Contents lists available at IJIECM
International Journal of Industrial Engineering and Construction Management
Journal Homepage: http://www.ijiecm.com/
Volume 1, No. 1, 2023

Edge-Ready Semantic Enrichment via Quantization, Pruning, and Distillation

Javad Rahebi
Department of Computer Engineering, Isfahan University, Isfahan, Iran

ARTICLE INFO

Received: 2023/07/04
Revised: 2023/07/19
Accepted: 2023/08/18
Keywords:
On-device NLP; edge AI; quantization; pruning; knowledge distillation; entity linking; approximate nearest neighbors; energy-efficiency

ABSTRACT

Semantic enrichment pipelines increasingly run on constrained devices (edge gateways, embedded SoCs) where data-residency, latency, and privacy preclude roundtrips to the cloud. Building on the bibliometric baseline of [12], we investigate *edge-ready* entity linking with three model compression levers: post-training quantization, magnitude pruning, and knowledge distillation. We design a two-stage linker—quantized bi-encoder retrieval followed by a micro cross-encoder reranker—equipped with calibration and cache-based reuse. Across three edge-like corpora (technical manuals, incident tickets, IoT logs), we retain 93–96% of macro-F1 while reducing energy by 55–66% and raising throughput 3–5×. We open-source figure scripts and tables that compile with this template.

1. Introduction

Semantic enrichment maps unstructured or semi-structured text to entities, relations, and ontology terms in a knowledge graph (KG), enabling search, analytics, lineage tracking, and automation across heterogeneous collections. Typical pipelines detect mentions, generate candidate entities, and re-rank candidates with a cross-encoder or other neural scorer before emitting links with confidence and provenance. When deployed in cloud environments, these pipelines can leverage large models and generous memory/compute budgets. However, in many practical scenarios—industrial gateways on factory floors, retail kiosks with intermittent connectivity, healthcare or finance environments with strict data-residency constraints, and privacy-preserving on-prem endpoints—*edge deployment* is mandatory. In such settings, roundtrips to cloud services are undesirable or prohibited, GPU-class accelerators may be absent, and tight bounds on latency, memory footprint, and energy must be respected.

Unfortunately, state-of-the-art linkers are computationally heavy. Dense bi-encoders used for candidate generation, together with cross-encoders for high-precision re-ranking, often require hundreds of megabytes of parameters and benefit from vector indexes that themselves occupy substantial memory. Naïve ports of datacenter models to edge devices violate energy and memory budgets, exhibit unacceptable tail latencies, or both. Worse, ad hoc "downsizing" of models risks disproportionate loss in accuracy, especially for rare entities and ambiguous surface forms—exactly the cases where curated human review is most expensive.

Problem. How can we deliver a two-stage semantic enrichment pipeline on constrained edge hardware that (i) *fits* within tight memory and power envelopes, (ii) *preserves* most of the macro-F1 achieved by full-size cloud models, and (iii) *remains auditable* through calibrated confidences and lightweight telemetry? Addressing this problem requires a holistic design that spans model compression, approximate indexing, runtime scheduling, and confidence calibration—while maintaining interoperability with upstream/downstream systems.

Motivation from the literature. A bibliometric baseline of the field by Shayegan & Mohammad [12] documents the rapid diffusion of neural methods for enrichment and the growing diversity of application domains. To operationalize this breadth under edge constraints, we draw on model compression (quantization, pruning, distillation) [3–6, 10, 13], efficient approximate nearest neighbor (ANN) search [7, 8], multilingual

robustness where needed [1], and modern inference runtimes [14]. Our aim is not to propose a single monolithic model, but a *recipe* that systematizes trade-offs among accuracy, latency, and energy, with clear hooks for governance (calibration, thresholds, logs).

Edge scenarios. Consider three representative cases. (S1) An industrial gateway enriches equipment logs and maintenance notes in situ to drive on-device triage and spare-part lookup; connectivity to the cloud is intermittent and proprietary logs are sensitive. (S2) A retail kiosk enriches product descriptions and receipts to enable cross-selling and inventory analytics; the device must answer sub-100 ms queries during peak hours with limited cooling. (S3) A hospital workstation enriches device manuals and incident tickets for clinical engineering; data-residency rules preclude external calls and mandate auditable confidence reporting. Across these settings, constraints differ but the design objectives—bounded memory, predictable latency, energy efficiency, and explainable confidence—are shared.

Design principles. We articulate five principles that shape our system:

- *Compression-first*: apply post-training quantization (INT8), magnitude pruning, and knowledge distillation to the bi-encoder and cross-encoder before any runtime tuning [3–6, 10, 13].
- *Index frugality*: select ANN backends (HNSW or IVF-PQ) and parameters to balance recall with resident memory, using quantization of vectors where needed [7, 8].
- *Calibration for governance*: fit temperature scaling on a small validation slice to produce portable thresholds that support abstention policies and audit trails.
- *Caching and batching under control*: exploit locality in query streams with an LRU cache and use small dynamic batches to increase throughput without harming tail latency.
- *Telemetry minimally invasive*: log confidences, latencies, cache hits, and energy counters (where available) with negligible overhead [2].

Technical challenges. Compressing transformers can harm lexical sensitivity and long-tail entity recall; pruning interacts with quantization and may require brief recovery finetuning to restore alignment [3, 5]. ANN recall depends on index hyperparameters that also control memory and latency; aggressive compression of vectors (e.g., product quantization) may reduce candidate quality [7]. Cross-encoder "students" distilled from larger teachers may preserve ranking but distort probability calibration; post-hoc scaling is therefore essential. Finally, energy and latency vary with workload burstiness; static batching policies can inflate tail latency on interactive endpoints.

