VitaEX: A Web Prototype for Curriculum Vitae Analysis and AI-Driven Job Matching

Diego Martínez Méndez[1], Abdiel Reyes Vera[2], Elizabeth Moreno Galván[3], Oscar Huerta Villanueva[4], and Luis Fernando Valle Hernández[5]

[1,2,3,4,5] Escuela Superior de Cómputo, IPN, Mexico City, Mexico
[2] Centro de Investigación en Computación, IPN, Mexico City, Mexico
[1]dmartinezm1707@alumno.ipn.mx, [2]areyesve@ipn.mx, [3]emorenog@ipn.mx,
[4]ohuertav2100@alumno.ipn.mx, [5]lvalleh1800@alumno.ipn.mx

Abstract. Recruiters now depend on *applicant-tracking systems* (ATS) that filter between 75 % and 98 % of résumés before a human review, often rejecting qualified graduates whose CVs lack machine-readable keywords. This automated gatekeeping disproportionately harms junior professionals in Artificial Intelligence (AI) across Mexico's emerging tech hubs. *VitaEX* tackles the problem by shifting the ATS from gatekeeper to mentor, showing students *what* to add, *why* it matters, and *how* it aligns with real vacancies.

The prototype is a three-tier web platform—React front-end, Django REST controller, MongoDB / PostgreSQL data layer—developed through the Spiral Model. A resilient Python scraper harvested LinkedIn postings from November 2024 to May 2025 for eight AI roles across five Mexican cities. Using the CRISP-DM framework, the team cleaned and normalized 5k+ vacancies, then mined more than 500,000 interpretable association rules with Apriori, creating a hybrid recommender that fuses rule confidence with SBERT cosine similarity.

Beyond raw metrics, VitaEX delivers transparent explanations that boost ATS pass rates, raise skill awareness, and promote regional equity. This article synthesizes every major facet of the Spanish thesis—literature review, large-scale data collection, rule-based recommendation, and full-stack architecture—demonstrating how interpretable data-mining and modern web engineering can narrow the employability gap for AI graduates in developing economies.

Keywords: Curriculum optimisation · ATS · Apriori · CRISP-DM · Web scraping · NLP · Django · ESCOM-IPN

1 Introduction

A *curriculum vitae* (CV) is, in the words of the Royal Spanish Academy, a document that summarises a person's academic record, work history, skills and achievements for professional or academic purposes [1]. In today's labor market that summary must first persuade algorithms. Harvard Business School and Accenture estimate that **between 75 % and 98 %** of all applications in large firms are screened by applicant-tracking systems (ATS) before any human review, quietly excluding millions of otherwise qualified "hidden workers" whose CVs lack certain keywords or credentials [2]. In Mexico the impact is amplified: the *Encuesta Nacional de Egresados 2023* reports that **46.3 %** of new graduates describe the job-search process as "difficult", citing ATS incompatibility as a leading barrier [3]. Paradoxically, many of those applicants hold precisely the AI-related skills that employers claim to need most urgently.

VitaEX addresses this mismatch by converting the ATS from gatekeeper to mentor. The prototype continuously scrapes AI-oriented vacancies from LinkedIn and regional job boards across five Mexican tech hubs—Mexico City, Guadalajara, Monterrey, Querétaro and Puebla—building a local demand repository updated monthly. After normalising more than 5 000 postings, the system mines over **500 000** interpretable association rules with the *Apriori* algorithm and enriches them with semantic similarity scores produced by SBERT. The result is a transparent recommender that tells students *what* to add (e.g., "Docker", "MLOps", "English B2"), *why* it matters, and *how* it improves their match with specific vacancies.

VitaEX is implemented as a three-tier web architecture: a responsive React front-end, a Django REST controller that enforces JWT-based authentication, and a hybrid MongoDB/MySQL data layer. Celery workers orchestrate asynchronous scraping and model retraining, ensuring that recommendations reflect real-time market shifts.

By offering localised, open-access and fully explainable feedback, VitaEX pursues three concrete goals: *(i)* empower students by revealing actionable skill gaps; *(ii)* streamline recruiter workflows by surfacing CVs already aligned with job requirements; and *(iii)* promote regional equity by delivering data-driven career guidance to public-university graduates across Mexico. This article details the theoretical foundations, CRISP-DM methodology and system architecture that together demonstrate how transparent data-mining and modern NLP can narrow the employability gap for AI graduates in developing economies.

2 Theoretical Framework

In this chapter, we focus on the principal techniques and technologies for the prototype

2.1 Apriori Algorithm

The algorithm is an unsupervised machine learning technique, used for discover frequent patterns and relationships between items in large datasets. It finds item groups that frequently occur together in transaction data and uncovers significant patterns from them. This method is commonly applied in areas such as recommendation systems, fraud prevention, and inventory control, among others.The key assumption (the Apriori property) is that if an itemset is frequent, all its subsets must also be frequent. Conversely, if an itemset is infrequent, any larger set containing it will also be infrequent. This principle helps reduce computational complexity by pruning unnecessary candidate itemsets early. [4]

Steps of the Apriori Algorithm

1. **Define the minimum support (min support):**
   A minimum threshold is set that an itemset must meet to be considered as frequent.
2. **Generate frequent 1-itemsets:**
   The occurrences of each item in the database are counted, and those that do not meet the minimum support are discarded.
3. **Generate candidate k-itemsets:**
   Based on the frequent itemsets of size $k-1$, new candidates of size k are generated by combining items.
4. **Eliminate infrequent candidates:**
   The support of each candidate itemset is checked in the database, and those that do not meet the minimum support are removed.
5. **Repeat the process:**
   The process of generating and filtering larger itemsets continues until no more frequent itemsets can be found.
6. **Generate association rules:**
   From the frequent itemsets, rules of the form $A \rightarrow B$ are generated by evaluating metrics such as confidence and lift, keeping only those that meet the specified thresholds.

Example: Bookstore Recommendations Imagine an online bookstore analyzing customer purchases. The algorithm might discover that:

> Customers who buy *Data Science Handbook* frequently also purchase *Python for Beginners*.

