Colorado Workforce Intelligence Platform

Technical Documentation & Methodology Report

GW Hackathon 2026 | Problem Statement 4 | National Labor Exchange (NLx)

10,482	**165,664**	**9**
Job Postings Analyzed	Skill Records Processed	NLP Techniques Implemented

Dataset: National Labor Exchange (NLx) Colorado, January 2026
Primary Files: colorado.csv (57 columns) | colorado_processed.csv (6 columns)
Deployment: Streamlit on Hugging Face Spaces | Docker containerized

Table of Contents

1. Problem Statement

1.1 Background

The National Labor Exchange (NLx), developed by DirectEmployers Association and the National Association of State Workforce Agencies (NASWA), is one of the largest aggregators of job postings in the United States. It ingests data from thousands of employers and state workforce systems, covering millions of postings annually across all sectors of the American economy.

Despite the scale and richness of this data, a fundamental challenge persists: the most valuable workforce intelligence is buried in unstructured text. Job titles, skills, education requirements, salary ranges, and experience expectations are written by individual recruiters in freeform prose. There is no enforced schema. The same skill appears as "Python", "Python 3", "Python programming", "proficiency in Python", and "Python (3+ years)" across different postings. Researchers and policymakers who need to answer the question "what does Colorado actually need from its workforce right now?" cannot do so reliably from raw job posting text alone.

1.2 Official Problem Statement

> **Problem Statement 4 GW Hackathon 2026**
>
> Teams are invited to develop tools or pipelines that transform unstructured job posting text into structured, analyzable labor market data. The National Labor Exchange (NLx) maintains a largescale dataset of job postings across the United States. However, much of this data exists in unstructured text formats, limiting policymakers' and researchers' ability to extract workforce trends, skill demand patterns, and economic insights.

The scope of acceptable submissions was broad, explicitly including:

- NLPbased skill extraction frameworks
- Cleaned and structured datasets
- Reusable Python libraries or processing pipelines
- Transparent documentation of modeling assumptions

All solutions were required to consider bias, data limitations, and methodological transparency not just technical performance.

1.3 Our Response

We interpreted this problem statement as having four distinct user populations who all need structured access to the same underlying data, but for fundamentally different purposes:

User	Pain Point	What Structured Data Enables
Job Seeker	Cannot find jobs matching skills without knowing exact titles	Skillvector matching across all postings simultaneously
Student	Degree decisions based on anecdote, not market data	Actual employer skill frequency and salary distribution by field
Veteran	Military skills have no civilian equivalent in job search systems	MOCtocivilian skill translation + cosine matching
Policymaker / Researcher	Cannot measure credential inflation, skill gaps, or ghost job prevalence at scale	NLPderived metrics on populationlevel posting patterns

Table 1.1 Four user populations served by this platform

2. Dataset Description

2.1 Source

The NLx Colorado dataset represents a snapshot of job postings active on or around January 2026. It was provided as two CSV files for the hackathon:

File	Rows	Columns	Description
colorado.csv	10,482	57	One row per job posting. Contains all employerauthored fields title, description, salary, education, experience, company name, location, and a set of structured classification columns populated by NLx automated processing.
colorado_processed.csv	165,664	6	One row per skill per job posting. The result of NLx's own NLP pipeline mapping raw employer skill text to official ESCO and O*NET taxonomy categories. Contains the key Correlation Coefficient column.

Table 2.1 Source files and their structure

2.2 Key Columns Used

From colorado.csv (selected columns)

Column	Description and Use
system_job_id	Primary key. Used to join both files on the Research ID in colorado_processed.csv.
title, description	Employerauthored text. Used for ghost job language analysis (TFIDF) and description quality scoring.
city, zipcode	Geographic filtering, citylevel salary analysis.
parameters_salary_min / max	Numeric salary range. Present in approximately 60% of postings.
requirements_min_education	Education requirement string. Normalized and used for credential inflation detection.
requirements_experience	Experience requirement string. Used for description quality scoring.
classifications_onet_code	O*NET occupation code. Primary grouping key for credential inflation analysis.
classifications_naics_code	NAICS industry code. First two digits used for sectorlevel adjacent career discovery.
ghostjob	Boolean. NLxlabeled flag indicating a potentially nonactive posting. Central to ghost job analysis.

