Agent-Align: Multi-Agent Preference Synthesis for Semantic Routing and Out-of-Scope Detection in Industrial RAG Systems

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) systems rely on a front-end semantic router to send each query to the right knowledge source, making routing accuracy critical. In industrial settings, training such routers is difficult because labeled data are scarce, intent boundaries are fine-grained, and out-of-scope (OOS) queries must be rejected rather than force-matched. Existing solutions either call large language models (LLMs) at inference time, incurring latency, cost, and privacy concerns, or use synthetic augmentation that rarely improves boundary discrimination. We propose Agent-Align, a two-stage framework that uses a generator–critic agent pair as an offline boundary annotator. The cloud-side stage produces accepted examples, chosen–rejected preference pairs, and near-boundary OOS examples; the edge-side stage distills these signals into a compact router with a preference-guided supervised contrastive objective (Pref-SCL), avoiding online LLM dependence. Across five benchmarks, Agent-Align achieves 93.0–95.5% routing accuracy on public in-scope datasets and 81.6% F1 on CLINC150's official OOS split, outperforming adapted baselines.

1 Introduction

Industrial RAG systems rely on a front-end semantic router to direct each user query to the correct knowledge source before generation begins (Lewis et al., 2020; Gao et al., 2023). While large language models can perform this routing via in-context learning (Ong et al., 2024), their latency, cost, and data-privacy constraints make them impractical for edge deployment where routing must complete in single-digit milliseconds. The practical alternative is a compact, locally deployed classifier, but training such a router to reliably distinguish fine-grained intents under few-shot supervision remains an open challenge.

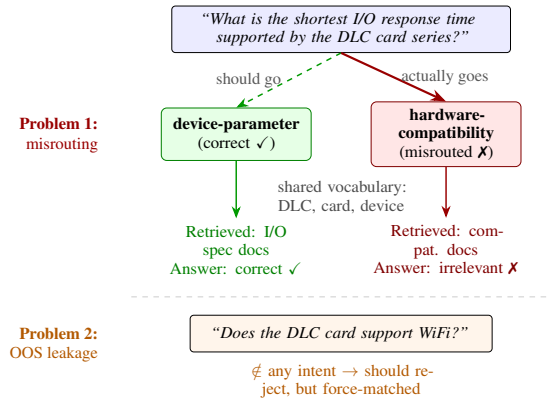

Figure 1: The dual boundary problem. **Top**: A device-parameter query is misrouted to hardware-compatibility due to shared terminology. **Bottom**: An out-of-scope query is force-matched rather than rejected.

The central difficulty is *intent-boundary discrimination*. When boundaries are blurred, the router makes two costly errors (Figure 1): (1) it *misroutes* queries to a semantically adjacent but incorrect knowledge source because both intents share surface vocabulary, causing downstream retrieval to return irrelevant documents; (2) it *fails to reject* queries outside all supported intents, forcing hallucinated answers instead of a graceful fallback. This dual boundary problem is pervasive in fine-grained benchmarks such as BANKING77 (Casanueva et al., 2020) and CLINC150 (Larson et al., 2019).

Existing few-shot methods (Zhang et al., 2021; Mehri and Eric, 2021; Tunstall et al., 2022) optimize for class-center separation but do not probe the confusable margins between adjacent intents. Data-augmentation strategies (Wang et al., 2023) increase training volume, yet generated examples cluster around canonical expressions and rarely venture into the boundary zones where errors actually occur. Without supervision that explicitly targets decision margins, a lightweight router remains brittle at the points that matter most.

Our key observation is that a *generator–critic agent pair* can serve as an offline boundary annotator, decoupling large-model judgment quality from online inference latency. Building on this insight, we propose Agent-Align, a two-stage framework. In the *cloud-side synthesis* stage, a generator–critic loop produces three forms of contrastive supervision: accepted task examples, chosen–rejected preference pairs, and near-boundary OOS examples. In the *edge-side training* stage, these signals are distilled into a compact router via Pref-SCL, a preference-guided supervised contrastive objective that pulls accepted examples together, repels rejected variants, and separates OOS items.

We evaluate on five benchmarks in a 5-shot setting. Agent-Align achieves 93.0–95.5% routing accuracy on public datasets, outperforming the adapted CoDa baseline by 6–18 points, while attaining 81.6% OOS F1 on CLINC150's official out-of-scope split.

Our contributions are:

- We propose Agent-Align, whose generator–critic agent pair explicitly probes intent-confusion zones and produces preference pairs encoding where decision boundaries should lie (addressing in-scope misrouting).
- We introduce Pref-SCL, which unifies same-intent attraction, rejected-pair repulsion, and OOS separation in a single contrastive loss (addressing OOS leakage).
- We validate on five benchmarks, demonstrating strong routing accuracy and OOS F1 against adapted baselines in the few-shot setting, with ablation confirming each component's contribution.

2 Related Work

Few-shot intent routing. Benchmarks such as CLINC150, BANKING77, and HWU64 (Larson et al., 2019; Casanueva et al., 2020; Liu et al., 2021) study fine-grained intent prediction under limited supervision. Prior work explores contrastive pre-training (Zhang et al., 2021) and example-driven matching (Mehri and Eric, 2021). Our setting differs by targeting RAG-style routing with OOS robustness as a first-class objective. LLM-based data generation (Wang et al., 2023) can bootstrap supervision from small seed sets, but typically treats synthetic data as undifferentiated augmentation. We instead preserve critic judgments that turn synthesis into a boundary-collection process.