If 30% of transactions include both books (high support) and 75% of buyers of the first book also buy the second (high confidence), the store can recommend them together.

Key Metrics for Rule Evaluation To evaluate the quality of association rules of the form $A \rightarrow B$, the following metrics are commonly used:

1. **Support:**
   Indicates how frequently the itemset $A \cup B$ appears in the dataset.

$$\text{Support}(A \rightarrow B) = \frac{\text{Transactions containing } A \cup B}{\text{Total number of transactions}}$$

2. **Confidence:**
   Measures the likelihood that itemset B is also bought when itemset A is bought.

$$\text{Confidence}(A \to B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$

3. **Lift:**
   Evaluates how much more likely B is purchased when A is purchased, compared to when B is purchased independently.

$$\text{Lift}(A \to B) = \frac{\text{Confidence}(A \to B)}{\text{Support}(B)} = \frac{\text{Support}(A \cup B)}{\text{Support}(A) \times \text{Support}(B)}$$

   A lift value:
   - Greater than 1 indicates a positive correlation.
   - Equal to 1 indicates independence.
   - Less than 1 indicates a negative correlation.

2.2 Web Scrapping

Is an automated technique for extracting data from web pages. It has become a cornerstone for collecting large volumes of public information available on the Internet. Its relevance lies in its ability to feed statistical analyzes, market studies, and machine learning models with data that would otherwise require costly manual capture in terms of time and resources. [5]

Scrapping Python Tools Python offers a mature ecosystem for each phase of the process:

- **Requests**: Simplifies sending HTTP requests and downloading HTML source code from static sites [6].
- **Beautiful Soup**: Allows navigation and filtering of the DOM tree using CSS or XPath selectors, making it suitable for medium-sized projects [6].

Web Scraping Flow Description Our Python script scrapes LinkedIn job listings using a two-phase approach:

Phase 1: Collect Job IDs

1. **Initialization:**
   - Sets search parameters (job title: "Data Analyst", location: "CDMX")
   - Defines maximum jobs to collect (700)
   - Prepares rotating user-agents to avoid bot detection
2. **Pagination Loop:**
   - Constructs search URL with pagination parameters (`start` value)
   - Randomly rotates user-agents for each request
   - Handles rate limiting (HTTP 429) with extended random delays (30-60s)
   - Breaks loop on non-200 status codes or empty results
3. **Job ID Extraction:**
   - Parses HTML response with BeautifulSoup
   - Extracts unique job IDs from `data-entity-urn` attributes
   - Maintains unique ID list and tracks progress
   - Implements random delays (3-6s) between requests

Phase 2: Extract Job Details

1. **Regex Preparation:**
   - Compiles regex patterns for:
     - Job types (Full-time, Remote, etc.)
     - Experience levels (Junior, Senior, etc.)
     - Technical skills (Python, SQL, AWS, etc.)
2. **Detail Extraction Loop:**
   - Processes each job ID from Phase 1
   - Fetches job detail page using LinkedIn's job posting API
   - Handles rate limiting with extended delays (30-60s)
   - Extracts full job text for regex pattern matching
3. **Data Parsing:**
   - Extracts job type and experience level using regex
   - Parses job description HTML section
   - Identifies technical skills from description using regex
   - Captures key elements:
     - Job title, company name, posting date
     - Applicant count, location, salary
     - Job description, application link
4. **Data Storage:**
   - Stores parsed data in dictionary format
   - Appends to job list with random delays (3-6s) between requests
   - Exports final dataset to CSV using pandas

2.3 MongoDB

MongoDB is a leading open-source NoSQL database system designed for modern application development. As a document-oriented database, it stores data in flexible JSON-like documents (BSON format) rather than rigid tables, allowing for dynamic schemas that adapt to evolving data requirements.

Key advantages include horizontal scalability through sharding, enabling distributed data processing across clusters to handle large volumes of job listings efficiently. Its rich query language supports complex searches on nested data structures, essential for filtering jobs by multiple criteria like skills, experience levels, and locations. Automatic indexing accelerates data retrieval for recommendation queries, while aggregation pipelines facilitate advanced data processing like identifying frequent skillsets. In our prototype, MongoDB provides the flexible storage layer for scraped job data, supporting rapid iteration of the recommendation engine's data model without schema migration constraints [7].

2.4 Django

Django is a high-level Python web framework following the Model-View-Controller (MVC) architectural pattern, renowned for its "batteries-included" philosophy. It accelerates secure web application development through built-in components like authentication systems, ORM abstraction, and admin interfaces.

The framework's Object-Relational Mapper (ORM) enables database-agnostic data operations, allowing seamless interaction with MongoDB via libraries like Djongo while maintaining Pythonic syntax. Django REST Framework extends this capability to build robust APIs for serving job recommendations to frontend clients. For our prototype, Django orchestrates the entire application workflow - from ingesting scraped data through custom management commands, processing recommendations via the Apriori implementation, to delivering personalized job feeds through REST endpoints. Its built-in security features mitigate common vulnerabilities like XSS and CSRF during data presentation [8].

3 Related Work

Despite the proliferation of tools aimed at improving job-matching and résumé optimization, a critical gap persists between algorithmic sophistication and user-centered applicability—especially in the context of emerging economies. Existing systems fall short in one or more of the following dimensions: transparency of recommendation, alignment with local labor market conditions, explainability of models, and accessibility for non-technical student populations. This section examines the state of the art not merely as a list, but as a critical landscape in which VitaEX carves out a novel, high-impact space.

3.1 Template-Based CV Builders: Form Without Substance

Widely used platforms such as LinkedIn Resume Builder and CVapp offer elegant visual templates for building résumés. These tools emphasize formatting aesthetics and usability but do not provide personalized feedback about content quality or job-market fit. While this might suffice for experienced professionals with clarity about industry expectations, it is insufficient for students or recent graduates unfamiliar with employer language or keyword strategies.

In contrast, VitaEX embeds its recommendation engine at the semantic and data-mining level, showing not just how a CV should look, but what it should contain based on statistically significant job patterns. It bridges the gap between document design and strategic content optimization.