Column	Description and Use
fedcontractor	Boolean. Whether the employer is a federal contractor subject to VEVRAA veteran hiring requirements.
rapids_codes	Registered Apprenticeship Program codes. Presence indicates an earnwhileyoutrain pathway exists.
moc_codes	Military Occupational Codes. Some employers tag their postings with these to signal veteran hiring intent.
cip_codes	Classification of Instructional Programs codes. Links postings to academic degree program categories.

Table 2.2 Key columns from colorado.csv

From colorado_processed.csv (all 6 columns)

Column	Description and Use
Research ID	Foreign key joining to system_job_id in colorado.csv.
Raw Skill	Exactly what the employer typed. E.g. 'Python 3', 'ML', 'machine learning experience'.
Taxonomy Skill	Official ESCO or O*NET category the raw skill was mapped to. E.g. 'machine learning'.
Taxonomy Description	Humanreadable definition of the taxonomy category.
Taxonomy Source	Whether the mapping came from ESCO or O*NET. Used for sourcelevel skill frequency breakdown.
Correlation Coefficient	NLx confidence score (0.0 to 1.0) for the rawtotaxonomy mapping. The key variable for emerging skills detection.

Table 2.3 All columns from colorado_processed.csv

2.3 The Correlation Coefficient Central Insight

The Correlation Coefficient in colorado_processed.csv is not a standard statistical correlation in this context. It is the NLx system's internal confidence score for how well a raw employerwritten skill maps to an official taxonomy category. A score of 1.0 means the raw text is an exact or nearexact match to a taxonomy entry. A score of 0.3 means the system had to make a significant inferential leap.

This score becomes the foundation of our Emerging Skills Detector: when a skill appears frequently across multiple employers but has consistently low Correlation Coefficient values, it means the taxonomy has no established category for it. The skill is genuinely newer than the taxonomy itself. This is a signal that official workforce training databases are missing active employer demand.

3. System Architecture

3.1 Technology Stack

Layer	Technology	Purpose
Frontend / UI	Streamlit 1.32+	Interactive web interface with 5 userspecific tabs
NLP / ML	scikitlearn (TFIDF, cosine similarity)	Core skill vectorization and matching
Data Processing	pandas 2.0+, NumPy	CSV loading, preprocessing, aggregation
Containerization	Docker (python:3.11slim)	Reproducible deployment on Hugging Face Spaces
Hosting	Hugging Face Spaces	Public access, persistent data files, Docker runner

Table 3.1 Technology stack

3.2 File Structure

The codebase was split into purposespecific modules after initial development in a single app.py. The final structure is:

Project File Tree

```
colorado-workforce/
│
├── app.py                      # Entry point: page config, header, tab scaffol
├── requirements.txt            # Python dependencies
├── Dockerfile                  # Container definition for Hugging Face Spaces
│
├── colorado.csv                # 10,482 job postings, 57 columns (NLx data)
├── colorado_processed.csv      # 165,664 skill records, 6 columns (NLx data)
│
├── .streamlit/
│   └── config.toml             # Theme configuration (navy/teal palette)
│
├── core/
│   ├── __init__.py
│   ├── data_loader.py          # CSV loading, preprocessing, feature
engineering
│   ├── nlp.py                  # All 9 NLP techniques + MOC dictionary
│   └── style.py                # CSS injection
│
└── tabs/
    ├── __init__.py
    ├── tab1_jobseeker.py       # Job Seeker interface
    ├── tab2_student.py         # Student interface
    ├── tab3_veteran.py         # Veteran interface
    ├── tab4_market.py          # Market Intelligence interface
    └── tab5_recruiter.py       # Recruiter Tools interface
```

3.3 Data Flow

On application startup, the following sequence runs exactly once and the results are cached by Streamlit for the duration of the session:

1. load_data() reads both CSV files, selects the 24 analytically relevant columns from colorado.csv, fills NaN values with empty strings, normalizes boolean flag columns (ghostjob, fedcontractor) from string/int representations to Python booleans, computes derived boolean columns (has_apprenticeship, has_cip, has_moc), converts salary columns to float, computes the twodigit NAICS sector prefix, and aggregates colorado_processed.csv into one skilltext string per job via groupby on Research ID.

