Dataset	No. of Intents	No. of Queries		
		Train	Valid	
			In Scope	OOS
ALC	8	150	338	128
ADP	13	683	803	91
OADP	13	-	430	56

Table 2: Data Statistics for AID3 dataset

training sentences, these are expected to help the model avoid latching onto any spurious patterns and help overall learning.

3.1.2 Adaptive ICL + CoT based Intent Detection using LLMs

Fig 2 shows how we use LLMs with adaptive ICL and CoT prompting for intent detection. During offline processing, we embed all training examples using a sentence transformer model and store the embedding vectors in a DB. Additionally, we generate and store descriptions for every intent from training data using LLM. During inference, we embed the user query using the same transformer model and retrieve top-k most similar queries per intent with similarity >t, where t is retriever threshold. We construct prompt for LLM using retrieved ICL examples, stored intent descriptions and static task specific instructions.

3.1.3 Uncertainty based Query Routing

High compute and latency costs of LLMs make them prohibitively expensive to use in production at scale.[3] Hence, we propose a hybrid system which routes incoming queries to LLMs for intent detection only if SetFit model is uncertain. We sample M predictions from the SetFit model using Monte Carlo (MC) dropout (Gal and Ghahramani, 2016) and use variance of the predictions as an uncertainty estimate.

3.2 Datasets

We use SOFMattress, Curekart and Powerplay11 datasets from HINT3 (Arora et al., 2020). We also use **AID3**[4], a collection of three internal multi-label datasets shown in Table 2 - ALC, ADP and OADP, each containing diverse set of PII redacted in-scope and OOS real world queries from shopping domain. Both ALC and ADP contain queries from deployed shopping assistant, whereas OADP contains queries from single turn QnA forum. We use OADP to test out of distribution generalization while using ADP train set. See Appendix A.1 for more details on AID3. Label space size across HINT3 and AID3 datasets varies from 8 till 59 and all these datasets are real world intent detection datasets from deployed TODS which mimic real world scenarios and production challenges like handling intents with very-broad to very-specific scopes, imbalanced training datasets with very few examples per intent. By evaluating on HINT3 and AID3 datasets we include scenarios where there are large number of intents (59 being the maximum label space size) and also include multi-label scenarios (3 out of 6 of our datasets are also multi-label), which makes our evaluation more comprehensive.

3.3 Experiment Setup

SetFit. We use MPNet (Transformers, 2021; Song et al., 2020) as the backbone and use linear layer with sigmoid as differentiable head. We do hyperparameter search over search space given in Table 6 using Optuna (Akiba et al., 2019) and report best valid set results across all datasets. For MC Sampling, we use 0.1 dropout across hidden and attention layers in the backbone.

LLMs. We use BGE sentence transformer (BAAI, 2023) as the retriever and do grid search over k and t with search space specified in Table 7 and report best valid set results. To prevent LLMs from using any spurious patterns from intent label names, especially for open source datasets, we randomly mask them to Label-xx, where xx is some random integer. We use Claude v3 Sonnet to generate label descriptions for each intent for all datasets and keep them consistent across all LLMs.

Metrics. We use F1-Score as the primary performance metric. Additionally, we use OOS Recall (Larson et al., 2019) and OOS AUCROC to compare model's OOS detection capabilities and use in-scope accuracy to compare their in-scope performance.

See Appendix A.2 for more details on implementation and experiment setup across models.

3.4 Results

Evaluation results from 7 SOTA LLMs across two LLM families (Claude, Mistral) are shown in Table 1. Overall Claude v2, v3 LLMs and Mistral Large have similar performance, but Claude v3 Haiku is better amongst them with respect to la-

[3]Mechanisms like caching can help somewhat but we skip their discussion for brevity.

[4]The splits of all three datasets in AID3 were prepared specifically for experiments done as part of this work and performance on them does not reflect our production system's performance.

