Zero-shot Slot Filling in the Age of LLMs for Dialogue Systems

Mansi Rana
Kadri Hacioglu
Sindhuja Gopalan
Maragathamani Boothalingam
Uniphore
{mansi.rana,kadri.hacioglu,sindhuja,maragathamani}@uniphore.com

Abstract

Zero-shot slot filling is a well-established subtask of Natural Language Understanding (NLU). However, most existing methods primarily focus on single-turn text data, overlooking the unique complexities of conversational dialogue. Conversational data is highly dynamic, often involving abrupt topic shifts, interruptions, and implicit references that make it difficult to directly apply zero-shot slot filling techniques, even with the remarkable capabilities of large language models (LLMs). This paper addresses these challenges by proposing strategies for automatic data annotation with slot induction and black-box knowledge distillation (KD) from a teacher LLM to a smaller model, outperforming vanilla LLMs on internal datasets by 26% absolute increase in F1 score. Additionally, we introduce an efficient system architecture for call center product settings that surpasses off-the-shelf extractive models by 34% relative F1 score, enabling near real-time inference on dialogue streams with higher accuracy, while preserving low latency.

1 Introduction

Slot filling for product-centric business use cases involves extracting essential information from customer interactions such as inquiries, complaints or feedback, and organizing it into predefined slots like product name, issue type, customer details, and resolution status. This approach enables customer service teams to quickly identify the nature of problems, streamline responses, and enhance the overall customer experience by automating certain aspects of the process, resulting in faster and more efficient support. Slot filling also enables thorough real-time and after-call analysis by organizing and making key conversation details easily accessible. This approach is often used to enhance after-call summaries, thereby reducing the need for agents to spend time manually writing reports.

The motivation to implement Zero-Shot NLU (Bapna et al., 2017; Palatucci et al., 2009), for slot filling lies in addressing traditional limitations (Mehri and Eskenazi, 2021) like high time to value (TTV), reliance on labeled data, and costly iterative training. Zero-shot models (Touvron et al., 2023) allow for immediate deployment without prior training. However, applying these methods to conversational data is challenging due to ambiguity, lack of context, and interruptions. For instance, correctly mapping implicit mentions or resolving references like "she," "he," or "it" is difficult without strong contextual understanding. Conversational shifts, slot values spanning across multiple turns, and slang expressions further complicate this task. While LLMs have improved contextual understanding, they have also introduced new challenges in deployment, such as latency, concurrency, and maintaining cross-domain functionality (Shi et al., 2023).

Combining slot descriptions with a small set of example slot values improves the model's ability to generalize across different domains, though its reliance on accessible and representative examples remains a limitation (Shah et al., 2019). To deal with ambiguity, they were reformulated into concrete questions by Du et al. (2021), but poorly framed questions can negatively impact accuracy in slot prediction. Prompting techniques (Li et al., 2023; Luo and Liu, 2023) enhanced adaptability by offering explicit cues or feedback, and improved adaptability while increasing complexity and computational overhead.

To reduce computational overhead, this task was treated as a joint problem by combining intent detection with slot filling (Zhang and Wang, 2016), but errors in one task can adversely impact the other. Retrieval augmented generation (RAG) approaches demonstrate strong generalization in low data settings (Glass et al., 2021), but the effectiveness of the retrieval mechanism can be a bottle-

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 697–706
January 19–24, 2025.

Training set	Samples
Multi-domain data	13700
In-house telecom data	2179
In-house insurance data	9240

Table 1: Training datasets for baseline fine-tuning.

Test set	Samples
Multi-domain	3432
Seen domain, Unseen source	550
Unseen domain: Finance	2522

Table 2: Test datasets for baseline fine-tuning.

neck. Contrastive learning techniques (Wang et al., 2021; Zhang and Zhang, 2023), shared embedding spaces (Siddique et al., 2021; Shi et al., 2023) enable adaptability across new domains, but have their own challenges such as demanding significant computational resources and good embedding quality.