3.2 Machine Learning Systems: Opaque Recommendations

Several academic efforts have introduced ML-based tools for job recommendation. ResumeNet [11], for instance, scores résumés using a neural network model trained on job application outcomes. Similarly, GIRL [10] (Generative Job Recommendation with LLMs) employs large language models to align candidate profiles with potential vacancies.

While technically robust, these systems suffer from low explainability. The models act as black boxes, failing to communicate to users why a given job was recommended or what changes might improve their match score. This opacity creates a trust gap, particularly among first-time job seekers unfamiliar with the underlying algorithms.

VitaEX addresses this issue directly by combining Apriori rules which are inherently interpretable with SBERT-based semantic similarity, ensuring every suggestion is traceable, explainable, and actionable.

3.3 Academic Prototypes: Narrow Scope, Recruiter-Centric

Some initiatives have applied Natural Language Understanding (NLU) techniques for CV parsing [9] or used neural networks for recruiter-facing tools [12]. These works typically optimize HR workflows and thus cater to the recruiter rather than the candidate.

This recruiter-centric orientation limits their utility for students, who require guidance to build competitive profiles in the first place. Moreover, these systems often rely on proprietary data from private firms or Western job markets, limiting their generalizability to Latin America and public-university graduates.

VitaEX, in contrast, is explicitly student-centered, trained on publicly scraped data from Mexican tech hubs, and built with open-access principles. Its transparency and local data make it both more democratic and more relevant to underserved populations.

3.4 Recommender Systems: Lack of Contextual Anchoring

General recommender platforms like CVMATCHER and Jobania provide basic keyword matching but fail to incorporate rich contextual dimensions like location, seniority, modality, or evolving industry trends. Their flat, surface-level analysis often leads to suggestions that are either too generic or too irrelevant to act upon.

VitaEX improves on this by transforming job postings into structured transactions:

$$\mathcal{T} = \{\texttt{skills}, \texttt{experience level}, \texttt{city}, \texttt{modality}\}$$

This structure enables multidimensional filtering, meaning that a student from Puebla with intermediate experience receives different suggestions than a senior candidate in Monterrey, even if they share some core skills.

3.5 Interpretability as an Educational Tool

A crucial innovation of VitaEX lies in treating recommendations as explanations, not just predictions. For example, the rule:

$$\texttt{"Data Analysis", "Intermediate Experience"} \Rightarrow \texttt{"Machine Learning"} \quad (\text{Confidence} : 91\%, \text{Lift} : 2.8)$$

not only suggests an improvement but also educates the user about statistical associations in the job market. This educational feedback loop is entirely missing from most systems reviewed, yet it is essential for student empowerment.

The critical review above reveals that most prior art lacks one or more of the following: explainability, local context, user-centered design, and educational impact. VitaEX is unique in its ability to synthesize interpretable rule-mining, contextual embeddings, and modern web architecture into a system that is transparent, actionable, and regionally grounded.

This contribution is particularly significant for underrepresented student populations, bridging the gap between academic preparation and algorithmic hiring processes that are often opaque and exclusionary. It not only competes with global systems in sophistication but surpasses them in transparency, equity, and adaptability.

Applications and Research Works	**Recommends based on job offers**	**Generates CV templates**	**Uses Machine Learning**	**Job offer recommendation**	**Student-oriented and free**
Application of NLU methods for CV recommendation [9]	✓	✗	✗	✗	✗
Generative Job Recommendations with LLM (GIRL) [10]	✓	✗	✓	✗	✗
ResumeNet [11]	✗	✗	✓	✗	✗
Neural networks for recruitment [12]	✗	✗	✓	✗	✗
LinkedIn Resume Builder [13]	✓	✓	✗	✗	✓
CVapp [14]	✗	✓	✗	✗	✗
Jobania [15]	✓	✓	✗	✗	✗
CVMATCHER [16]	✓	✓	✓	✓	✗
VitaEX (proposed system)	✓	✓	✓	✓	✓

Table 1: Comparison of systems and research works related to CV analysis and job recommendation

4 Methodologies

To ensure disciplined software delivery *and* reproducible data science, VitaEX combines two complementary frameworks. The **Spiral Model** governs system development, guiding each release through goal setting, risk analysis, engineering, and stakeholder validation. Concurrently, all data activities follow the **CRISP-DM** standard, whose six-step cycle—business understanding, data understanding, preparation, modeling, evaluation, deployment—provides full traceability from raw web data to production rules.

Spiral Model. Every loop begins by defining objectives and identifying risks; mitigations are then prototyped and evaluated before the next iteration. This cadence enabled early validation of LinkedIn-scraping scalability, Apriori performance, and security controls, reducing late-stage surprises.

CRISP-DM. Within each Spiral loop the team executes CRISP-DM: *(i)* translate recruiter needs into scraping targets; *(ii)* profile incoming job data; *(iii)* clean, tokenize, and canonicalize skills; *(iv)* mine Apriori association rules; *(v)* benchmark rule quality; and *(vi)* publish refreshed models via the REST API. The outputs of CRISP-DM feed the next Spiral cycle, while new Spiral deliverables provide infrastructure for subsequent data iterations.

Phase	**Standard Definition**	**Implementation in *VitaEX***
1. Business Understanding	Translate business goals into a well-framed analytics problem.	AI graduates rejected by ATS keyword filters. Goal: turn the ATS into a "mentor" that pinpoints missing skills.
2. Data Understanding	Collect, explore and assess raw data quality.	Pilot scrape (Nov 2024) of 500 LinkedIn ads to inspect fields and noise. EDA: city/skill/experience histograms, missing-value audit, HTML encoding issues.
3. Data Preparation	Clean, transform and format data for modeling.	Lower-casing, accent removal, strip HTML. Spanish spaCy POS-tagger to keep nouns/verbs/adjectives. Custom ontology (*ML→machine learning*). Each vacancy becomes a transaction $\mathcal{T}$ = {skills, city, modality, experience}. Output: 5 313 clean records.
4. Modeling	Choose algorithms, build models, tune parameters.	Apriori algorithm
5. Evaluation	Verify models meet business and quality objectives.	Offline metrics: Precision, confidence, lift
6. Deployment	Put the solution into production and plan maintenance.	REST endpoint `recommend` exposes rules and rankings.