	SOFMattress	Curekart	PowerPlay11	ALC	ADP	OADP	Avg Score
Claude v1 Instant	0.229	0.241	0.122	0.742	0.143	0.000	0.246
Claude v2	0.688	0.701	0.580	0.945	0.330	0.232	0.579
Claude v3 Haiku	**0.736**	0.716	0.561	**0.961**	0.593	0.036	0.601
Claude v3 Sonnet	0.479	0.436	0.402	0.953	0.440	0.036	0.458
Mistral 7B	0.465	0.376	0.205	0.781	0.154	0.018	0.333
Mixtral 8x7B	0.382	0.391	0.455	0.914	0.264	0.036	0.407
Mistral Large	0.646	**0.771**	0.602	0.945	**0.615**	**0.268**	**0.641**
SetFit (Baseline)	0.563	0.293	**0.798**	0.594	0.022	0.000	0.378
SetFit + Neg Aug	0.681	0.592	0.665	0.844	0.154	0.000	0.489

Table 3: Out of Scope Recall at best F1 Score of various SOTA LLMs with fine tuned sentence transformer models across AID3 and HINT3 datasets

	M	SOF Mattress	Cure Kart	Power Play11	ALC	ADP	OADP	Avg score		Avg latency	Delta Avg score		Latency fraction
									w/o OADP			w/o OADP	
SNA	**-**	0.672	0.709	0.639	0.848	0.625	0.459	0.658	0.698	0.030	-0.078	-0.061	0.013
v3 Haiku	**-**	0.815	0.775	0.646	0.849	0.715	0.619	0.736	0.760	2.345	0.000	0.000	1.000
SNA + v3 Haiku	5	0.719	0.734	0.654	0.849	0.653	0.473	0.680	0.722	0.748	-0.056	-0.038	0.319
	10	0.740	0.747	0.671	0.863	0.666	0.489	0.696	0.737	1.005	-0.040	-0.022	0.429
	20	0.730	0.756	0.690	0.855	0.668	0.485	0.697	0.740	1.287	-0.039	-0.020	0.549
Mistral-L	-	0.767	0.779	0.668	0.907	0.688	0.601	0.735	0.762	3.867	0.000	0.000	1.000
SNA + Mistral-L	5	0.712	0.739	0.648	0.872	0.651	0.481	0.684	0.724	1.063	-0.051	-0.037	0.275
	10	0.726	0.747	0.668	0.879	0.662	0.497	0.696	0.736	1.453	-0.038	-0.025	0.376
	20	0.719	0.761	0.692	0.872	0.664	0.498	0.701	0.742	1.657	-0.034	-0.020	0.428

Table 4: Table showing F1 score of two best LLMs (Claude v3 Haiku and Mistral Large) and SetFit + Neg Aug (SNA) hybrid system with varying number of samples (M) from MC dropout.

tency. We see that adding negative augmentation to baseline SetFit improves performance by >5%, but still has ∼8% poor predictive performance with respect to best performing LLM. SetFit is about 56 times faster than overall best LLM (v3 Haiku). Additionally, all models see lower performance for OADP as compared to ADP but SetFit has one of the largest drop in performance (∼15%) for OADP as compared to ADP. This shows lack of generalization ability of smaller SetFit models in comparison to LLMs. Table 3 shows that all models including LLMs struggle with OOS detection with poor OOS recall across datasets.

Table 4 shows hybrid system results for two best performing LLMs. We see that with the hybrid system we are able to bring performance gap further down to ∼2% (from ∼6%) for all datasets for which train and test data were from same distribution (i.e. except OADP) and down to ∼4% (from ∼8%) including OADP at ∼50% reduced latency[5]. Increasing number of samples (M) in MC dropout does not increase performance significantly.

[5]Latency would reduce further if we do MC sampling in batches. See latency discussion in Appendix A.2.

