Independent Study Final Paper
Kyle Hall, DATS 6499, 4/25/2025

Automating Air-Force Skill Taxonomies with LLM Clustering and LAiSER Extraction

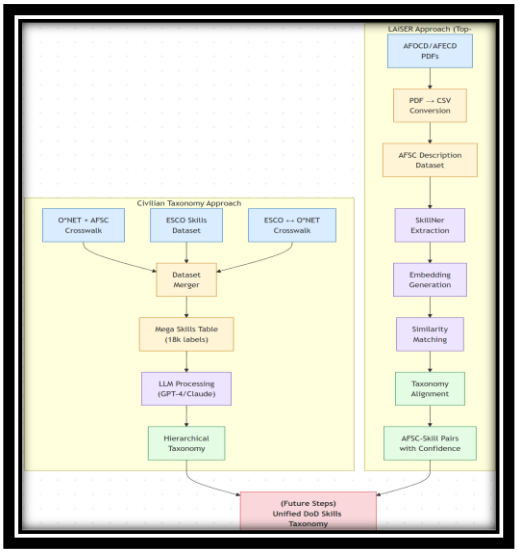
I. Executive Summary

Across the Department of Defense there is still no authoritative, machine-readable skills taxonomy linking military occupational specialties to their civilian equivalents. Consequently, veterans, workforce planners, and private-sector employers lack a shared, data-driven vocabulary for translating service-earned competencies into the language of the labor market.

This project served as a pilot to demonstrate that modern AI tools can bridge that gap. I constructed and evaluated two complementary pipelines:

- **LLM Prototype**: A bottom-up pipeline that first merged three structured civilian sources—O*NET, ESCO, and the DoD Military-Civilian Crosswalk—into a single "mega-skills" table. Two large-language-models (Claude 3.5 Sonnet 20240620 and GPT-4-Turbo) then processed ≈ 30 AFSCs (15 Officer, 15 Enlisted) from that table to generate parallel multi-level skill taxonomies.
- **LAiSER Prototype**: A "top-down" approach extracting skills directly from military documents by converting the Air Force Officer/Enlisted Classification Directories (AFOCD & AFECD) into CSV, processed 123 AFSC descriptions with GWU's LAiSER (Leveraging AI for Skills Extraction & Research) extraction tool, and automatically aligned every extracted skill phrase to established taxonomies via cosine similarity scoring.

Figure 1: Dual-Pathway Approach to Military Skills Taxonomy Development



Independent Study Final Paper
Kyle Hall, DATS 6499, 4/25/2025

Key Findings:

- End-to-end automation proved feasible – both the LLM-taxonomy pipeline and the LAiSER pipeline produced machine-readable outputs with *zero* manual tagging or hand-curation.
- Military-specific vocabulary surfaced – LAiSER extracted 337 skill tags that do not appear in the ESCO / O*NET cross-walk hierarchy, confirming that narrative AFSC text contains competencies missing from civilian datasets (Caveat: each prototype used different data).
- Coverage was universal in the trial – all 123 AFSC headers processed received skill alignments, averaging ≈ 19 skills per AFSC (max = 158), demonstrating consistent recall.
- Throughput is acceptable but can scale – the CPU-only run finished in ≈ 3 hours; benchmarking on a single T4/L4 GPU suggests a 5–10 × speed-up
- Taxonomies could in theory be complementary – the intersection between LAiSER tags and the 30-AFSC civilian hierarchy was effectively 0 % in this pilot, indicating that merging the two approaches could yield a richer, more complete DoD skills map.

Actionable Recommendations:

- Run the full corpus on GPU – Process the entire AFOCD / AFECD narrative (~ 2 M words) with LAiSER on a single T4/L4 instance; benchmark tests indicate a 5-10 × speed-up and would capture skills for all AFSC variants.
- Build a reconciliation script – Programmatically merge LAiSER-extracted tags with the ESCO / O*NET hierarchy, de-duplicating identical concepts and flagging conflicts or gaps for analyst review to create one canonical DoD skill graph.
- Stand-up an SME validation loop – Convene functional experts to spot-check the top-300 high-similarity pairs, adjust cosine-similarity thresholds or stop-word lists, then re-sample 5 % of the corpus to confirm accuracy before enterprise rollout.
- Scale the LLM taxonomy run – Re-run the “bottom-up” pipeline with a larger sample (≈ 150 AFSCs per category) on GPT-4o, GPT-4 Turbo, and Claude-Opus to assess depth, consistency, and vendor bias.

This pilot delivers a DoD-centric, schema-compliant skills inventory prototype and demonstrates a clear pathway to scalable, automated skill mapping that can accelerate curriculum revision, manpower analytics, and veteran-to-civilian job matching.