2. build_vectorizer() fits a TFIDF vectorizer on the aggregated skilltext strings, producing a sparse matrix of shape (10,482 jobs × 15,000 features). This matrix stays in memory for the entire session.

3. Platformlevel statistics (total jobs, ghost count, federal count, apprenticeship count, MOC count) are computed from the loaded data and passed into each tab module.

4. All five tab modules receive the shared jobs_clean, skill_profiles, processed, vectorizer, and job_matrix objects as function arguments. No tab imports data directly this keeps caching clean and prevents redundant computation.

4. NLP Methodology

Nine distinct NLP and analytical techniques are implemented across the platform. They are documented here in order of architectural dependency techniques built upon earlier ones are described after the techniques they depend on.

4.1 Technique 1 TFIDF Vectorization

What it does

Converts each job's skill profile into a numeric vector in a shared 15,000dimensional space, enabling mathematical comparison between any user input and any job posting.

How it works

The 165,664 rows of colorado_processed.csv are first aggregated: all Taxonomy Skill values for a given job are joined into a single spaceseparated string. This becomes the "document" representing that job. A TFIDF vectorizer is then fitted on these 10,482 documents with the following parameters:

Parameter	Value	Rationale
stop_words	english	Removes articles, prepositions, and other nonskill words that would add noise to skill vectors
ngram_range	(1, 2)	Captures both single skills ('python') and compound skills ('machine learning', 'project management') as single features
min_df	2	Ignores terms appearing in only one document likely typos or employerspecific jargon with no generalization value
max_features	15,000	Caps vocabulary size to manage memory while retaining the full range of meaningful skill terms
sublinear_tf	True	Applies log(1 + tf) instead of raw tf. Prevents a skill mentioned 50 times from being 50x more important than a skill mentioned once. Critical for job descriptions that repeat the same term.

Table 4.1 TFIDF vectorizer parameters and rationale

Mathematical formulation

For term t in document d across corpus D:

   TFIDF Formula (with sublinear scaling)

TFIDF(t, d, D) = log(1 + tf(t,d)) × log(N / df(t)) where: tf(t,d) = raw count of term t in document d
N   = total number of documents (10,482)   df(t)   = number of documents containing term t The
IDF component log(N / df(t)) is high for rare skills (specific, valuable signal) and near zero for skills
that appear in almost every job (e.g. 'communication').

4.2 Technique 2 Cosine Similarity Matching

What it does

Given a user's freetext skill description, finds the job postings whose skill profiles are most similar
regardless of exact word matches or job title.

How it works

The user's input text is transformed by the same fitted TFIDF vectorizer into a vector in the identical
15,000dimensional space. Cosine similarity is then computed between this user vector and every
row in the precomputed job matrix:

Cosine Similarity Formula

similarity(A, B) = (A · B) / (||A|| × ||B||) = cos(θ) where θ is the angle between the two vectors
Range: 0.0 (no shared terms) to 1.0 (identical profile) Key property: anglebased, not
magnitudebased. A short skill description and a long one can achieve high similarity if they point in
the same direction. This means 'Python SQL machine learning' matches just as well as a 500word
description with the same three core skills.

Implementation detail the buffer

When filters are active (hide ghosts, federal only, apprenticeship only, city filter), the raw topN result
set would often be depleted below the requested number after filtering. To prevent this, we retrieve
top_n × 4 candidates before filtering, then trim to top_n after. This ensures that requesting 10
results always returns 10 results when 10 are available, even if 75% of the top matches are filtered
out.

4.3 Technique 3 Skill Gap Analysis

What it does

For each matched job, shows the user which of the job's most important required skills they already
have and which they need to develop. The output is not just a list it is ranked by importance.

How it works

The Correlation Coefficient in colorado_processed.csv serves a second purpose here. A high
Correlation Coefficient means the NLx system was very confident in mapping that raw skill to its

taxonomy category implying it is a clearlydefined, wellestablished skill that employers are asking for with precision. We use this as a proxy for skill importance within the job.

For each matched job, we retrieve all its skills from colorado_processed.csv, sort by Correlation Coefficient descending, and take the top 12. We then perform keyword matching against the user's text: a skill is considered matched if any word from the skill name (longer than 3 characters) appears in the user's description. Matched skills are shown as blue pills. Unmatched skills are shown as red pills. The ordering by Correlation Coefficient means the first red pill is always the highestpriority skill gap.