Addressing these issues calls for innovative solutions that can bridge the gap between zero-shot adaptability and practical deployment in real-world applications. Our contribution in this is two-fold:

- First, we propose a tailored data annotation strategy incorporating slot induction (Nguyen et al., 2023) followed by black-box knowledge distillation (KD) (Wang, 2021; Nguyen et al., 2022; Finch et al., 2024), that transfers knowledge from the teacher model without accessing its internal architecture or parameters. This fine-tuning approach results in a model that is both highly generalizable and robust to conversational data.
- Secondly, we demonstrate that the performance of a standard off-the-shelf extractive model commonly used in products can be significantly improved by integrating it as a preprocessing step alongside a fine-tuned model like ours and applying slot-specific constraints. This layered approach achieves near real-time performance with enhanced accuracy, while maintaining low latency and outperforming standalone extractive or LLM methods.

2 Approach

In this section we describe our framework for zero-shot slot filling. We developed a data annotation strategy based on black-box KD and slot induction and an architecture that leverages an extractive model to improve the accuracy and latency of the fine-tuned model. Using black-box KD, knowledge from a large foundation model, in our case, a commercial LLM with over 70 billion parameters, is transferred to a relatively small model through fine tuning using predictions from larger model with significantly reduced requirements for human annotation. In this approach, we follow a slot induction approach (instead of using predefined slot names), where we instructed the teacher model to predict all possible slot label-value pairs from the input text. This approach enhances the model's ability to generalize well across domains. We also used context along with the input text in the instruction to make the model context-aware. The training dataset comprises context and input text of varying lengths, which enables configuration flexibility for inference. The input to the system can be a single turn or multi turn. The following steps are required for this process:

1: Creation of an annotated dataset using a large commercial LLM with more than 70 billion parameters
2: Conversion of the annotated dataset into an instruction dataset
3: Instruction fine-tuning of a smaller model
4: Refining and aligning predictions with human annotations

3 Data Collection

3.1 Annotation of data

The raw dataset is a collection of call transcripts, capturing conversations between agents and customers, sourced from contact center interactions. A specific prompt for the LLM to facilitate the targeted "slot filling" task was developed (see Appendix 8.1). For slot induction, the LLM is instructed to discover novel slot labels with each annotation request. Iteratively, we obtain a growing list of slot labels as new ones are discovered. To refine/align these annotations with human annotations, we also employ consistent annotation guidelines across the same datasets for human annotators.

Our dataset comprises three diverse and distinct sources: 1) a balanced, multi-domain dataset sourced externally, encompassing five external domains of banking, insurance, telecommunications, retail and healthcare retail in multiple English accents; 2) an in-house dataset from the telecommunications domain; and 3) an in-house dataset from

the insurance domain. While the first dataset is externally sourced, the in-house datasets are collected and maintained internally by our organization.

All data used in our study has been anonymized, i.e., real identifiers and sensitive information were systematically replaced with fictitious equivalents to preserve confidentiality and comply with privacy standards, while maintaining the complexity of these dialogues. Appendix section 8.2 presents some examples of annotated data.

3.2 Creation of instruction fine-tuning dataset

After all the transcripts were annotated turn-by-turn by the LLM (teacher) and/or humans, with slot labels and their corresponding values, we created an instruction dataset that is used for fine-tuning a smaller student model. An instruction sample consists of two parts: 1. the instruction, comprising a brief description of the system, the task description, and the input as context or text to be used for slot filling, and 2. the response, comprising the slot labels and their corresponding values. See Appendix 8.3 for the template used for creating instruction samples.

To allow the system to be robust to any particular strategy of slot-filling, for every turn in a transcript, we randomize the number of turns in the "context" and "text" to create an instruction sample. "Context" here refers to the text in the transcript preceding the text from which the slots are to be extracted. This way, the model is able to generalize to different lengths of input context and text. We also randomize the type and the number of "distractor slot labels", which refer to slots that are not present in the input text but are used as distractors in the input query. This also serves as a data augmentation strategy for creating more training data, reflecting the same completion under different "context", "text" and "distractor slot labels". See Table 1 for an overview of training data where each sample is an instruction sample extracted from a turn in a transcript as explained above.

