projectfit.ai

isar innovations

projectfit.ai Technical Whitepaper

Automatic Skill Clustering at Scale: From 30k to 300k+ Labels

Nino Wagensonner

info@isar-innovations.dev

March 2026

Abstract

projectfit.ai clusters 307,126 bilingual skill labels for search, matching, and personalization. The previous production pipeline (Stella 1.5B → PCA → t-SNE → HDBSCAN) looked acceptable on a 30k sample but failed at full scale because t-SNE did not scale and the embedding space collapsed on German labels. We replaced it with qwen3-embedding, a mutual k-nearest-neighbor graph, and Leiden community detection. The new pipeline scales to the full corpus, achieves a 100% pass rate on the curated business-rule suite, and under a single-judge evaluation protocol reaches a mean semantic coherence score of 9.0/10 across 40 sampled clusters. The production deployment now runs with 14,523 clusters and provides the semantic backbone for later matching personalization.

1 Executive Summary

projectfit.ai maintains a mixed German/English skill corpus with 307,126 unique labels. Those labels range from single technologies such as "Java" and "React" to longer phrases such as "SAP S/4HANA Customizing" and "Agile Projektmanagement". The clustering system must do three things at once:

1. scale well beyond 30k labels,
2. preserve real ecosystem structure,
3. stay robust on bilingual short labels.

The old pipeline served as a practical intermediate solution for a long time, but it eventually ran into production-scale limits. The new pipeline meets those constraints:

- 307,126 labels processed with no dimensionality reduction,
- 100% business-rule pass rate at practical resolutions,
- 9.0/10 mean score under a 40-cluster single-judge protocol,
- production deployment at 14,523 clusters,
- 2,680 singletons handled defensively rather than forced into noise.

The key technical insight is that the bottleneck was not simply "choose a better clustering algorithm". The old stack remained useful while the corpus was smaller and the target granularity was coarser. The real break appeared when the system had to scale to the full corpus and separate closely related ecosystems such as JavaScript and TypeScript more reliably. Replacing Stella with qwen3-embedding and replacing plain kNN with mutual kNN turned that older pipeline into a more stable community-detection workflow.

2 Production Problem

2.1 What the system had to support

projectfit.ai uses skill clusters for at least three product surfaces:

- project-to-freelancer matching,
- search and filtering,
- downstream personalization based on user feedback.

The corpus contains 307,126 unique labels and mixes German and English. That matters because short technical labels are not well covered by standard embedding benchmarks, and bilingual false friends are common.

2.2 Why the old pipeline broke

The previous production stack was:

```
Stella 1.5B -> StandardScaler -> PCA -> t-SNE -> HDBSCAN
```

This stack was operationally useful for a long stretch of product work and produced many reasonable clusters at smaller scale. Its limits became clear only once the corpus grew and the product needed finer ecosystem boundaries. At full scale, two failure modes became hard to ignore:

1. **t-SNE did not scale.** The similarity matrix and repeated optimization became impractical at 307k labels.
2. **Fine-grained bilingual separation became unreliable.** The old embedding setup remained usable on many broad categories, but it no longer separated some close technical ecosystems cleanly enough, especially in German/English short-label regions and in JavaScript versus TypeScript neighborhoods.

Because evaluation data were English-heavy, this failure remained hidden long enough for the production pipeline to appear healthy.

3 Embedding Evaluation

3.1 German collapse was the decisive test

We evaluated eight embedding models on the full corpus. The most revealing test was simple: how many labels collapse to near-identical vectors around an arbitrary German skill label?

Table 1: German collapse test on the 307k production corpus.

Model	Dim	Collapsed	German-capable
qwen3-embedding	1024	1	Yes
bge-m3	1024	1	Yes
snowflake-arctic-embed2	1024	1	Yes
mxbai-embed-large	1024	2,407	No
all-minilm:33m	384	2,407	No
nomic-embed-text	768	31,046	No
stella_en_1.5B_v5	1024	185	No*

The original Stella setup was not useless. It was good enough to support the earlier product phase and to get the clustering effort off the ground. But for the full 307k-label bilingual corpus, it was no longer reliable enough. In the measured configuration, 257,483 of 307,126 labels showed cosine similarity above 0.99 to an arbitrary German label, which is strong evidence that the bilingual subspace was too compressed for the production task.

