Speed Without Sacrifice: Fine-Tuning Language Models with Medusa and Knowledge Distillation in Travel Applications

Daniel Zagyva[2], Emmanouil Stergiadis[1], Laurens van der Maas[2], Aleksandra Dokic[2]
Eran Fainman[1], Ilya Gusev[1], Moran Beladev[1],
[1]Booking.com [2]Amazon AWS Professional Services
zagyvad@amazon.com, emmanouil.stergiadis@booking.com, laurensv@amazon.com, dokica@amazon.com, eran.fainman@booking.com, ilya.gusev@booking.com, moran.beladev@booking.com

Abstract

In high-stakes industrial NLP applications, balancing generation quality with speed and efficiency presents significant challenges. We address them by investigating two complementary optimization approaches: Medusa for speculative decoding and knowledge distillation (KD) for model compression. We demonstrate the practical application of these techniques in real-world travel domain tasks, including trip planning, smart filters, and generating accommodation descriptions. We introduce modifications to the Medusa implementation, starting with base pre-trained models rather than conversational fine-tuned ones, and developing a simplified single-stage training process for Medusa-2 that maintains performance while reducing computational requirements. Lastly, we present a novel framework that combines Medusa with KD, achieving compounded benefits in both model size and inference speed. Our experiments with TinyLlama-1.1B as the student model and Llama-3.1-70B as the teacher show that the combined approach maintains the teacher's performance quality while reducing inference latency by 10-20x.

1 Introduction

Rapid growth of digital applications has intensified the demand for real-time natural language processing (NLP) capabilities. Although recent large language models (LLMs) have achieved remarkable generation quality through billion-scale parameters (Chowdhery et al., 2022; Zhang et al., 2022; Hoffmann et al., 2022; OpenAI, 2023; Google, 2023; Llama team, 2024), their increased inference latency poses significant challenges for production deployment. Studies have shown that even slight increases in latency (100-400 ms) can measurably decrease user engagement (Brutlag, 2009). Combined with the high computational costs of large models, these factors emphasize the need to optimize both speed and efficiency for practical NLP deployment in time-sensitive applications.

This paper explores two complementary approaches: the Medusa framework (Cai et al., 2024), a novel approach for speculative decoding, and KD (Hinton et al., 2015) for model compression. While Medusa accelerates inference without modifying the original model, KD creates smaller, efficient models that maintain performance. We integrate these techniques to improve both the speed and efficiency of NLP systems in travel applications.

Our study makes three key contributions: First, we analyze the implementation of Medusa and KD techniques in real-world NLP tasks. Second, we present a modified Medusa implementation that begins with base pre-trained models and introduces a simplified single-stage training process for Medusa-2. Third, we demonstrate the complementary benefits of Medusa with KD for performance and speed. In addition, we provide practical insights and best practices for production deployment.

2 Real-World Applications

The travel domain offers numerous applications that can benefit from fine-tuned LLMs. At Booking.com, a leading online travel agency, we fine-tune and deploy LLMs to improve various aspects of the user experience. In this section, we describe three such applications. Each application goes through the process of online A/B testing that is conducted on real production traffic over multiple weeks to measure its effectiveness.

2.1 AI Trip Planner

The *AI Trip Planner* (AITP) is a conversational travel assistant that transforms trip planning by integrating LLMs with internal recommendation systems. As illustrated in Figure 1a, this chatbot provides personalized hotel and destination recommendations by extracting structured travel features from user interactions.

To enable seamless integration with our internal recommendation models, we employ the *JSON Travel Entity Extraction* model, which extracts key travel parameters from user conversations. An example of a conversation and its extracted travel entities is provided in Appendix A.

Since this use case requires real-time responses, low-latency inference is critical. Deploying our in-house fine-tuned distilled LLM with Medusa acceleration allowed us to significantly decrease latency. The A/B tests against OpenAI GPT-3.5 showed a **+2.9%** increase in clicks on the recommendation cards, indicating an improved precision of feature extraction and retrieval.