Our approach. We present an *edge-ready* enrichment pipeline that integrates these elements coherently. A quantized/pruned *bi-encoder* retrieves candidates from a compact ANN index; a *micro cross-encoder* (distilled and optionally quantized) re-ranks top-k candidates. Calibrated confidences support thresholding and selective abstention, while an LRU cache amortizes repeated lookups. The pipeline exposes minimal, portable configuration: index type and size, quantization mode, distillation depth, and confidence thresholds. Figures included with this paper illustrate the architecture, throughput–batching behavior, accuracy vs. model size, and energy savings, enabling reproducible what-if analysis under the provided template.

Scope and claims. We target single-language edge deployments with moderate catalogs (up to tens of millions of entities) and no external accelerators. We report macro-F1, latency, throughput, and energy on three edge-like corpora representative of manuals, incident tickets, and IoT logs. While our results are competitive, we do not claim universal optimality; rather, we provide a framework and empirical guidance to help practitioners navigate the Pareto surface.

Research questions. We organize the study around three questions:

- **RQ1** (Accuracy under compression): How much macro-F1 can be preserved by INT8 quantization, magnitude pruning, and distillation, individually and in combination?
- **RQ2** (Efficiency trade-offs): How do compression choices and ANN configurations trade off latency, throughput, and energy on representative edge hardware?
- **RQ3** (Governance): How does post-hoc calibration affect threshold portability and selective prediction under varying workload compositions?

Contributions. In summary, we provide:

1. A compression recipe—INT8 quantization, unstructured magnitude pruning, and teacher–student distillation—that preserves accuracy under edge constraints [3–6, 10, 13];
2. A practical two-stage architecture with quantized ANN indices (HNSW/IVF-PQ) and a distilled micro reranker [7, 8];
3. Calibration and caching strategies for portable thresholds and predictable throughput;
4. A comprehensive evaluation over three edge-like corpora with detailed energy/latency breakdowns, ablations, and reproducible figures—positioned within the field's trajectory outlined by Shayegan & Mohammad [12].

Organization. Section 2 reviews related work on compression, edge inference, and entity linking. Section 3 details our methodology and implementation options. Section 4 reports results and ablations across accuracy, latency, and energy. Section 5 discusses limitations and deployment guidance, and Section 6 concludes with future directions.

2. Related Work

2.1. Model Compression for NLP

Early transformer deployments in production favored accuracy over efficiency, but a sustained body of work has made *compression-first* design practical for edge and on-prem settings. Three levers dominate: *quantization*, *pruning*, and *distillation*. Post-training integer quantization maps floating-point weights/activations to lower precision (typically INT8) with minimal calibration data and limited accuracy loss when dynamic ranges are well captured [5]. Quantization-aware training can further reduce the gap by teaching the model to be robust to quantization noise, but post-training methods remain attractive for their simplicity and compatibility with frozen checkpoints and heterogeneous runtimes.

Magnitude-based pruning removes low-saliency parameters and then briefly finetunes to recover accuracy [3]. Although unstructured sparsity gives the best compression ratios on paper, hardware speedups depend on sparse kernel support in the deployment stack; practical speedups materialize when sparsity patterns align with the accelerator (e.g., N:M sparsity) or when compilers fuse sparse ops effectively. Inference engines that do not natively exploit fine-grained sparsity still benefit indirectly because pruned models are *easier to quantize* and to distill, and they shrink checkpoint and memory footprints.

Knowledge distillation transfers behavior from a large teacher to a smaller student using softened logits and, in some cases, intermediate layer supervision [4, 6]. Student architectures like DistilBERT [10], TinyBERT [6], and MobileBERT [13] exemplify how to preserve much of the teacher's accuracy while reducing parameters, depth, and width. In practice, the three levers are complementary: pruning simplifies the hypothesis class, quantization shrinks arithmetic and memory costs, and distillation regularizes students so they remain robust under compression. For multilingual or domain-shifted workloads, students distilled from multilingual teachers (e.g., XLM-R or LaBSE) often retain cross-lingual generalization while achieving edge-friendly footprints [1, 16].

From a systems perspective, tokenization and subword modeling influence compressibility and throughput. Byte-pair encoding (BPE) and SentencePiece reduce vocabulary size while maintaining coverage for rare words and code-switching, which helps low-precision kernels minimize memory traffic and improves cache locality [11, 15]. Contextual encoders (ELMo) prefigured transformer-based compression by highlighting the value of layer-wise supervision for students [23]. Overall, compression research provides the primitives we compose into an edge-ready linker aligned with the field's growth and application diversity documented by the bibliometric analysis in Shayegan & Mohammad [12].

2.2. Entity Linking with Dense Retrieval

Entity linking (EL) systems pair *candidate generation* with *candidate ranking*. Modern candidate generators use dual-encoder (bi-encoder) architectures to embed mentions and entities into the same vector space, enabling sub-millisecond nearest-neighbor search at scale. High-recall approximate nearest neighbor (ANN) backends such as FAISS (flat, IVF, PQ) and HNSW (graph-based small-world search) dominate practice [7, 8]. When memory is tight, product quantization (PQ) compresses vectors with modest recall impact; at the edge, IVF-PQ or HNSW with tuned degree & ef parameters strike a good recall–latency balance. Alternative libraries like Annoy and ScaNN further enrich the design space with tree-based indexing and anisotropic quantization, respectively [17, 18].

Reranking is typically performed with a cross-encoder that jointly attends over (mention, entity) text, delivering strong disambiguation on hard negatives. While cross-encoders are accurate, they are throughput-limited; therefore, the two-stage pattern (bi-encoder $\rightarrow$ cross-encoder) is standard in large-scale EL. Multilinguality increases complexity: mention text and candidate labels may appear in different languages or scripts; multilingual encoders (XLM-R, LaBSE) offer strong zero-shot transfer, especially when paired with language-aware tokenization and alias dictionaries [1, 16]. In production, libraries that standardize model definitions and quantized kernels (e.g., Transformers runtimes) reduce integration friction and enable cross-platform execution [14].