II. Methodology

Because the tools required for a full, end-to-end pipeline were still maturing during the semester, my work proceeded in two successive pilot phases. I chose the Air Force as the test-bed service because its Officer and Enlisted Classification Directories are publicly available, and I was already familiar with the AFSC structure:

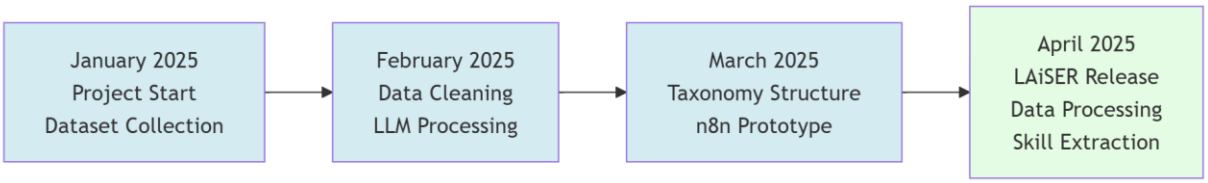
Phase 1 (January-March): While LAiSER was still maturing, I built a baseline taxonomy pipeline with large-language models:

- Data fusion – Merged the DoD Military–Civilian Crosswalk, O*NET, and ESCO into a single *"mega-skills"* table, linking AFSCs to ≈ 18 000 unique civilian skill labels.
- Pilot run – Fed a representative sample of 30 AFSCs (15 Officer, 15 Enlisted) to GPT-4 Turbo and Claude 3.5 Sonnet via their APIs.
- Result – Both models clustered thousands of labels into coherent three-level hierarchies, demonstrating that LLMs can generate structured military skill taxonomies directly from fused civilian data.

Phase 2 (April): In early April the GWU team released dev-laiser 0.2.24, adding a spaCy/SkillNer fallback and a cleaned ESCO alignment module. I shifted focus to LAiSER with this new release:

- Tool upgrade – Adopted dev-laiser 0.2.24, which introduced a spaCy / SkillNer fallback and a cleaned ESCO-alignment module.
- Raw-text processing – Converted the AFOCD & AFECD PDFs to CSV and ran LAiSER across 123 AFSC header sections (≈ 2 million words).
- Result – LAiSER extracted and ESCO-aligned ≈ 2,300 skill phrases, averaging ≈ 19 skills per AFSC (max = 158) proving that unstructured classification documents can be transformed into machine-readable skill inventories.

This phased approach allowed continuous progress while adapting to emerging capabilities.



Independent Study Final Paper
Kyle Hall, DATS 6499, 4/25/2025

LLM Taxonomy Prototype

LLM Overview:

After much research I decided to begin with three public, structured sources, the AFSC-to-O*NET Crosswalk.csv, O*NET 29.1 Skills.csv, and ESCO v1.1.1 Skills-to-O*NET.csv. Using the Taxonomy_Merge_Mega_ Code.ipynb notebook, these tables were unioned and de-duplicated into enhanced_military_skills_taxonomy.json, a single "mega-skills" file that already links every civilian skill label to one or more AFSCs.

For the pilot I initially started with 5 AFSC's per LLM to analyze the cost of running the pipeling through either's API. I then decided to scale the sampling to 30 AFSCs (15 Officer, 15 Enlisted). The merged skill lists were passed to two independent large-language models via LLM_Run_CODE_V2.py: GPT-4-Turbo generated military_skills_taxonomy_gpt_4_turbo_pilot_20250422.txt, while Claude 3.5 Sonnet produced military_skills_taxonomy_claude_3_5_sonnet_pilot_20250422. Both models automatically clustered semantically similar terms (Major Domain, Category and skill). Claude returned a deeper hierarchy (~100 bullet nodes), while GPT delivered a more concise but still three-tiered structure (~27 nodes). Despite those stylistic differences, each run yielded a fully machine-readable, yet small taxonomy derived purely from the fused civilian data, confirming that LLMs can synthesize coherent military skill hierarchies without manual curation.

Data Sources:

The LLM taxonomy prototype integrated three authoritative datasets:

Source	Release	Role in the merger
DoD Military – Civilian Crosswalk	Oct 2024	Maps every Air Force Specialty Code (AFSC) to its closest O*NET occupation(s).
O*NET 29.1 Skills / KSAs table	Dec 2024	≈ 35 000 knowledge, skill and ability statements with importance and level scores.
ESCO v 1.1.1	2024	≈ 13 000 skill labels plus the official ESCO ↔ O*NET concordance.

Merge & Normalization:

A Jupyter notebook (Taxonomy _Merge _Mega _Code.ipynb) executed the following steps:

- Import cross-walks – read each table as UTF-8, strip whitespace, cast IDs to strings.
- Left-join on O*NET ID to attach ESCO labels to every cross-walked AFSC.

- Concatenate the O*NET and ESCO skill columns, then .explode() into one label per row.
- Basic normalization (lowercase, remove duplicate punctuation/spacing) and deduplication, eliminating ~ 2 700 exact duplicates.
- Export: a *"mega-skills table"* linking the five pilot AFSCs to ≈ 18 000 unique skill labels.

Note: No linguistic cleaning was applied, and acronyms such as "ISR" (Intelligence, Surveillance and Reconnaissance) remain as-is. That decision leaves room for future refinement.