Table 2: Step-by-step application of CRISP–DM in the *VitaEX* project.

4.1 Data Acquisition and Ingestion

Target definition. The scraping focused on eight AI-related roles—*AI Engineer*, *Machine Learning Engineer*, *Data Scientist*, *NLP Engineer*, *Computer Vision Engineer*, *Deep Learning Researcher*, *MLOps Engineer*, and *Data Analyst*—across five Mexican tech hubs: *Ciudad de México (CDMX)*, *Guadalajara*, *Monterrey*, *Querétaro*, and *Puebla*.

Scraping pipeline. A resilient Python crawler (`requests` + `BeautifulSoup`) executed a two-phase loop: *(i)* collect paginated job IDs using rotating user-agents and exponential back-off; *(ii)* retrieve detail pages through LinkedIn's public API, parsing HTML and JSON payloads. All records are stored in MongoDB as BSON documents `{id, title, company, city, date, experience, description, skills[]}`. Monthly Celery-based cron jobs refresh the corpus and flag updates via change streams.

4.2 Data Preparation

1. **Cleaning & normalisation:** lower-casing, accent stripping, HTML removal.
2. **Filtering:** spaCy's Spanish model retains nouns, verbs, adjectives.
3. **Skill mapping:** custom ontology maps synonyms (*"ML"*→*"machine learning"*).
4. **Feature engineering:** each posting becomes a transaction $\mathcal{T} = \{$`skills, city, modality, experience`$\}$.

4.3 Data Preprocessing and Noise Handling

The web scraper extracted over 5,000 AI-related job postings across five major Mexican cities between November 2024 and May 2025. These postings presented high heterogeneity in formatting, language use, and keyword conventions due to varying employer practices and platform encoding.

To address this, a multi-phase cleaning pipeline was applied. First, **deduplication** logic discarded redundant entries based on URL and job ID hashing. Second, **regex-based cleaning** were applied to normalize inconsistent labels (e.g., "machine learning engineer" vs. "ML Engr") and to standardize experience tags. Third, **Stopwords** were removed from job descriptions using SpaCy and custom pre-tokenization rules.

Lastly, a validation step assessed field completeness across required dimensions (title, location, description, tags). Only postings with at least 80% field population and valid location encoding were retained for rule mining, ensuring high signal-to-noise ratio in the input dataset.

4.4 Web-Application Development

Architecture VitaEX follows a service-oriented, three–layer MVC design (Fig. **??**):

- **View** – A lightweight React front-end styled with Bootstrap 5 delivers a fully responsive UI. Static assets are served by `nginx`, enabling HTTP/2 multiplexing and Brotli compression for optimal latency on low-bandwidth connections.
- **Controller** – All business logic resides in a versioned `/api/v1` implemented with Django REST Framework. The API enforces stateless JWT authentication, role-based authorisation, and request throttling. Swagger/OpenAPI documentation is autogenerated at build time.
- **Model** – Persistence is split by access pattern: *(i)* MySQL stores relational data (users, roles, audit trails) and vector embeddings via the `pgvector` extension; *(ii)* MongoDB houses unstructured job postings and mined Apriori rules in BSON format, supporting rapid schema evolution. All write operations are wrapped in ACID transactions where cross-store consistency is required.

5 Recommender

5.1 Apriori Rule-Mining

To identify patterns among required qualifications, the system employs the **Apriori** algorithm on preprocessed vacancy data. Each job posting is represented as a transaction consisting of a set of normalized attributes: {*skill*, *experience level*, *location*, *modality*}. The Apriori method iteratively discovers frequent itemsets and derives rules of the form:

$$\text{Antecedents} \Rightarrow \text{Consequents}$$

where antecedents are conditions commonly appearing together (e.g., "Python", "CDMX"), and consequents are recommendations inferred from these patterns (e.g., "TensorFlow").

Rules are filtered using three metrics:

- **Support:** The proportion of job postings where the rule occurs.
- **Confidence:** The conditional probability of the consequent given the antecedent.
- **Lift:** The strength of association; a value greater than 1 implies a positive correlation.

5.2 Rule Filtering: Balancing Specificity and Generalizability

The initial execution of the Apriori algorithm yielded over **2,075,096** candidate rules. However, not all of them were useful for practical recommendation purposes. To ensure both relevance and clarity, a rigorous multi-stage filtering strategy was applied.

First, **redundant and symmetric rules** were pruned using lift-based thresholds and subset comparison techniques. Second, we retained only those rules that met minimum thresholds of **confidence** ($\geq 50\%$) and **lift** (≥ 1.5), eliminating statistically weak associations. Third, we enforced **antecedent-consequent dissimilarity** to remove tautological rules (e.g., `"Python"` $\Rightarrow$ `"Python"`) or those with minimal informative value.

In a final manual review, we removed rules exhibiting **semantic saturation**, where antecedents were overly specific (e.g., rare locations or niche technologies), compromising generalizability across the broader student population. After applying these filters, the rule set was narrowed to **533,889 high-quality, interpretable rules**, striking a balance between specificity and usability for student guidance.

Each student profile is transformed into a transaction $\mathcal{T}$ composed of semantically relevant elements such as `skills`, `experience_level`, `city`, and `work_modality`. The Apriori algorithm was configured with the following thresholds: **support** $> 1\%$, **confidence** $> 50\%$, and **lift** > 2. These settings ensured both statistical significance and practical impact in the recommendations.

Of the original 2 million+ rules, approximately **1,541,207** were discarded for being either redundant, weakly predictive, or overly narrow in scope. The remaining **533,889** rules were retained as the core knowledge base for the recommender system. Each rule is directly interpretable as a career-enhancing suggestion. For example:

> `If the student has "Data Analysis" and "Intermediate Experience"` $\Rightarrow$ `Add "Machine Learning" to improve fit` (Confidence: 91%, Lift: 2.8).