3.3 Test Data

To ensure our model's generalization across multiple domains without compromising training, we benchmark it using three distinct datasets. The first is a "multi-domain" dataset, which comprises transcripts from the same sources included in our training set. The second, labeled "seen domain, unseen source" belongs to the Telecommunications domain which is seen in training but originates from a different source. The third, labeled "unseen domain" belongs to the Finance domain, which is not represented in our training dataset. See Table 2 for an overview of the test data, where each sample is an instruction sample extracted from a turn in a transcript.

After baseline fine-tuning, we further introduce two new internal datasets from different domains (healthcare and financial services). Table 3 provides an overview of these datasets that are used in upcoming section 4.6 Model Generalization.

Training set	Samples
Healthcare domain	1846
Financial Services domain	2209

Test set	Samples
Healthcare domain	8832
Financial Services domain	11487

Table 3: Overview of additional training and test datasets for extended fine-tuning and testing.

4 Model Development and Evaluation

4.1 Fine-Tuning, Inference, Optimization

Fine-tuning was performed on NVIDIA A10G GPUs, and the software ecosystem was primarily based on the Huggingface framework. To optimize the fine-tuning process, we employed several advanced techniques and libraries, including PEFT (parameter-efficient fine-tuning), QLORA (quantized low-rank adaptation), Accelerate, and DeepSpeed. For optimization, we used the AdamW optimizer. We used F1 as our primary metric for model selection. After fine-tuning, we implemented an efficient inference pipeline for evaluation using the open source vLLM (Virtual Large Language Model) library designed for efficient inference of LLMs (Kwon et al., 2023). Prior to our primary experiments, we performed a series of preliminary tests that focused on optimizing a select set of critical hyper-parameters. See Appendix 8.4 for a detailed overview of the hyperparameters and configurations used at the fine-tuning and inference stages.

4.2 Evaluation

Given the generative nature of LLMs, the metrics based on "exact match" of ground truth to the model responses are inadequate. These metrics often penalize responses that are semantically correct

Model	Multi-domain			Seen domain, Unseen source			Unseen domain			Average
	P	R	F1	P	R	F1	P	R	F1	F1
Mistral v0.3	0.45	0.44	0.44	0.82	0.79	0.80	0.41	0.26	0.32	0.52
Llama 3 8B	0.46	0.49	0.47	0.80	0.77	0.78	0.43	0.24	0.31	0.52
Phi3-mini	0.45	0.49	0.47	0.82	0.73	0.77	0.30	0.22	0.25	0.50
Gemma 2B	0.43	0.28	0.34	0.71	0.61	0.66	0.28	0.14	0.19	0.40

Table 4: Baseline performance (Precision, Recall, F1) for pretrained models over three datasets and their average

but differ syntactically or lexically. To address this, we adopt a more flexible evaluation strategy using lenient matching, i.e., if the system response is partially correct or incomplete, it receives a credit. Despite this, in cases where the model responses are in normalized form (e.g., for dates or emails), we do not obtain a "lenient match". To address this, we applied inverse text normalization (ITN) to both the ground truths and responses before evaluation.

All metrics reported in this paper—lenient precision, recall, and F1 scores—are calculated after applying ITN. Henceforth, "F1" refers to this modified metric. We elaborate on the lenient metric with a few examples in Appendix 8.5.

4.3 Baseline Model Performances

In this section, we evaluate multiple foundational LLMs and report their performance without applying fine-tuning. Due to computational constraints, we prioritized model selection based on two key criteria: the model's ability to follow instructions for structured output, and its baseline performance.

Results in Table 4 show that Mistral v03 and Llama 3 achieved identical average F1 scores, while Phi3-mini demonstrated competitive performance despite its smaller size. By contrast, the Gemma 2B model underperformed compared to the other LLMs. We observed that Mistral occasionally failed to generate valid JSON outputs, which occurred 2-3 times more frequently than Llama 3. Although Phi3 showed promising results, its primary focus on English and limited multilingual capabilities made it less suitable for our planned language expansions. Consequently, we selected Llama 3 8B model as the base of our subsequent experiments.