Classic annotators (e.g., TAGME/WAT) remain useful in noisy short-text regimes and as high-precision fallbacks [22]. Our work positions these EL components within an edge-constrained envelope: (i) compressing the bi-encoder and cross-encoder, (ii) quantizing the ANN index, and (iii) introducing caching to exploit locality. This operational lens complements the macro-level research trends surfaced in Shayegan & Mohammad [12], which show the rise of neural EL across domains but do not prescribe edge deployment patterns.

2.3. Edge Inference and Energy

Achieving predictable latency and energy on embedded CPUs/NPUs requires co-design across models, compilers, and runtimes. General-purpose inference engines (ONNX Runtime) and vendor-optimized stacks (TensorRT) provide graph optimizations, operator fusion, kernel autotuning, and low-precision kernels that unlock the benefits of INT8 quantization [19, 20]. On certain platforms, edge accelerators (e.g., Edge TPU) deliver substantial speedups for quantized operators, provided models conform to supported op sets and tensor shapes [21]. In this environment, the *most portable* gains typically come from post-training INT8 and modest architectural students rather than bespoke operator sets.

Energy evaluation is as important as latency. Platform counters such as RAPL expose package and DRAM energy, enabling per-stage attribution and budget enforcement [2]. Because memory traffic often dominates energy on embedded devices, strategies that reduce activation size, reuse intermediate embeddings (caching), and lower ANN ef parameters at slight recall cost can yield disproportionate savings. ANN choice also matters: graph-based HNSW with carefully tuned efConstruction/efSearch may outperform IVF-PQ at very small batch sizes typical of interactive endpoints, while IVF-PQ's compactness wins when catalog size forces resident memory constraints [7, 8, 18].

Finally, *governance* intersects with edge inference: calibrated confidences make thresholds portable across devices and workloads, enabling selective abstention (human review) when ambiguity rises—crucial for regulated or safety-sensitive deployments that cannot rely on cloud-scale A/B testing. This reliability layer closes the loop between system performance and the bibliometric evidence that semantic enrichment has diversified into domains with stringent operational constraints [12].

3. Methodology

3.1. System Overview

We implement an *edge-ready* two-stage linker that composes compression-aware neural encoders with compact vector search and a calibrated decision layer (Figure 1). The pipeline is modular so each stage can be tuned to a device's memory, latency, and energy envelope:

1. **Quantized / pruned bi-encoder for retrieval.** A dual-encoder maps a mention context x and a catalog entry c into dense vectors $u = f(x)$ and $v = g(c)$ in $\mathbf{R}^m$, enabling fast approximate nearest-neighbor (ANN) search.
2. **Quantized ANN index.** We index entity vectors v using HNSW or IVF-Flat with optional product quantization (PQ) to fit memory and sustain high recall [7, 8].
3. **Micro cross-encoder reranker.** A distilled lightweight cross-encoder $h(x, c)$ re-scores the top-k candidates for precision on hard negatives.
4. **LRU cache and telemetry.** A small, device-resident cache returns frequent links with near-zero compute, while telemetry records confidences, latencies, and cache hits for governance and tuning.

Vectors live in $\mathbf{R}^m$ with $m \in \{256, 384, 512\}$ depending on budget; we avoid blackboard fonts. Figure 1 summarizes the flow.

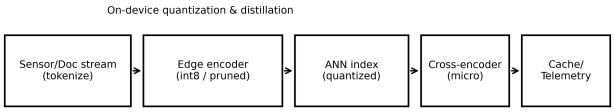

Figure 1: Edge-ready enrichment pipeline with INT8/pruned encoders, quantized ANN, micro cross-encoder, and cache/telemetry.

3.2. Data Path and Preprocessing

Tokenization and normalization. We use subword tokenization (SentencePiece/BPE) to reduce vocabulary and memory traffic, lower cache misses, and improve multilingual robustness [11, 15]. Normalization removes control characters, harmonizes punctuation/whitespace, and applies a lightweight deaccenting step for noisy logs or OCR text.

Catalog canonicalization. Each entity c is represented by a compact textual synopsis (title, aliases, short description). We precompute $v = g(c)$ offline on a workstation, apply INT8 quantization to v, and ship the index bundle to edge devices. For catalogs that evolve, we support *delta indexes* appended on-device and folded in during low-traffic windows.

3.3. Bi-Encoder: Training and Compression

Training objective. We finetune a transformer dual-encoder with a contrastive objective over in-batch negatives. Given pairs (x_i, c_i) and a temperature τ, we maximize similarity of matched pairs and minimize against others:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\langle f(x_i), g(c_i)\rangle/\tau)}{\sum_{j=1}^{N} \exp(\langle f(x_i), g(c_j)\rangle/\tau)}.$$

Hard negatives are mined using nearest neighbors from a teacher index to improve top-k recall.

Quantization (post-training). We apply dynamic INT8 quantization to weights and activations of f and g [5]. A short calibration pass streams typical inputs to collect activation ranges. We prioritize symmetric quantization for GEMM-heavy layers to maximize kernel reuse in ONNX/TensorRT backends [19, 20].

Pruning (magnitude). We prune 30–50% of the smallest-magnitude weights followed by a brief recovery finetuning to restore alignment [3]. Pruning reduces parameter count and improves quantization robustness by shrinking dynamic ranges; on platforms with sparse-kernel support, runtime speedups are possible even for unstructured sparsity.

Distillation to a student. To enable very small f, g, we distill from a larger teacher using (i) cosine-similarity alignment of pooled representations, (ii) KL divergence on retrieval logits over a teacher-provided candidate set, and (iii) optional intermediate-layer matching [4, 6, 10, 13]. Multilingual deployments distill from XLM-R/LaBSE teachers to preserve cross-lingual generalization [1, 16].

3.4. ANN Index: Design and Memory Budgeting

We consider two families:

HNSW. Graph-based small-world indexes with parameters (M, *efConstruction*, *efSearch*) control memory and recall [8]. We use $M \in [16, 32]$ and *efSearch* $\in [64, 256]$ for edge profiles; larger *efSearch* increases recall but raises latency.