OOS detection and contrastive learning. Prior OOS methods include energy-based scoring (Liu et al., 2020), Mahalanobis distance (Podolskiy et al., 2021), and learned thresholds (Zhan et al., 2021), but rely on encoder representation quality without boundary-adjacent training signal. Our chosen–rejected structure resembles preference-learning pipelines (Rafailov et al., 2023), but we distill preferences into a discriminative router rather than optimizing a generator. The geometry extends supervised contrastive learning (Khosla et al., 2020) with two routing-specific signals: rejected-pair repulsion encoding terminology drift, and near-boundary OOS separation.

3 Method

3.1 Task Formulation

We formulate the front-end router as a semantic-routing model over an intent schema. Given a user utterance x, the router predicts an intent label $y \in \mathcal{Y} \cup \{\texttt{__oos__}\}$ and, when slot annotations are available, token-level slot tags used as metadata filters for downstream retrieval.

3.2 Virtual Schema Construction

Rather than relying on hand-written knowledge bases, Agent-Align constructs a *virtual routing schema* from the label inventory and seed examples. For each intent, the schema-construction LLM is prompted with the target label, a domain hint, and a small set of potentially confusable intents. The output is a structured entry containing an intent description, boundary statements that distinguish it from adjacent intents, and representative keywords. This schema serves two purposes: it conditions the generator during synthesis and gives the critic explicit routing boundaries to enforce.

3.3 Generator–Critic Preference Synthesis

The core of cloud-side synthesis is a two-agent loop.

- The **generator** (Qwen-72B) proposes candidate user queries for each intent, conditioned on the schema entry and K seed examples.
- The **critic** (DeepSeek-R1-Distill-Llama-70B) inspects each candidate and returns one of three judgments: `chosen`, `rejected`, or `oos`.

We intentionally instantiate the generator and critic with different model families to reduce *same-source bias*: if a single LLM family both proposes candidates and validates them, generation artifacts and critic judgments may share correlated blind spots.

If a candidate is judged `chosen`, it is added as a synthetic training example. If it is `rejected`, the critic also returns a corrected version; the corrected query is stored as the chosen example and the original as its rejected partner, together with structured rejection metadata (rejection type, confusable intent, slot corrections). This produces the chosen–rejected preference pairs used by the contrastive objective in Section 3.5. Each preference record contains one chosen text and at most one rejected partner; if multiple rejected variants are corrected to the same chosen text, they are kept as separate records rather than merged into a multi-rejection anchor. If the critic marks a candidate as `oos`, it is stored as a near-boundary OOS example.

To ensure sufficient boundary-probing examples, Agent-Align includes a targeted hard-negative pass. When the initial synthesis does not produce enough rejected pairs for an intent, the generator is explicitly prompted to create terminology-drift and boundary-confusion candidates near adjacent intents until a minimum rejected-pair quota is reached.

3.4 Near-Boundary OOS Synthesis

Beyond critic-triggered OOS cases, Agent-Align uses the full Qwen API to explicitly generate near-boundary OOS queries conditioned on each intent's schema for datasets without official OOS splits. These examples are designed to remain close to in-domain vocabulary while falling outside all supported intents, providing training signal for the OOS separation term in the contrastive loss. For BANKING77, HWU64, SNIPS, and JCZ-dedup, synthesized OOS examples are partitioned into disjoint train, validation, and test splits. For CLINC150, which provides official OOS splits, the reported OOS-aware protocol relies on the official OOS data rather than synthetic or broad-OOS generation.

3.5 Preference-Guided Supervised Contrastive Training

The edge-side router is a compact pre-trained encoder (Liu et al., 2019; Devlin et al., 2019) with two heads: an intent classification head for routing and, when applicable, a token-level slot tagging head.

Task loss. Each mini-batch is partitioned into in-scope examples $\mathcal{B}_{\text{in}}$ and OOS examples $\mathcal{B}_{\text{oos}}$ (the latter present only when the dataset includes synthesized or official OOS training data). We define $c_{\text{slot}}{=}1$ if the dataset provides slot annotations and 0 otherwise, and a batch-level indicator $c_{\text{oos}}^{\text{batch}}{=}1$ if $\mathcal{B}_{\text{oos}} \neq \emptyset$ in the current batch and 0 otherwise. The same routing head predicts labels in $\mathcal{Y} \cup \{\texttt{__oos__}\}$; no separate binary OOS classifier is trained. The task loss is:

$$\mathcal{L}_{\text{task}} = \underbrace{\mathcal{L}_{\text{route}}^{\text{in}}(\mathcal{B}_{\text{in}})}_{\text{CE over in-scope labels}} + c_{\text{slot}} \underbrace{\mathcal{L}_{\text{slot}}(\mathcal{B}_{\text{in}})}_{\text{slot tagging}} + c_{\text{oos}}^{\text{batch}} \underbrace{\mathcal{L}_{\text{route}}^{\text{oos}}(\mathcal{B}_{\text{oos}})}_{\text{CE to the \texttt{__oos__} route}},$$

where

$$\mathcal{L}_{\text{route}}^{\text{oos}} = -\frac{1}{|\mathcal{B}_{\text{oos}}|} \sum_{x \in \mathcal{B}_{\text{oos}}} \log p_\theta(\texttt{__oos__} \mid x).$$

Here $\mathcal{L}_{\text{route}}^{\text{in}}(\mathcal{B}_{\text{in}})$ is the standard routing cross-entropy on in-scope labels, and $\mathcal{L}_{\text{slot}}(\mathcal{B}_{\text{in}})$ is the token-level slot tagging loss. Since OOS examples have no gold slot labels, no slot loss is applied to $\mathcal{B}_{\text{oos}}$. For datasets without slot annotations (CLINC150, BANKING77, HWU64), all slot-related terms are zero.

