top:0px;"> </div> </div>	:MATION-TECHNOLOGY
328	39413067	ınology Internship	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
329	48037995	ıed and Self-Motivated	:/div> </div> </div> </div>	:MATION-TECHNOLOGY
330	15802627	s Security, Biochemistry	:/div> </div> </div> </div>	:MATION-TECHNOLOGY

Fig. 2. Job Dataset

Authentic Resumes. In addition to Kaggle datasets, we also used additional real resumes from the authors to evaluate the difference in similarity metrics for the embedding models and to compare the precisions of the TF-IDF and semantic embedding models. The reason for using real resumes is to produce practical relevance based on the authors' personal preferences and job search goals. Along with using resumes to search for jobs, we also support the inverse direction—using job descriptions as queries to retrieve semantically aligned candidate resumes. This bidirectional retrieval design enables comprehensive evaluation of both directions.

The resumes are one page each in PDF format:

- Shang Andrews Resume.pdf
- Sue Yang Resume.pdf

Shang has working experience in DevOps and is interested in DevOps software engineering roles. Sue has data analytics experience and is interested in data science roles.

V. Implementation

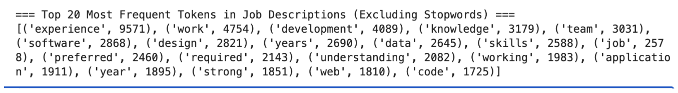
Preprocessing

The project is implemented using Python in Jupyter notebook, where we first load and process both resume and job description datasets into data frames using the pandas library. Pandas operations are applied to rename columns and filter resume datasets to include only tech-related roles. Text in all processed datasets are converted to lowercase for consistency, and white space is stripped before further processing.

We used the PyMuPDF library for data extraction from PDF to process the authentic resumes in PDF format.

TF- IDF Model

We applied the stop-word remover and tokenization function to the pre-processed text and build inverted index that has each index term as rows along with their total frequency and document frequency value. We checked the top 20 most frequent tokens for job descriptions as follows.



```
=== Top 20 Most Frequent Tokens in Job Descriptions (Excluding Stopwords) ===
[('experience', 9571), ('work', 4754), ('development', 4089), ('knowledge', 3179), ('team', 3031),
('software', 2868), ('design', 2821), ('years', 2690), ('data', 2645), ('skills', 2588), ('job', 257
8), ('preferred', 2460), ('required', 2143), ('understanding', 2082), ('working', 1983), ('applicatio
n', 1911), ('year', 1895), ('strong', 1851), ('web', 1810), ('code', 1725)]
```

Fig. 3. Top 20 Most Frequent Tokens.

We created a function compute_tfidf_scores to implement TF-IDF-based cosine similarity for information retrieval. Given a search query, and an inverted index of the documents, it returns the top k most relevant documents, ranked by cosine similarity between the query and each document. Optionally, scikit-learn's built-in tools can be easily implemented to perform the same task.

To enable user interaction, we created a function match_pdf_resume_tfidf to use the existing TF-IDF scoring system to retrieve the top k job matches for a user-uploaded PDF resume. We will use this function to generate search results for authentic resumes. We also implemented functions match$_{j}ob_{t}o_{r}esumes_{t}fidf()andmatch_{j}ob_{t}o_{r}esumes_{l}lm()toallowjobdes$
$kresumesfromthecandidatedataset, enablingrecruiter -$
$facingsearchfunctionalitywithinthesamesystem.$

To save the search results, we created a function to retrieve the top matching job documents and save the results to a CSV file. The results include the similarity scores and will be used to calculate the average similarity score for all queries.

The overview of the process is as follows.

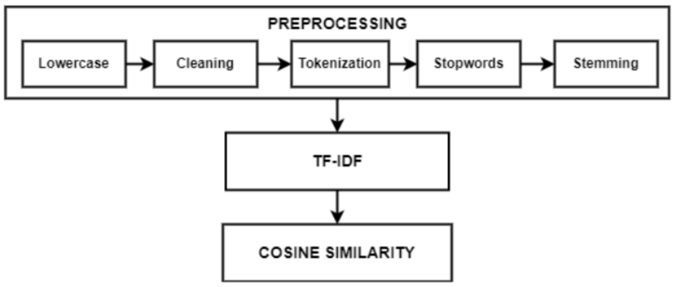


Fig. 4. TF-IDF Process

Semantic Embedding Model

We use a pre-trained sentence embedding model, all_MiniLM_L6_v2 from the sentence-transformers library, to embed text and compute semantic similarity. It is a compact, yet powerful transformer model pretrained for semantic similarity tasks. It has 6 transformer layers, making it lightweight and fast. The model converts the query or document into a dense vector that captures semantic meaning.