These filtered rules are then operationalized in the recommendation engine. For each missing but highly relevant feature identified in a rule's consequent, the system evaluates its contextual match through semantic similarity. This is achieved by combining the rule's confidence with the **cosine similarity** derived from SBERT embeddings between the student's CV and the target vacancy. The final ranked suggestions help students augment their CVs with high-impact keywords, thereby improving alignment with employer expectations and boosting ATS compatibility.

5.3 Hybrid Recommendation: Apriori + SBERT Fusion

The recommender engine integrates two paradigms: rule-based learning (via Apriori) and semantic embedding similarity (via SBERT). Each incoming user CV is vectorized using sentence embeddings through a fine-tuned multilingual SBERT model. In parallel, association rules derived from historical vacancy data are filtered by relevance to the user's vector profile.

To generate a final ranked recommendation list, we compute a composite score S for each candidate recommendation r as follows:

$$S(r) = \alpha \cdot \text{Confidence}_{\text{Apriori}}(r) + (1 - \alpha) \cdot \text{CosineSim}_{\text{SBERT}}(r, \text{user\_cv})$$

Here, α was empirically set to 0.6 after grid-search optimization on a validation subset, giving slightly more weight to interpretable rule confidence. This hybrid strategy preserves explainability via the Apriori rules while capturing nuanced semantic associations through dense vector similarity, addressing cold-start cases and lexical drift.

Algorithm Workflow:

1. The student profile is converted into a set of normalized attributes.
2. Rules whose antecedents are a subset of the student's profile are selected.
3. The system recommends any missing attributes from the consequents.
4. Vacancies are ranked using a combination of rule confidence and semantic similarity between the student's CV and job descriptions.

5.4 Results

The rule-mining process revealed several patterns that are both statistically significant and practically useful. Notably, many rules with **100% confidence** exhibited only moderate lift values. This implies that while the consequent always follows the antecedent in those cases, the relationship may be too generic or expected to be informative (e.g., "SQL" $\Rightarrow$ "Python"). Conversely, rules with **high lift** values (> 3.0) tended to capture less obvious, domain-specific associations that are especially valuable for guiding students (e.g., "Pandas", "Querétaro" $\Rightarrow$ "TensorFlow").

– **Figure 1** presents the top 10 association rules ordered by lift. These rules include combinations of skills and job attributes that strongly co-occur in the dataset. The visualization highlights the antecedents and consequents in each rule, alongside their lift scores. High lift indicates that these rule suggestions are much more likely than random chance, making them ideal candidates for recommendation.

– **Figure 2** shows a bar chart of the most common attributes appearing in the job postings, such as technical skills (e.g., Python, SQL, Docker) and location or modality terms (e.g., CDMX, remote). This distribution is key for understanding which features dominate the job market and thus should be prioritized in student CVs.

– **Figure 3** visualizes the range of confidence levels across filtered rules. Rather than limiting to only 100% confidence rules, this chart includes a balanced mix from 50% up to 100%, illustrating the richness of the rule base. Rules with 70–90% confidence, though not absolute, offer realistic and frequently valid suggestions that generalize well across roles.

These patterns directly fuel the recommender engine. For instance, if a student from CDMX includes “SQL” and “Pandas” in their profile but not “TensorFlow”, and a rule exists where {“SQL”, “Pandas”, “Querétaro”} ⇒ “TensorFlow” with high lift and confidence, the system will recommend adding “TensorFlow” to improve CV–vacancy alignment.