4.4 Technique 4 Emerging Skills Detection

What it does

Identifies skills that Colorado employers are actively requesting which do not have an established category in the ESCO or O*NET taxonomy. These represent gaps between what employers need right now and what the official skill databases which workforce training programs are built on currently capture.

How it works

This technique directly exploits the Correlation Coefficient. When the NLx system processes an employerwritten skill and produces a low Correlation Coefficient, it is signaling that it could not find a confident taxonomy match. The raw skill text is genuinely ambiguous with respect to the existing taxonomy which happens when a skill is newer than the taxonomy.

The algorithm:

5. Filter colorado_processed.csv to rows where Correlation Coefficient is below the userspecified threshold (default: 0.65).
6. Remove rows where Raw Skill is purely numeric, shorter than 5 characters, or empty these are data quality issues, not real emerging skills.
7. Group by Raw Skill and aggregate: count distinct employers requesting it, compute average Correlation Coefficient, identify the most common Taxonomy Skill mapped to it (the closest official category).
8. Filter to skills requested by at least min_employers distinct employers a skill appearing once could be a typo; appearing in multiple independent postings is a real signal.
9. Return the top 30 results sorted by employer count.

Policy significance

Why This Matters

ESCO (European Skills, Competencies, Qualifications and Occupations) is updated on a multiyear cycle. O*NET is reviewed periodically by the U.S. Department of Labor. Both are authoritative but inherently backwardlooking. A skill that emerged in the last 18 months may not appear in either database yet. Workforce development agencies that build training curricula exclusively from these

taxonomies are systematically training workers for yesterday's job market. The Emerging Skills Detector makes this gap visible and quantifiable.

4.5 Technique 5 Ghost Job Language Analysis

What it does

Uses NLP to compare the textual patterns in ghost job postings versus real job postings, turning a binary labeled column into an NLP research finding about the language characteristics of nongenuine postings.

How it works

The ghostjob boolean column in colorado.csv is treated as a class label. All job descriptions in each class are concatenated into two large text documents. Two separate TFIDF models (unigrams and bigrams, top 5,000 features) are fitted on these documents. Term weights are then compared: terms with high weight in the ghost document but low weight in the real document are ghostdominant terms. The inverse identifies realjobdominant language.

This technique answers: do ghost postings use systematically different language than active ones? If recruiters or automated systems could detect these patterns, ghost job filtering could be improved beyond the current binary flag.

4.6 Technique 6 Credential Inflation Detection

What it does

Identifies job postings where education requirements are significantly higher than what comparable employers require for the same role a practice known as credential inflation or degree inflation.

How it works

Jobs are grouped by O*NET occupation code, which represents a standardized occupational category. Within each group, we establish a minimum education baseline (the lowest education level required by any employer for that O*NET code, across at least 3 postings). Any posting requiring 2 or more education levels above that baseline is flagged.

Education levels are ranked on a 0–7 scale:

Level	Education	Notes
0	No Formal Education Required	
1	High School Diploma or GED	

Level	Education	Notes
2	Some College Coursework Completed	
3	Associate's Degree	Aliases: Associates Degree, Associate Degree
4	Bachelor's Degree	Aliases: Bachelors Degree, Bachelor Degree
5	Master's Degree	Aliases: Masters Degree, Master Degree
6	Doctoral Degree / PhD	
7	PostDoctoral Training	

Table 4.2 Education level ranking used for credential inflation detection

Bug encountered and fixed

The education strings in colorado.csv use inconsistent formatting apostrophes, capitalization, and spelling vary. An initial implementation using exact string matching silently dropped the majority of rows because "Bachelor's Degree" (with curly apostrophe) did not match "Bachelor's Degree" (with straight apostrophe). The fix normalizes all education strings to title case and maintains an alias dictionary covering common variants before matching.

4.7 Technique 7 MOC Code Matching and Veteran Career Translation

What it does

Translates a veteran's military occupational code and service description into civilian job matches, using a twostage approach that prioritizes direct employer intent before falling back to skill similarity.

Stage 1 Direct MOC matching

The moc_codes column in colorado.csv contains military occupational codes that some Colorado employers have explicitly added to their postings, signaling that they actively seek veterans with those backgrounds. Stage 1 performs a vectorized string search on this column for the veteran's MOS code. These matches are shown first they represent employers who already understand the veteran's training.