Using the pre-trained model, we created match_pdf_resume_llm function to return the top k job matches for a user-uploaded PDF resume, same to the TF-IDF function retrieval function.

The overview of the LLM retrieval process is as follows.

Similarity Measurement

In information retrieval (IR) systems, the ability to accurately measure the similarity between a query and a document is critical for ranking relevance. In both classical TF-IDF–based retrieval and modern embedding-based semantic

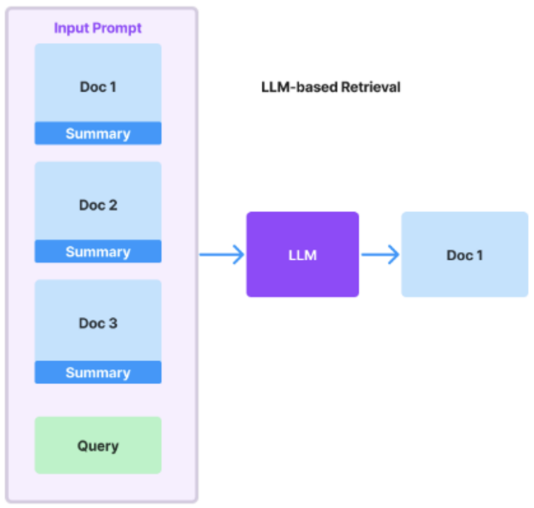


Fig. 5. LLM Process

search, cosine similarity is one of the most widely adopted metrics due to its effectiveness, efficiency, and geometric interpretability.

For TF-IDF based retrieval, documents are represented as high-dimensional sparse vectors where each dimension corresponds to a term's importance in the document. This project implements our own cosine similarity calculation for TF-IDF model; however, it can be easily substituted with scikit-learn functions. For the semantic model based retrieval, we use the cosine similarity function from the sentence- transformers library for the calculation.

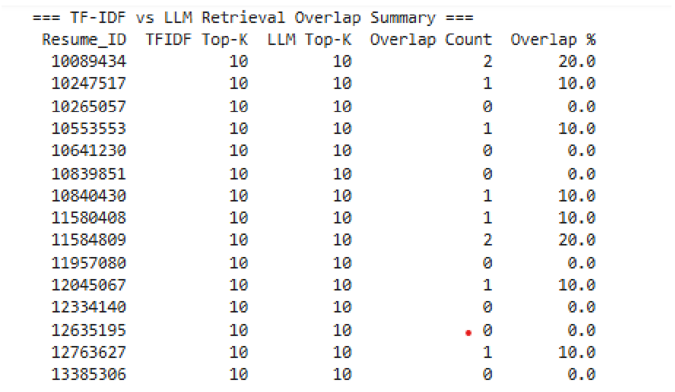
Metrics for Evaluation

- **Average Similarity Scores (Resume-to-Job and Job-to-Resume)**
  We use resumes as queries and return the top 10 most similar job postings for each resume. We then compute the average similarity scores for both TF-IDF and a semantic embedding model (LLM-based). TF-IDF is grounded in surface-level token matching, while LLM embeddings rely on contextual similarity and semantic understanding. This metric allows us to assess how strongly each model aligns resumes with job descriptions on average. We compute these scores in both directions: resumes retrieving jobs, and job descriptions retrieving resumes.
- **Overlap @K**
  Because TF-IDF and LLM produce scores on different scales, comparing raw values is not meaningful. Instead, we evaluate Top-K overlap: how many of the top 10 results from each model agree. This shows how often both retrieval systems arrive at the same job recommendations and offers insight into model convergence and reliability.
- **Precision @K**
  To directly assess relevance, we used three authentic resumes (Shang, Sue, RDH) and hand-labeled job titles that would be considered good matches. We then calculated precision based on whether relevant jobs appeared in the top 10 returned by each model. This provides grounded insight into practical utility for real users.