2.2 Smart Filters

Our *Smart Filters* feature empowers users to refine searches through natural language queries, allowing more flexible and personalized searches. Users enter free-text queries, and we employ the *JSON Travel Entity Extraction* model that extracts structured entities to apply relevant filters. Figure 1b illustrates this process.

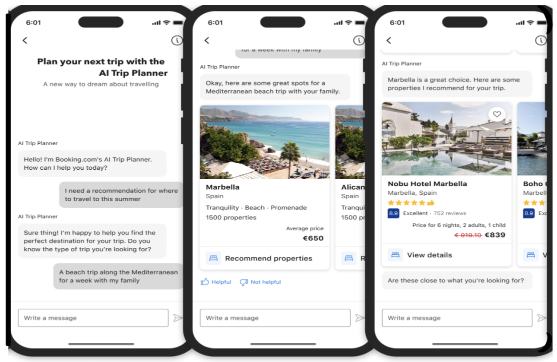
This feature enhances search results by enabling fast query resolution, which is crucial to user experience. The deployment of our distilled model with Medusa heads achieved 15x faster response times in terms of $p99$, increasing scalability.

2.3 Accommodation-Level Description Generation

Traditionally, accommodation descriptions are generated using structured templates based on accommodation attributes (see Appendix B.1 for an example). Although templates ensure consistency, they come with several challenges: maintaining them is complex, especially with multiple templates across accommodation segments and evolving business rules. They can also be repetitive for users, potentially lowering engagement, and limit the integration of unstructured data, such as free text inputs from accommodation owners or user-generated content for personalization.

To overcome these limitations, we introduce a generative AI-based approach capable of dynamically tailoring descriptions based on the existing set of accommodation attributes provided by the partners. An example screenshot is provided in Appendix B.2. The A/B testing of the generative descriptions against template-based versions demonstrates a **+1.4%** increase in helpful votes, validating the improved user experience and relevance. Given the need to process descriptions for 3 million accommodations and update them as properties change, computational efficiency is crucial. Optimized inference enables large-scale generation within reasonable time as detailed in Section 7.

(a)



Hello! I'm Booking.com's AI Trip Planner. How can I help you today?

I need a recommendation for where to travel to this summer

Sure thing! I'm happy to help you find the perfect destination for your trip. Do you know the type of trip you're looking for?

A beach trip along the Mediterranean for a week with my family

Okay, here are some great spots for a Mediterranean beach trip with your family.

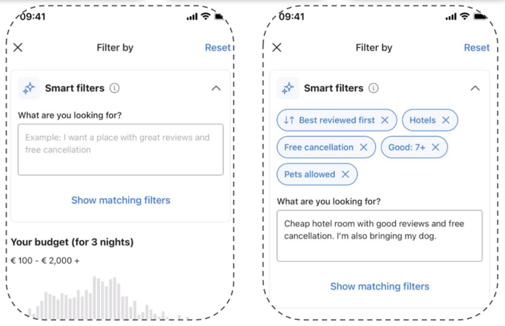
Marbella, Spain, Tranquility · Beach · Promenade, 1500 properties, Average price €650

Marbella is a great choice. Here are some properties I recommend for your trip.

Nobu Hotel Marbella, Marbella, Spain, 8.9 Excellent · 752 reviews, Price for 6 nights, 2 adults, 1 child ~~€ 919.10~~ €839

Are these close to what you're looking for?

(b)



Example: I want a place with great reviews and free cancellation

Cheap hotel room with good reviews and free cancellation. I'm also bringing my dog.

Figure 1: (a) The *AI Trip Planner* providing personalized hotel and destination recommendations by extracting structured features. (b) The *Smart Filters* application with user input and the extracted structured filters.

3 Related Work

3.1 Speculative Decoding

Speculative decoding has emerged as a promising approach to address inference bottlenecks in autoregressive LLMs. The core principle is to predict multiple tokens in parallel followed by verification, transforming sequential operations into more hardware-efficient batched computations (Leviathan et al., 2023). This approach maintains the original model's output distribution while providing significant speedups, typically 2-4x, without compromising generation quality.