For thresholded OOS inference, we also define the log-sum-exp confidence score

$$c(x) = \log \sum_{y \in \mathcal{Y}} \exp g_y(x),$$

where $g_y(x)$ is the routing logit for in-scope label y. When enabled, the optional OOS margin term is

$$\mathcal{L}_{\text{margin}}^{\text{oos}} = \max\left(0, m + \bar{c}_{\text{oos}} - \bar{c}_{\text{in}}\right),$$

with $\bar{c}_{\text{oos}} = |\mathcal{B}_{\text{oos}}|^{-1} \sum_{x \in \mathcal{B}_{\text{oos}}} c(x)$ and $\bar{c}_{\text{in}} = |\mathcal{B}_{\text{in}}|^{-1} \sum_{x \in \mathcal{B}_{\text{in}}} c(x)$. This margin encourages in-scope examples to have larger log-sum-exp confidence than OOS examples.

Pref-SCL. Let $h(x)$ denote the mean-pooled encoder representation of utterance x, and let $z(x) = h(x)/\|h(x)\|_2$ be its ℓ_2-normalized embedding. Let $\mathcal{B}_c$ be the set of chosen in-scope examples in a mini-batch and $\mathcal{B}_o$ be the set of OOS examples. For an anchor $i \in \mathcal{B}_c$ with label y_i, we define:

$$P(i) = \{p \in \mathcal{B}_c \setminus \{i\} : y_p = y_i\},$$
$$A(i) = \{n \in \mathcal{B}_c \setminus \{i\} : y_n \neq y_i\}.$$

Let $\mathcal{I} = \{i \in \mathcal{B}_c : |P(i)| > 0\}$; anchors without in-batch positives are skipped. The OOS set $O(i) = \mathcal{B}_o$ consists of all OOS examples in the mini-batch; OOS examples are not included in $A(i)$. Each mini-batch row has at most one rejected partner $\tilde{z}_i$, matching the one-partner preference records described above; when present, $\tilde{z}_i$ participates only through the rejected-pair term and is excluded from $A(i)$. The preference-guided supervised contrastive loss is:

$$\mathcal{L}_{\text{pref}} = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i^\top z_p/\tau)}{\mathcal{D}_i},$$

where

$$\begin{aligned}\mathcal{D}_i = &\sum_{p \in P(i)} \exp(z_i^\top z_p/\tau) \\ &+ \sum_{n \in A(i)} \exp(z_i^\top z_n/\tau) \\ &+ \mathbf{1}[m_i]\, \alpha \exp(z_i^\top \tilde{z}_i/\tau) \\ &+ \beta \sum_{k \in O(i)} \exp(z_i^\top z_k^{\text{oos}}/\tau).\end{aligned}$$

Here m_i indicates the presence of a rejected partner for anchor i. The four denominator terms serve distinct geometric roles: (1) same-intent attraction, (2) inter-class repulsion, (3) *rejected-pair repulsion* that pushes accepted examples away from their terminology-drift variants weighted by α, and (4) OOS *separation* that pushes in-scope examples away from boundary-adjacent OOS items weighted by β.

Training uses class-aware batching to ensure that high-cardinality intent sets still present in-batch positives for contrastive learning. The full objective is:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda\, \mathcal{L}_{\text{pref}} + \delta_{\text{margin}}\, c_{\text{oos}}^{\text{batch}}\, \mathcal{L}_{\text{margin}}^{\text{oos}},$$

where $\delta_{\text{margin}} \in \{0, 1\}$ denotes whether the optional confidence-margin term is enabled.

3.6 Checkpoint Selection and Inference

Early stopping uses a weighted validation metric computed exclusively on validation data. We define dataset-level indicators $c_{\text{slot}}^{\text{val}}$=1 if the validation split contains slot annotations and $c_{\text{oos}}^{\text{val}}$=1 if it contains OOS gold labels (distinct from the batch-level $c_{\text{oos}}^{\text{batch}}$ used during training). The selection score is:

$$s = \frac{0.8\, a_{\text{rt}} + 0.1\, c_{\text{slot}}^{\text{val}}\, f_{\text{slot}} + 0.1\, c_{\text{oos}}^{\text{val}}\, f_{\text{oos}}}{0.8 + 0.1\, c_{\text{slot}}^{\text{val}} + 0.1\, c_{\text{oos}}^{\text{val}}},$$

where a_{rt} is routing accuracy, f_{slot} is slot F1, and f_{oos} is OOS F1.

Confidence-threshold inference. At test time, OOS decisions use the same log-sum-exp confidence score defined above:

$$c(x) = \log \sum_{y \in \mathcal{Y}} \exp g_y(x),$$

where $g_y(x)$ is the logit for intent y. A query is classified as OOS when its confidence falls below a threshold γ:

$$\hat{y} = \texttt{__oos__} \quad \text{if } c(x) < \gamma.$$

The threshold γ is calibrated on the validation split to maximize OOS F1.