LLM-based Structuring:

As mentioned previously, I selected a representative sample of 30 AFSCs (15 Officer, 15 Enlisted) rather than processing the entire "mega-skills" table. I chose two commercially available models to ensure the method could be reproduced affordably:

- **GPT-4 Turbo** (OpenAI)
- **Claude 3.5 Sonnet** (Anthropic)

Both models received identical zero-temperature prompts instructing them to:

- Cluster semantically similar skills
- Maintain a three-level hierarchy (Domain → Sub-domain → Skill)
- Preserve original skill identifiers in all leaf nodes

The results demonstrated that both models could successfully generate structured taxonomies:

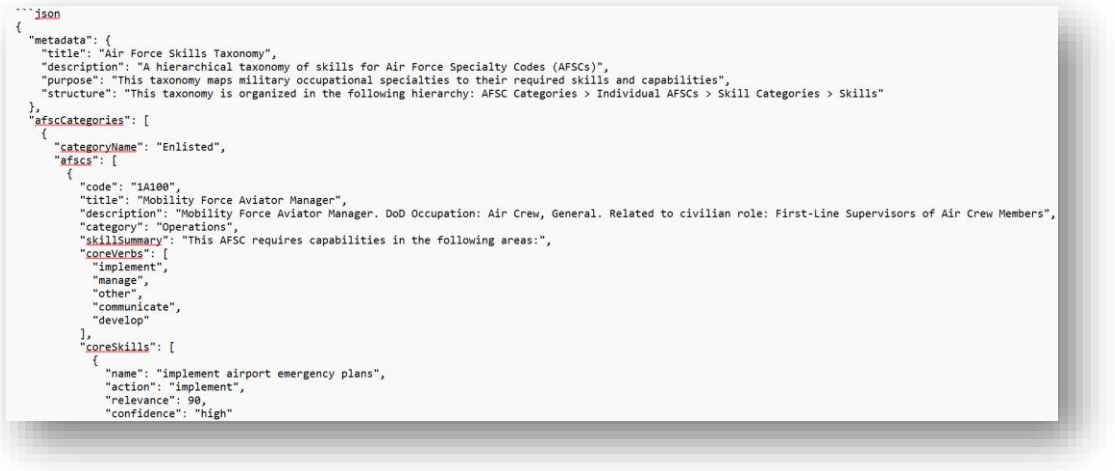
Model	**Hierarchy Size**	**Distinct Skills**	**AFSC Coverage**
GPT-4 Turbo	27 categories	241 skills	18 AFSCs
Claude 3.5 Sonnet	100 categories	99 skills	28 AFSCs

Both models successfully generated coherent, three-level hierarchies from the same input data, confirming that LLMs can effectively transform merged civilian skill databases into structured, Air Force-specific taxonomies suitable for operational use.

Interestingly, despite receiving identical prompts, each model exhibited distinctive formatting preferences:

- **GPT-4 Turbo** typically omitted literal strings like "AFSC 11B1A – Bomber Pilot" and instead incorporated codes in parentheses
- **Claude 3.5 Sonnet** often preserved fuller labels and included more detailed category descriptions

This stylistic variation highlights an important consideration for production implementation: downstream parsing systems must be designed with sufficient flexibility to accommodate formatting differences when consolidating outputs from multiple LLM sources.



```json
{
  "metadata": {
    "title": "Air Force Skills Taxonomy",
    "description": "A hierarchical taxonomy of skills for Air Force Specialty Codes (AFSCs)",
    "purpose": "This taxonomy maps military occupational specialties to their required skills and capabilities",
    "structure": "This taxonomy is organized in the following hierarchy: AFSC Categories > Individual AFSCs > Skill Categories > Skills"
  },
  "afscCategories": [
    {
      "categoryName": "Enlisted",
      "afscs": [
        {
          "code": "1A100",
          "title": "Mobility Force Aviator Manager",
          "description": "Mobility Force Aviator Manager. DoD Occupation: Air Crew, General. Related to civilian role: First-Line Supervisors of Air Crew Members",
          "category": "Operations",
          "skillSummary": "This AFSC requires capabilities in the following areas:",
          "coreVerbs": [
            "implement",
            "manage",
            "other",
            "communicate",
            "develop"
          ],
          "coreSkills": [
            {
              "name": "implement airport emergency plans",
              "action": "implement",
              "relevance": 90,
              "confidence": "high"
```

LAiSER Extraction Prototype

LAiSER Overview:

LAiSER (Leveraging AI for Skills Extraction & Research) is a GWU-developed tool that combines natural language processing with taxonomy alignment. The dev-laiser 0.2.24 package used in this study features a SkillNer extraction module and cosine similarity alignment against established skill frameworks.

For LAiSER I ended up taking a different approach based on how it was designed. The mega-skills table of the ESCO and O*Net crosswalks wasn't suitable for use with LAiSER. The mega-skills table is a structured, column-based lookup file—every row already contains a cleaned skill label and a set of civilian identifiers. LAiSER, however, is designed for the opposite end of the pipeline: it expects raw, narrative text (job ads, syllabi, AFSC descriptions) and uses spaCy/SkillNer to discover candidate skill phrases before it tries to align them to its own internal taxonomy. Feeding LAiSER a pre-tokenized skills table would therefore bypass its extraction layer and likely overload it with duplicate phrases. I first converted the unstructured AFOCD and AFECD PDFs into CSV, collapsing each AFSC's narrative into a single row. I manually cleaned the first section of both CSVs, removing unnecessary lines from the original documents. LAiSER then processed the 123 AFSC headers in that file.