Medusa (Cai et al., 2024) represents a significant advance in speculative decoding by eliminating the need for separate draft models. Instead, Medusa augments the base LLM with additional lightweight prediction heads that forecast tokens at specific future positions. The architecture employs a tree-structured attention mechanism that ensures that tokens only attend to their predecessors in the same continuation path, maintaining the autoregressive property while enabling efficient parallel processing. Medusa offers two training strategies: Medusa-1, which fine-tunes only the additional heads while keeping the backbone frozen, and Medusa-2, which jointly trains both components for a larger speedup.

Recent improvements in the Medusa framework include Hydra (Xia et al., 2024), which improves speculation accuracy by making the draft heads sequentially dependent rather than independent. Other active research directions include adaptive speculation that dynamically adjusts the number of speculative tokens based on context, multi-modal speculation extensions, and hardware-specific optimizations (Miao et al., 2023).

3.2 Knowledge Distillation

KD addresses the deployment challenges of large, parameter-heavy models by transferring knowledge from larger "teacher" models to smaller "student" models (Hinton et al., 2015). This approach enables significant model compression while preserving much of the original performance. Sequence KD (SeqKD) (Kim and Rush, 2016) represents a specialized form of KD designed for sequence generation tasks. Unlike traditional word-level distillation that matches probability distributions at each position, SeqKD focuses on transferring knowledge at the sequence level. The process involves three main steps: (1) training a large teacher model, (2) generating new training data using the teacher's highest-scoring outputs, and (3) training the smaller student on this teacher-generated dataset. This approach allows student models to achieve performance comparable to that of teacher models.

4 Approach

4.1 Single-stage Medusa

Our methodology differs from the original Medusa study in two aspects. While the original work used models fine-tuned on conversations (Vicuna, trained on ShareGPT), we start with base pre-trained models. Furthermore, rather than targeting general conversation, our implementation focuses on specific travel domain tasks.

Unlike the original Medusa paper that trained Medusa heads directly on an already fine-tuned model, our Medusa-1 implementation follows a two-stage process: first fine-tuning the base model for our tasks, and then training the Medusa heads while keeping the fine-tuned model frozen. By keeping the fine-tuned model frozen during the second stage, this approach achieves lossless acceleration.

For Medusa-2, we implement a single-stage approach that contrasts with the two-stage methodology presented in the original Medusa paper. In the original work, the Medusa heads required more extensive training than the already fine-tuned base model. This discrepancy led to larger gradients from the Medusa heads, distorting the base model's parameters. To mitigate this issue, they employed a two-stage process: first training Medusa heads only, and then jointly training both the base model and Medusa heads with a warm-up strategy.

Our implementation demonstrates that a single-stage process suffices when starting from a base pre-trained model. Specifically, we attach the Medusa heads to the pre-trained model and fine-tune the entire architecture end-to-end in a single training phase. As both our base model and Medusa heads begin at a similar level of task-specific training, we hypothesize that the gradient disparities would be less pronounced. This single-stage method achieves comparable performance and maintains Medusa's latency reduction benefits while streamlining implementation and reducing computational requirements.

4.2 SeqKD

For the entity extraction task, we use a traditional SeqKD approach and incorporate both labeled and unlabeled data in the student training process. We first fine-tune the teacher model on a human-labeled dataset, then generate predictions on additional unlabeled data. The final student training dataset is created by combining these teacher-generated samples with the human-labeled dataset, allowing the student to learn from both expert-curated and teacher-generated examples.

For the accommodation description generation task, we use GPT-4 (OpenAI, 2023) as a teacher model to generate training samples from unlabeled data, which serves as our sole training dataset.

4.3 SeqKD with Medusa

While speculative decoding primarily targets inference speed, and KD focuses on model size reduction, these approaches can be complementary rather than mutually exclusive. Combining these techniques offers potential for compounded benefits: smaller distilled models with accelerated inference.