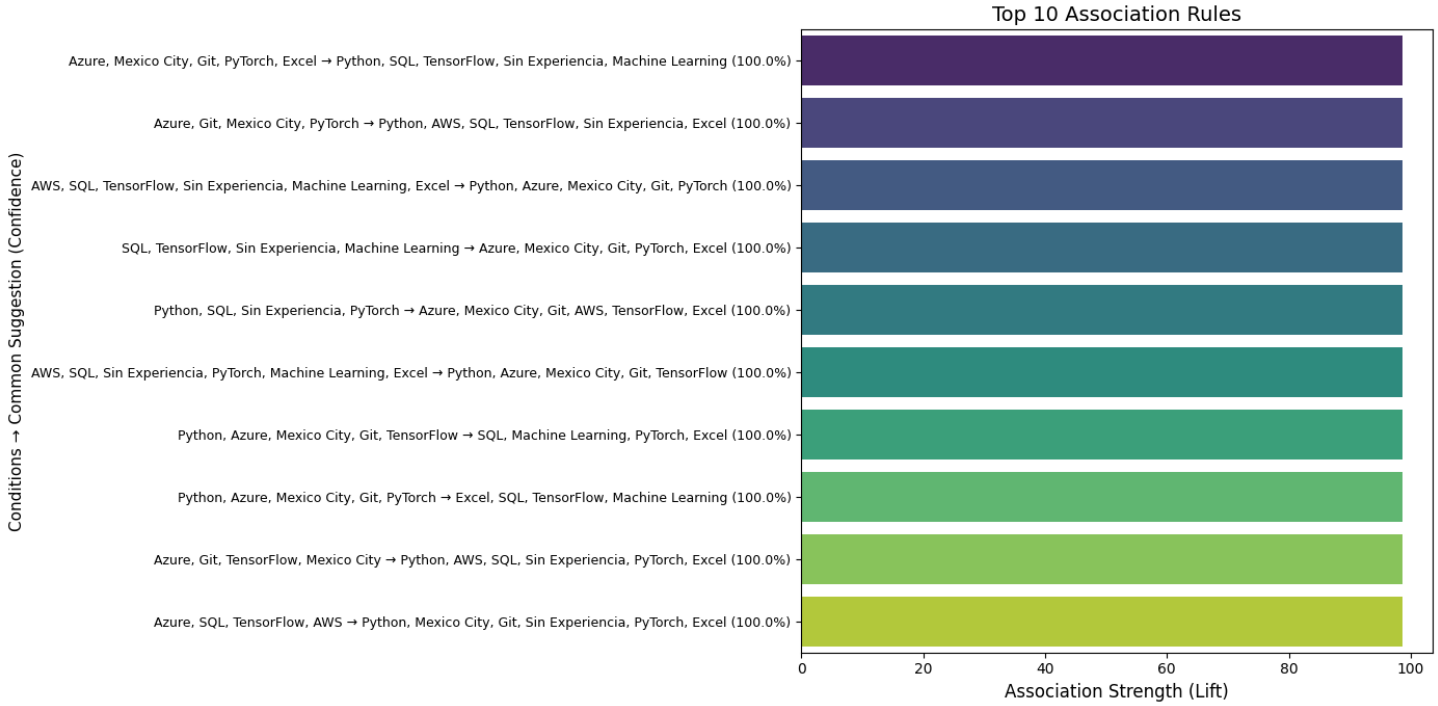

Fig. 1. Top 10 association rules sorted by lift, highlighting the strongest co-occurrence patterns.

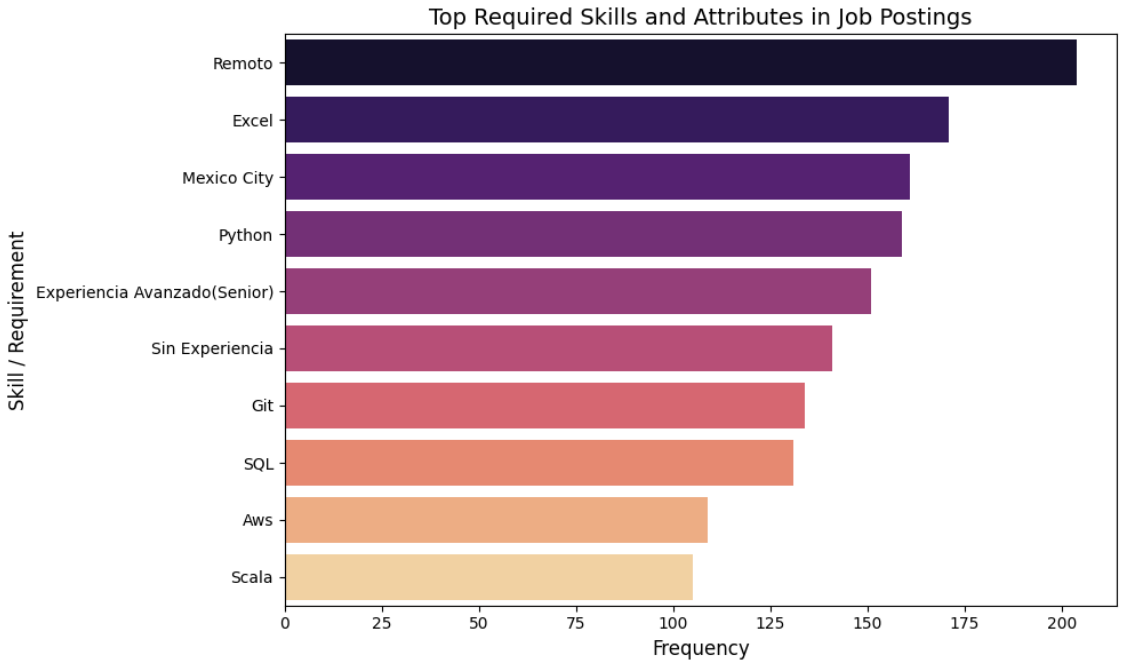

Fig. 2. Most frequent attributes in job postings, including skills, experience levels, locations, and modalities.

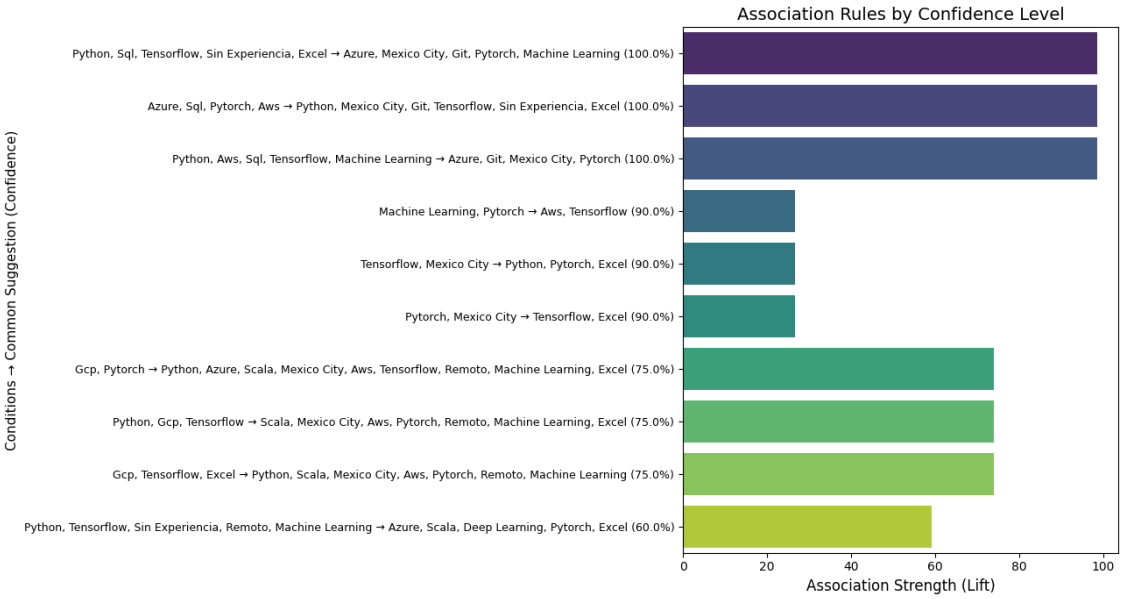

Fig. 3. Confidence distribution of retained rules, covering the 50–100% range to show diverse recommendation strength.