VI. Result

- **Average Similarity Score Comparison**

LLM-based embeddings significantly outperformed TF-IDF in terms of average similarity score across all resumes. While TF-IDF achieved an average of 0.3402, the LLM model reached 0.5942—nearly 75 percent higher. This gap supports the claim from Ghali et al. (2025) that generative retrieval models improve matching by leveraging contextual understanding. While this figure highlights resume-to-job matching, job-to-resume retrieval showed similarly strong performance trends for LLM-based methods, indicating bidirectional robustness.



```
=== TF-IDF vs LLM Retrieval Overlap Summary ===
 Resume_ID  TFIDF Top-K  LLM Top-K  Overlap Count  Overlap %
  10089434           10         10              2       20.0
  10247517           10         10              1       10.0
  10265057           10         10              0        0.0
  10553553           10         10              1       10.0
  10641230           10         10              0        0.0
  10839851           10         10              0        0.0
  10840430           10         10              1       10.0
  11580408           10         10              1       10.0
  11584809           10         10              2       20.0
  11957080           10         10              0        0.0
  12045067           10         10              1       10.0
  12334140           10         10              0        0.0
  12635195           10         10              0        0.0
  12763627           10         10              1       10.0
  13385306           10         10              0        0.0
```

Fig. 6. Top-K Overlap Between TF-IDF and LLM Job Matches (K=10)

- **Job Matches for Resume ID: 36856210**

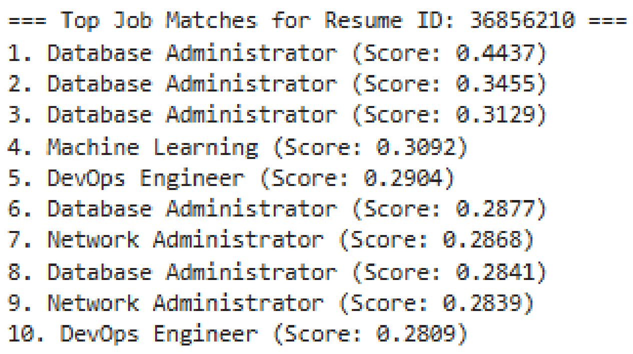
The job matches for Resume ID 36856210 provide a striking contrast between methods. TF-IDF returns a variety of roles—Database Administrator, DevOps Engineer, Network Administrator—without strong consistency. LLM, however, focuses sharply on Network Administrator, suggesting it identified a strong role alignment based on deeper context.

- **Resume Matches for Job #0 (Flutter Developer)**

For the Flutter Developer role, the top-matching resumes differ significantly across methods. TF-IDF prioritizes resume IDs with surface overlaps like ”Flutter” or “developer,” while LLM identifies resumes with relevant mobile development experience, even if Flutter is not explicitly mentioned.

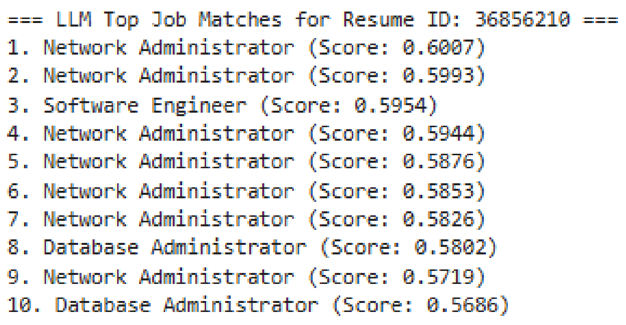
- **Top-K Overlap**

The overlap between models was relatively low, with an average Top-K intersection of 8.75 percent. For many resumes, TF-IDF and LLM returned completely different sets of jobs. This reinforces findings from the GTR-T paper, where combining semantic search with vector databases revealed that overlap-based evaluation is more reliable than raw score comparison.



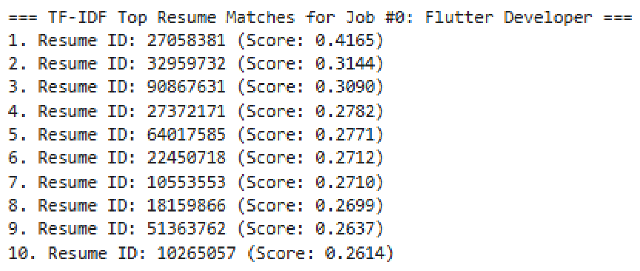
```
=== Top Job Matches for Resume ID: 36856210 ===
1. Database Administrator (Score: 0.4437)
2. Database Administrator (Score: 0.3455)
3. Database Administrator (Score: 0.3129)
4. Machine Learning (Score: 0.3092)
5. DevOps Engineer (Score: 0.2904)
6. Database Administrator (Score: 0.2877)
7. Network Administrator (Score: 0.2868)
8. Database Administrator (Score: 0.2841)
9. Network Administrator (Score: 0.2839)
10. DevOps Engineer (Score: 0.2809)
```