We present a unified framework that integrates several efficiency techniques to produce compact, high-performance language models with reduced latency. The framework enables efficient implementation of both techniques in a streamlined process, making it practical for deployment in production environments.

The proposed integration of these techniques is the following three-stage pipeline that starts with pre-trained base models. The two-step student's training dataset generation follows the SeqKD approach described previously, after which the student model undergoes a single training phase using the Medusa-2 architecture. Although Medusa-1 can be used, it requires an additional training step and offers no significant advantages over Medusa-2, which we recommend for its simplicity of implementation and superior speedup ratios.

5 Experimentation

5.1 Experimental Setup

Our experiments encompasses three investigation paths: Medusa acceleration techniques, KD methods and their combination. Each training experiment uses full model fine-tuning, as opposed to parameter-efficient methods.

In our implementation of the Medusa framework, we build five Medusa heads, with each head consisting of a single ResNet layer. For all predictions and evaluations, we employ greedy decoding, which leads to a deterministic acceptance scheme, where candidates are accepted only if the base model generates the same sequence.

In the extraction task, we additionally experiment with SeqKD. These experiments use a fine-tuned Llama-3.1-70B as the teacher model, generating 17,000 pseudolabels on an unseen dataset. This dataset is derived from Booking.com production environment and covers various query patterns to ensure robust generalization. The entire data generation process by doing inference on the teacher model takes approximately 10 hours. We then combine these pseudo-labeled examples with the original 9,000 observation training set. We explore different mix ratios of the datasets and achieve the best result by up-sampling the original 9,000 observations until it matches the generated one in size, leading to 34,000 samples of which 50% are annotated by humans and 50% by the teacher model. This dataset is then used to fully fine-tune a TinyLlama-1.1B student, a process that takes two hours.

5.2 Tasks and Datasets

Our experimental setup includes a dataset for each real-world application in Section 2 and include: User preference extraction across **(1.1)** dialog and **(1.2)** query inputs and **(2)** Accommodation Description Generation.

This set covers a wide range of use cases often encountered in industrial settings: structured (1.1 and 1.2) versus natural language (2) outputs; multi-turn dialog (1.1) versus single-turn inputs (1.2 and 2); and short versus long input/outputs that may require truncation.

For entity extraction, we use human annotators to extract up to 35 different fields from AI Trip Planner dialogues and search queries, see Table 1 for exact sizes. The full list of extracted fields is shown in Appendix C. For description generation, we prompt the GPT-4 (teacher) model with information on 10,000 accommodations, and a set of instructions is provided by the content experts team in our organization. The content experts team edits another small set of 273 GPT-4 outputs to meet all guidelines. The cost of GPT-4 generation is $354.68.

Metric	AITP	Smart Filters	Desc Gen
Size Train	5,562	4,230	9,500
Size Dev	310	300	500
Size Test	310	300	273
Input Mean	222	29	1,100
Input Max	6,955	87	2,795
Input Min	21	21	222
Output Mean	66	30	1,339
Output Max	194	185	2,051
Output Min	9	2	774

Table 1: Statistics for the datasets used in entity extraction. During training and inference we truncate inputs to the last 1024 tokens when sequences exceed this length.

5.3 Evaluation Metrics

Entity Extraction Our main performance metrics are precision and recall, and we use their harmonic mean (F1 score) aggregated among topics. We use micro-averaging to address class imbalance as certain topics are very rare.

In addition to performance, the main metric we wish to improve is latency. We report the median (p50) and the 99th percentile (p99) measured using a TGI (Text Generation Inference, 2023) container at moderate load (1 RPS). Since we are interested in real-time use cases, we do not use batching (each request consists of a single query).

Accommodation Description Generation We report ROUGE metrics (Lin, 2004) and BERTScore (Zhang et al., 2019). For BERTScore, we use DEBERTA-XLARGE-MNLI (He et al., 2020) as the backbone model [1], which currently shows the strongest correlation to human judgment (BERTScore, 2023). We report the F1 score without the TF-IDF weighting.