4 Experimental Setup

4.1 Datasets

We evaluate on four public intent-routing benchmarks and one internal industrial dataset. **CLINC150** (Larson et al., 2019) contains 150 intents across 10 domains with an official out-of-scope (OOS) test split of 1,000 examples, making it the standard benchmark for joint intent classification and OOS detection. **BANKING77** (Casanueva et al., 2020) is a single-domain fine-grained dataset with 77 banking intents that exhibit substantial lexical overlap. **HWU64** (Liu et al., 2021) covers 64 intents across 21 domains, collected for evaluating conversational NLU services. **SNIPS** (Coucke et al., 2018) is a widely used multi-domain benchmark with 7 intents and slot annotations. **JCZ** is an internal Chinese industrial support dataset from the laser industrial processing domain, covering control software, system and hardware configurations, and enterprise-level technical Q&A for laser equipment. It contains 42 intents and 33 slot types; we report results on a deduplicated version (JCZ-dedup) that removes train/test leakage.

Table 2 summarizes the dataset statistics. Table 1 details the OOS evaluation protocol for each dataset. For datasets without an official OOS split, we construct OOS test sets exclusively from LLM-synthesized near-boundary examples generated by the same prompt family as training-time negatives but held out strictly from the training and validation sets. These held-out examples share the synthesis distribution and thus represent a *near-boundary* evaluation; they do not include naturally occurring open-world queries.

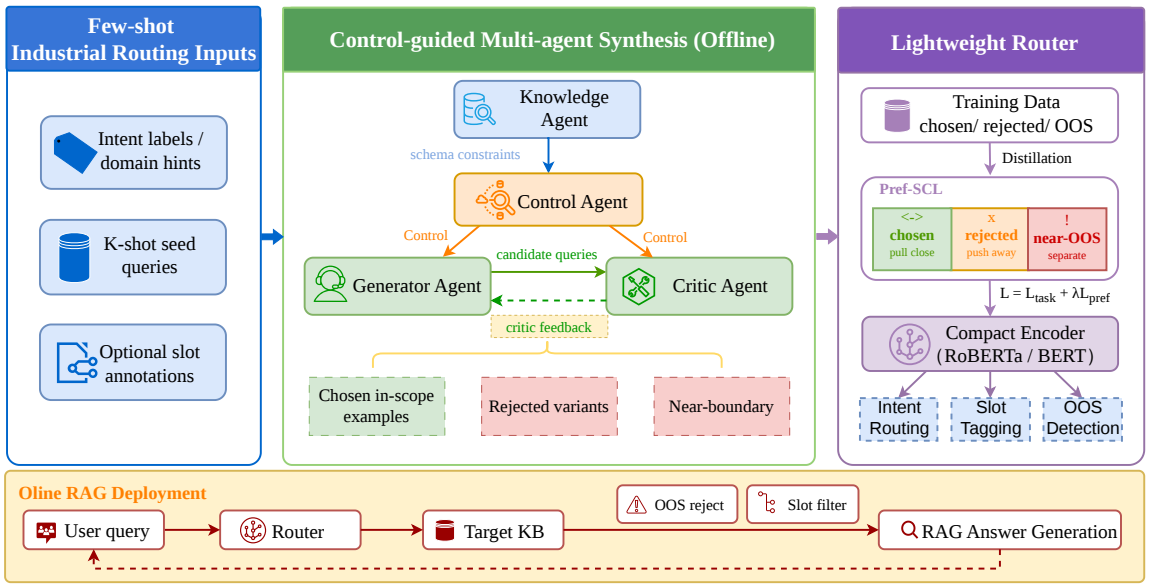

Figure 2: Overview of Agent-Align. **Left**: few-shot industrial routing inputs (intent labels, K-shot seed queries, and optional slot annotations) are fed into the offline synthesis stage. **Center**: a control-guided multi-agent system where a Knowledge Agent provides schema constraints to the Control Agent, which orchestrates a Generator Agent and a Critic Agent in a feedback loop. The generator produces candidate queries; the critic evaluates them and returns feedback, yielding three output types: chosen in-scope examples, rejected variants, and near-boundary OOS examples. **Right**: the synthesized data is distilled into a lightweight encoder (RoBERTa/BERT) via Pref-SCL, which pulls chosen examples close, pushes rejected variants away, and separates near-OOS items, producing joint intent routing, slot tagging, and OOS detection heads. **Bottom**: online RAG deployment where the router directs user queries to the target knowledge base with OOS rejection and slot-based filtering before answer generation.

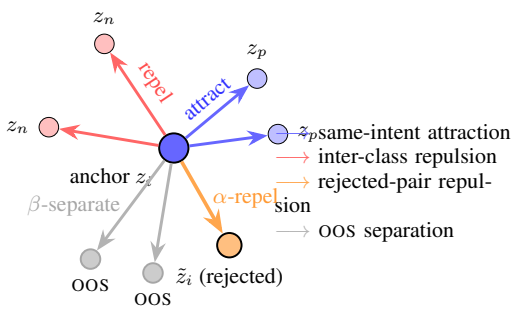

Figure 3: Geometric interpretation of Pref-SCL. The anchor (accepted example) is attracted toward same-intent positives, repelled from different-intent negatives, strongly repelled from its paired rejected variant (weighted by α), and separated from near-boundary OOS examples (weighted by β).

Dataset	OOS source	OOS generator	Shared schema?	Split
CLINC150	Official split	Human	No	1,000 test
BANKING77	Synthesized	Full Qwen API	Yes	Held-out
HWU64	Synthesized	Full Qwen API	Yes	Held-out
SNIPS	Synthesized	Full Qwen API	Yes	Held-out
JCZ-dedup	Synthesized	Full Qwen API	Yes	Held-out

Table 1: OOS evaluation protocol. "Synthesized" test sets are near-boundary examples generated with the full Qwen API and held out from training, sharing the same prompt family and schema as training-time negatives. This means reported OOS F1 on these datasets reflects near-boundary rejection ability, not open-world robustness.