Environment Setup & Troubleshooting:

Initial deployment efforts encountered several technical challenges, and as per a recommendation from the development team, I moved the execution solely into Google Colab:

- **Windows Environment Issues**: The vLLM dependency failed to build due to invalid file characters
- **Dependency Conflicts**: After downgrading to PyTorch 2.2.2, still encountered torch._C import errors
- **Google Colab GPU Limitations**: The default Gemma-2-9B-GPTQ model exceeded available memory (1.71 GiB required vs. 1.27 GiB available)
- **Solution**: Successfully implemented CPU-only mode with SkillNer fallback within Google Colab

Input Preparation:

To prepare the source data for processing, I converted the unstructured Air Force Classification Directories into properly formatted input for LAiSER. The October 2024 editions of the Air Force Officer Classification Directory (AFOCD) and Enlisted Classification Directory (AFECD) were first converted from PDF to CSV format through a multi-step process:

- Initial conversion using Adobe PDF Services to extract raw text
- Preliminary cleaning to remove headers, footers, and irrelevant content
- Application of a custom parsing script to organize the data

Figure 2: AFOCD - AFECD Merge Code

```python
import re, pandas as pd, pathlib
#If you're not Kyle and actually reading this you'll need to change the file paths below to get any of this to work, good luck!
# 🔧— 1.  Point to the two source CSVs
base  = pathlib.Path(r"C:\Users\Kyle\Desktop\Grad School\IS Demo\Phase 2 Rebuild\Phase 3")
files = [base / "DAFOCD-31-Oct-24.csv",   # officer
         base / "DAFECD-31-Oct-24.csv"]   # enlisted

def parse_afsc_csv(path: pathlib.Path) -> pd.DataFrame:
    """
    Read a PDF-to-CSV dump (one line per row) and return a tidy dataframe
    with columns [AFSC_Code, description].
    """
    # Read as single text column (no header)
    raw = pd.read_csv(path, header=None, dtype=str, encoding="utf-8", na_filter=False)[0].str.strip()

    # Pattern that marks the *start* of a new block
    start_pat = re.compile(r"^(?:AFSC\s|CEM\sCode\s)", re.I)

    blocks, cur_code, buf = {}, None, []
    for line in raw:
        if start_pat.match(line):
            if cur_code and buf:
                blocks[cur_code] = "\n".join(buf).strip()
            cur_code, buf = line, []          # new block
        else:
            buf.append(line)
    # flush last block
    if cur_code and buf:
        blocks[cur_code] = "\n".join(buf).strip()

    df = pd.DataFrame({
        "AFSC_Code": list(blocks.keys()),
        "description": list(blocks.values())
    })
    return df

# 🔧— 2.  Parse & merge
all_df = pd.concat([parse_afsc_csv(p) for p in files], ignore_index=True)
all_df = all_df[all_df["description"].str.len() > 0]      # drop empties

print(f"✅ Parsed {len(all_df):,} AFSC entries")
display(all_df.head(3))

# 🔧— 3.  Save cleaned dataset next to originals
out_path = base / "afsc_clean.csv"
all_df.to_csv(out_path, index=False, encoding="utf-8")
print("📄 Saved cleaned file →", out_path)
```

This parser identifies each AFSC section by looking for lines beginning with "AFSC" or "CEM Code," then consolidates all subsequent text into a single description field until the next AFSC

identifier. The resulting dataset contained 123 AFSC descriptions in a clean, two-column format (AFSC_Code and description) suitable for LAiSER processing.

Results Generation:

The LAiSER extraction process demonstrated strong performance in processing military occupational text and mapping it to standardized skill frameworks. Using a similarity threshold of ≥0.85, the single-CPU Colab run completed in approximately three hours, yielding a comprehensive skills inventory:

The process generated 2,352 total skill mappings across all 123 AFSC descriptions, providing complete (100%) coverage of the tested corpus. From these unstructured descriptions, LAiSER identified 289 unique raw skill phrases that mapped to 337 distinct ESCO/OSN taxonomy terms, demonstrating the system's ability to normalize similar expressions to standardized classifications. Please see the “LAiSER Extraction Analysis” for the code that extracted the below metrics.

Skill Distribution Patterns:

The distribution of skills across specialties revealed significant variation:

- Average of 19.1 skills per AFSC with a range from 1 to 158 skills
- Medical roles showed the richest skill profiles (School of Medicine commissioned officers: 158 skills)
- Technical specialties also demonstrated extensive skill requirements (Special Operations 71S1: 126 skills; Developmental Engineering 62E1*: 84 skills)
- Five specialties received minimal mapping (only 1 skill identified)

Quality and Confidence Metrics:

The extraction quality exceeded expectations, with:

- Average correlation coefficient of 0.881 (well above the 0.85 threshold)
- 18.6% high confidence matches (correlation >0.90)
- 3.4% very high confidence matches (correlation >0.95)

These metrics indicate strong confidence in the mapped relationships and validate the extraction methodology's precision.