Another important metric is the cost per million input and output tokens. For the GPT-4 model we report the official cost mentioned by OpenAI [2]. For our fine-tuned models, we report the estimated hardware cost measured using a TGI container at moderate load (1 RPS).

5.4 Hardware Requirements

We use Amazon SageMaker AI g5.2xlarge instances with NVIDIA A10G GPUs (24GB GPU memory) for TinyLlama-1.1B full fine-tuning, which takes 2-3 hours. Llama-3.1-70B full fine-tuning requires 2 p4d.24xlarge instances with 16 NVIDIA A100 GPUs (640GB total GPU memory) using DeepSpeed ZeRO Stage 3 with CPU offload (DeepSpeed team, 2021), completing in 7-8 hours. The Medusa experiments are conducted on the same fine-tuning infrastructure, with five Medusa heads (each a feed-forward layer with residual connection) adding approximately 750 million parameters ($5 \times (d \cdot V + d^2)$, where $d = 4096$ is the hidden dimension and $V = 32000$ is the vocabulary size) without requiring additional GPU resources. The generation of KD data using the fine-tuned Llama-3.1-70B model is performed on g5.48xlarge instances equipped with 8 NVIDIA A10G GPUs (192GB total GPU memory).

6 Experiment Results and Analysis

6.1 Entity Extraction

Table 2 presents the entity extraction results for the dialog and search query distributions using the TinyLlama-1.1B model. We also present GPT-4o and GPT-4o-mini as proprietary model baselines, evaluated both in a zero and 3-shot setting.

We observe that trained models perform well, significantly exceeding even the few shot GPT-4o baseline, which confirms that our training pipeline is effective and the backbone model has sufficient capacity despite its relatively small size for LLM standards. We then implement both Medusa variants (Medusa-1 and Medusa-2) and achieve significant improvements. Medusa-1's deterministic acceptance mechanism maintains performance metrics (within floating-point precision) while reducing latency by factors of 2.0x and 3.6x for search and dialog tasks, respectively. Medusa-2 achieves comparable efficiency gains while requiring only a single training stage, making it particularly attractive for practical applications. The relatively smaller improvement in p50 measurements for single queries is due to their short output lengths (see 1), which limit the utilization of the Medusa heads. In particular, note that 18% of the samples in the Search test set include fewer than 5 tokens in their output, rendering at least 1 Medusa head completely useless. Medusa speedup relies on the assumption that the output distribution is long enough to utilize the added heads; the less this assumption holds, the smaller the speedup can be expected.

		AI Trip Planner			Smart Filters		
Technique	**Model**	**Micro-F1**	**P50**	**P99**	**Micro-F1**	**P50**	**P99**
SFT	TinyLlama-1.1B	89.4	449	995	85.8	89	406
SFT + M1	TinyLlama-1.1B	89.4	171	379	85.8	53	196
M2	TinyLlama-1.1B	89.9	160	316	87.7	**53**	180
SFT	Llama-3.1-70B	91.7	3149	7167	**89.1**	669	2502
SeqKD	TinyLlama-1.1B	91.4	475	1022	88.8	138	416
SeqKD + M1	TinyLlama-1.1B	91.3	170	359	88.7	78	184
SeqKD + M2	TinyLlama-1.1B	**91.8**	**150**	**293**	88.0	73	**162**
				OpenAI			
Zero Shot	GPT-4o-mini	46.4	1262	5645	46.7	819	4979
Few Shot	GPT-4o-mini	74.2	1290	5070	70.4	890	6305
Zero Shot	GPT-4o	54.9	1328	6122	53.8	1012	8899
Few Shot	GPT-4o	77.8	1652	6657	66.1	982	11597

Table 2: Performance and efficiency results across two use cases. We report vanilla supervised fine-tuning (**SFT**), Medusa-1 applied on top of SFT (**SFT + M1**), and Medusa-2 trained in a single step (**M2**). We report (**SeqKD**) from our teacher model (Llama-3.1-70B) to the student. Micro F1 is presented in % and P50 and P99 represent latency in ms.

