Modeling Assumptions Documentation

This document describes the explicit and implicit modeling assumptions used in the Colorado Workforce Intelligence project.

1) Scope and Modeling Objective

The system is designed for **job discovery and prioritization**, not for causal inference or hiring outcome prediction.

Primary objective: - Rank job postings by semantic similarity between user-entered skills and job skill profiles.

Non-objectives: - Predict interview probability or hiring probability. - Infer wage outcomes or career trajectories. - Perform fairness-optimized ranking.

---

2) Data Assumptions

2.1 Source and Structure

Assumptions encoded in hackathon/core/data.py: - Core inputs are NLx Colorado files (`colorado.csv`, `colorado_processed.csv`). - If required columns are missing, they are created with empty string defaults. - Empty string is treated as a valid placeholder for missing categorical/text values.

2.2 Record Identity

- `system_job_id` is treated as the stable join key between job records and derived artifacts.
- `Research ID` in processed skill mentions is assumed to map to `system_job_id`.

2.3 Missingness

- Missing values are filled with empty strings before downstream processing.
- Missing salary min/max implies "Not listed" in UI.
- Missing requirements fields trigger NLP inference only where pattern matches exist.

2.4 Representativeness

- Job posting frequency is used as a proxy for demand (especially in Student views).
- Posting data is assumed to represent labor market demand trends, acknowledging sampling bias by employer/region.

---

3) NLP Skill Extraction Assumptions

Assumptions encoded in hackathon/core/nlp_pipeline.py:

3.1 Text Normalization

- Lowercasing and punctuation stripping preserve enough semantic signal for matching.
- Whitespace normalization and token simplification are sufficient.

3.2 Skill Catalog Construction

- Only skills with normalized frequency >= 3 are retained (`min_frequency=3`).
- Catalog size cap (`max_skills=3000`) is assumed to keep performance tractable without losing high-value signal.
- Tokens with very short length are implicitly downweighted by preprocessing and frequency behavior.

3.3 Mention Extraction

- TF-IDF with unigrams+bigrams (`ngram_range=(1,2)`) captures enough lexical context.
- Similarity threshold (`min_similarity=0.08`) distinguishes meaningful skill-job relationships.
- Top K skills per job (`top_k=15`) is assumed to be sufficient for profile richness.
- Batch size (`batch_size=256`) is performance-oriented and does not materially alter ranking semantics.

3.4 Fallback Behavior

- If NLP mention extraction returns empty, fallback uses source processed correlations (`Correlation Coefficient`) as proxy scores.

---

4) Education and Experience Inference Assumptions

Assumptions encoded in hackathon/core/nlp_pipeline.py:

- Existing dataset values for requirements have higher trust than inferred values.
- Regex pattern matching can adequately infer coarse categories for education/experience.
- Experience phrases are normalized to canonical labels (e.g., `Entry level / no experience`, `x-y years`).
- Absence of pattern match is surfaced as `not_specified` rather than estimated/imputed.

Source-confidence assumptions: - `dataset` > `nlp_inferred` > `not_specified`

---

5) Matching Model Assumptions

Assumptions encoded in hackathon/core/matching.py:

5.1 Representation

- Job relevance can be approximated by TF-IDF vectors built from aggregated job skill text.
- Lexical overlap with weighting is a practical proxy for skill fit.

5.2 Similarity Metric

- Cosine similarity is appropriate for ranking textual skill vectors.
- Vector direction similarity is more informative than vector magnitude for this task.

5.3 Ranking Behavior

- Top-N (`top_n`) truncation is acceptable for user-facing recommendation lists.
- Ranking ignores non-skill business factors (company preference, recency weighting, commute distance, etc.) unless explicitly filtered elsewhere.

5.4 Cache/Indexing

- Reusing a cached vectorizer+matrix across queries is assumed to preserve deterministic behavior while improving latency.

---

6) Skill Gap Analysis Assumptions

Assumptions encoded in hackathon/core/matching.py:

- Most informative job skills are approximated by top-scoring mention rows.
- User-skill presence can be approximated by lexical token overlap (substring check) on tokens with length > 3.
- The method is explanatory and heuristic, not a strict competency assessment.