4.4 Fine-Tuned Model Performances

Our initial fine-tuning experiment utilized the training sets from three internal datasets to assess the improvements over baseline models. Table 5 shows the substantial performance gains achieved through fine-tuning. These results indicate that smaller LLMs, when used out-of-the-box, are inadequate for slot filling tasks that demand extensive world knowledge and robust language understanding for generating structured outputs consistently. The significant performance boost underscores the critical importance of fine-tuning for specific domains. The mediocre performance of pretrained foundational models was notably enhanced by 25% absolute to achieve acceptable results post fine-tuning. This relative difference between a mediocre performance of a "generalist" model and stellar performance of a "domain/task expert" model is achieved by fine-tuning.

Model	**Dataset**	**F1**
Baseline	Multi-domain	0.47
	Seen domain, Unseen source	0.78
	Unseen domain	0.31
	Average	0.52
Fine-tuned	Multi-domain	0.61
	Seen domain, Unseen source	0.92
	Unseen domain	0.78
	Average	0.77

Table 5: Comparison of F1 scores between baseline and fine-tuned versions of the Llama 3 8B LLM

4.5 Model Alignment

Human annotated data can often serve as high-quality reference for fine-tuning NLP models, allowing models to learn from and align with human behavior. Additionally, human annotated data can be used to identify and reduce potential inconsistencies and noise in the training data and models' outputs, to improve accuracy and consistency. To evaluate the impact of human annotations, we perform fine-tuning with only human annotations, and then also with both LLM-generated and human-labeled data. Table 6 presents the results, showing the average F1 score improvement across the three test datasets previously considered.

The baseline model trained solely on human annotations significantly under-performed compared to the model trained exclusively on LLM annotations. This is in line with the empirical analysis we conducted on human annotations compared to LLM-annotations over a subset of 20 transcripts.

Fine-tuning Strategy	Avg F1
Human Annotations Only	0.74
LLM Annotations Only	0.77
Both Annotations	0.78

Table 6: Comparison of fine-tuning results with and without human annotations

Experiment	Avg. F1
Baseline	0.73
Fine-tuning (+ new datasets)	0.74

Table 7: Generalization Experiment Result

Humans are good with named entities and often create more specific labels, like breaking down "monthly insurance cost" into "Old Monthly Cost" and "New Monthly Cost". However, they miss out on abstractive slots like reason for call, product mentions, survey participation, etc. We quantified the number of missed labels—those not identified or annotated in the transcript—and found that human annotators tend to miss twice as many labels compared to LLMs. Specifically, human annotators missed an average of 8 labels in a transcript, whereas LLMs only missed 4. This performance gap highlights the broader knowledge transfer achieved by LLMs compared to human annotations.

Combining both annotations resulted in a modest 1% absolute improvement, which we term as "human alignment". The results suggest that the benefit of fine-tuning on human plus machine-generated data is due the volume of added data, not higher quality of the labels. These findings imply the costly and time-consuming human annotation and alignment processes could potentially be bypassed, with only a modest sacrifice in accuracy.

4.6 Model Generalization

We evaluated the model's performance on additional datasets summarized in Table 3 without further fine-tuning, establishing a baseline. The difference between this baseline and the performance after fine-tuning with these datasets demonstrated the "generalization gap".

Table 7 summarizes the result using F1 scores averaged across our five test sets. The final result demonstrates the model's generalization capability. The increase in average F1 score from 0.73 to 0.74 indicates a marginal generalization gap, demonstrating strong generalization capability of the model on other domains.

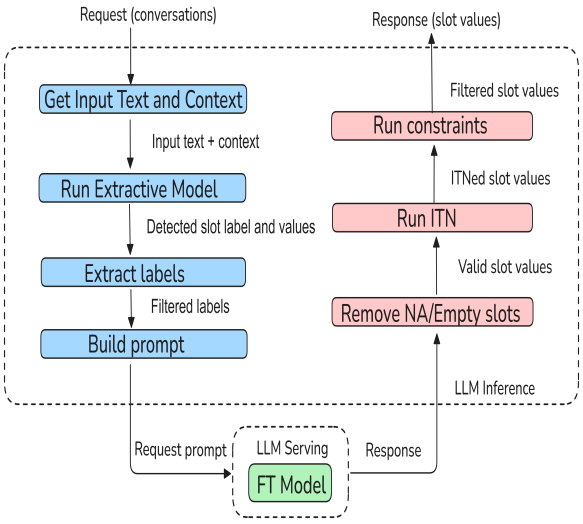

Figure 1: System Architecture: The LLM Inference module handles both preprocessing and postprocessing tasks. The fine-tuned (FT) LLM model is served by LLM Serving module.