For KD we additionally train a much larger teacher model: Llama-3.1 70b. Due to its scale, the teacher model achieves optimal performance but exhibits prohibitive latency for online deployment, not to mention its sizable cost and memory footprint.

We then use SeqKD, where the teacher's greedy decoding output gets concatenated with the original

[1]Model hashcode: MICROSOFT/DEBERTA-XLARGE-MNLI_L40_NO-IDF_VERSION=0.3.12(HUG_TRANS=4.43.1)-RESCALED

[2]https://openai.com/api/pricing/

human annotated dataset to train the small student model. We observe that in this setup, the distilled model almost eliminates the performance gap relative to the teacher model, effectively combining efficiency with high performance.

To investigate the complementarity between Medusa and SeqKD, we then apply both Medusa-1 and Medusa-2 to the distilled student model. The results demonstrate complete performance retention with consistent speed improvements, confirming the complementarity of these approaches. The final models maintain the teacher model's performance with negligible degradation while achieving substantial inference speed-ups of 10-20x.

6.2 Accommodation Description Generation

		Quality			Cost ($) / 1M tokens	
Technique	**Model**	**R-1**	**R-2**	**BERTScore**	**Input**	**Output**
Zero Shot	GPT-4	57.5	25.6	53.2	30.00	60.00
SFT	TinyLlama-1.1B	58.4	27.3	53.4	0.063	3.476
SFT + M1	TinyLlama-1.1B	58.3	27.3	53.3	0.063	1.810
M2	TinyLlama-1.1B	58.3	27.2	53.1	0.063	1.095

Table 3: Results for the Accommodation Description Generation task. We report vanilla supervised fine-tuning (**SFT**), Medusa-1 applied on top of SFT (**SFT + M1**), and Medusa-2 (**M2**). R-1 and R-2 stands for the ROUGE-1 and ROUGE-2 metrics, respectively. Quality metrics are presented in %.

Table 3 presents the results of the Accommodation Description Generation task using the TinyLlama-1.1B model variations. For comparison, we also include GPT-4 baseline evaluated in a zero-shot setting.

Our results show that fine-tuned models perform as well as or better than GPT-4 in terms of quality, demonstrating the effectiveness of our training pipeline and the performance of the TinyLlama-1.1B model despite its relatively small size. Furthermore, both Medusa variants achieve substantial computational efficiency, reducing costs by 1.9 and 3.2 times, respectively. Cost estimates were performed using a single g5.2xlarge machine.

7 Model Serving and Deployment

Our models are deployed on Amazon SageMaker AI g5.2xlarge instances using the TGI 2.2.0 container for optimized inference (Ifs et al., 2023). To ensure scalability and efficiency, we employ an auto-scaling mechanism that dynamically adjusts the number of instances based on request volume. This approach improves system robustness by efficiently handling peak loads while reducing costs during off-peak periods. Model-serving performance metrics (e.g., throughput, inference time) are continuously monitored through in-house dashboards and Amazon CloudWatch to maintain reliability and optimize resource utilization. The final models support both real-time prediction services and batch-based backfilling workflows.

7.1 Real-time Invocations

Both *AI Trip Planner* and *Smart Filters* require real-time inference, as user queries and conversations arrive dynamically and demand low-latency responses. To support this, we deploy a real-time service that: (1) Receives and processes user inputs (queries/conversations); (2) Invokes the appropriate model endpoint for inference; (3) Processes the model output before delivering the final response to the user.

7.2 Batch Invocations

For Accommodation Description Generation, inference runs in batch mode when property metadata is updated. This process consumes metadata events and triggers predictions asynchronously, using event-driven batch processing for efficient scaling and throughput optimization. To manage batch workloads, the system auto-scales model endpoints to handle peak demand while optimizing resource usage. Batch inference results are stored in a dedicated data pipeline before being consumed by the front-end application. This setup allows for controlled updates and periodic backfilling, ensuring that predictions remain accurate and up-to-date as new data becomes available.

