Group 85 Progress Report: CareerCompass Resume Title Normalization

Imran Chowdhury Karim Elbasiouni Rami Abu Sultan
Department of Computing and Software
McMaster University
{chowdi13,elbasik,abusultr}@mcmaster.ca

November 11, 2025

Abstract

CareerCompass is an end-to-end pipeline for converting raw resume text into normalized job titles to support program advising and labour market analytics. During the first half of the term we implemented a fully scripted data processing flow that anonymizes personally identifiable information, aggregates multi-section resumes, and exports reproducible train/validation/test splits. We then added a TF–IDF + LinearSVC baseline that meaningfully outperforms a majority-class heuristic while remaining interpretable and lightweight for deployment in advising dashboards. This progress report summarizes the dataset, preprocessing choices, model inputs, implementations, and preliminary evaluation results, and outlines how TA feedback is steering us toward transformer-based models, expanded label coverage, and richer error analysis in the final milestone.

1 Introduction

Academic advisors increasingly need structured views of student experience in order to recommend co-op placements and upskilling pathways. Manually reviewing unstructured resumes is slow, subjective, and does not scale beyond small cohorts. Our project reframes resume understanding as a resume-to-title classification task: ingest raw resumes, scrub PII, and map each document to a canonical job title and occupational family. The resulting labels can power dashboards that highlight dominant skill profiles and align students with industry demand. Building on our proposal, we committed to fully automating data preparation and establishing baselines that can be iterated upon for the final submission.

2 Related Work

Classical text categorization pipelines often pair TF–IDF features with linear Support Vector Machines, a combination that remains competitive on high-dimensional sparse problems [Joachims, 1998]. Term-weighting schemes such as TF–IDF have been studied extensively for document retrieval and classification [Salton and Buckley, 1988]. More recent resume and job-title normalization systems leverage distributional semantics and contextual encoders to capture skill synonyms and phrasing variation [Xu and Dredze, 2021]. Large language models such as BERT offer strong transfer learning capabilities for professional profile understanding [Devlin et al., 2019]. Data-centric approaches recommend aggressive text cleaning and PII masking when handling resumes to protect privacy while preserving downstream signal [Chowdhury, 2020]. Finally, real-world deployments often combine heuristic baselines with machine-learned models to guarantee minimum service

quality while explaining decisions to stakeholders [McCallum and Nigam, 1998]. These works guided our choice of baselines, preprocessing, and planned transformer experiments.

3 Dataset

We curated a 3,500-resume JSONL corpus sourced from the public "Resume Dataset" collection [Dodia and Pawar, 2020]. Each entry includes identifiers, contact fields, free-form narrative sections (Summary, Skills, Experience, Education), and a noisy category label derived from the job title on the document. There are 36 normalized titles, with "Java Developer," "Data Science," and "Python Developer" each covering roughly 200 resumes, while the long tail contains specialized roles such as "Network Security Engineer." The dataset is balanced enough for stratified evaluation but still exhibits class imbalance, motivating macro-averaged metrics.

Preprocessing follows `build_clean_dataset.py`. We first drop rows missing all major text fields, retaining the full 3,500 resumes. We then scrub emails and phone numbers with deterministic regular expressions, join the major sections into a single newline-delimited blob, and derive 'text_norm' by stripping whitespace. Label creation normalizes raw categories to a curated set of canonical job titles and assigns broader occupational families (e.g., "Computers / IT" vs. "Business / Finance"). Finally, we guarantee a stable "resumeid" for downstream joins and persist both Parquet and CSV outputs (Parquet generation requires the optional `pyarrow` dependency, which we install inside a project virtual environment on development machines). The cleaning step reduces vocabulary noise and aligns noisy titles before modeling.

4 Features

Our primary feature representation is a TF–IDF bag-of-ngrams vector built with `src/features/tfidf_build.py`. We retain unigrams and bigrams, cap document frequency at 0.9 to downweight cross-cutting templates, and discard tokens appearing in fewer than three resumes. Text is accent-stripped to unify characters across international CVs. The result is a sparse matrix with roughly 75,000 active features. We additionally persist an index that tracks resume IDs and split membership for reproducible training. Although we do not yet employ contextual embeddings, we plan to extend this stage with domain-adapted transformer encoders (e.g., fine-tuned BERT) and skill-level entity features, as suggested by related work [Devlin et al., 2019, Xu and Dredze, 2021].