3.2 Why qwen3-embedding won

mxbai-embed-large scored best on a small English known-pairs benchmark, but it failed badly on German labels. qwen3-embedding delivered the best overall trade-off:

- no meaningful German collapse,
- strong local ecosystem neighborhoods,
- stable geometry for graph construction,
- production-usable latency through Ollama.

This decision mattered more than any later clustering hyperparameter.

4 New Pipeline

4.1 Architecture

The replacement pipeline is:

```
Skills (Parquet) -> qwen3-embedding -> Mutual kNN Graph -> Leiden -> MongoDB
```

4.2 Reproducibility profile

The original internal rollout prioritized production migration over external reproducibility. For the method itself, however, the operative configuration is compact and should be stated explicitly:

Table 2: Exact protocol used for the reported production-style run.

Component	Setting
Dataset	`skills_307k.parquet`, 307,126 bilingual labels
Embedding model	`qwen3-embedding:0.6b` via Ollama
Graph construction	mutual kNN on normalized embeddings
k for mutual kNN	30 in the reported experiment scripts
Community detection	Leiden `RBConfigurationVertexPartition`
Leiden resolution sweep	50, 200, 500, 1,000, 1,200, 3,000, 5,000
Production deployment	resolution 1,200
Leiden iterations	`n_iterations=-1` (run to convergence)
Random seed	42 for Leiden and judge sampling
Judge model	`mistral-small-latest`
Judge sampling	10 largest non-singleton clusters + 30 random non-singletons
Judge decoding	temperature 0.3, max tokens 512, JSON output

The core implementation lives in the experiment scripts for spectral pipeline construction and cluster judging under `experiments/skill_clustering`. That does not yet replace a cleaned public artifact with pinned package versions and a one-command reproduction script, but it removes ambiguity about the actual protocol used in this whitepaper.

4.3 Why mutual kNN matters

Direct distance clustering and naive kNN graphs both suffered from hub effects. Labels with broad connectivity bridged otherwise unrelated ecosystems. Mutual kNN fixes this by requiring a bidirectional neighborhood relation: label i must consider j a neighbor, and label j must also consider i a neighbor.

This single design change eliminated the monster-cluster behavior that had made earlier graph attempts unusable.

4.4 Why Leiden matters

Leiden was chosen over Louvain because it guarantees internally connected communities and behaves more predictably across large sparse graphs. The graph view also made PCA and t-SNE unnecessary. Those steps had been compensating for HDBSCAN limitations, not adding product value.

5 Evaluation

5.1 Resolution sweep

The resolution parameter controls granularity. The important result is not one magic number but a stable region in which business constraints hold.

Table 3: Resolution sweep for qwen3-embedding + mutual kNN + Leiden.

Resolution	Clusters	Singletons	Business Rules
50	4,813	2,682	75%
200	7,069	2,682	87%
500	9,905	2,682	100%
1,000	13,318	2,682	100%
1,200	14,500	2,682	100%
3,000	23,712	2,747	100%
5,000	34,797	3,751	87%

The production deployment uses resolution 1,200, yielding 14,523 clusters. This sits inside the stable regime while keeping clusters operationally useful.

5.2 Quality metrics

Table 4: Key production outcomes of the new clustering pipeline.

Metric	Result
Production corpus size	307,126 labels
Production clusters	14,523
Singleton clusters	2,680
Business-rule pass rate	100%
LLM judge mean rating	9.0 / 10
Sampled clusters for judging	40
Largest cluster at res=500	135 labels

The judge evaluation complements business rules. Rules test a small number of must-link and must-separate constraints. The judge test checks broader semantic coherence. The two together are much more credible than either alone.

5.3 Judge protocol and calibration caveat

The 9.0/10 score should be read as a useful but limited semantic signal, not as a definitive research-grade ground truth. The protocol uses a single LLM judge, a single prompt template, and one deterministic sample split. Concretely, the judge sees a cluster anchor label plus the top cluster members and is asked to return JSON with a 1–10 rating, a short rationale, and suspected outliers.