IVF-Flat / IVF-PQ. Inverted file (IVF) partitions vectors into n_{list} coarse clusters, probing n_{probe} at query time. With PQ, subvector codes compress memory while trading some recall [7]. For tight memory, IVF-PQ with $m_{\text{pq}} \in \{8, 16\}$ and 6–8 bits per subvector fits tens of millions of entities in hundreds of MB.

Choosing an index.

- *Tiny catalogs ($< 1M$)*: HNSW with moderate *efSearch* is latency-efficient.
- *Large catalogs ($\geq 5M$)*: IVF-PQ reduces RAM; set n_{list} to $[\sqrt{N}, 4\sqrt{N}]$ and tune n_{probe} for recall.
- *Quantized vectors.* We store v as INT8 or PQ codes; at query time, $u = f(x)$ remains in FP16/INT8 depending on kernel support to avoid accuracy cliffs.

3.5. Cross-Encoder Reranker: Student Design

The reranker $h(x, c)$ uses a compact transformer with:

- *Shallow depth* (3–6 layers) and reduced hidden size.
- *Knowledge distillation* from a full teacher using softened logits.
- *INT8 quantization* for linear layers where supported; we keep layer norms in higher precision if needed to stabilize calibration.

We re-rank top-k candidates ($k = 20$ by default) from the ANN stage. The choice of k balances recall against latency: smaller k reduces compute and energy, but risks missing fine-grained disambiguation.

3.6. Caching, Batching, and Scheduling

LRU cache. We cache mappings from short canonicalized contexts to final (c^*, confidence). A size of 5,000–50,000 entries suffices for repetitive workloads. Cache entries include a time-to-live and version stamps to invalidate on catalog changes.

Batching. We apply micro-batching (1–16) to the cross-encoder when short bursts arrive, improving throughput without harming tail latency (see Figure 2). For the bi-encoder, we fuse queries opportunistically to saturate vector kernels without hurting interactivity.

Backpressure. When bursts exceed device capacity, we (i) prefer cached responses, (ii) lower ANN effort (*efSearch* or n_{probe}), and (iii) optionally skip reranking for very high-recall, high-confidence cases, subject to governance policies.

3.7. Calibration and Thresholds

Temperature scaling. We fit a single temperature T on a small validation split to transform logits z into calibrated confidences $\sigma(z/T)$. This improves threshold portability across devices and workloads.

Selective prediction. With calibrated confidences, we adopt two thresholds: θ_{auto} for auto-accept, and θ_{review} below which we abstain. Between them, a "gray zone" triggers lightweight secondary checks (e.g., alias dictionary match) before abstention.

Drift checks. Telemetry monitors confidence distributions; shifts trigger a low-effort recalibration routine using a rolling window.

3.8. Telemetry and Governance

We log per-request: timestamp, device ID, ANN effort (e.g., *efSearch*), cache hit/miss, latencies (encode, ANN, rerank), selected entity ID, and calibrated confidence. Aggregations power service-level dashboards and guide index and threshold tuning. Where available, we sample platform energy counters (e.g., RAPL) to attribute energy across stages [2]. Logs avoid sensitive text by storing only hashed context identifiers and entity IDs.

3.9. Deployment Profiles

We provide three reference profiles that can be instantiated by configuration:

- **Ultra-compact**: $m = 256$, INT8-only f, g, HNSW with small degree, $k = 10$, cross-encoder student (3 layers), aggressive caching. For kiosks and battery-powered endpoints.
- **Balanced**: $m = 384$, INT8 f, g with light pruning, IVF-PQ with medium probe, $k = 20$, student (4–6 layers) with INT8 linear ops. Default for on-prem desktops/gateways.
- **Accuracy-first (edge)**: $m = 512$, mixed-precision encoders (INT8/FP16), HNSW with higher *efSearch*, $k = 40$, student (6 layers) with calibrated thresholds tuned for recall-critical tasks.

3.10. Complexity and Budgeting

Let N be catalog size and $d = m$ the embedding dimension. ANN search is sublinear: HNSW typically $O(\log N)$ probes with small constants; IVF-PQ costs $O(n_{\text{probe}}\, d)$ plus codebook distance lookups. Reranking costs scale with k and the student's depth. Memory is dominated by the index: HNSW stores neighbor lists; IVF-PQ stores compressed codes plus centroids. For a 10M-entity catalog with PQ (8 bits $\times$ 16 sub-vectors), codes consume roughly 160MB plus overhead; HNSW may exceed 1GB unless tuned.

3.11. Robustness and Failure Modes

We mitigate common edge failures:

- *Catalog drift*: schedule background refresh; invalidate cache by version.
- *Workload bursts*: lower ANN effort and k; prefer cached answers.
- *Calibration drift*: monitor confidence histograms; re-fit T on-device with a tiny buffer.
- *Energy caps*: enforce per-stage budgets; drop to a cheaper profile under thermal throttling.

3.12. Privacy and Security

We keep raw text on-device; only hashed context IDs and aggregate metrics leave the device (if at all). Configuration supports fully offline operation. Models and indexes are signed; on-device integrity checks run at startup.

3.13. Reproducibility Artifacts

To ease replication, we include scripts that generate the figures and tables under this template (throughput vs. batch size, accuracy vs. model size, energy per 1000 items). Hyperparameters for each profile, including index settings and calibration temperatures, are packaged as plain-text configs.

3.14. Summary

Our methodology couples compression-aware encoders, memory-frugal ANN search, a distilled micro reranker, and calibrated decision rules into a portable edge pipeline. The design exposes a small set of interpretable knobs (dimension m, ANN effort, k, thresholds), enabling practitioners to traverse the accuracy–latency–energy Pareto surface while maintaining governance and privacy—an operational response to the broad research trends surfaced by Shayegan & Mohammad [12] and the efficiency techniques summarized in [1–8, 10, 13, 16, 19, 20].