4 LLMs and OOS Detection

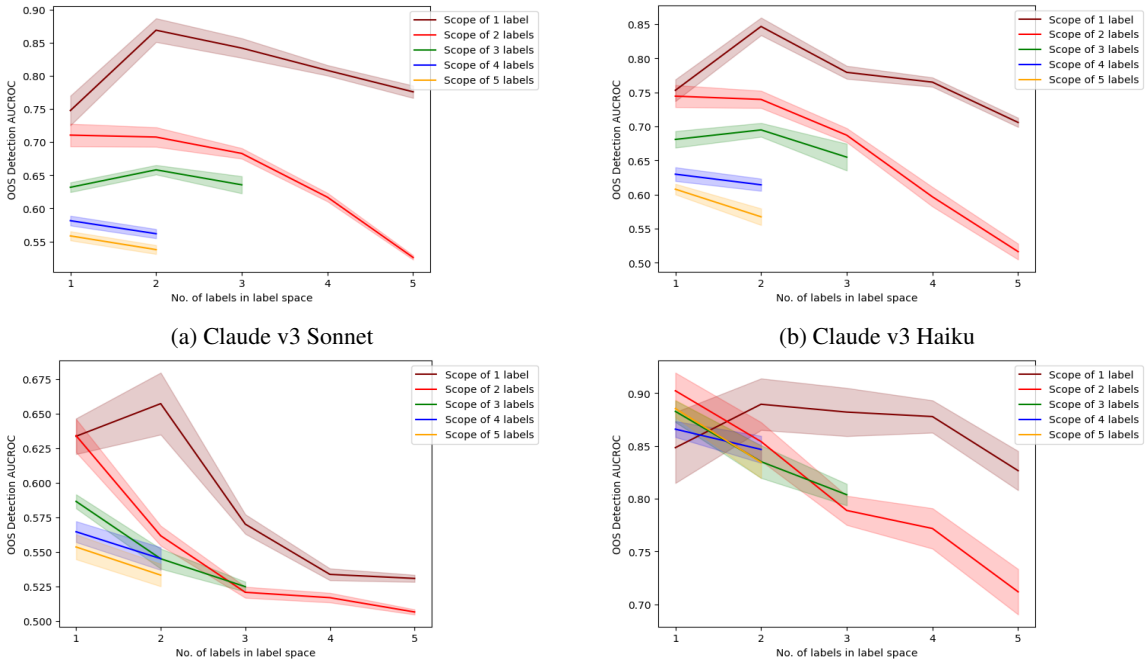
Evaluation results in Sec 3.4 showed that LLMs struggle with OOS detection. Hence, in this section we do a controlled study to better understand behavior of LLM based intent detection with special focus on their OOS detection capabilities (Sec 4.1) and based on the insights propose a novel methodology for OOS detection to improve LLMs performance (Sec 4.2).

4.1 Analyzing LLMs OOS Detection Abilities

We first describe how we setup a controlled experiment to understand how varying "scope of intents" and "no. of labels" in the label space affects LLM performance, and then share our analysis results.

Dataset. We hand curate a dataset with hierarchical label space consisting of 20 leaf intents/labels and two unique parent intents as shown in Table 8. From it, we create new intents with varying scope of $S \in [1, 5]$ labels by randomly combining S leaf intents from the same parent, without replacement. This is realistic because in real world intent scope is driven by bot usecases and scope of APIs/systems which TODS can access.

Experiment Setup. We experiment by varying



(a) Claude v3 Sonnet

(b) Claude v3 Haiku

(c) Mistral 7B

(d) SetFit + Neg Aug (SNA)

Figure 3: Change in OOS detection performance with number of labels in label space and scope of labels.

"scope of intents" by choosing intents from the newly created intents with scope of S labels with $S \in [1, 5]$ and experiment with varying "no. of labels" in label space by randomly picking L different intents of the required scope with $L \in [1, 5]$. Higher S leads to intents with broader scope. We report results based on runs on 10 randomly created datasets for every experiment. See Appendix A.3 for more details on the setup.

Results and Analysis. Fig 3[6] shows how OOS detection AUCROC for LLMs is affected with change in "scope of intents" and "no. of labels" in the label space. We see significantly more performance degradation across all LLMs in comparison to SNA model with increase in "scope of intents" and "no. of labels" in label space. This highlights greater importance of class design for LLMs and suggests that fine grained labels and smaller label spaces are better for LLM's OOS detection capabilities. From Fig 5 in Appendix A.3 we see that in-scope accuracy of LLMs is relatively immune to change in "scope of intents" but degrades with increase in label space size. However, degradation in OOS detection AUCROC is worse than in-scope accuracy degradation with increase in label space size. SNA model on the other hand does show degradation in in-scope accuracy as well with both increase in "scope of intents" and "no. of labels" in label space.

[6]Curves with scope of label > 2 are truncated because we sample and combine leaf nodes without replacement to create non-conflicting intents with bigger scope.

4.2 OOS Detection using LLMs Internal Representations

Motivated by the insights from controlled experiment, we propose a two step methodology using LLM's internal representations to improve its performance which we describe in this section.

4.2.1 Methodology

Fig 4 shows our proposed methodology. During offline processing, we generate representation of each sentence in the training data by obtaining LLM decoder layer's last prompt token's representation. Then during inference, we perform following steps.

Step 1. Firstly, we prompt the LLM to predict one of the in-scope labels without asking it to predict out of scope by completely discarding out of scope from label space given to LLM in the prompt.

Step 2. Then, based on in-scope label predicted from the previous step, we generate incoming query's representation in similar way as done during offline processing using LLM's decoder layer. We then compare this representation with representations of training instances of predicted in-scope label from the first step.

This ensures reduced label space for OOS detection but adds low latency overhead for generating