8 Conclusions

This work demonstrates the practical viability of combining speculative decoding and model compression techniques to optimize industrial scale NLP systems. Our experiments show that the proposed combined framework delivers an improvement of inference latency larger than an order of magnitude while maintaining the performance of the best and largest open-source LLMs.

References

BERTScore. 2023. Bertscore default layer performance on wmt16.

Jake Brutlag. 2009. Speed matters. https://research.google/blog/speed-matters/. Accessed: 2025-03-04.

Tianle Cai, Yuhong Gao, Zhengyan Li, Hongyi Yang, Jiang Li, Jungo Kasai, Matei Zaharia, and Percy Liang. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

DeepSpeed team. 2021. Zero-offload: Democratizing billion-scale model training. https://www.deepspeed.ai/2021/03/07/zero3-offload.html. Accessed: 2024-03-18.

Google. 2023. Palm 2 technical report.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hassan Ifs, Philip Schmid, and Nicolas Patry. 2023. Text generation inference.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. *arXiv preprint arXiv:2308.00264*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Llama team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Xupeng Miao, Yujie Gu, Zhihao Huang, Xin Qu, Miao Huang, Xiaoyi Li, Zhihui Chen, Tong Zhang, Yihua Lin, Mingyu Jin, et al. 2023. Specinfer: Accelerating generative large language model serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Text Generation Inference. 2023. Text generation inference (tgi). https://github.com/huggingface/text-generation-inference.

Tianhua Xia, Patrick Lewis, and Baolin Peng. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A AI Trip Planner Input and Output Example

Conversation:

```
Assistant: Hello! I’m the AI Trip Planner.
How can I help you?
User: I want to travel in August.
Assistant: Great!
Where are you thinking of going?
User: Paris.
```

Model Output:

```
{"location": {"country": "France",
              "city": "Paris"},
 "checkin_month": 8}
```

B Accommodation Description

B.1 Template Example

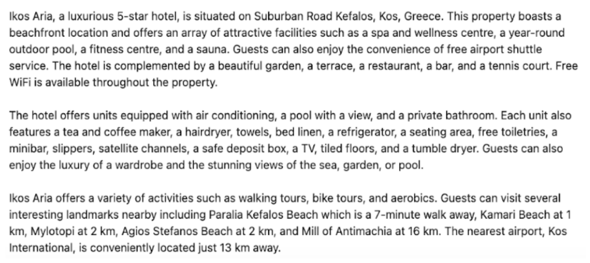
Hotel {{name}} is a {{num_of_stars}}-star hotel, located in {{city_name}}. The hotel provides {{facility_1}}, ..., and {{facility_n}}. Rooms include {{room_amenity_1}}, ..., and {{room_amenity_n}}. The nearest airport is {{nearest_airport_name}}, located {{distance_from_nearest_airport}} km away.

B.2 Description Generation Example

See Figure 2.

C Extracted Fields

The full schema, including filter names, types and where applicable the list of valid values, is shown below in JSON format.


Ikos Aria, a luxurious 5-star hotel, is situated on Suburban Road Kefalos, Kos, Greece. This property boasts a beachfront location and offers an array of attractive facilities such as a spa and wellness centre, a year-round outdoor pool, a fitness centre, and a sauna. Guests can also enjoy the convenience of free airport shuttle service. The hotel is complemented by a beautiful garden, a terrace, a restaurant, a bar, and a tennis court. Free WiFi is available throughout the property.

The hotel offers units equipped with air conditioning, a pool with a view, and a private bathroom. Each unit also features a tea and coffee maker, a hairdryer, towels, bed linen, a refrigerator, a seating area, free toiletries, a minibar, slippers, satellite channels, a safe deposit box, a TV, tiled floors, and a tumble dryer. Guests can also enjoy the luxury of a wardrobe and the stunning views of the sea, garden, or pool.

Ikos Aria offers a variety of activities such as walking tours, bike tours, and aerobics. Guests can visit several interesting landmarks nearby including Paralia Kefalos Beach which is a 7-minute walk away, Kamari Beach at 1 km, Mylotopi at 2 km, Agios Stefanos Beach at 2 km, and Mill of Antimachia at 16 km. The nearest airport, Kos International, is conveniently located just 13 km away.