5 System Architecture

Figure 1 describes our system architecture. The two main components in our architecture are LL-inference and LL-serving. LLM-inference handles the pre/post processing of the data, while LLM-serving serves the (LLM) model. Once the system receives a conversation during call processing, the preprocessing module selects the input text and context. This text is passed to an extractive model, specifically GLiNER (Zaratiana et al., 2024), which extracts slot values. GLiNER excels in recall for extracting slot values but lacks precision and cannot extract abstractive slots (e.g., Call Reason or Claim Issue). However, GLiNER can identify whether the conversation pertains to these labels. Slot values extracted by the lightweight model (GLiNER) are used to narrow down the list of slots requested from the LLM.

A prompt is constructed containing the input text, context, and the pre-determined, reduced set of slot labels by GLiNER. This prompt is sent to the fine-tuned LLM, and the LLM's response is processed to eliminate any empty or NA responses. Subsequently, ITN is applied to the results to filter out false positives and improve precision using constraints. A constraint is a predefined rule or user-defined parameter that is designed to minimize erroneous value extractions by zero-shot models. Predefined constraints are set for each entity

Method	Gain %
Only GLiNER	2%
GLiNER + Constraints	16%
GLiNER + ft-llm + Constraints	34%

Table 8: Comparison of different pipelines of our system over the legacy product baseline on unseen data.

type, while user-defined constraints apply to individual slots. Predefined constraints include matching entity types like date, duration, cardinal, money, email and standard named entities, whereas user-defined constraints, like lengths of values, can be applied to the numerical and alphanumeric entities (e.g., Dosage slot should have "partial cardinal" as a constraint).

Table 8 compares the percentage gains in F1 scores compared to current product's baseline, which is a legacy extractive model. This table shows the incremental improvements achieved at each stage of system enhancement. Initially, off-the-shelf GLiNER model delivers a modest 2 % increase in the F1 score. However, with the introduction of a constraints stage, this improvement becomes more pronounced, boosting the score by 16 %. Notably, integrating GLiNER as a preprocessing step before the fine-tuned large language model (ft-LLM) and applying constraints results in a substantial 34% improvement in the F1 score.

More details on the system performance are provided in section 8.9. This staged approach demonstrates how integrating the extractive model as a preprocessing module before using the fine-tuned LLM, and applying constraints as postprocessing can lead to substantial improvements in both accuracy and reliability for complex industrial applications.

6 Limitations and Future Work

6.1 Expansion to other languages

There are ongoing experiments to expand our approach to multiple languages including Spanish, Hindi, French, German and Arabic. Spanish and Hindi experiments indicate that the performances of vanilla Llama3 8b model trained with each language separately are comparable to those of a single fine-tuned model trained with all languages. Thus, we illustrate the possibility of a single multilingual model capable of slot filling in multiple languages.

6.2 Exploration of smaller models

Initial experiments involving much smaller models like Phi3-mini have demonstrated that there is room for improvement by training smaller and compute-efficient models that exhibit enhanced capabilities. These advancements not only hold the promise to support a wider array of languages but also enable faster inference and reduced computational cost.

6.3 More robust evaluation metrics

We discussed how the current evaluation metrics do not fully capture semantics and are not correlated with our small-scale human evaluations, underestimating model performance. Future work could benefit from adopting more form and content aware evaluation metrics. Ongoing work considers 1) weighted average of lenient F1 scores, ROUGE and BERTScores (Zhang et al., 2020), and 2) slot-type specific evaluation using relevant metrics, such as ROUGE for form-insensitive slots and BERTScore for semantic slots.