Stage 2 Cosine similarity with civilian translation

A dictionary of 24 common MOS codes maps each code to a civilian role label and a curated string of equivalent civilian skill terms. This civilian skill string is concatenated with any freetext description the veteran provides, then passed through the standard TFIDF + cosine similarity pipeline to find the top 12 skillsimilar jobs across the full posting database.

Bug encountered and fixed

If the veteran entered no MOS code (empty string), the Stage 1 string search would match every row in the dataset because Python evaluates "" in any_string as True. Additionally, the original implementation used iterrows() for the search, which is O(n) on 10,482 rows and too slow for interactive use. Both issues were fixed: an if moc_upper guard prevents emptystring matching, and the search was replaced with vectorized str.contains() using the re.escape()sanitized MOS code.

4.8 Technique 8 Skill Frequency Analysis

What it does

Computes the most indemand skills across the entire Colorado job market, and within specific career fields, directly from employer postings rather than survey data or occupational projections.

How it works

The Taxonomy Skill column in colorado_processed.csv is used directly. Since each row represents one employer's requirement for one skill in one posting, counting value frequencies gives the number of postings requiring each skill. This is not a projection it is a direct count of active January 2026 Colorado employer demand. The same approach is applied to subsets of postings matching a specific career field query to produce fieldspecific skill rankings.

4.9 Technique 9 Job Description Quality Scoring

What it does

Evaluates any job description against five measurable dimensions that research links to application volume, candidate quality, and workforce diversity outcomes. Returns a 0–100 score with dimensionlevel breakdown.

Dimension	Max Points	Measurement Method
Length	25	Word count. 150–600 words = 25 pts. 75–150 or 600–900 = 12 pts. Outside this range = 4 pts. Research basis: descriptions under 75 words provide insufficient information; over 900 words reduce completion rate.
Salary Transparency	30	Presence of a numeric minimum salary value. 30 pts if salary > 0, 0 if missing. Research basis: salary disclosure increases qualified application volume by approximately 35%.
Education Clarity	15	Whether requirements_min_education field is populated and nonempty. Binary: 15 or 0 pts.
Experience Clarity	15	Whether requirements_experience field is populated and nonempty. Binary: 15 or 0 pts.
Specificity	15	Regex count of quantified requirements in the description text (e.g., '3+ years', '2 months'). 2+ instances = 15 pts, 1 instance = 8 pts, 0 = 0 pts.

Table 4.3 Job description quality scoring rubric

Table 4.3 Job description quality scoring rubric

5. Platform Features

The platform is organized into five tabs, each serving a distinct user type. All features draw from the same two source files and the same NLP pipeline.

5.1 Tab 1 Job Seeker

- Skillbased job matching using TFIDF + cosine similarity (Techniques 1 and 2)
- Ghost job filter: hides all postings where ghostjob = True from results
- Federal contractor badge: flags employers subject to VEVRAA veteran hiring preference
- Registered apprenticeship badge: surfaces RAPIDScoded earnwhileyoutrain pathways
- Perjob skill gap analysis showing matched skills (blue) and missing skills (red), ranked by Correlation Coefficient (Technique 3)
- Overqualification detection: warns when advanceddegree users match disproportionately with loweducationrequirement postings
- Adjacent career discovery: returns one result per NAICS sector to surface roles across different industries that share high skill overlap

5.2 Tab 2 Student

- Career field analysis: 12 preset field queries plus custom input, showing required skill frequency, salary distribution, and learning priority recommendation
- Trending skills: top 30 skill frequency across all 10,482 postings, broken down by ESCO vs O*NET taxonomy source
- Emerging skills detector: configurable threshold and minimum employer count (Technique 4)
- Nodegree pathways: full table of jobs with RAPIDS codes, showing that professional pathways exist that do not require a fouryear degree

5.3 Tab 3 Veteran

- MOS code reference table covering 24 military occupational codes
- Stage 1 direct MOC matching: finds employers who explicitly tagged their posting with the veteran's MOS code (Technique 7, Stage 1)
- Stage 2 skill similarity: translates the MOS code through a civilian skill dictionary and runs cosine similarity matching (Technique 7, Stage 2)
- Federal contractor priority filter: limits results to VEVRAAcovered employers
- Skill gap showing transferable military skills vs. civilian bridge skills needed