This is adequate for catching obvious semantic failures and was directionally validated with simple adversarial probes, but it still leaves three real risks:

1. calibration drift across prompts and models,
2. prompt bias toward generic technical coherence,
3. limited evidence about inter-rater consistency.

For that reason, the judge score in this paper should be interpreted as support for deployment readiness, not as the sole basis for a strong research claim.

5.4 Ablations expected for a research version

The production result already identifies a strong end-to-end recipe, but a research paper would need component isolation. The most direct ablation suite is:

1. **Embedding ablation.** Hold graph construction and Leiden fixed, then compare qwen3-embedding against bge-m3, snowflake-arctic-embed2, and the previously deployed Stella setup on the same bilingual corpus.
2. **Graph ablation.** Hold embeddings fixed, then compare plain kNN, mutual kNN, and possibly radius-graph variants to quantify hub removal, business-rule compliance, and giant-component behavior.
3. **Clustering ablation.** Hold embeddings and graph fixed, then compare Leiden against Louvain and the previous HDBSCAN pipeline to isolate the gain from connected-community detection.
4. **Judge ablation.** Re-run the semantic evaluation with prompt variants, a second LLM judge, and human spot checks on the same cluster sample to measure rating stability rather than only mean score.

Those ablations would answer the natural reviewer question: how much of the observed improvement comes from bilingual embedding quality, how much from hub suppression, and how much from the final partitioning algorithm?

5.5 What the spectral analysis told us

The normalized graph Laplacian showed no meaningful "natural" cluster count. That is a useful product result: cluster granularity is a business decision, not something the data uniquely determine. In practice, this means we can choose a resolution that balances browsing, matching, and personalization needs without pretending the graph itself dictates a single correct answer.

6 Deployment

6.1 Runtime

Table 5: Observed runtime profile of the clustering pipeline.

Step	**Runtime**
Parquet load	< 1s
Embedding generation	$\tilde{4}0$ min
Mutual kNN + Leiden	$\tilde{7}$ min
MongoDB export	$\tilde{3}$ min
Total first run	$\tilde{5}0$ min
Total with cached embeddings	$\tilde{1}0$ min

The deployment is operationally simple:

- embeddings are cached,
- graph construction is reproducible,
- MongoDB output stays backward-compatible,
- no t-SNE, PCA, or HDBSCAN stack remains in the critical path.

6.2 Why this matters beyond clustering

This pipeline is not only a clustering subsystem. It is a structural dependency for the matching personalization work. The later cluster-importance boost in the matching stack depends on these clusters being semantically stable. That is why the clustering system had to be production-grade first.

7 Limitations and Next Steps

The current deployment is strong, but not perfect.

1. **Reproducibility is not yet artifact-complete.** The scripts and operative settings exist, but a public repository snapshot with pinned dependencies, one-command reruns, and frozen judge prompts would make exact reproduction substantially easier.
2. **Next.js remains a known error case.** qwen3 places Next.js too close to Vue.js in some neighborhoods.
3. **Singletons deserve more analysis.** Many are correct, but some may still be merge candidates under additional constraints.
4. **Judge evaluation should be diversified.** The 9.0/10 mean score is useful, but human spot checks, a second judge, and agreement analysis would make the semantic evidence much stronger.
5. **Ablations are still missing.** The production migration proves the combined pipeline works, but it does not yet isolate the marginal contribution of qwen3-embedding, mutual kNN, and Leiden.
6. **Cluster monitoring should continue.** The matching system now depends on cluster quality, so cluster drift should be tracked as a first-class metric.

8 Conclusion

The migration succeeded because of three deliberate decisions:

1. treat bilingual embedding quality as a production requirement,
2. remove hub effects with mutual kNN,
3. replace low-dimensional projection plus density clustering with direct graph community detection.

In practical terms, the system moved from a sample-only pipeline that broke at full scale to a production deployment over 307k labels with 14,523 clusters, 100% compliance on the curated business-rule suite, and 9.0/10 judged quality. Just as importantly, the resulting cluster graph now supports later matching improvements instead of blocking them.