Fig. 7. TF-IDF Job Matches for Resume ID: 36856210



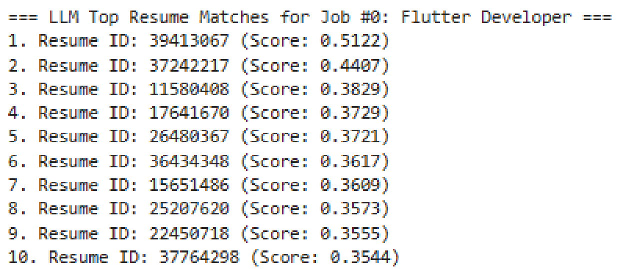
```
=== LLM Top Job Matches for Resume ID: 36856210 ===
1. Network Administrator (Score: 0.6007)
2. Network Administrator (Score: 0.5993)
3. Software Engineer (Score: 0.5954)
4. Network Administrator (Score: 0.5944)
5. Network Administrator (Score: 0.5876)
6. Network Administrator (Score: 0.5853)
7. Network Administrator (Score: 0.5826)
8. Database Administrator (Score: 0.5802)
9. Network Administrator (Score: 0.5719)
10. Database Administrator (Score: 0.5686)
```

Fig. 8. LLM Job Matches for Resume ID: 36856210



```
=== TF-IDF Top Resume Matches for Job #0: Flutter Developer ===
1. Resume ID: 27058381 (Score: 0.4165)
2. Resume ID: 32959732 (Score: 0.3144)
3. Resume ID: 90867631 (Score: 0.3090)
4. Resume ID: 27372171 (Score: 0.2782)
5. Resume ID: 64017585 (Score: 0.2771)
6. Resume ID: 22450718 (Score: 0.2712)
7. Resume ID: 10553553 (Score: 0.2710)
8. Resume ID: 18159866 (Score: 0.2699)
9. Resume ID: 51363762 (Score: 0.2637)
10. Resume ID: 10265057 (Score: 0.2614)
```

Fig. 9. TF-IDF Resume Matches for Job #0 (Flutter Developer)



```
=== LLM Top Resume Matches for Job #0: Flutter Developer ===
1. Resume ID: 39413067 (Score: 0.5122)
2. Resume ID: 37242217 (Score: 0.4407)
3. Resume ID: 11580408 (Score: 0.3829)
4. Resume ID: 17641670 (Score: 0.3729)
5. Resume ID: 26480367 (Score: 0.3721)
6. Resume ID: 36434348 (Score: 0.3617)
7. Resume ID: 15651486 (Score: 0.3609)
8. Resume ID: 25207620 (Score: 0.3573)
9. Resume ID: 22450718 (Score: 0.3555)
10. Resume ID: 37764298 (Score: 0.3544)
```

Fig. 10. LLM Resume Matches for Job #0 (Flutter Developer)

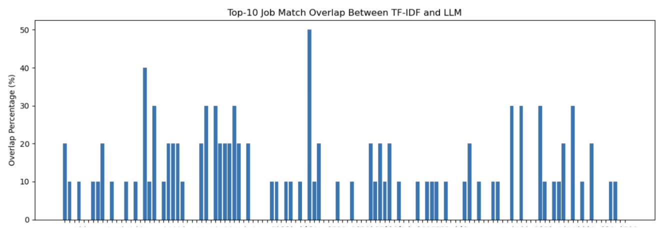
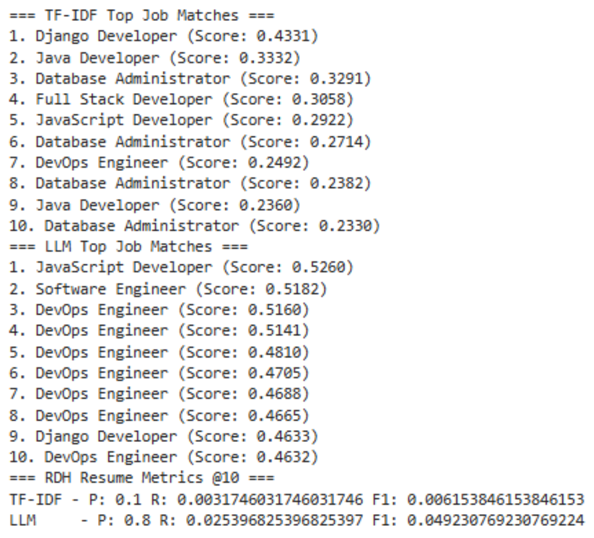