5.4 Tab 4 Market Intelligence

- Credential inflation scanner: groups by O*NET code, flags postings requiring 2+ education levels above the minimum for that occupation (Technique 6)

- Salary fairness map: citylevel salary statistics (average, median, P25, P75, max) for cities with 3+ postings
- Rolespecific salary lookup: cosine similarity finds comparable jobs, extracts salary distribution for any role
- Ghost job population analysis: volume statistics, city distribution, employer distribution, salary comparison between ghost and real postings
- Ghost job language analysis: TFIDF language comparison showing distinguishing terms (Technique 5)
- Educationjob alignment: CIP code frequency showing which academic program categories have high Colorado employer demand

5.5 Tab 5 Recruiter Tools

- Job description quality scorer: 0–100 score across 5 dimensions with actionable breakdown (Technique 9)
- Skill profile preview: shows which NLx taxonomy skills the description would attract via cosine similarity
- Salary benchmarking: compares planned offer to market percentiles for comparable roles with colorcoded positioning card (top quartile, above median, below median, bottom quartile)

6. Technical Implementation Notes

6.1 Critical Bugs Found and Fixed

Seven bugs were identified during development that would have caused crashes or silent data errors in production. All were fixed before deployment.

#	Bug	Root Cause	Fix Applied
1	groupby().apply() crash with include_groups=False	Parameter only exists in pandas 2.2+; HF Spaces ran an earlier version	Applied groupby on single column series, removing the parameter entirely
2	Regex dot in O.NET matches any character	str.contains('O.NET') the dot is a regex wildcard	Escaped to r'O\.NET' with regex=True
3	value_counts().reset_index() wrong column names	pandas 2.0+ changed output column names from ['index', col] to [col, 'count']	Set column names explicitly with .columns = [...] after reset_index()
4	Empty string passed to TFIDF in ghost analysis	If no ghost jobs have descriptions, ' '.join([]) produces '' which causes a meaningless allzero vector	Added guard: if not ghost_text.strip(): return None, None
5	Education string matching drops most rows silently	Exact string match failed on apostrophe and capitalization variants	Normalize to titlecase + expanded alias dictionary before matching
6	float() crashes on commaformatted salary input	Recruiter typing '65,000' raises ValueError in float()	Strip commas with str.replace(',', '') before float() conversion
7	Empty MOS code matches every row	"" in any_string is always True in Python; also iterrows() over 10K rows is too slow	Added if moc_upper: guard; replaced iterrows() with vectorized str.contains()

Table 6.1 Bugs identified and resolved

6.2 Caching Strategy

Streamlit's caching system is used at two levels:

- @st.cache_data on load_data(): The 104 MB of CSV data is read and preprocessed once per session. The decorated function receives a hash of its arguments; since load_data() has no arguments, it runs exactly once and its outputs are returned from memory on subsequent calls.

- @st.cache_resource on build_vectorizer(): The TFIDF matrix is a sparse numpy array approximately 180 MB in memory. @st.cache_resource keeps it as a shared object across reruns, unlike @st.cache_data which would serialize and copy it. This is the correct decorator for large nonserializable objects.

- Functions that accept dataframes use the underscore prefix convention (_jobs_clean, _processed) to signal to Streamlit's hasher that these arguments should not be hashed hashing a 165Krow dataframe on every call would be slower than just recomputing the result.

6.3 Hugging Face Deployment Fix

On initial deployment, the application crashed with ModuleNotFoundError: No module named 'tabs.tab4_market' even though the file existed. The root cause was that Docker sets WORKDIR /app in the Dockerfile, but Python's sys.path does not automatically include the working directory when modules are imported as packages. The fix adds an explicit sys.path.insert(0, APP_DIR) at the top of app.py before any imports, where APP_DIR is derived from os.path.abspath(__file__). This runs before any module import and ensures Python always knows where to look for core/ and tabs/ regardless of the container's working directory configuration.

7. Bias, Limitations, and Methodological Transparency

The problem statement explicitly required solutions to address bias, data limitations, and methodological transparency. This section documents all known limitations of the platform and the design choices made in response to them.