7 Conclusion

In this paper, we have proposed a comprehensive, practical, and scalable approach for high-performance zero-shot slot filling using black-box knowledge distillation for conversational data. Through comprehensive experimentation, we demonstrated the effectiveness of using a larger LLM (teacher model) for creating data and transferring knowledge (through fine-tuning) to smaller LLMs (student models). In addition, we showed that fine-tuning significantly improved domain-specific performance, with the Llama 3 8B model outperforming the other foundation models we explored, achieving a 26% absolute improvement in F1 over its vanilla version. We also demonstrated a flexible and scalable deployment architecture supporting multiple use cases, including agent assistance, automated self-service, and post-call analytics. We used preprocessing with off-the-shelf GliNER model and postprocessing with slot constraints to improve the baseline system performance by 34% relative F1. We highlighted future work for language expansion, the use of more efficient smaller models and developing human-correlated metrics to better assess real-world performance. This work contributes to the growing body of research on practical applications of LLMs in customer service domains and provides a practical foundation for future developments in this field.

Acknowledgements

We sincerely thank Andreas Stolcke, Roberto Pieraccini, Aravind Ganapathiraju, Malolan Chetlur and Neha Gupta for their valuable discussions that greatly influenced this work. We thank the product team at Uniphore for their product related insights. We also express our appreciation to Manickavela Aruguman for throughput and latency experiments.

References

Ankur Bapna, Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *Proceedings of INTERSPEECH*, pages 2476–2480.

Xinya Du, Luheng He, Qi Li, Dian Yu, Panupong Pasupat, and Yuan Zhang. 2021. Qa-driven zero-shot slot filling with weak supervision pretraining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 2, pages 654–664.

James D. Finch, Boxin Zhao, and Jinho D Choi. 2024. Transforming slot schema induction with generative dialogue state inference. In *Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 317–324.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021. Robust retrieval augmented generation for zero-shot slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1939–1949.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Xuefeng Li, Liwen Wang, Guanting Dong, Keqing He, Jinzheng Zhao, Hao Lei, Jiachi Liu, and Weiran Xu. 2023. Generative zero-shot prompt learning for cross-domain slot filling with inverse prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 825–834.

Qiaoyang Luo and Lingqiao Liu. 2023. Zero-shot slot filling with slot-prefix prompting and attention relationship descriptor. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, volume 37(11), pages 13344–13352.

Shikib Mehri and Maxine Eskenazi. 2021. Gensf: Simultaneous adaptation of generative pre-trained models and slot filling. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 489–498.

Dang Nguyen, Sunil Gupta, Kien Do, and Svetha Venkatesh. 2022. Black-box few-shot knowledge distillation. In *Proceedings of the European Conference on Computer Vision*, pages 196–211.

Hoang Nguyen, Chenwei Zhang, Ye Liu, and Philip Yu. 2023. Slot induction via pre-trained language model probing and multi-level contrastive learning. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 470–481.

Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 1410–1418.

Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490.

Yuanjun Shi, Linzhi Wu, and Minglai Shao. 2023. Adaptive end-to-end metric learning for zero-shot cross-domain slot filling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6291–6301.

A. B. Siddique, Fuad Jamour, and Vagelis Hristidis. 2021. Linguistically-enriched and context-aware zero-shot slot filling. In *Proceedings of the Web Conference 2021 (WWW '21)*, pages 3279–3290.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, volume arXiv preprint arXiv:2302.13971.

Liwen Wang, Xuefeng Li, Jiachi Liu, Keqing He, Yuanmeng Yan, and Weiran Xu. 2021. Bridge to target domain by prototypical contrastive learning and label confusion: Re-explore zero-shot learning for slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9474–9480.

Zi Wang. 2021. Zero-shot knowledge distillation from a decision-based black-box model. In *Proceedings of the 38th International Conference on Machine Learning, PMLR*, volume 139, pages 10675–10685.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. Gliner: Generalist model for

named entity recognition using bidirectional transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 5364–5376.

Junwen Zhang and Yin Zhang. 2023. Hierarchical contrast: A coarse-to-fine contrastive learning framework for cross-domain zero-shot slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14483–14503.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of the International Conference on Learning Representations (ICLR)*. ArXiv:1904.09675.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2993–2999.