4.2 Evaluation Protocol

All experiments use a 5-shot setting: K=5 seed examples per intent are sampled from the training split. The cloud-side synthesis stage uses Qwen-72B as the Generator Agent and DeepSeek-R1-Distill-Llama-70B as the Critic Agent, while near-boundary OOS examples, when synthesized, are generated with the full Qwen API. Using different model families for generation and criticism reduces same-source bias in boundary annotation. Each run follows the same protocol:

1. Construct a virtual schema from the label inventory and seed examples.
2. Run generator–critic synthesis to produce accepted examples, chosen–rejected preference pairs, and, for non-CLINC150 datasets, near-OOS examples.
3. Train the router with the joint objective described in Section 3.

Dataset	Intents	Slots	Test	OOS test
CLINC150	150	–	4,500	1,000
BANKING77	77	–	3,080	synth.
HWU64	64	–	1,076	synth.
SNIPS	7	39	700	synth.
JCZ-dedup	42	33	898	synth.

Table 2: Dataset statistics. "synth." indicates OOS evaluation uses held-out synthesized examples; CLINC150 uses its official OOS split. See Table 1 for detailed OOS source descriptions.

Dataset	Pref. examples	Rejected pairs	Rej. rate
CLINC150	1,741	451	25.9%
BANKING77	904	225	24.9%
HWU64	785	205	26.1%
SNIPS	124	32	25.8%
JCZ-dedup	456	94	20.6%

Table 3: Multi-agent synthesis statistics. Rejected pairs are used as contrastive signals in Pref-SCL.

4. Select the best checkpoint using the normalized composite validation score s (Section 3), then evaluate on held-out test data.

For CLINC150, which provides official OOS development and test splits, validation and test performance are measured on the official splits and no broad-OOS synthesis is used in the reported run. For other datasets, synthesized near-OOS examples are partitioned (80%/10%/10%) into disjoint train/validation/test subsets using the synthesis seed.

4.3 Baselines

We compare Agent-Align against the following methods:

- **CoDa adapted** (Evuru et al., 2024): A generation-augmentation approach that uses LLM-generated paraphrases to expand the seed set, followed by standard supervised fine-tuning. We adapt this method to the same 5-shot routing setting for fair comparison.
- **DETER adapted** (Lin et al., 2024): A diverse example generation strategy that produces varied synthetic training data through prompt diversification, adapted to our routing task.
- **CPFT reported**: Contrastive pre-training and fine-tuning results as reported in prior work (Zhang et al., 2021), representing the few-shot intent detection state of the art without synthetic augmentation.
- **Direct LLM Router**: Qwen3-32B (Qwen Team, 2025) used directly for zero-shot label-list prompting, where the prompt provides the full intent label inventory (names and schema descriptions) and the input utterance, without seed exemplars. This represents a strong but costly LLM-routing baseline that is impractical for edge deployment (see Section 5.6 for a quantitative comparison).
- **Vanilla Encoder**: Seed-only supervised fine-tuning without any data augmentation or preference training, representing the few-shot lower bound. English datasets use `roberta-base` (Liu et al., 2019); JCZ-dedup uses `bert-base-chinese` (Devlin et al., 2019).

4.4 Training Details

The edge-side router uses `roberta-base` (Liu et al., 2019) for English datasets and `bert-base-chinese` (Devlin et al., 2019) for JCZ-dedup. Training uses AdamW with a learning rate of 2e-5 and batch size of 32 with class-aware sampling. The contrastive temperature is τ=0.07; preference weighting coefficients are α=2.0 (rejected-pair repulsion), β=1.0 (OOS separation), and λ=0.5 (loss balancing). Early stopping monitors the composite validation score (Section 3) with patience of 10 epochs. At inference time, OOS decisions use energy-score thresholding (Liu et al., 2020) as described in Section 3.

4.5 Metrics

We report three metrics: **Routing Accuracy**, defined as the fraction of in-scope test queries assigned to the correct intent (excluding gold OOS examples); **OOS F1**, defined as the F1 score for detecting out-of-scope queries; and **Slot F1**, defined as token-level slot tagging F1, reported only for datasets with slot annotations (SNIPS, JCZ-dedup).

4.6 Synthesis Statistics

Table 3 reports the volume and composition of data produced by the multi-agent synthesis stage. The stable rejected-pair rate across datasets (20–26%) suggests that the critic consistently produces preference supervision at a similar density regardless of domain.

5 Results and Analysis

5.1 Main Results

Table 4 presents the main routing results across all five benchmarks in the 5-shot setting. Agent-Align achieves 93.0–95.5% routing accuracy on the four public datasets and 87.8% on the more challenging industrial JCZ-dedup dataset, consistently outperforming all adapted baselines.

Three observations emerge from the main results. First, Agent-Align matches or exceeds the Direct LLM Router (Qwen3-32B, 32.8B parameters) on four of five datasets while requiring only a compact encoder at inference time. On CLINC150, Agent-Align trails by a small margin of 0.5 points (93.0 vs. 93.5). This demonstrates that the accuracy gap can be closed while eliminating the latency, cost, and privacy overhead of online LLM routing. Second, the gap between Agent-Align and the generation-augmentation baselines is substantial: +6.2 to +44.6 over CoDa and +20.7 to +66.4 over DETER on routing accuracy, indicating that preference-guided boundary supervision provides fundamentally stronger training signal than volume-only augmentation. Third, Agent-Align maintains strong OOS rejection alongside high in-scope accuracy: on CLINC150's official OOS split, Agent-Align achieves 81.6% OOS F1 while preserving 93.0% routing accuracy, demonstrating that these two objectives are complementary rather than conflicting under our framework.