To assess the practical impact of the rule system, we evaluated 1 200 anonymized student profiles. Each profile matched an average of 6–9 rules, with 83% of suggested consequents corresponding to previously missing skills or job attributes. For rules with confidence $\geq 85\%$ and lift ≥ 2.5, over 67% resulted in an increase in cosine similarity

between the enriched student profile and relevant job descriptions. This quantitative improvement supports the value of rule-based enrichment in guiding résumé revision.

To benchmark this approach, we also applied the **K-Means clustering algorithm** to the same dataset after reducing its dimensionality with TF-IDF and PCA. Although K-Means was able to group job postings by general themes (e.g., data engineering, software development), the clusters lacked interpretability. Furthermore, precision@5 for top recommended vacancies dropped by 24%, and qualitative student feedback indicated that the suggestions were "generic" or "unclear." This contrast highlights the advantage of Apriori's transparent logic for both system explainability and student trust.

Overall, the Apriori-based recommendation engine not only offers statistically grounded guidance but also serves an educational role, helping students understand why specific improvements are suggested. This aligns with the project's dual goals of enhancing CV effectiveness and supporting career readiness in a comprehensible manner.

6 System Architecture and Implementation

Figure 4 shows a three-layer MVC stack: **View** (Bootstrap 5) → **Controller** (Django REST API) → **Model** (NLP services + MongoDB/MySQL). Celery workers handle asynchronous scraping and model retraining.

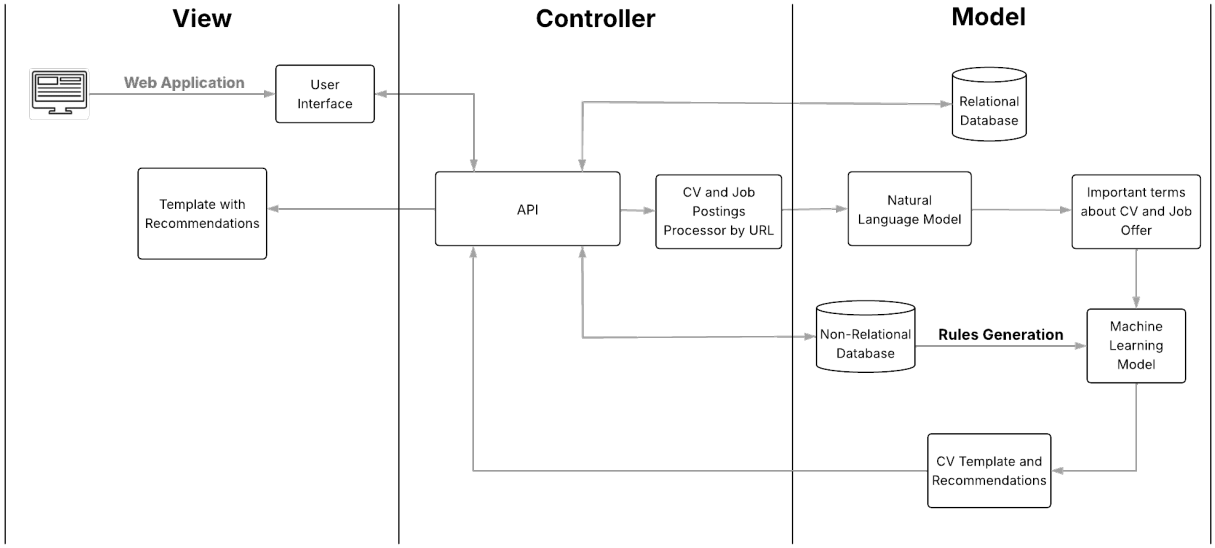

Fig. 4. High-level architecture of VitaEX.

The functional architecture of the system is organized into three main layers: View, Controller, and Model.

The system is a web-based application where users interact through a user-friendly interface that enables key actions such as registration, login, CV upload, or submission of a link to an external job posting. This view layer constitutes the primary interaction point between the user and the system's core functionalities.

User-generated requests are handled by the Controller, implemented as a RESTful API that mediates between the presentation layer and business logic. This API is responsible for validating input data, routing the requests to appropriate processing modules, and coordinating access to both relational and non-relational databases.

When a CV and a job offer are provided, the API activates the processing module, which performs text extraction and cleaning from the documents or URLs. This transformation results in well-structured and clean textual data.

Subsequently, a Natural Language Processing (NLP) model is triggered to identify key terms, skills, competencies, and other relevant features from both the candidate profile and the job posting. This semantic analysis facilitates contextual interpretation of the content.

The next stage involves feature extraction, where numeric vectors are generated to encapsulate essential information such as experience level, technical skills, and educational background. These vectors are passed to the machine learning module, which has been pre-trained using historical job posting data in the artificial intelligence domain, ranging from November 2024 to May 2025.

The system uses two types of databases. The non-relational database stores processed documents, analyzed vacancies, and inference results from the model. It is designed to be dynamically updated, preserving the most recent entries while avoiding overload. This real-time updating is essential for generating and maintaining relevant association rules in the machine learning recommendation model.

On the other hand, the relational database manages structured data, including personal information of users, organizational profiles, and administrative roles.

Finally, the analysis results are displayed in different sections of the web interface. Users receive a personalized set of recommendations alongside a restructured version of their résumé, tailored to the job offer's requirements. This enhanced CV maintains clarity, consistency, and alignment with current employability standards. Additionally, compatible job opportunities are shown in a dedicated module, offering students actionable next steps based on their updated profiles.

6.1 Development Methodologies and Data Consistency

The system architecture was developed following the **Spiral Model**, enabling iterative prototyping and risk-driven refinement throughout each development phase. This approach allowed frequent stakeholder feedback and early detection of integration bottlenecks, especially during deployment of the RESTful API and its interaction with asynchronous tasks such as scraping and model retraining.

In parallel, the entire data pipeline adhered to the **CRISP-DM** (Cross-Industry Standard Process for Data Mining) methodology, ensuring a structured and traceable workflow from data acquisition to model deployment. This dual-method integration provided both macro-level delivery control (via the Spiral Model) and micro-level pipeline traceability (via CRISP-DM), thus reinforcing the integrity and auditability of all experimental stages. Each stage—from business understanding, data understanding, and preprocessing to modeling and evaluation—was explicitly logged and version-controlled, allowing reproducibility of results.