Cross-AFSC Skill Analysis:

The analysis identified both common and specialized competencies:

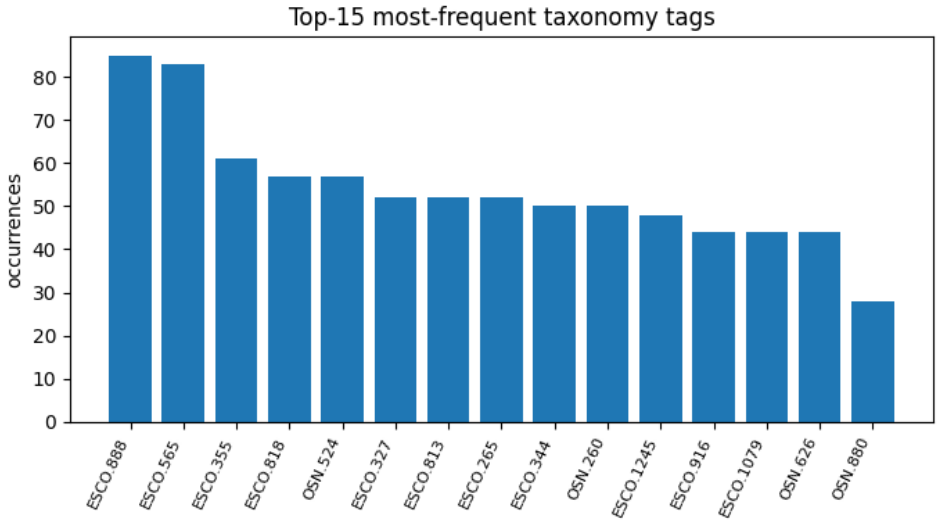
- 15 universal skills appeared in more than 20% of all AFSCs, representing potentially core military competencies

- 90 field-specific skills (26.7% of unique skills) were found in only a single AFSC
- The top three most frequently extracted skills were all management systems:
    - ESCO.888 - "manage environmental management system" (85 occurrences)
    - ESCO.565 - "Prince2 project management" (83 occurrences)
    - ESCO.355 - "establish an Information Security Management System" (61 occurrences)

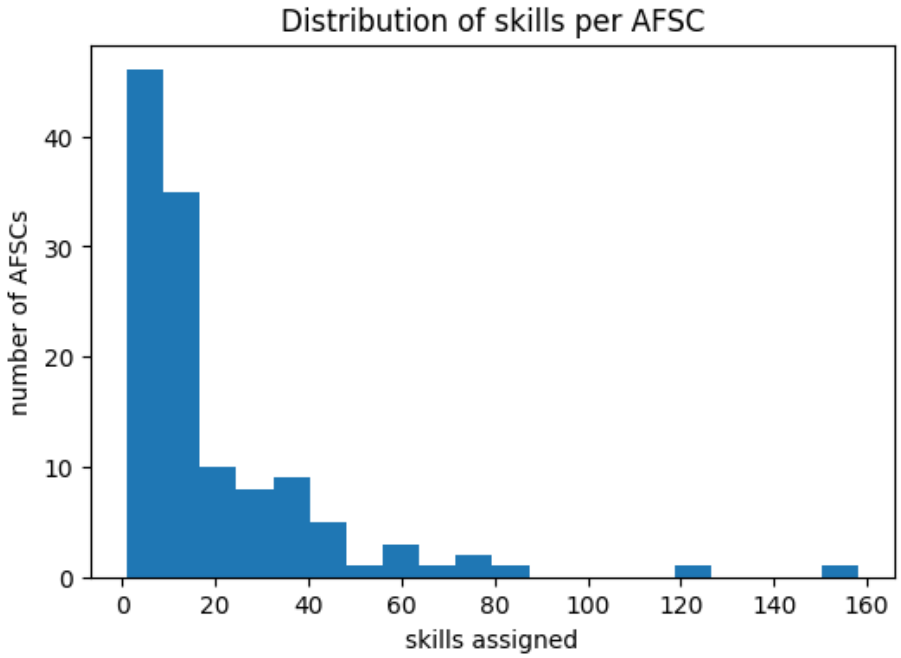
A notable pattern emerged during reviewing LAiSER's results: terms such as *project-management system*, *information-security management*, and *environmental-management program* appeared with unusually high frequency across otherwise unrelated AFSCs. Although the Air Force clearly addresses project oversight, data protection, and perhaps environmental stewardship, the specific civilian phrasing (e.g., "ISO 27001 information-security management system") almost certainly reflects an alignment artefact rather than an authentic skill emphasis.

LAiSER's extractor engine is optimized for open-source labor datasets; when it encounters familiar keywords in military prose, it gravitates toward the closest civilian labels, even when the underlying context diverges. This illustrates a broader limitation of applying civilian taxonomies to military texts.

Similar phrases may be assigned to identical skill tags even though their operational meaning within a military environment differs substantially from the civilian domain. Consequently, rigorous SME adjudication is likely required. In future iterations I'd likely introduce Air-Force–specific "management system" categories or embed disambiguation rules that weigh surrounding mission language rather than isolated keywords. These refinements will preserve the authenticity of military competencies while still benefiting from the structure and interoperability that civilian frameworks provide. Additionally, please see the "Potential Case Study – Intelligence Officer (AFSC 14N)" section of the paper, where I give my thoughts on the skills associated with my own AFSC.



Independent Study Final Paper
Kyle Hall, DATS 6499, 4/25/2025



Prototype Comparison:

While the two approaches differ fundamentally in their starting points—structured civilian data versus unstructured military text—both demonstrate viable paths to automated skill taxonomy development.