7.1 Data Limitations

Limitation	Impact and Mitigation
Employerposted jobs only	Positions filled through internal promotion, referral, or headhunter are not represented. The platform cannot speak to the hidden job market, which research estimates represents 30–50% of actual hiring. This is disclosed in the methodology footer.
Geographic concentration	Denver and the Front Range are heavily overrepresented relative to rural Colorado. A job seeker in Pueblo or Grand Junction will see fewer results. City filter helps but does not solve this imbalance.
Ghost job labeling opacity	The NLx ghostjob flag's criteria are not publicly documented. We treat it as a binary label without knowing the underlying detection model's precision, recall, or false positive rate. Results from ghost job analysis inherit this uncertainty.
Salary completeness	Approximately 40% of postings have no salary data. All salary features exclude these postings, which means the salary distributions shown are from the subset of employers who choose to disclose pay potentially a nonrepresentative sample skewed toward larger or more competitive employers.
January 2026 snapshot	This is a pointintime dataset, not a realtime feed. Postings may have been filled, pulled, or modified since the snapshot date. All findings should be interpreted as representing the state of the Colorado job market in January 2026.

Table 7.1 Data limitations and impacts

7.2 NLP Methodology Limitations

Limitation	Impact and Mitigation
Bagofwords model	TFIDF treats text as an unordered set of words. It cannot capture semantic relationships: 'Python' and 'Java' are as different in vector space as 'Python' and 'cooking'. A user describing React experience will not automatically match jobs requiring Vue.js, even though the skills are closely related. Word embeddings (Word2Vec, BERT) would capture this but require significantly more compute.
MOC dictionary coverage	The MOS translation dictionary covers 24 of the 190+ active MOS codes in the U.S. Army alone, and does not include Navy rates, Air Force AFSCs, or Marine MOSs beyond those that overlap with Army codes. Veterans with uncovered codes fall back to generic military skill matching.

Limitation	Impact and Mitigation
Correlation Coefficient interpretation	We use the NLx Correlation Coefficient as a proxy for both skill importance (Technique 3) and taxonomy novelty (Technique 4). These are two distinct interpretations of the same score and they may sometimes conflict. The score is not independently validated against external benchmarks.
Skill gap keyword matching	The skill gap analysis uses simple keyword overlap rather than semantic matching. 'ML' will not match 'machine learning'; 'JS' will not match 'JavaScript'. Users who use abbreviations will see their skills underreported.
Credential inflation threshold	The twolevel gap threshold for flagging credential inflation (2+ education levels above the minimum in the same O*NET group) is a reasonable heuristic but is not derived from a statistical significance test or external validation. Some flagged postings may reflect legitimate specialization requirements not captured in the O*NET grouping.

Table 7.2 NLP methodology limitations

7.3 Equity Considerations

Several design decisions were made specifically to surface equityrelevant information that is typically hidden from job seekers:

- The ghost job filter is on by default. Job seekers who apply to ghost jobs waste time and experience rejection that is not reflective of their qualifications. Making the filter the default protects users who may not know ghost job detection is possible.
- Credential inflation is framed as an equity issue in the UI. The warncard explicitly states that inflated education requirements 'disproportionately lock out lowerincome workers, immigrants, and career changers.' This framing follows the research literature on degree inflation's disparate impact.
- The Registered Apprenticeship pathway tab directly counters the assumption that professional roles require fouryear degrees. Making RAPIDS codes visible is an equity intervention these pathways exist but are invisible on every major job board.
- Salary transparency is presented as a workerprotective feature, not just a convenience. The benchmark tool explicitly tells recruiters that posting below the 25th percentile reduces qualified application volume framing salary disclosure as a selfinterested hiring practice, not just a regulatory or fairness concern.

8. Key Findings

The following findings emerge directly from the NLx Colorado dataset and the NLP analysis implemented in this platform. All are reproducible by running the platform against the January 2026 data.

8.1 Ghost Job Prevalence

A nontrivial proportion of Colorado job postings in January 2026 are flagged by the NLx system as ghost jobs. The exact percentage is visible in the platform's Market Intelligence tab. The Ghost Job Language Analysis (Technique 5) surfaces whether ghost postings use systematically different language a finding with implications for automated ghost job detection systems that go beyond binary flagging.