Figure 2: Illustration of Accommodation Description Generation.

```
[
    {
        "key": "price_sensitivity",
        "type": "str",
        "valid": [
            "Cheap",
            "Luxurious"
        ]
    },
    {
        "key": "currency",
        "type": "str",
        "valid": [
            "Euro",
            "US Dollar",
            "British Pound",
            "SG Dollar"
        ]
    },
    {
        "key": "property_type",
        "type": "str",
        "valid": [
            "hostel",
            "hotel",
            "apartment",
            "villa",
            "chalet/cabin/lodge"
        ]
    },
    {
        "key": "facilities",
        "type": "List[str]",
        "valid": [
            "Swimming pool",
            "Bed (King/Queen)",
            "Bed (Double)",
            "Bed (Twin)",
            "Spa",
            "Jacuzzi/hot tub",
            "Airport service (shuttle)",
            "Airconditioning",
            "Garden",
            "Private bathroom",
            "Shower",
            "Wifi",
            "Parking",
            "Breakfast",
            "Restaurant",
            "Kitchen",
            "Sauna",
            "Balcony"
        ]
    },
    {
        "key": "city_center",
        "type": "bool"
    },
    {
        "key": "deals",
        "type": "bool"
    },
    {
        "key": "free_cancellation",
        "type": "bool"
    },
    {
        "key": "sustainability",
        "type": "bool",
        "test_only": true
    },
    {
        "key": "lgbt_friendly",
        "type": "bool"
    },
    {
        "key": "family_friendly",
        "type": "bool",
        "test_only": true
    },
    {
        "key": "pet_friendly",
        "type": "bool"
    },
    {
        "key": "nature_trip",
        "type": "bool"
    },
    {
        "key": "accessibility",
        "type": "bool"
    },
    {
        "key": "beach_trip",
        "type": "bool"
    },
    {
        "key": "ski_trip",
        "type": "bool"
    },
    {
        "key": "length_of_stay",
        "type": "int"
    },
    {
        "key": "num_adults",
        "type": "int"
    },
    {
        "key": "num_children",
        "type": "int"
    },
    {
        "key": "max_price_per_night",
        "type": "float"
    },
    {
        "key": "min_price_per_night",
        "type": "float"
    },
    {
        "key": "max_price_total",
        "type": "float",
        "test_only": true
    },
    {
        "key": "chain_name",
        "type": "str",
        "test_only": true
    },
    {
        "key": "hotel_name",
        "type": "str"
    },
    {
        "key": "landmark",
        "type": "str"
    },
    {
        "key": "district",
        "type": "str"
    },
    {
        "key": "airport",
        "type": "str"
    },
    {
        "key": "city",
        "type": "str"
    },
    {
        "key": "region",
        "type": "str",
        "test_only": true
    },
    {
        "key": "country",
        "type": "str"
    },
    {
        "key": "continent",
        "type": "str"
    },
    {
        "key": "checkin",
        "type": "str"
    },
    {
        "key": "checkout",
        "type": "str"
    },
    {
        "key": "strategy",
        "type": "str",
        "valid": [
            "Popular",
            "Nearby",
            "Deals",
            "Attractive",
            "Similar"
        ]
    },
    {
        "key": "month",
        "type": "int",
        "test_only": true
    },
    {
        "key": "romantic",
        "type": "bool"
    },
    {
        "key": "season",
        "type": "str",
        "valid": [
            "winter",
            "summer",
            "spring",
            "fall"
        ],
        "test_only": true
    },
    {
        "key": "num_beds",
        "type": "int"
    },
    {
        "key": "num_bedrooms",
        "type": "int"
    },
    {
        "key": "num_bathrooms",
        "type": "int"
    },
    {
        "key": "minimum_stars",
        "type": "float"
    },
    {
        "key": "minimum_review",
        "type": "int"
    },
    {
        "key": "sorter",
        "type": "str"
    }
]
```