5 Implementation

The code base is intentionally modular. Configuration centralizes filesystem paths and ensures required directories exist at import time. Data ingestion (`src/load_dataset.py`) prints schema diagnostics on demand, while `src/data_filter.py` enforces the "at least one major section" rule. Text cleaning and label creation live in dedicated modules to encourage reuse in future experiments. The dataset builder orchestrates these pieces, writing parquet and CSV artifacts for other teams.

Modeling scripts live under `src/models/`. We implemented two baselines:

1. **Majority vote**: Always predict the most frequent training label. This establishes the accuracy floor and provides a sanity check on label extraction.

2. **LinearSVC**: Train a class-weighted Linear Support Vector Machine on TF–IDF features with optional C tuning over $\{0.25, 0.5, 1.0, 2.0\}$. The script automatically reuses splits, persists the trained model and label encoder, and generates confusion-matrix plots.

Table 1: Preliminary validation and test metrics.

Model	Split	Acc.	Macro-F1
Majority	Val	0.057	0.011
Majority	Test	0.057	0.010
LinearSVC	Val	0.824	0.812
LinearSVC	Test	0.808	0.798

During local development we encountered the macOS system-Python PEP 668 restriction, which prevented installing optional packages globally. We resolved this by provisioning a project-specific virtual environment on a TA-managed Linux host; all results reported here come from that environment. The repository also includes evaluation utilities for accuracy, macro-F1, top-k accuracy, and confusion visualization, alongside a `run_check` helper for data sanity checks. While the dataset builder already emits a `y_family` column, the current release focuses on job-title prediction; the multi-task head that jointly predicts occupation families is partially implemented and scheduled for Milestone 3 together with the transformer baseline.

6 Results and Evaluation

We adopt a 80/10/10 stratified train/validation/test split generated with a fixed random seed to ensure reproducibility. Evaluation reports accuracy, macro-F1, and top-k hits (for $k \in \{1, 3\}$) using the provided utilities.

Table 1 summarizes the baselines. The majority-class predictor performs as expected, underscoring the need for stronger models. The LinearSVC baseline delivers an order-of-magnitude improvement, achieving 0.824 validation accuracy (macro-F1 0.812) and 0.808 test accuracy (macro-F1 0.798). Top-3 accuracy exceeds 0.90 (not shown), indicating that most misclassifications fall within a short list of plausible titles. Qualitative error analysis reveals confusion between adjacent software roles (e.g., "Python Developer" vs. "Data Science") and between DevOps and Cloud Engineering categories, suggesting benefits from incorporating skill embeddings or hierarchical labels.

7 Feedback and Plans

Our TA encouraged us to (1) tighten privacy guarantees by hashing resume IDs before sharing artifacts, (2) pursue a contextual baseline (e.g., fine-tuned BERT or a domain-specific sentence transformer) to benchmark against modern approaches, and (3) expand the evaluation protocol with calibration plots and qualitative case studies for stakeholder communication. In response, we are:

- Refactoring the artifact writer to optionally hash identifiers and store a lookup table in a secure directory.
- Extending the feature pipeline to cache raw text for transformer fine-tuning via HuggingFace, with early experiments scheduled for the next sprint.
- Building an interactive error-analysis notebook that surfaces misclassifications and their top-k predictions for advisor review.

We also plan to explore class aggregation for sparse labels, apply light data augmentation (synonym replacement) to bolster minority classes, and integrate evaluation summaries into the advisory dashboard mock-up.

Team Contributions

Imran Chowdhury led the data ingestion and cleaning modules, implementing the PII scrubbing routines and label normalization tables. **Karim Elbasiouni** owned the feature engineering and modeling scripts, including TF–IDF generation, SVM training, and metrics logging. **Rami Abu Sultan** coordinated infrastructure, configured the project virtual environment, ran baseline experiments, and drafted this report alongside the literature review. All members participated in weekly design discussions and jointly reviewed TA feedback.

References

Gobinda Chowdhury. Ethics of using artificial intelligence in human resources. *Online Information Review*, 44(7):1413–1428, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Transactions of the Association for Computational Linguistics*, 7:417–433, 2019.

Nimisha Dodia and Aditya Pawar. Resume dataset. Kaggle dataset, 2020. URL `https://www.kaggle.com/datasets/nishitjain/resume-dataset`. https://www.kaggle.com/datasets/nishitjain/resume-dataset.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142. Springer, 1998.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

Zeyu Xu and Mark Dredze. Job title classification with contextual embeddings. In *Proceedings of the 4th Workshop on Natural Language Processing in HR and Recruitment*, pages 12–21, 2021.