Limitations implied by assumptions: - Synonyms/paraphrases may be missed. - Negation and contextual nuance are not modeled.

---

7) Veteran Translation Assumptions

Assumptions encoded in hackathon/core/veterans.py:

- If `moc_codes` contains the user code, that is treated as a high-confidence direct match.
- Expanded MOC dictionary text provides a meaningful civilian-skill bridge for similarity search.
- Unknown MOC fallback (`operations leadership management`) is acceptable as a broad baseline query.

---

8) Student/Career Demand Assumptions

Assumptions encoded in hackathon/core/student.py and hackathon/app.py:

- Skill demand is approximated using raw frequency counts of `Taxonomy Skill`.
- Static field keyword prompts are sufficient to seed field-specific exploration.
- Demand tiers (High/Medium/Low by rank bands) are ordinal UX categories, not calibrated probabilistic classes.

---

9) Analytics and Monitoring Assumptions

Assumptions encoded in hackathon/core/analytics_logger.py and hackathon/app.py:

- Event-level logging (visit/search/recommendation) is sufficient for operational usage insights.
- SQLite is reliable for lightweight local persistence; CSV mirror improves portability.
- Timestamp ordering and event completeness are adequate for trend charts.

Non-assumptions: - This is not a privacy-preserving telemetry system for multi-tenant production.

---

10) UI/Presentation Assumptions

Assumptions encoded in hackathon/app.py:

- Users can interpret match score percentages as **relative ranking signals**, not absolute success probabilities.
- Card and table formats improve scanability for different user preferences.
- Displaying requirement source transparency improves user trust and interpretability.

---

11) Key Hyperparameters and Heuristics

Component	Parameter	Current Value	Assumption
Skill catalog	`min_frequency`	3	Rare singleton skills are likely noise for global matching
Skill catalog	`max_skills`	3000	Practical upper bound for performance and utility
Mention extraction	`top_k`	15	Top 15 skills adequately summarize job skill profile
Mention extraction	`min_similarity`	0.08	Filters weak lexical associations
Mention extraction	`batch_size`	256	Balanced memory/performance for sparse multiplication
Matching model	`ngram_range`	(1,2)	Bigrams add useful phrase context
Matching model	`stop_words`	english	Function words add low value to skill matching
Skill gap	`limit`	12 (default)	Top skills provide enough explanatory signal
Student demand tiers	rank bands	1–6 / 7–14 / 15–20	Tiering is UX-oriented ordinal grouping

---

12) Risk, Bias, and Reliability Notes

Potential risk sources from assumptions: - Employer and region sampling bias from posting data. - Under-representation of non-standard wording in regex and token overlap logic. - Skill trend overemphasis for high-posting sectors.

Reliability notes: - Deterministic matching under same artifacts and query. - Results depend on freshness of generated processed artifacts.

---

13) Recommended Validation Strategy

To ensure assumptions remain valid over time:

1. **Data drift checks**
   - Monitor volume, missingness, and column integrity in raw files.
2. **Retrieval quality checks**
   - Build a small benchmark set of expected query-to-job relevance pairs.
3. **Skill-gap sanity checks**
   - Human review of matched/missing outputs on sampled recommendations.
4. **Veteran mapping review**
   - Periodic SME review of MOC dictionary terms and outcomes.
5. **Threshold tuning**
   - Re-evaluate `min_similarity`, `top_k`, and tier boundaries quarterly.

---

14) Change Control Guidance

When updating modeling assumptions: - Update this document and docs/PROJECT_DOCUMENTATION.md. - Track parameter changes and rationale in PR descriptions. - Re-run baseline smoke tests for data loading, matching, and UI rendering.

---

15) Summary

The project uses pragmatic, transparent NLP retrieval assumptions to support exploration-oriented workforce matching. The current design favors: - interpretability, - responsiveness, - and operational simplicity,

while explicitly acknowledging that outputs are recommendation aids, not predictive guarantees.