Fig. 11. TF-IDF vs LLM: Resume-Job Match Overlap at K=10

- **Precision @K – Real Resume Evaluation**

We selected three resumes and manually labeled relevant job types. For RDH, the LLM achieved a precision of 0.8, while TF-IDF scored just 0.1—highlighting the latter's struggle with semantic nuance.



```
=== TF-IDF Top Job Matches ===
1. Django Developer (Score: 0.4331)
2. Java Developer (Score: 0.3332)
3. Database Administrator (Score: 0.3291)
4. Full Stack Developer (Score: 0.3058)
5. JavaScript Developer (Score: 0.2922)
6. Database Administrator (Score: 0.2714)
7. DevOps Engineer (Score: 0.2492)
8. Database Administrator (Score: 0.2382)
9. Java Developer (Score: 0.2360)
10. Database Administrator (Score: 0.2330)
=== LLM Top Job Matches ===
1. JavaScript Developer (Score: 0.5260)
2. Software Engineer (Score: 0.5182)
3. DevOps Engineer (Score: 0.5160)
4. DevOps Engineer (Score: 0.5141)
5. DevOps Engineer (Score: 0.4810)
6. DevOps Engineer (Score: 0.4705)
7. DevOps Engineer (Score: 0.4688)
8. DevOps Engineer (Score: 0.4665)
9. Django Developer (Score: 0.4633)
10. DevOps Engineer (Score: 0.4632)
=== RDH Resume Metrics @10 ===
TF-IDF - P: 0.1 R: 0.0031746031746031746 F1: 0.006153846153846153
LLM    - P: 0.8 R: 0.025396825396825397 F1: 0.04923076923076924
```

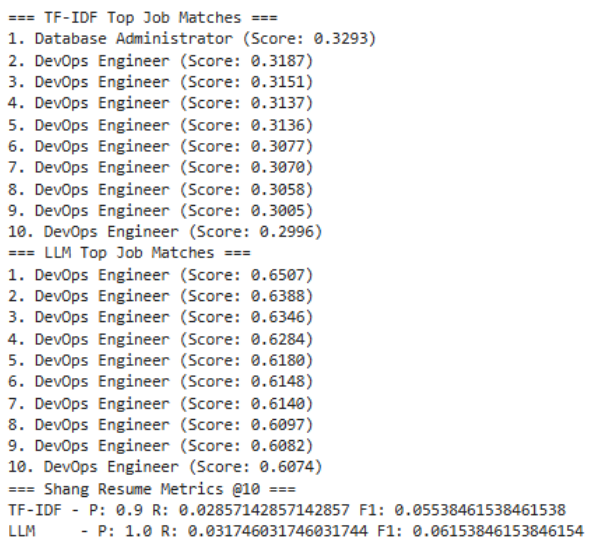
Fig. 12. Evaluation Metrics for RDH Resume

For Shang, relevant roles included DevOps and Software Engineer. TF-IDF precision was 0.9, while LLM reached 1.0 with a more diverse yet accurate top-10 list. This case shows that TF-IDF can perform well in narrow skill matches but lacks consistency across broader semantics.

Sue's resume, with broader phrasing and more generalized experience, demonstrated the LLM model's ability to capture implicit relevance. LLM achieved perfect precision and F1 score, while TF-IDF fell short at 0.5.