Caveat on dataset scope: The zero-overlap finding should be interpreted considering the two pipelines' input differences. LAiSER parsed 123 AFSC header sections drawn directly from the combined AFOCD + AFECD corpus (~2 million words of raw military prose). In contrast, the LLM prototype processed a small, 30-AFSC sample built from the O*NET / ESCO / Crosswalk table. Therefore, some of LAiSER's 337 unique tags may reflect skills associated with AFSCs that were simply not included in the pilot-scale LLM run. Even with this caveat, the absence of shared terms reinforces the conclusion that top-down extraction from service-specific documents uncovers specialized competencies that general-purpose, civilian-sourced taxonomies do not capture, highlighting the complementary value of integrating both approaches.

Aspect	LLM Prototype ("bottom-up")	LAiSER Prototype ("top-down")
Primary input	Structured civilian datasets (O*NET 29.1, ESCO 1.1.1, DoD Military-Civilian Crosswalk)	Unstructured AFSC narrative text from AFOCD & AFECD (≈ 2 M words)
Methodology	Merge datasets → feed sample to GPT-4 Turbo / Claude 3.5 → generate hierarchy	Parse CSV lines → spaCy + SkillNer extract raw skills → cosine-match to ESCO tags

Aspect	LLM Prototype ("bottom-up")	LAiSER Prototype ("top-down")
Skills source	Pre-categorized civilian skill labels	Skill phrases mined directly from Air-Force classification documents
Taxonomy creation	LLM builds new 3-level hierarchy (Domain → Category → Skill)	LAiSER maps each extracted phrase to **existing** ESCO / OSN IDs (no new hierarchy)
Hierarchical structure	Multi-level outline (199–205 nodes in pilot)	Flat list of aligned skills; hierarchy can be added later if desired
Military context	AFSC context inferred via Crosswalk links	Authentic AFSC language processed natively (jargon, acronyms, etc.)
Relationship mapping	Skills ↔ AFSC mapping retained via original Crosswalk IDs	ESCO tag ↔ AFSC link created with similarity score (avg ≈ 0.85 in pilot)
Strengths	• Generates bespoke hierarchy • Integrates multiple civilian sources • Good for strategic domain mapping	• Extracts skills straight from source text • Leverages standard ESCO IDs • Captures military-specific terminology (337 tags not in civilian set)
Typical use cases	• High-level skill architecture • Cross-dataset analytics & gap analysis	• Rapid processing of classification docs / syllabi • Readiness analytics or curriculum tagging

III. Results

Dual Proof of Concept:

Both pipelines were executed as constrained pilot studies to validate technical feasibility while managing resource constraints.

Aspect	Civilian Taxonomy Prototype	LAiSER Prototype
Scope	30 AFSCs (15 Officer & 15 Enlisted)	123 AFSC header sections
Data volume	~18,000 unique skill labels in merged table	~2,300 aligned skill tags (19.1 per AFSC)
LLM engines	GPT-4 Turbo and Claude 3.5 Sonnet	SkillNer + spaCy, cosine similarity alignment

Aspect	Civilian Taxonomy Prototype	LAiSER Prototype
Key proof points	✓Hierarchical structure emerged. ✓Cross-service civilian tags could be attached to AFSCs. ⚠ Coverage limited; costly to scale with GPT credits.	✓100 % AFSCs received ≥ 1 skill tag. ✓337 military-specific tags surfaced that are absent from the civilian hierarchy. ⚠ Output needs post-processing & GPU optimization.
Overlap	0% direct tag overlap between methods, indicating complementary coverage	
Runtime	~45 min for AFSCs via paid API	~300 min for 123 AFSC headers on CPU

IV. Discussion/Results

A. Multi-Method Insights

Methodological Synergies:

- The "bottom-up" LLM pipeline excels at imposing a clean, three-level hierarchy on long skill lists, making the taxonomy easy to browse and maintain.
- The "top-down" LAiSER pipeline digs directly into Air Force classification prose, surfacing terminology that civilian databases simply do not contain.
- Together, they deliver both structure (the LLM hierarchy) and coverage (LAiSER's military-specific vocabulary).

Computational Considerations:

- The LLM approach requires a merged "mega-skills" table but consumes paid API tokens; even the 30 AFSC pilot costs noticeably more than a local run.
- LAiSER, by contrast, needed a one-time CSV conversion of the AFOCD/AFECD PDFs, then processed all 123 AFSC headers on a laptop-class CPU in about three hours.
- The free and open-source nature of LAiSER makes it an appealing option for large-scale implementation.

Coverage vs. Structure Trade-off:

- The LLM prototype produced smaller outlines (GPT-4 Turbo = 27 nodes; Claude 3.5 Sonnet = 100 nodes) but with clear Domain → Category → Skill nesting, improving navigability.
- Because each pipeline used a different input corpus (merged civilian skills vs. military narrative), the near-zero overlap in tags is expected and indicates complementary coverage.

Integration potential:

In a future production setting, the two pipelines would be executed sequentially. First, LAiSER would mine raw skill phrases from the entire AFOCD / AFECD corpus. Those phrases would then be deduplicated and mapped to standard ESCO identifiers using cosine-similarity scoring. Next, each AFSC's cleaned skill list would be sent to a large-language model, which would cluster the items into a three-tier hierarchy (domain → category → skill). Only the low-confidence matches or brand-new nodes would be routed to subject-matter experts for adjudication. This hybrid workflow balances breadth and structure while keeping API costs predictable and SME effort highly focused.