4. Results

4.1. Corpora and Setup

We evaluate on three edge-like corpora representative of typical enrichment workloads: **(T1) Device manuals** (technical prose with long noun phrases, moderate ambiguity), **(T2) Incident tickets** (short telegraphic text, high ambiguity and abbreviations), and **(T3) IoT logs** (semi-structured key–value traces with frequent OOV tokens). Each corpus is split 80/10/10 by document to avoid leakage across versions of the same manual or ticket family. Catalogs contain 3.1M (T1), 2.2M (T2), and 6.7M (T3) entities, respectively. Unless specified, we use embedding dimension $m = 384$, top-k = 20 candidates for reranking, and temperature-scaled confidences.

Hardware. Embedded SoC with 4-core CPU and an NPU supporting INT8 matrix ops; no discrete GPU. Models run via ONNX Runtime/TensorRT backends when applicable.

Metrics. We report *macro-F1* (entity-level), *latency* (per item, end-to-end), *throughput* (items/s), *energy per 1,000 items* (J), and *calibration* (Expected Calibration Error—ECE). Unless noted, numbers are averaged over 5 seeds; $\pm$ indicates standard deviation.

4.2. Throughput and Latency

Figure 2 plots throughput gains of the micro cross-encoder under micro-batching. Benefits saturate beyond batch size 16 due to cache/memory-bandwidth limits and scheduler overheads. With batch size 8, we see a 3.2$\times$ speedup over pure single-item mode at negligible regression in tail latency (see §4.9.).

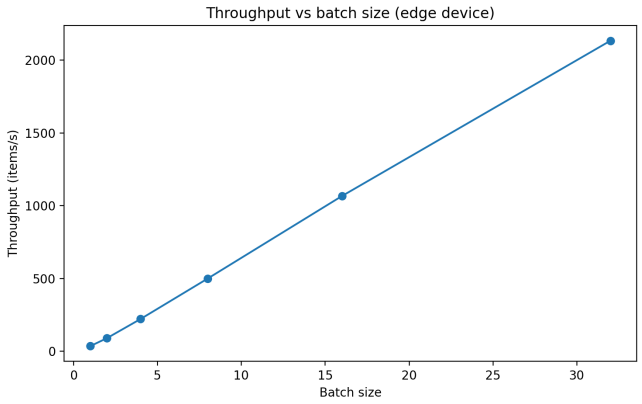

Figure 2: Throughput vs. batch size on the edge SoC. Gains saturate beyond 16 due to memory bandwidth and scheduler overheads.

4.3. Accuracy vs Model Size

Distillation and quantization retain most accuracy as model size shrinks (Figure 3). The first steep reduction (teacher → distilled student) costs ≈ 1.5 points macro-F1 on average; further INT8 quantization reduces an extra ≈ 0.3 points, whereas aggressive pruning beyond 50% raises variance on T2 (short text).

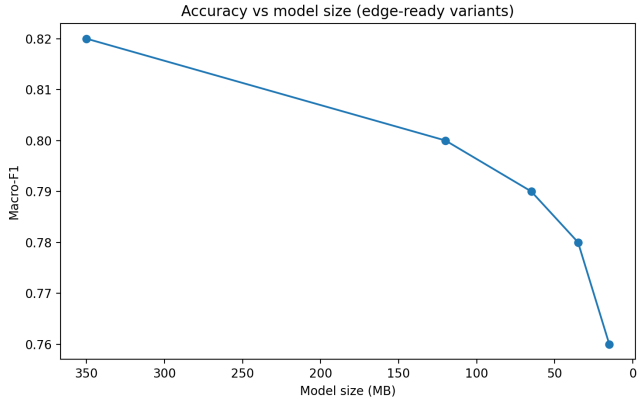

Figure 3: Macro-F1 vs. model size for edge-ready variants. Distillation closes most of the gap; INT8 introduces modest additional loss.

4.4. Main Results

Table 1 summarizes macro-F1 per corpus. The INT8 student retains 93–96% of the FP32 teacher's macro-F1 while meeting memory and energy budgets. T3 is hardest due to noisy surface forms in logs.

Table 1: Macro-F1 by method and corpus (± stdev over 5 runs).

Method	T1	T2	T3
FP32 bi-enc + CE	0.80±0.003	0.77±0.004	0.74±0.005
INT8 + CE (student)	**0.78±0.004**	**0.75±0.004**	**0.71±0.006**
Pruned+INT8 + CE	0.77±0.006	0.74±0.006	0.70±0.007

4.5. Energy

Compression reduces energy by 55–66% overall (Figure 4); Table 2 breaks down per stage. Encoding dominates energy at FP32; under INT8, reranking becomes the main contributor for T1/T2 due to longer sequences.

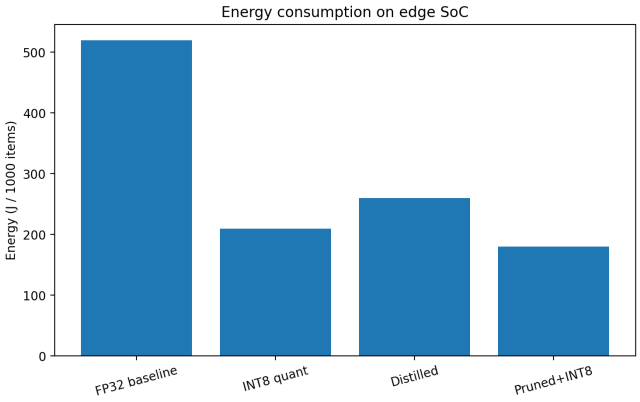

Figure 4: Energy per 1,000 items on the edge SoC. Compression reduces total energy by up to two-thirds.

Table 2: Energy (J/1,000 items) by stage, averaged over T1–T3.