5.2 Few-Shot Scaling

Table 5 examines how performance scales with annotation budget across all five datasets.

Even in the 1-shot setting, Agent-Align achieves 75–88% routing accuracy (+24 over seed-only fine-tuning), and scales to 90–96% at 10-shot. The gap is largest for OOS F1, indicating that near-boundary synthesis provides benefits that additional seed examples alone cannot replicate.

5.3 Out-of-Scope Detection

Table 6 presents the OOS detection performance in detail.

OOS evaluation setup. CLINC150 uses its official OOS split (⋆), which contains diverse naturally occurring open-world queries. For other datasets, OOS test sets consist exclusively of LLM-synthesized near-boundary examples held out from training (Table 1). Because these test items share the synthesis distribution with training-time negatives, the reported OOS F1 on non-CLINC150 datasets reflects near-boundary rejection ability rather than open-world robustness.

On synthesized-OOS datasets, Agent-Align outperforms CoDa by 19.5–31.0 points and DETER by 46.9–56.8 points. On CLINC150's official OOS split, the margin over CoDa is smaller (+2.9), likely because CLINC150's diverse open-world OOS examples are less sensitive to near-boundary synthesis. In our adapted implementations, generation-only augmentation underperforms the full method on OOS metrics, consistent with the hypothesis that explicit boundary supervision helps.

5.4 Ablation Study

Appendix Table 8 presents ablation results across all five datasets in the 5-shot setting.

The ablation reveals a clear hierarchy. **Pref-SCL** is the most critical: removing it causes 13.7–34.2 point routing accuracy drops. **Near-OOS synthesis** most strongly affects OOS F1 (drops 26–41 points) and also degrades routing accuracy by 7.8–25.2 points, indicating that OOS boundary awareness sharpens in-scope boundaries. **Hard negatives** contribute 9.1–25.1 point gains, largest on CLINC150 and JCZ-dedup where adjacent intents share terminology. The **schema**, **feedback**, and **critic** components each contribute 5.8–18.1 points, with importance increasing on complex datasets.

5.5 Slot Tagging and Error Analysis

For datasets with slot annotations, Agent-Align achieves 94.2% slot F1 on SNIPS and 82.3% on JCZ-dedup, showing that Pref-SCL does not degrade token-level tagging.

Residual routing errors on BANKING77 (68% of cases) involve confusion clusters where intent pairs share nearly identical surface patterns and differ only in pragmatic implication, requiring world knowledge beyond few-shot synthesis. On JCZ-dedup, errors concentrate on pairs distinguished by numeric parameter ranges rather than lexical cues, suggesting numeric reasoning in the critic is a promising future direction.

5.6 Deployment Efficiency Analysis

Appendix Table 9 compares deployment cost between Agent-Align's distilled routers and the Direct LLM Router. The distilled routers achieve sub-10 ms mean latency on a single A10 GPU, satisfying single-digit-millisecond requirements for on-

Method	BANKING77	CLINC150	HWU64	SNIPS	JCZ-dedup
Routing Accuracy (%)					
Vanilla Encoder	78.1	58.8	80.2	86.1	64.9
DETER adapted	59.8	69.1	56.0	74.2	21.4
CoDa adapted	75.7	86.8	76.7	83.3	43.2
CPFT reported	80.9	92.3	82.0	–	–
Direct LLM Router	91.5	**93.5**	93.0	94.0	86.0
Agent-Align (ours)	**93.5**	93.0	**95.5**	**94.9**	**87.8**
OOS *F1 (%)*					
DETER adapted	33.8	25.1	42.1	51.6	29.6
CoDa adapted	61.2	78.7	70.2	74.2	53.4
Agent-Align (ours)	**80.7**	**81.6**	**98.9**	**99.2**	**84.4**

Table 4: Main results in the 5-shot setting. Seed sampling uses seed = 42. **OOS-aware protocol**: checkpoint selection uses the composite score s including OOS F1 when a validation OOS split is available; OOS test uses CLINC150's official split or synthesized near-boundary holdouts for other datasets (see Table 1). Best results in **bold**. CPFT results are reported by prior work and not rerun under our protocol.

		Agent-Align			Vanilla Encoder		
Dataset	Metric	1-shot	5-shot	10-shot	1-shot	5-shot	10-shot
CLINC150	Rout. Acc	79.8	93.0	91.1	48.2	58.8	68.2
	OOS F1	64.3	81.6	71.3	38.6	44.1	50.3
BANKING77	Rout. Acc	85.2	93.5	94.9	61.2	78.1	84.2
	OOS F1	68.4	80.7	83.2	42.1	55.2	60.1
HWU64	Rout. Acc	87.4	95.5	96.3	64.6	80.2	85.0
	OOS F1	87.1	98.9	99.1	50.3	64.1	70.1
SNIPS	Rout. Acc	88.4	94.9	95.8	73.1	86.1	88.9
	OOS F1	84.6	99.2	100	49.2	61.3	65.8
JCZ-dedup	Rout. Acc	74.8	87.8	90.1	51.2	64.9	70.4
	OOS F1	70.1	84.4	86.2	36.6	48.2	52.1

Table 5: Few-shot scaling results

Dataset	Agent-Align	CoDa	DETER
OOS *F1 (%)*			
CLINC150*	**81.6**	78.7	25.1
BANKING77	**80.7**	61.2	33.8
HWU64	**98.9**	70.2	42.1
SNIPS	**99.2**	74.2	51.6
JCZ-dedup	**84.4**	53.4	29.6

Table 6: OOS F1 comparison. *CLINC150 uses its official open-world OOS split; other datasets use held-out synthesized near-boundary examples (Table 1).

line routing. Compared to the Direct LLM Router, Agent-Align achieves comparable or higher routing accuracy with over two orders of magnitude fewer parameters and roughly 70–84× lower peak VRAM under the reported FP16 estimates.