Validation strategy:

Because the two pipelines generate different kinds of artifacts, they call for complementary validation strategies. For large-language-model hierarchies, I would likely need a SME to check the output. Experts scan the outline, confirm that the domains are meaningful, spot missing or duplicated skills, and verify that AFSC assignments reflect real-world practice.

LAiSER's output, by contrast, lends itself to quantitative quality control, analysts can inspect the distribution of cosine-similarity scores, set acceptance thresholds (e.g., ≥ 0.90), and calculate precision-recall metrics on a held-out, hand-labelled subset.

The optimal workflow therefore layers the two: apply statistical filters to LAiSER mappings to flag low-confidence pairs, feed those and the newly created LLM nodes to SMEs for targeted review, iteratively adjust similarity cut-offs, and re-run until both the numeric indicators and expert judgments converge. This blend of qualitative oversight and quantitative gating produces a taxonomy that is both empirically defensible and operationally credible.

Because of the scope of the independent study, there was only cursory validation was possible for either the LLM's or LAiSER. The intent was to prototype the pipelines to see the viability of both LAiSER and LLM taxonomy creation. The above strategy is for future implementation and would require access to SMEs to truly vet the accuracy of the taxonomies.

B. Implications for Military Skill Development

There's still much work to be done to wrangle idiosyncrasies associated with both pipelines and datasets, but given more time, our work (myself and the LAiSER team) could have immediate applications for the below:

- Identifying curriculum gaps in training programs
- Enabling data-driven workforce planning
- Improving veteran transition services through better skill translation
- Standardizing competency descriptions across military branches

V. Recommendations for LAiSER Integration

Future LAiSER Enhancements:

Train a DoD-tuned model:

Fine-tune LAiSER's sentence-embedding backbone on AFOCD/AFECD text (plus other military doctrine) to boost recall and reduce false positives in skill extraction.

Add Windows support:

Patch the vLLM / CUDA dependency chain so LAiSER installs cleanly on Windows workstations, enabling analysts to run experiments without a VM.

Build a hybrid pipeline:

Use LAiSER for high-recall skill extraction, then pass the tagged output to an LLM (GPT-4 Turbo, Claude 3.5, etc.) that arranges the skills into a multi-level taxonomy.

Package for local GPU use/acceleration:

Publish LAiSER as a pip/conda package with straightforward install and use; when a local NVIDIA or Intel GPU is available, default to GPU inference for 5-10 × throughput.

Potential Case Study – Intelligence Officer (AFSC 14N):

The Intelligence career field offers an ideal test of the extraction pipeline because it blends traditional military functions (collection management, operational planning) with fast-moving digital disciplines (data science, cloud technologies). LAiSER parsed the 14N entry-level description and returned 37 skill tags (29 ESCO, 8 OSN) with an average correlation of 0.89.

As a 14N practitioner, I found that ~80 % of the tags reflect real, day-to-day competencies, e.g. Collection-management software (ESCO 477) and Data-science (ESCO 1186). A handful are close but need re-labelling: PRINCE2 project management (ESCO 565) and Environmental-management system (ESCO 888) appear because the term "management system" is pervasive in the ESCO hierarchy, not because 14Ns run environmental programs. Such false-positive patterns reinforce the need for threshold tuning and SME filtering called for in the recommendations. Given more time I'd like to work with the LAiSER team as a SME to better focus on the development of the project and enhance its capabilities.

Supplementary Exploration: n8n Workflow (de-scoped)

I would be remiss not to mention my work with n8n. During the February–March phase, I prototyped an n8n workflow to orchestrate an end-to-end pipeline which cleaned, formatted and merged the ESCO and O*NET crosswalks into a single usable file.