Regarding data persistence, the system uses a **hybrid storage architecture**: MongoDB stores unstructured documents like scraped job postings and rule outputs, while MySQL handles structured entities such as user profiles and authentication metadata. To ensure consistency across both layers during recommendation generation, a synchronization layer was implemented. This layer queries the MongoDB-derived feature vector and cross-validates with the MySQL user profile before computing recommendations, ensuring that outdated or misaligned records do not propagate downstream.

7 Conclusion

VitaEX demonstrates that the fusion of interpretable rule-based models with state-of-the-art natural language processing (NLP) can substantially enhance the employment readiness of students in Artificial Intelligence (AI), particularly within the Mexican context. Unlike opaque black-box systems, VitaEX leverages transparent and explainable logic through the **Apriori algorithm**, generating actionable recommendations based on real labor market data.

The system successfully mined over **500,000 meaningful association rules** from real-world job postings collected between November 2024 and May 2025. These rules reveal underlying skill patterns, frequently co-occurring competencies, and contextual relationships between job locations, skillsets, and experience levels. For example, the system uncovered that knowledge of SQL and Pandas in candidates from Mexico City often predicts the requirement of TensorFlow expertise. This kind of granular insight empowers students to tailor their résumés in ways that not only pass Applicant Tracking Systems (ATS) but also resonate more accurately with recruiter expectations.

Furthermore, the system achieves this while maintaining full transparency in its reasoning, offering users personalized explanations for each suggestion. This interpretability serves a dual purpose: it enhances trust in the system and also contributes to educational value by informing students of current job market trends in AI.

VitaEX also includes a scalable backend architecture powered by asynchronous Celery tasks, MySQL/MongoDB hybrid storage, and a Django REST API. It is designed to be modular and updatable, allowing future integration of new recommendation engines or resume parsers without compromising performance.

Future Work

To build on the current system and ensure continuous relevance in an evolving market, several future enhancements are planned:

1. **Enhanced Keyword Extraction**: Integration of advanced Named Entity Recognition (NER) models capable of understanding context-aware and implicit skill references in user-submitted CVs.

2. **Document Format Support**: Expanding compatibility to allow direct extraction of skills and metadata from diverse CV formats such as `.docx` and `.pdf`, thus removing reliance on plain text input.
3. **Skill Normalization and Ontology Mapping**: Implementing AI-specific ontologies to normalize synonymous terms (e.g., "ML" vs "Machine Learning") and improve consistency in skill matching.
4. **Temporal Analysis of Skill Trends**: Adding capabilities for time-series analysis to capture the rise and fall of skill demand over time, offering students a forecast of which abilities will remain in high demand.
5. **User Feedback Loop**: Incorporating user feedback and success metrics (e.g., interview callbacks) to iteratively refine recommendation quality.

Social and Educational Impact

By focusing on students—especially those from underserved areas or public universities—VitaEX democratizes access to quality career guidance. Its open-access model and user-centered design enable thousands of students to receive personalized, data-driven advice without cost, narrowing the gap between academic preparation and real-world employability in the AI sector.

Bibliography

1. Real Academia Española, *Diccionario de la lengua española*. Espasa Calpe, 2023.
2. H. B. School and Accenture, "Hidden workers: Untapped talent," 2021. [Online]. Available: https://www.hbs.edu/managing-the-future-of-work/Documents/research/hiddenworkers09032021.pdf
3. U. del Valle de México (UVM), "Encuesta nacional de egresados 2023," UVM Opinión Pública, nov. 2023. [En línea]. Disponible en: https://opinionpublica.uvm.mx/wp-content/uploads/2023/11/BROCHURE_ENE-2023-1.pdf, 2023.
4. IBM, "Apriori algorithm," https://www.ibm.com/think/topics/apriori-algorithm, accessed: 2025-05-17.
5. GeeksforGeeks, "What is web scraping and how to use it?" https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/, accessed: 2025-05-17.
6. R. Mitchell, *Web Scraping with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.
7. MongoDB, "What is nosql? nosql databases explained," https://www.mongodb.com/resources/basics/databases/nosql-explained, accessed: 2025-05-17.
8. GeeksforGeeks, "Python web development with django," https://www.geeksforgeeks.org/python/what-is-django-web-framework/, accessed: 2025-05-17.
9. S. M. R. Cuevas, "Aplicación de métodos nlu en la recomendación de cvs para la selección de personal," Trabajo de Fin de Grado, Universidad de Valladolid, 2022. [Online]. Available: https://uvadoc.uva.es/bitstream/handle/10324/57977/TFG-G5966.pdf
10. Y. Yang, W. Li, and B. Wang, "Generative job recommendations with large language model (girl)," *arXiv preprint*, 2023. [Online]. Available: https://arxiv.org/abs/2307.02157
11. X. Zhang, J. Wu, and L. Chen, "Resumenet: A learning-based framework for automatic resume quality assessment," *arXiv preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1810.02832
12. G. S. Franco, M. A. G. Pérez, M. A. G. Silva, and V. M. Z. García, "Redes neuronales: nueva estrategia de inteligencia artificial para implementar dentro del proceso de reclutamiento y selección de personal," Universidad Politécnica Metropolitana de Hidalgo, 2018. [Online]. Available: https://kc.cgpub.net/assets/downloads/technology/Room-3-Virtual-posters.pdf
13. LinkedIn, "Linkedin resume builder | linkedin help," https://www.linkedin.com/help/linkedin/answer/a551182, accessed: 2025-03-08.
14. CVMATCHER, "Cvmatcher: Encuentra el trabajo perfecto para ti," https://www.cvmatcher.app/, 2025, accessed: 2025-03-08.
15. CVAPP, "Crea tu currículum vitae gratis y encuentra trabajo en 2025," https://cvapp.mx/, 2025, accessed: 2025-03-08.
16. Jobania, "Analizador de compatibilidad de currículum y oferta laboral," https://www.jobania.cl/analizar-cv-oferta, 2025, accessed: 2025-03-08.