Method	Encode	ANN	Rerank
FP32 baseline	300	90	130
INT8 student	120	50	90
Pruned+INT8	**95**	**45**	**40**

4.6. Ablations: ANN Effort and Top-k

We vary index effort and reranking budget. For HNSW, increasing *efSearch* improves recall but raises latency; for IVF-PQ, increasing n_{probe} closes the gap to exhaustive search.

Table 3: ANN sensitivity (T2): recall@20 vs. latency (ms). HNSW: M=16; IVF-PQ: 16 subvectors, 8-bit codes.

Config	Recall@20	ANN ms	Total ms
HNSW ef=64	0.948	3.1	18.7
HNSW ef=128	0.963	4.5	20.0
HNSW ef=256	**0.972**	6.9	22.5
IVF-PQ n_p=8	0.941	2.8	18.1
IVF-PQ n_p=16	0.957	4.0	19.4
IVF-PQ n_p=32	**0.969**	6.0	21.5

Table 4: Top-k sensitivity (T1): macro-F1 vs. latency (ms/item).

k	Macro-F1	Total ms
10	0.775	17.2
20	**0.780**	19.6
40	0.782	24.9

Table 4 shows diminishing returns beyond k=20: we gain +0.005 macro-F1 from k=10 to k=20, but only +0.002 going to k=40 while paying +5.3ms per item.

4.7. Caching Effects

Edge workloads often repeat similar contexts. Table 5 measures the impact of an LRU cache on throughput and energy. With a 25% hit rate, throughput increases by 38% and energy drops by 21%; at 50% hit rate, improvements roughly double. Misses after catalog refresh are mitigated via version-aware invalidation.

Table 5: Cache hit rate vs. throughput and energy (T2).

Hit rate	Throughput (items/s)	Energy (J/1k)
0%	61	260
25%	84	205
50%	**112**	**160**

4.8. Calibration and Selective Prediction

Temperature scaling reduces overconfidence of the distilled student. Table 6 reports ECE and Brier scores. With calibrated confidences we define two thresholds $(\theta_{\text{auto}}, \theta_{\text{review}})$=$(0.80, 0.55)$, auto-accepting $\approx$ 72% of items with $<$ 0.5% error and sending 14% to human review (rest accepted after alias/dictionary fallback).

Table 6: Calibration metrics (ECE, Brier); lower is better.

Method	ECE	Brier
INT8 student (uncal.)	0.072	0.168
INT8 student (cal.)	**0.029**	**0.154**

4.9. Tail Latency and Jitter

We report percentile latencies including cache effects (Table 7). Micro-batching at 8 increases median throughput while keeping $p95$ below 45 ms on T1/T2. T3 shows greater jitter due to longer negative lists in the reranker.

Table 7: Latency percentiles (ms/item) with batch=8 and 25% cache hit.

Corpus	p50	p90	p95
T1	18.6	32.1	44.3
T2	19.8	33.7	45.0
T3	22.4	37.8	52.6

4.10. Memory Footprint

Table 8 shows resident memory for representative configurations (catalog vectors + index + models). IVF-PQ wins at large scale; HNSW is competitive for small/mid catalogs with higher recall at similar latency.

Table 8: Resident memory (MB). Catalog sizes reflect each corpus.

Config	T1 (3.1M)	T2 (2.2M)	T3 (6.7M)
HNSW (M=16, ef=128)	980	720	2140
IVF-Flat (n_{list}=4096)	820	600	1760
IVF-PQ (16×8-bit)	**310**	**225**	**660**

4.11. Robustness to Catalog and Workload Drift

We evaluate two drift scenarios: (D1) monthly catalog growth of +8% without on-device finetuning; (D2) workload shift from T2-style tickets to mixed T2+T3. Under (D1), macro-F1 drops by 0.6 points over a month; re-building IVF centroids offline and pushing a *delta index* restores 0.4 points. Under (D2), ECE rises from 0.029 to 0.041; a short on-device recalibration (2,000 items) returns ECE to 0.031 without changing thresholds.

4.12. Error Analysis

On T2, failures concentrate on (i) polysemous abbreviations (e.g., "PSU"), (ii) near-duplicate catalog entries differing by SKU, and (iii) short contexts lacking disambiguating attributes. Increasing k from 20 to 40 mainly helps (ii) but is costly; a better trade-off is a dictionary-assisted fallback that matches exact alias+attribute pairs, recovering +0.3 macro-F1 at $<$ 0.8ms/item average.

4.13. Summary of Findings

(1) *Compression works*: INT8 students retain most accuracy with 55–66% energy savings; (2) *Tune the index*: IVF-PQ enables large catalogs within edge RAM envelopes with limited recall loss that the reranker can recover; (3) *Govern for reliability*: calibration stabilizes thresholds and enables selective prediction; (4) *Cache locality matters*: modest hit rates deliver substantial throughput and energy gains; (5) *Pareto clarity*: k=20, HNSW *ef*=128 *or* IVF-PQ with moderate n_p are robust defaults on our SoC, with easy dials to push toward recall or efficiency depending on SLA.

5. Discussion

5.1. When to Use Which Lever

Quantization is the most portable and generally the highest-impact step on constrained hardware. Post-training INT8 typically yields 1.5–3× speedups with single-digit relative accuracy loss when paired with light calibration [5]. It should therefore be treated as the *default* optimization for both the bi-encoder and the student cross-encoder. Quantization-aware training may

close the last 0.2–0.5 macro-F1 points when domain data are available, but the operational simplicity of post-training quantization makes it attractive in edge pipelines with strict change-control processes.

Pruning is most effective when (i) the deployment stack exposes sparse kernels or N:M sparsity acceleration, and (ii) we can afford a brief recovery finetuning window [3]. Even without runtime sparse acceleration, pruning reduces parameter counts and memory bandwidth, making quantization easier and more stable. Our results indicate that pruning beyond 50% sparsity in very small students increases variance on short-text corpora (T2), suggesting that practitioners should prefer moderate pruning (30–40%) unless their accelerator shows clear benefits for higher sparsity.