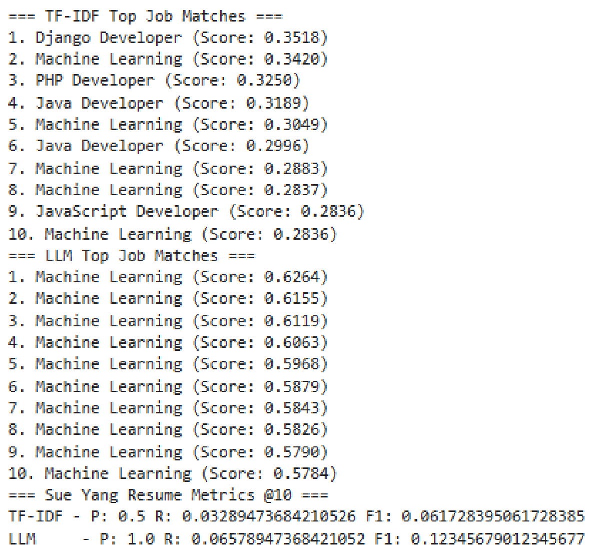
- **Inverted Index Summary**

We also explored term frequency from the inverted index to understand common patterns across resumes and job descriptions. ”Experience”, ”development”, and ”skills” were dominant—terms that TF-IDF heavily favors but may cause bias toward generic or senior-level roles. This connects to fairness concerns in the “Writing Style Matters” paper (WSDM 2025), which showed how lexical models penalize non-traditional phrasing.



```
=== TF-IDF Top Job Matches ===
1. Database Administrator (Score: 0.3293)
2. DevOps Engineer (Score: 0.3187)
3. DevOps Engineer (Score: 0.3151)
4. DevOps Engineer (Score: 0.3137)
5. DevOps Engineer (Score: 0.3136)
6. DevOps Engineer (Score: 0.3077)
7. DevOps Engineer (Score: 0.3070)
8. DevOps Engineer (Score: 0.3058)
9. DevOps Engineer (Score: 0.3005)
10. DevOps Engineer (Score: 0.2996)
=== LLM Top Job Matches ===
1. DevOps Engineer (Score: 0.6507)
2. DevOps Engineer (Score: 0.6388)
3. DevOps Engineer (Score: 0.6346)
4. DevOps Engineer (Score: 0.6284)
5. DevOps Engineer (Score: 0.6180)
6. DevOps Engineer (Score: 0.6148)
7. DevOps Engineer (Score: 0.6140)
8. DevOps Engineer (Score: 0.6097)
9. DevOps Engineer (Score: 0.6082)
10. DevOps Engineer (Score: 0.6074)
=== Shang Resume Metrics @10 ===
TF-IDF - P: 0.9 R: 0.02857142857142857 F1: 0.05538461538461538
LLM    - P: 1.0 R: 0.031746031746031744 F1: 0.06153846153846154
```

Fig. 13. Evaluation Metrics for Shang Andrews' Resume



```
=== TF-IDF Top Job Matches ===
1. Django Developer (Score: 0.3518)
2. Machine Learning (Score: 0.3420)
3. PHP Developer (Score: 0.3250)
4. Java Developer (Score: 0.3189)
5. Machine Learning (Score: 0.3049)
6. Java Developer (Score: 0.2996)
7. Machine Learning (Score: 0.2883)
8. Machine Learning (Score: 0.2837)
9. JavaScript Developer (Score: 0.2836)
10. Machine Learning (Score: 0.2836)
=== LLM Top Job Matches ===
1. Machine Learning (Score: 0.6264)
2. Machine Learning (Score: 0.6155)
3. Machine Learning (Score: 0.6119)
4. Machine Learning (Score: 0.6063)
5. Machine Learning (Score: 0.5968)
6. Machine Learning (Score: 0.5879)
7. Machine Learning (Score: 0.5843)
8. Machine Learning (Score: 0.5826)
9. Machine Learning (Score: 0.5790)
10. Machine Learning (Score: 0.5784)
=== Sue Yang Resume Metrics @10 ===
TF-IDF - P: 0.5 R: 0.03289473684210526 F1: 0.061728395061728385
LLM    - P: 1.0 R: 0.06578947368421052 F1: 0.12345679012345677
```

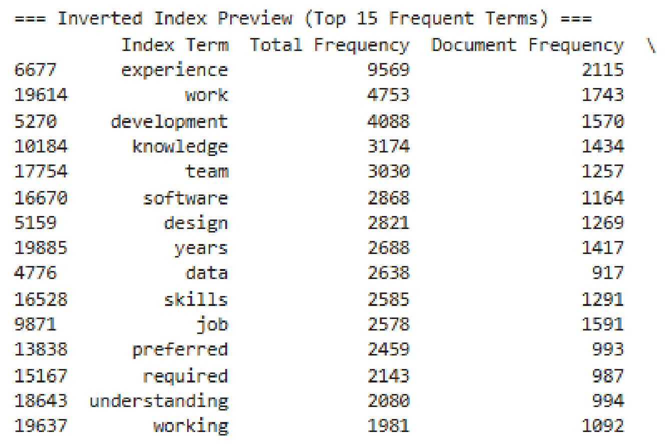
Fig. 14. Evaluation Metrics for Sue Yang Resume

VII. Discussion

The results consistently demonstrated the superiority of LLM-based semantic embeddings over traditional TF-IDF in both average similarity and precision metrics. While TF-IDF occasionally performed well for highly keyword-specific resumes (e.g., Shang's DevOps resume), it struggled with generalization and semantic nuance, as seen with RDH and Sue's resumes.