8.2 Emerging Skills Gap

The Emerging Skills Detector (Technique 4) consistently surfaces skills with low taxonomy confidence that are being requested by multiple independent Colorado employers. These represent active market demand that the ESCO and O*NET databases the standard inputs to workforce training program design have not yet catalogued. Any workforce development agency building curricula exclusively from official taxonomy data is systematically missing these signals.

8.3 Credential Inflation Patterns

Within multiple O*NET occupation codes in the Colorado dataset, postings exist that require two or more education levels above the minimum seen for comparable postings in the same occupation category. This inflation is not uniformly distributed across sectors the sector bar chart in the platform shows which O*NET major groups have the highest concentration of credential inflation. This is a direct, datadriven input for evidencebased credential reform policy.

8.4 Geographic Salary Variance

The Salary Fairness Map shows that the same role matched by skill profile, not just title can pay significantly more in Denver than in smaller Colorado cities. Workers who accept the first offer without knowing their geographic market value consistently leave money on the table. This finding is directly actionable at the individual level through the Salary Fairness lookup tool.

8.5 Apprenticeship Pathway Invisibility

RAPIDS codes exist in a nontrivial number of Colorado postings, representing federallycertified earnwhileyoutrain pathways. These pathways are entirely invisible on major job boards including Indeed and LinkedIn. The assumption that professional employment requires a fouryear degree is

contradicted by the data itself the platform makes this visible for the first time in a jobsearch interface.

9. Future Development

9.1 NLP Enhancements

- Replace TFIDF with sentencelevel embeddings (BERT, sentencetransformers) to capture semantic similarity between skills. This would solve the JavaScript/TypeScript gap and similar abbreviation/synonym problems.
- Train a named entity recognizer on NLx data to extract skills, years of experience, salary ranges, and education requirements from unstructured job descriptions addressing the core structuring problem beyond what the existing taxonomy mapping achieves.
- Implement temporal skill trend analysis: compare skill frequency in January 2026 data against a historical baseline to measure the growth rate of specific skills over time, not just current prevalence.

9.2 Data Expansions

- Extend to all 50 states to enable interstate wage and credential comparison a particularly valuable tool for policymakers considering interstate labor market effects of training investments.
- Integrate O*NET occupationlevel wage data to provide an independent salary benchmark that does not depend on employers disclosing pay.
- Expand the MOC dictionary to cover all 190+ Army MOS codes, all Navy rates, Air Force AFSCs, and Marine MOSs.

9.3 Methodological Improvements

- Validate the Emerging Skills Detector against ground truth: compare skills flagged as emerging in January 2026 against taxonomy updates published in the following year to measure whether low Correlation Coefficient is a reliable predictor of taxonomy inclusion lag.
- Statistical validation of the credential inflation threshold: use survival analysis or propensity score matching to determine whether 2+ education levels above the O*NET minimum is genuinely inflated versus legitimately specialized.
- Uncertainty quantification: add confidence intervals to skill gap analysis and salary benchmarks to communicate to users the reliability of the estimates given sample size.

10. References and Resources

Data Sources

- National Labor Exchange (NLx) Colorado Job Postings January 2026. Provided by DirectEmployers Association and NASWA for GW Hackathon 2026, Problem Statement 4.
- ESCO (European Skills, Competencies, Qualifications and Occupations) taxonomy European Commission. https://esco.ec.europa.eu
- O*NET (Occupational Information Network) U.S. Department of Labor Employment and Training Administration. https://www.onetcenter.org
- RAPIDS (Registered Apprenticeship Program Information Data System) U.S. Department of Labor. https://www.dol.gov/agencies/eta/apprenticeship
- CIP (Classification of Instructional Programs) National Center for Education Statistics. https://nces.ed.gov/ipeds/cipcode

Key Libraries

- scikitlearn: Pedregosa et al. (2011). Scikitlearn: Machine Learning in Python. JMLR 12, pp. 2825–2830.
- pandas: The pandas development team (2020). pandasdev/pandas: Pandas. Zenodo.
- Streamlit: Streamlit Inc. (2019). Streamlit The fastest way to build data apps. https://streamlit.io

Deployment

- Platform live at: https://huggingface.co/spaces/aksharatarikere/coloradoworkforce
- Source code: All files available in the Hugging Face Space repository under the Files tab.