8 Appendix

8.1 Annotation Prompt

Following is the prompt template with variables "labels" and "text", which are filled with a seed list of slot names (that reflects our knowledge/experience of slot labels across several domains) and the call transcripts, respectively:

```
You are an expert in Natural
    Language Processing.

Your task is to identify all
    named slot values in the
    given dialogue text, in
    which agent turn starts with
     ''Agent says:'' and
    customer turn starts with ''
    Customer says:''.

Return the output in a json
    format for every line in the
     dialogue where key is text
    and value is dict of slot
    types and values. If there
    are no slot types in the
    line, return NA.

To get started, here is the list
     of slot types available to
    you: {labels}.

Do not be restricted by this
    list. You should also
    extract slot types that are
    not in this list but present
     in the text.

Dialogue Text: {text}
```

8.2 Data Annotation Examples

Following are some examples of labeled utterances derived from our dataset. These examples illustrate typical interactions in a call center setting, along with the corresponding labeled information (slots) extracted from the dialogue.

```
Example 1:
Text: "Thank you for calling Net Company.
       How can I assist you today?"
Slots: {"Company Name": "Net Company"}


Example 2:
Text: "Yes, uh I'm John Doe, and the
       account number is 123456.
       My wifi doesn't work."
Slots: {"Customer Name": "John Doe",
       "Account Number": "123456",
       "Reason for call": "wifi
        doesn't work"}
```

8.3 Fine-tuning Prompt

Following is the template for creating training samples for the instruction fine-tuning task:

```
<s>[INST]<<SYS>>
You are an honest and helpful
    information extractor.
<</SYS>>
Your task is to extract values
    for the following slot
    labels in the Main Text
    delimited by triple
    backticks: {target slot
    labels}, {distractor slot
    labels}. Format your
    response as a JSON object
    with slot labels as the keys
     and slot values in a list.
    Only return the slots found
    the Main text. Use the
    following dialogue only as
    context support to extract
    slots from the Main text
    delimited by triple
    backticks:
{context text}
```
```
Main text: {text}
```
```
[/INST] {completion text}
```

The variables in this template are:

domain	domain of the call transcript
target slot labels	labels in the text of interest
distractor slot labels	labels that don't exist in text
context text	the textual information that precedes the text
text	the textual information to be processed for slot filling of target slot labels
completion text	the response part of the prompt that the model will be trained to generate given all the information between the tags [INST] and [/INST]

8.4 Fine-tuning and Inference Setups

The table presents a summary of the hyperparameters and configuration settings used in our fine-tuning steps, including hardware specifications, LORA settings, optimization parameters, and training details. Prior to our main experiments, we performed a series of preliminary experiments for optimizing a select set of critical hyperparameters. Specifically, we examined the effects of varying the number of training epochs, the size of the training dataset, the rank of the LORA (Low-Rank Adaptation) matrices, and the neural network layers subjected to fine-tuning. After the fine-tuning process, we implemented an efficient inference pipeline to evaluate our fine-tuned models using compute efficient vLLM inference engine. We have used greedy search with temperature 0. The maximum number of new tokens was set to 512. This temperature setting without sampling is particularly useful when we want consistent, high-confidence responses from the model. The choice of 512 tokens allows for reasonably lengthy responses while preventing excessively long generations.

Parameter	Value
GPUs	4
GPU Memory	24GB per GPU
LORA Rank	16
LORA Alpha	32
Dropout Rate	0.05
Batch Size per GPU	1
Gradient Accumulation Steps	4
Effective Batch Size	16
Maximum Learning Rate	2e-4
Number of Epochs	5
Warm-up	10% of iterations
AdamW beta1	0.9
AdamW beta2	0.999
AdamW epsilon	1e-8
Weight Decay	Not applied
Gradient Clipping Threshold	1.0
Adaptation Layers	All linear layers

Table 9: Fine-tuning Hyperparameters and Configuration

8.5 Lenient Metric Examples

In this section, we present some examples of lenient metrics for slot matching in dialogue systems. While traditional extractive systems select spans from user utterances, modern generative systems may rephrase or reformulate the extracted information, making lenient matching particularly crucial. The following examples demonstrate how generative systems could produce variations of the same semantic content, requiring more flexible evaluation metrics compared to exact span matching used in extractive approaches. Lenient matching allows for partial matches and semantic equivalence, providing a more realistic evaluation of slot filling systems compared to strict matching. Consider the following reference slot values:

```
Reference: {
  "time": "7:00 PM",
  "people": "2 people",
  "restaurant": "Joe's Pizza & Italian
                 Restaurant"
}
```