One factor influencing performance was tokenization. TF-IDF heavily relied on token frequency, and its effectiveness diminished with varied phrasing. On the other hand, the LLM embedding model (all-MiniLM-L6-v2) effectively captured deeper relationships without exact word overlap.

The relatively low Top-K overlap (8.75 percent) suggests that the two models retrieve qualitatively different results, emphasizing that raw similarity scores are not interchangeable. This confirms findings in recent work like GTR-T, which promotes ranking evaluation based on result alignment rather than vector distance alone.



```
=== Inverted Index Preview (Top 15 Frequent Terms) ===
          Index Term  Total Frequency  Document Frequency  \
6677      experience             9569                2115
19614           work             4753                1743
5270     development             4088                1570
10184      knowledge             3174                1434
17754           team             3030                1257
16670       software             2868                1164
5159          design             2821                1269
19885          years             2688                1417
4776            data             2638                 917
16528         skills             2585                1291
9871             job             2578                1591
13838      preferred             2459                 993
15167       required             2143                 987
18643  understanding             2080                 994
19637        working             1981                1092
```

Fig. 15. Top 15 Most Frequent Terms in Resume and Job Corpus

Hyperparameters such as Top-K value (K=10), vector dimensionality, and resume filtering by job category ("Information-Technology") were critical to system performance. Experiments showed that including broader resume categories significantly diluted retrieval relevance.

Fairness concerns also emerged. TF-IDF favored generic corporate phrasing (e.g., "experience", "development") due to term frequency, introducing bias toward conventional writing styles. This aligns with the WSDM 2025 paper "Writing Style Matters," where resumes with an informal tone or creative expression ranked lower under lexical models.

Future improvements include testing different transformer models (e.g., BGE, E5-Large), integrating reranking strategies, or combining lexical and semantic scores for ensemble matching. Expanding evaluation to more annotated real-world resumes would improve result generalization and fairness metrics.

VIII. Conclusion

While TF-IDF provides a fast and interpretable baseline, semantic embeddings offer richer representations and improved retrieval quality. However, the two models are not directly comparable by raw similarity scores due to their differing vector spaces and scales. Retrieval overlap analysis and normalized metrics are better suited for evaluation. The evaluation with authentic resumes supports the conclusion that using LLMs in an information retrieval system improves retrieval quality by introducing deeper semantic understanding.

Our bidirectional setup not only supports resume search but also enables job descriptions to serve as queries for identifying strong candidates. This flexibility makes the system applicable to both job seekers and hiring platforms. This project provides a baseline for further research on including LLMs in the resume-job match domain. Future studies could explore hybrid or ensemble methods for more robust job-to-resume matching systems. In particular, combining TF-IDF's efficiency with LLMs' semantic strength may yield a scalable yet context-aware matching pipeline. Expanding evaluation to more real-world annotated data and diverse applicant profiles would also improve model robustness and fairness assessments.

References

[1] Rubén Alonso, Danilo Dessí, Antonello Meloni, Diego Reforgiato Recupero, A novel approach for job matching and skill recommendation using transformers and the O*NET database, Big Data Research, Volume 39, 2025, 100509, ISSN 2214-5796, https://doi.org/10.1016/j.bdr.2025.100509.

[2] Mohammed-Khalil Ghali, Abdelrahman Farrag, Daehan Won, Yu Jin, Enhancing knowledge retrieval with in-context learning and semantic search through generative AI, Knowledge-Based Systems, Volume 311, 2025, 113047, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2025.113047.

[3] Zinjad, Saurabh and Bhattacharjee, Amrita and Bhilegaonkar, Amey and Liu, Huan. (2024). ResumeFlow: An LLM-facilitated Pipeline for Personalized Resume Generation and Refinement.

[4] Hongliu Cao. 2025. Writing Style Matters: An Examination of Bias and Fairness in Information Retrieval Systems. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25). Association for Computing Machinery, New York, NY, USA, 336–344. https://doi.org/10.1145/3701551.3703514