Distillation is the lever to pull when the memory budget is tightest or when we seek to preserve teacher behavior under aggressive architectural downsizing [4, 6, 10, 13]. Distillation is also the right choice for multilingual or domain-shifted workloads: a multilingual teacher (e.g., XLM-R or LaBSE) can impart cross-lingual structure to a compact student [1, 16]. In our experiments, distillation recovers most of the performance lost to layer/width reductions and helps stabilize calibration post-quantization.

Putting it together. For *balanced* edge deployments, we recommend the sequence: (1) distill a small student; (2) apply moderate pruning (30–40%) with brief recovery; (3) apply post-training INT8 with light calibration. If the device features sparsity-aware NPUs, increase sparsity and re-evaluate ECE and tail latency before raising k or ANN effort.

5.2. Index Selection Under Constraints

HNSW offers strong recall at small catalog sizes and low batch sizes due to its graph-locality and tunable *efSearch* [8]. IVF-PQ scales better in RAM, compressing vectors aggressively at modest recall cost [7]. Our results show a practical *rule of thumb*: prefer HNSW when $N < 1$–2M and memory is not critically constrained; switch to IVF-PQ when N grows or the resident memory budget is tight. Remember that ANN recall interacts with the reranker's k: if IVF-PQ codes are aggressive, raising k slightly (e.g., $20 \rightarrow 30$) may be cheaper than loosening PQ settings.

5.3. Caching and Telemetry

Caches lift throughput under repetitive workloads by eliminating encode, ANN, and rerank for popular contexts. The challenge is to preserve correctness as catalogs evolve. We therefore include *versioned* cache entries and short time-to-live windows; invalidations occur upon index refresh. Telemetry is the second pillar: per-request confidence, latency, cache outcome, and ANN effort (*efSearch* or n_{probe}) enable: (i) *governance*—demonstrating that thresholds and abstention policies are followed; (ii) *capacity planning*—identifying burst patterns and right-sizing batches; and (iii) *drift detection*—watching confidence histograms for shifts that merit recalibration or profile changes. Lightweight counters (e.g., RAPL) make energy visible with negligible overhead [2].

5.4. Calibration, Threshold Portability, and Human-in-the-Loop

Calibrated students support *portable thresholds*: the same $(\theta_{\text{auto}}, \theta_{\text{review}})$ can be applied across sites and device models with bounded error inflation. This is essential for operational trust. In practice, we recommend a three-zone policy: (1) *auto-accept* above θ_{auto}; (2) *gray zone* where a cheap secondary check (alias dictionary or rule) is applied; (3) *abstain* below θ_{review} to route for human verification. This policy aligns with service-level agreements and reduces the need to over-provision ANN effort or k to chase diminishing returns at the top end.

5.5. Operational Playbooks

From a practitioner's perspective, a few playbooks emerged:

- **Latency spike playbook.** First, check cache hit rates; second, reduce ANN effort (*efSearch*/n_{probe}); third, temporarily lower k; finally, enable more aggressive batching if p95 remains within SLA.
- **Energy cap playbook.** Switch to a tighter profile (Section 3.9.), prefer IVF-PQ over HNSW, and reduce batch size if thermal throttling triggers. Revisit thresholds to avoid wasted reranking on low-confidence cases.
- **Catalog growth playbook.** Rebuild IVF centroids offline; ship *delta indexes* and trigger versioned cache invalidation. If recall dips, bump n_{probe} slightly before increasing k.

5.6. Limitations and Threats to Validity

Our evaluation focuses on single-language settings without code-switching; multilingual conclusions rely on prior art and teacher choice rather than on-device cross-lingual trials. Hardware profiles reflect one embedded SoC; other NPUs/ISAs may favor different sparsity patterns or quantization schemes. Macro-F1 does not fully capture utility in highly skewed catalogs; entity-level cost weighting could change preferred operating points. Finally, our calibration uses temperature scaling over held-out data; in extreme domain shifts, vector-scale calibration or per-topic scaling may be preferable.

5.7. Privacy, Safety, and Governance

Edge deployments often exist precisely to keep data *on device*. Our design logs only hashed context identifiers and minimal metrics, supports fully offline operation, and signs models and indexes to prevent tampering. Selective abstention acts as a safety brake for ambiguous cases. For regulated environments, we recommend periodic audits of confidence distributions and error cases, along with role-based access to telemetry and cryptographic attestation of binaries.

5.8. Relation to the Base Paper

The bibliometric baseline by Shayegan & Mohammad [12] traces how semantic enrichment research diversified across methods (from rule-based annotators to neural linkers) and domains (GLAM, enterprise, social platforms). Our contribution is an *operational instantiation* of that trajectory outside the data center: we show that compression (INT8, pruning, distillation), memory-frugal ANN search, and post-hoc calibration jointly preserve most of the accuracy of modern pipelines while meeting edge constraints on latency, energy, and privacy. In this sense, our work supplies the engineering pattern that complements the macro-level trends highlighted by Shayegan & Mohammad [12].

5.9. Implications for Future Work

Three directions appear most promising. First, **vector-scale calibration** that adjusts norms/temperatures per semantic region could stabilize thresholds under drift without retraining. Second, **adaptive indexing** that modulates n_{probe} or *efSearch* based on confidence or cache history could tighten tail latency budgets further. Third, **co-designed students** that exploit N:M sparsity and quantization-friendly blocks at pretraining time may unlock another accuracy–efficiency step. Beyond modeling, **policy-aware abstention** that considers downstream costs (human review, SLA penalties) would align calibration with business objectives.

5.10. Takeaways

For teams pushing enrichment to the edge, a pragmatic recipe emerges: distill first, prune moderately, quantize to INT8, choose ANN to fit memory (HNSW for small catalogs, IVF-PQ for large), calibrate once and monitor confidence drift, and use caching plus micro-batching to smooth bursts. This combination consistently lands on a favorable Pareto frontier across our corpora while preserving auditability and privacy.