6 Conclusion

We presented Agent-Align, a two-stage framework that uses a generator–critic agent pair to synthesize boundary-targeted preference data offline, then distills it into a compact router via the Pref-SCL objective. Experiments on five benchmarks show 93.0–95.5% routing accuracy on public in-scope benchmarks and 81.6% OOS F1 on CLINC150's official split, outperforming the adapted CoDa baseline by 6–18 points on routing accuracy while requiring only sub-1 GB VRAM and single-digit-millisecond latency at inference. These results demonstrate that structured preference synthesis is a promising direction for few-shot industrial routing. Future work will explore dynamic intent inventories, multi-turn contexts, and cross-lingual transfer.

Limitations

Despite these encouraging results, several caveats should be considered when interpreting the findings.

First, OOS evaluation is not uniform across datasets. Only CLINC150 provides an official

OOS test split composed of diverse, naturally occurring open-world queries. For the remaining datasets (BANKING77, HWU64, SNIPS, JCZ-dedup), the OOS test sets consist of LLM-synthesized near-boundary examples that share the same prompt family and schema as training-time negatives (Table 1). Because these synthetic OOS items share distributional assumptions with training data, the reported OOS F1 on these datasets reflects near-boundary rejection ability and may overestimate robustness against genuinely novel, open-world out-of-distribution queries. Practitioners should validate OOS robustness against naturally collected rejection data before production deployment.

Second, the cloud-side synthesis stage requires access to capable LLM services: Qwen-72B for generation, DeepSeek-R1-Distill-Llama-70B for critic judgments, and, when synthetic near-boundary OOS examples are enabled, the full Qwen API for OOS synthesis. While this cost is incurred only once offline and does not affect inference latency, it introduces a dependency on large-model availability during the initial data preparation phase.

Third, JCZ is an internal industrial dataset that cannot be publicly released. Although we describe its structure and report results on a deduplicated evaluation split, full external replication on this dataset is not currently possible.

References

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Maïzi, and Vincent Zanichelli. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chandra Sekhar Evuru, Faeze Brahman, and Snigdha Chaturvedi. 2024. CoDa: Constrained generation based data augmentation for low-resource NLP. In *Findings of the Association for Computational Linguistics: NAACL 2024*. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint*.

I-Fan Lin, Faegheh Hasibi, and Suzan Verberne. 2024. Generate then refine: Data augmentation for zero-shot intent detection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 33.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. *Benchmarking Natural Language Understanding Services for Building Conversational Agents*, pages 165–183. Springer Singapore.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Shikib Mehri and Mihail Eric. 2021. Example-driven intent prediction with observers. In *Proceedings of*

the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2979–2992, Online. Association for Computational Linguistics.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint*.

Alexander Podolskiy, Dmitry Liber, Alexander Molchanov, and Benjamin Gusev. 2021. Revisiting Mahalanobis distance for transformer-based out-of-domain detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint*.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Li-Ming Zhan, Haowen Liang, Bo Liu, Lu Fan, Xiao-Ming Wu, and Albert Y.S. Lam. 2021. Out-of-scope intent detection with self-supervision and discriminative training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jianguo Zhang, Trung Bui, Seunghyun Yoon, Xiang Chen, Zhiwei Liu, Congying Xia, Quan Hung Tran, Walter Chang, and Philip Yu. 2021. Few-shot intent detection via contrastive pre-training and fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1906–1912, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Implementation Details

All experiments use Qwen-72B as the Generator Agent and DeepSeek-R1-Distill-Llama-70B as the Critic Agent, while near-boundary OOS examples, when synthesized, are generated with the full Qwen API. This heterogeneous assignment is designed to reduce same-source bias between candidate generation and candidate validation. The Direct LLM Router baseline uses Qwen3-32B with zero-shot label-list prompting: the prompt contains the full intent schema (label names and descriptions) and the input utterance, but no seed exemplars (K=0). For the edge-side router, we use `roberta-base` (Liu et al., 2019) for English datasets and `bert-base-chinese` (Devlin et al., 2019) for JCZ. Training uses AdamW with a learning rate of 2e-5, batch size of 32 with class-aware sampling, and early stopping based on the validation selection score. The contrastive temperature τ is set to 0.07, and the preference weighting coefficients are α=2.0, β=1.0, and λ=0.5.

B Synthesis Prompt Templates and Parameters

This section provides the prompt templates and sampling parameters used in the multi-agent synthesis stage, enabling reproducibility on public datasets.

LLM configuration. The Generator Agent uses Qwen-72B, the Critic Agent uses DeepSeek-R1-Distill-Llama-70B, and explicit near-boundary OOS synthesis, when enabled, uses the full Qwen API. All calls are made through OpenAI-compatible chat-completions endpoints with temperature 0.4, max output length 2048 tokens, and up to 3 JSON-parse retries. Every prompt instructs the model to return only valid JSON without Markdown fences.