In the following, we analyze two different predictions under both strict and lenient matching criteria:

```
Prediction 1: {
  "time": "7 PM",
  "people": "two",
  "restaurant": "joes pizza"
}
Prediction 2: {
  "time": "19:00",
  "people": "couple",
 "restaurant": "Joe's Italian Restaurant"
}
```

Under strict matching criteria:

- Prediction 1: 0/3 correct (no exact matches)
- Prediction 2: 0/3 correct (no exact matches)

Under lenient matching criteria:

- Prediction 1: 3/3 correct
  - "7 PM" matches "7:00 PM" (time format variation)

  - "two" matches "2 people" (numerical equivalence)
  - "joes pizza" matches "Joe's Pizza & Italian Restaurant" (partial name match, missing punctuation)

- Prediction 2: 3/3 correct
  - "19:00" matches "7:00 PM" (time format equivalence)
  - "couple" matches "2 people" (semantic equivalence)
  - "Joe's Italian Restaurant" partially matches "Joe's Pizza & Italian Restaurant" (partial name match)

The lenient matching implementation involves:

1. **Time normalization:** Converting different time formats to a standard representation
2. **Numerical equivalence:** Matching different representations of numbers (words, digits)
3. **Name normalization:**
   - Handling missing punctuation (Joe's vs Joes)
   - Handling partial matches (subset of full name)
   - Handling special characters (& vs and)
   - Case-insensitive matching
4. **Semantic match:** Using word embeddings or knowledge bases for semantic equivalence

It should be noted that in all our metrics presented in this paper we have not employed semantic similarity. However, in our limited internal experimentation, we have seen that the scores are further elevated with semantic similarity reflecting anecdotal human judgments better.

8.6 GLiNER

GLiNER is a compact NER model designed to efficiently extract various types of entities from text. Unlike larger autoregressive models, GLiNER uses a bidirectional language model to process text and extract entities. It uses bidirectional transformer encoder to perform parallel entity extraction, making it more efficient than sequential models. The general approach is to place both span and entity embedding in the same latent space and then assess their compatibility, enabling accurate identification of entities. It outperforms in zero-shot NER scenarios on multiple NER benchmarks. It is more efficient, scalable and versatile approach to NER.

8.7 System Framework

In our architecture, we have designed two key services to efficiently handle the zero-shot slot-filling system using an LLM for call center tasks. These services operate as:

1. **LLM-Inference Service**: This service is responsible for data preprocessing and postprocessing. It ensures that input data is properly formatted and contextualized before being passed to the LLM. Additionally, it manages GLiNER model integration for extractive tasks. GLiNER is utilized here to extract relevant slot values from the conversation, narrowing down the slots for the LLM to process.
2. **LLM-Serving Service**: This service focuses on serving the LLM model itself. It directly handles the inference requests and provides outputs based on the preprocessed data from the LLM-Inference service.

8.8 System Deployment

Both services (LLM-inference and LLM-serving) are deployed as Docker containers on **AWS**, taking full advantage of cloud-based infrastructure for scalability, reliability, and flexibility. The model serving is built using a combination of **Transformers** and **Seldon Core** libraries, ensuring robust performance and flexibility for various LLM use cases.

8.9 System Performance

To optimize the system for both **latency** and **throughput**, we benchmarked it across different configurations, including FP16 precision, GPTQ-4bit, and GPTQ-8bit. Among these, the **FP16 model** with **prefix caching** demonstrated the best trade-off between performance and resource utilization. The use of prefix caching significantly improved both concurrency and throughput. By caching common portions of input sequences, redundant computations during inference can be reduced, leading to a notable reduction in processing times. With the **FP16 + prefix caching** setup, we achieved a concurrency level of **100 requests** while maintaining an average latency of **3.8 seconds** on an **NVIDIA L40 GPU**. This level of performance ensures that the system can handle near real-time, high-volume call center conversations with minimal delay, improving the overall user experience.