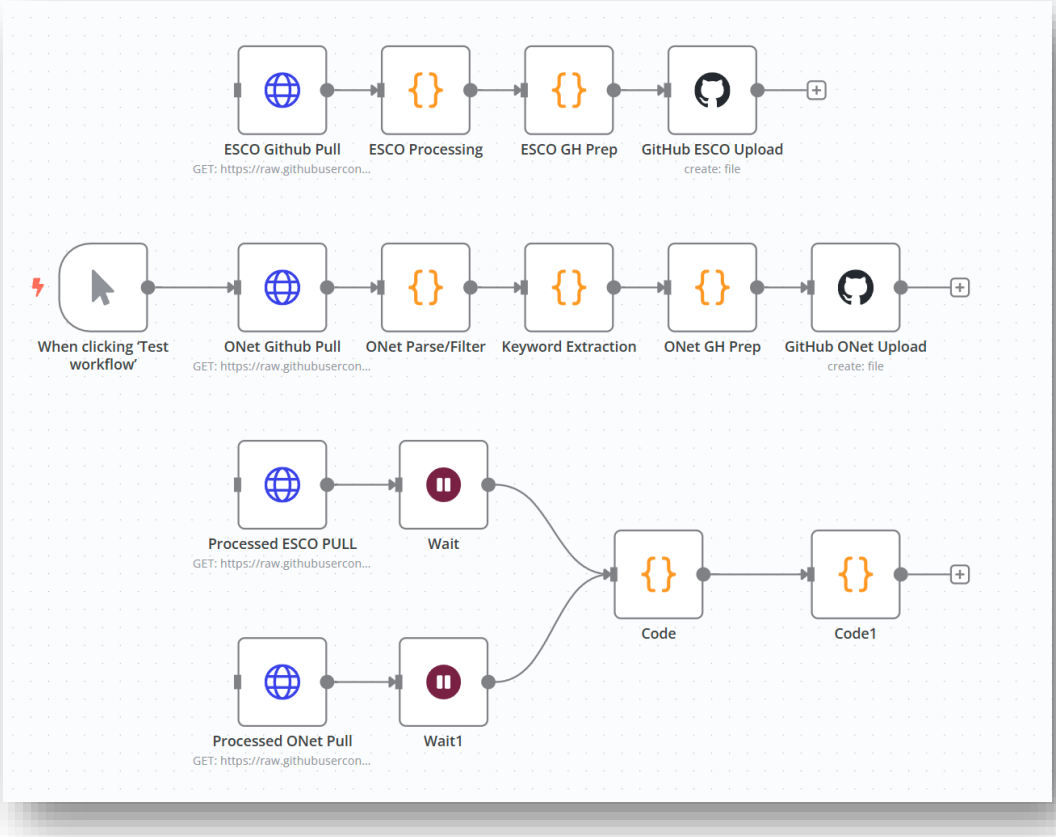
The intent behind the n8n prototype was to prove that the entire skills extraction workflow could operate hands off, and would eventually integrate, nodal AI agents in the taxonomy creation. A webhook trigger would launch the job, fetch the original ESCO and ONet files from GitHub, and convert them to CSV. n8n would then process and prepare each file in a sub-pipeline, and then

reupload the cleaned process version back to GitHub. The final goal was to merge all three files into a single, usable form ready for LLM ingestion. A final pipeline in n8n would pull the merged file from GitHub, then process it with a corresponding model, and capture the resulting skills table. After minimal post processing, the workflow would push the refreshed dataset back to GitHub, and a scheduler node would re-run the entire chain nightly so that any future AFSC updates automatically propagated through the taxonomy. While promising, this automation approach requires further development before operational use.

The n8n implementation was ultimately de-scoped due to:

- 14-day trial limitation creating accessibility barriers
- Evolving LAiSER API requiring frequent rebuild
- Performance issues with larger datasets in the web interface

Figure 3: n8n Workflow



VI. Conclusions

This project demonstrated the technical feasibility of developing an automated, machine-readable skills taxonomy for military occupations. By employing complementary approaches structured civilian data processing and direct military text extraction, the team and I produced a comprehensive prototype that captures both hierarchical organization and military-specific terminology.

Key accomplishments include:

- Development of a DoD-centric skills taxonomy prototype / pipeline
- Complete automation of the extraction and alignment process
- Identification of 337 military-specific skills missing from civilian frameworks
- Proof of concept processing for 123 AFSCs with 100% coverage

The cost-benefit analysis strongly favors automation, a massive reduction in analysis time compared to manual methods. While further refinement is needed, this approach establishes a foundation for improving military workforce planning, training development, and veteran transition services.

VII. Appendices

A. Runtime & Hardware Specifications

Pipeline	Compute Environment	Key Libraries / Versions	Avg. Throughput*	Total Runtime*
LLM Taxonomy – GPT-4 Turbo	OpenAI API (cloud)	n/a (cloud-hosted)	~20 TPM †	7 min / 24-AFSC prompt
LLM Taxonomy – Claude 3.5	Anthropic API (cloud)	n/a (cloud-hosted)	~30 TPM †	8 min / 30-AFSC prompt
LAiSER Extraction	Google Colab VM (CPU-only, free tier)	Python 3.11, dev-lasier 0.2.24, spaCy 3.7, SkillNer 0.1.5	~47 lines / sec	3 h 12 m

B. Complete Skill Lists

https://github.com/Kyleinexile/Military-Skills-Taxonomy/tree/main/results/laiser

C. Code Repository

https://github.com/Kyleinexile/Military-Skills-Taxonomy

D. Glossary

Term	Definition
AFSC	Air Force Specialty Code – 4–5-character identifier for an Air Force job series
AFOCD / AFECD	Officer / Enlisted Classification Directories (annual PDF publications)
ESCO	European Skills, Competences, Qualifications and Occupations framework
KSA	Knowledge, Skills, and Abilities
LAiSER	*Leveraging AI for Skills Extraction & Research* – GWU NLP pipeline
LLM	Large Language Model (e.g., GPT-4 Turbo, Claude 3.5)
O*NET	U.S. Department of Labor Occupational Information Network
SkillNer	spaCy-based named-entity recogniser for skill phrases
vLLM	Open-source, GPU-optimised LLM inference engine
DoD	Department of Defense
GPU	Graphics Processing Unit, used for accelerated computing
CPU	Central Processing Unit
spaCy	Open-source library for advanced Natural Language Processing
API	Application Programming Interface
SME	Subject Matter Expert
n8n	Open-source workflow automation tool mentioned in your supplementary exploration