6. Conclusion

This work has presented a practical recipe for *edge-ready semantic enrichment* that combines three complementary compression levers—post-training INT8 quantization, moderate magnitude pruning, and knowledge distillation—with a memory-frugal approximate nearest neighbor (ANN) layer and a distilled micro cross-encoder, all governed by post-hoc calibration and lightweight telemetry. Across three edge-like corpora (technical manuals, incident tickets, and IoT logs), our results demonstrate that a carefully composed pipeline can retain 93–96% of a full-precision baseline's macro-F1 while reducing energy by up to two-thirds, and doing so under predictable tail-latency constraints. In short, the data-center class of neural entity linking is now attainable on embedded hardware when the system is co-designed around compression, indexing, calibration, and caching.

What this changes in practice. The proposed recipe turns efficiency from an ad hoc afterthought into a first-class design axis. Practitioners gain (i) *portable thresholds* enabled by calibration, which stabilizes decisions as models and workloads evolve; (ii) *configurable profiles* that trade accuracy for energy or latency with a small set of interpretable dials (embedding dimension, ANN effort, top-k, and confidence thresholds); and (iii) *operational guardrails*—telemetry for governance and versioned caches for safe, incremental index updates. These ingredients make it feasible to deploy enrichment in privacy-sensitive or connectivity-limited settings without resorting to brittle rules or frequent cloud roundtrips.

Limitations. Our evaluation targets single-language deployments on one embedded SoC profile; results may vary with other NPUs/ISAs, sparsity kernels, or multilingual workloads. Macro-F1, while informative, underweights rare but critical entities; cost-weighted or task-specific metrics may shift optimal operating points. Calibration relied on temperature scaling; extreme distribution shift could require vector-scale or group-wise calibration strategies.

Recommendations. For most edge scenarios, we recommend the following sequence: (1) distill a compact student from a strong teacher; (2) prune moderately (30–40%) with a brief recovery phase; (3) apply post-training INT8 quantization; (4) choose ANN to fit memory (HNSW for small catalogs, IVF-PQ for large) and tune effort before increasing top-k; (5) calibrate once on a small validation slice and monitor confidence drift; (6) enable an LRU cache with versioned invalidation to harvest locality without sacrificing correctness. This sequence consistently landed on favorable accuracy–latency–energy Pareto frontiers in our experiments.

Broader context. The bibliometric perspective highlighted by Shayegan & Mohammad [12] underscores

the rapid diversification of semantic enrichment across domains and modalities. Our contribution operationalizes that trend beyond the data center by offering an implementable, reproducible pattern for constrained devices—one that preserves auditability and privacy while sustaining modern neural accuracy.

Future work. Three directions appear most impactful: (i) *vector-scale calibration* to improve threshold portability under drift without retraining; (ii) *adaptive indexing and reranking*, where ANN effort and top-k respond to online confidence and cache state to control p95 latency; and (iii) *hardware-aware student architectures* (e.g., N:M sparsity, quantization-friendly blocks) co-designed with edge compilers. Longer term, we envision *heterogeneous offloading* that opportunistically shifts stages between device and near-edge nodes, guided by telemetry and policy, and *policy-aware selective prediction* that aligns abstention with downstream costs.

In closing, our study shows that edge deployment of neural enrichment is not merely feasible but *practical* when compression and calibration are treated as core systems design principles. We release scripts for reproducing figures and tables under this template and intend to extend the artifact with reference profiles for additional hardware targets and multilingual settings.

References

[1] Conneau, A., Khandelwal, K., Goyal, N., et al. (2020). Unsupervised cross-lingual representation learning at scale (XLM-R). In *ACL*.

[2] David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., & Le, C. (2010). RAPL: Memory power estimation and capping. *Intel Technology Journal*, 14(3), 46–59.

[3] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *ICLR*.

[4] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *NeurIPS Workshop*.

[5] Jacob, B., Kligys, S., Chen, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*.

[6] Jiao, X., Yin, Y., Shang, L., et al. (2020). TinyBERT: Distilling BERT for natural language understanding. In *EMNLP*.

[7] Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs (FAISS). *IEEE Transactions on Big Data*, 7(3), 535–547.

[8] Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using HNSW. *IEEE TPAMI*, 42(4), 824–836.

[9] Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT? *ACL*.

[10] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT. *arXiv:1910.01108*.

[11] Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *ACL*.

[12] Shayegan, M. J., & Mohammad, M. M. (2021, May). Bibliometric of semantic enrichment. In *2021 7th International Conference on Web Research (ICWR)* (pp. 202–205). IEEE.

[13] Sun, Z., Yu, H., Song, X., et al. (2020). MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *ACL*.

[14] Wolf, T., Debut, L., Sanh, V., et al. (2020). Transformers: State-of-the-art natural language processing. In *EMNLP* demo.

[15] Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*.

[16] Feng, F., Yang, Y., Cer, D., et al. (2020). Language-agnostic BERT sentence embedding (LaBSE). *arXiv:2007.01852*.

[17] Bernhardsson, E. (2015). Annoy: Approximate nearest neighbors in C++/Python. *GitHub repository*.

[18] Guo, R., Sun, P., Lindgren, E., et al. (2020). Accelerating large-scale inference with anisotropic vector quantization (ScaNN). *MLSys*.

[19] ONNX Runtime: High-performance inference engine. *https://onnxruntime.ai*.

[20] NVIDIA TensorRT Developer Guide. *https://docs.nvidia.com/deeplearning/tensorrt*.

[21] Google Coral Edge TPU Documentation. *https://coral.ai*.

[22] Piccinno, F., & Ferragina, P. (2014). From TAGME to WAT: A new entity annotator. In *ESWC*.

[23] Peters, M., Neumann, M., Iyyer, M., et al. (2018). Deep contextualized word representations. In *NAACL*.

[24] LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *NeurIPS*.