Schema construction prompt (user turn).

```
Create a compact schema entry for one
intent label.
Intent label: {intent}
Domain hint: {domain_hint}
Adjacent or potentially confusable
labels: {peer_intents}
Return JSON with keys: intent,
description, boundaries, keywords.
```

Candidate generation prompt (user turn).

```
Target intent: {intent}
Target schema entry:
{schema_entry_json}
Few-shot seed examples:
{seed_examples_json}
Generate {num_candidates} candidate
queries for this intent.
Queries should be diverse, colloquial,
and close to real user phrasing.
Return a JSON array. Each item: {text,
slots}.
```

Critic judgment prompt (user turn).

```
Candidate query: {candidate_text}
Target intent: {target_intent}
Decide whether the query is: chosen / rejected / oos.
Return JSON with keys: status, preferred_text, rejected_text, reason, rejection_type, confusable_intent, slot_corrections.
rejection_type ∈ {terminology_drift, boundary_confusion, unsupported_slot, schema_mismatch, null}.
```

Near-boundary OOS generation prompt (user turn).

```
Reference intent: {intent}
Generate {num_near_oos_per_intent} near-boundary OOS queries.
They should sound close to domain vocabulary but be outside all supported intents. Return JSON array: {text, reason}.
```

Synthesis quotas. Table 7 summarizes the key public-data synthesis parameters. Candidate generation, critic judgment, and boundary-control parameters are shared across public datasets; synthetic OOS generation is used for public datasets without official OOS splits, while CLINC150 uses its official OOS splits and disables broad-OOS generation in the reported run.

Dataset	Cand.	Near-OOS	Boundary	Min rej.	Max rounds	Broad-OOS	OOS source
CLINC150	8	0	4	1	2	0	official
BANKING77	8	20	4	1	2	0	synthetic
HWU64	8	20	4	1	2	0	synthetic
SNIPS	8	20	4	1	2	0	synthetic

Table 7: Per-dataset synthesis quotas for public datasets. "Cand." is candidates per intent, "Near-OOS" is generated near-boundary OOS examples per intent, and "Boundary" is boundary candidates per retry round. Synthetic OOS examples are split 80/10/10 into train/validation/test for non-CLINC150 datasets; CLINC150 uses official OOS splits. All calls use temperature 0.4 and maximum output length 2048.

Filtering and deduplication. Generated text is normalized by removing BOS/EOS markers and collapsing whitespace. Candidate items with empty text or malformed JSON are skipped. After synthesis, task examples and OOS examples are deduplicated by exact `(text, intent)` key. The rejected-pair rate (20–26% across datasets, Table 3) indicates that the critic produces nontrivial boundary feedback; downstream value is validated through the ablation study (Table 8).

C Additional Results

This appendix provides the full ablation, deployment-cost, and JCZ-variant tables referenced from the main text.

Dataset	Configuration	Acc	OOS
CLINC150	Agent-Align (full)	93.0	81.6
	w/o Pref-SCL	58.8	48.5
	w/o near-OOS	67.8	40.9
	w/o hard neg.	67.9	59.1
	w/o schema	76.0	63.0
	w/o feedback	75.2	62.4
	w/o critic	74.9	59.6
BANKING77	Agent-Align (full)	93.5	80.7
	w/o Pref-SCL	75.6	59.1
	w/o near-OOS	81.9	54.8
	w/o hard neg.	80.4	65.0
	w/o schema	84.2	70.3
	w/o feedback	85.1	69.4
	w/o critic	83.1	68.1
HWU64	Agent-Align (full)	95.5	98.9
	w/o Pref-SCL	78.1	71.2
	w/o near-OOS	84.6	65.5
	w/o hard neg.	83.3	79.0
	w/o schema	86.7	84.2
	w/o feedback	87.2	83.6
	w/o critic	85.4	82.1
SNIPS	Agent-Align (full)	94.9	99.2
	w/o Pref-SCL	81.2	74.1
	w/o near-OOS	87.1	69.0
	w/o hard neg.	85.8	80.6
	w/o schema	89.1	87.6
	w/o feedback	88.7	86.1
	w/o critic	87.9	85.2
JCZ-dedup	Agent-Align (full)	87.8	84.4
	w/o Pref-SCL	62.1	55.3
	w/o near-OOS	69.7	50.4
	w/o hard neg.	68.3	61.2
	w/o schema	74.2	68.1
	w/o feedback	73.6	66.3
	w/o critic	71.1	64.2

Table 8: Ablation study across all datasets (5-shot). Acc = Routing Accuracy (%), OOS = OOS F1 (%). Each row removes one component while keeping all others intact.

Model	Params	Prec.	Mean lat. (ms)	p95 lat. (ms)	Peak VRAM (GB)
Agent-Align (EN)†	125M	FP16	8.1	9.4	0.94
Agent-Align (ZH)†	103M	FP16	7.7	8.9	0.78
Qwen3-32B‡	32.8B	FP16	∼75	–	≥65.6

Table 9: Deployment cost comparison. †Measured on a single NVIDIA A10 (batch 1, max length 96, N=1k queries). ‡Qwen3-32B values are FP16 estimates (weight memory alone: 32.8B × 2 bytes ≈ 65.6 GB); actual multi-GPU serving with quantization would reduce VRAM but requires additional infrastructure. Under these estimates, Agent-Align uses over two orders of magnitude fewer parameters and roughly 70–84× lower peak VRAM.

JCZ dataset variants. The main paper reports JCZ-dedup to avoid data leakage. The non-deduplicated JCZ variant reaches 93.2 routing accuracy, 75.6 Slot F1, and 82.4 OOS F1; JCZ-dedup reaches 87.8 routing accuracy, 82.3 Slot F1, and

84.4 OOS F1.