EasyChair Preprint

№ 13944

Leveraging Large Language Models to Build a Low-Resource Customer Support Chatbot in a Three-Sided Marketplace

Biagio Antonelli and Gonzalo Cordova

July 12, 2024

Leveraging Large Language Models to build a low-resource customer support chatbot in a three-sided marketplace

Biagio Antonelli
Glovo
Barcelona, Spain
biagio.antonelli@glovoapp.com

Gonzalo Cordova
Glovo
Barcelona, Spain
gonzalo.cordova@glovoapp.com

ABSTRACT

The development of fully autonomous conversational agents using large language models (LLMs) remains a significant challenge due to the inherent complexity and critical importance of customer-facing systems. This paper investigates a hybrid approach that integrates LLMs within traditional task-oriented systems to enhance performance while maintaining safety and reliability.

We propose a modular design, incorporating LLMs into specific modules of the system to leverage modern architecture capabilities while mitigating risks like hallucinations. Our work outlines the architectural design, assesses the performance of the latest LLMs on intent classification tasks, discusses unique challenges posed by three-sided marketplaces, and trade-offs in designing and deploying such systems at scale.

We demonstrate how this system can achieve real-world impact with minimal investments in architecture, modeling, and dataset collection, achieving up to a 35% reduction in workload while maintaining customer satisfaction.

CCS CONCEPTS

• **Computing methodologies** → *Natural language generation.*

KEYWORDS

Large Language Models, Dialogue Systems, Conversational Agents, Intent Classification, Generative AI

1 INTRODUCTION

The recent advances in LLMs, exemplified by the global success of ChatGPT[1], and new paradigms such as Retrieval-Augmented Generation (RAG) [10] and agentic applications [22], have made LLM-based workflows the preferred choice for designing modern dialogue management applications.

Despite the notable success of some applications and the impressive rate of new developments, LLM pipelines can still be challenging to deploy safely at scale, due to issues such as hallucinations, outdated knowledge, scalability, and cost [2, 22].

Glovo develops and operates a three-sided marketplace connecting vendors, customers, and riders. Customers use our mobile or web app to place orders from vendors, that are then delivered by riders to the customer's location. In this context, a distinctive challenge in designing a chatbot for order status queries lies in the real-time, dynamic nature of the context. Unlike conversational agents designed for static knowledge bases, the context of an order being delivered can evolve mid-conversation. The system must access real-time information regarding the geographical location of the rider, initial and current Estimated Time of Arrival (ETA) estimations, and store saturation indicators. Customer support agents typically handle an average of approximately 300,000 monthly support requests related to order status, with each chat averaging approximately seven messages, 13 tokens long, resulting in a monthly volume of up to 300 million tokens across more than 20 countries and languages.

In this paper, we describe the trade-offs encountered while developing and deploying a customer support chatbot for delivery marketplace at scale. To fully explain the rationale behind our design decisions, we first present an overview of potential patterns to use when building such systems, ranging from traditional task-oriented systems to modern LLM-based agentic workflows, including hybrid workflows. We then detail the design of our chosen solution, discussing trade-offs, costs, experimental results, and failure modes, and conclude with potential future directions.

2 RELATED WORK

2.1 Dialogue systems

Dialogue systems can be categorized into task-oriented and non-task-oriented [5]. Task-oriented systems interact with users with the goal of making them achieve a certain task, while non-task-based systems can entertain more open ended conversations. Typically task-based systems treat the dialogue as a structured pipeline consisting of the following components: a Natural Language Understanding (NLU) unit; a tracker, to keep track of all parameters needed to manage the dialogue; a policy (or in our case decision engine), responsible to make a decision given latest context and metadata acquired by the NLU; and finally a Natural Language Generation (NLG) unit, responsible for translating the action selected into human language and advance the conversation.

Such systems can be designed with different level of complexities, going from pure intent recognition systems [19] to end-to-end trained systems such as the one described in [11], where both NLU, policy selection and NLG are neural networks. On the other hand, in [28] the problem is defined as a dynamic state machine to carry out the conversation based on the detected user intent. Conversational frameworks such as Rasa [3] make the design of such systems more flexible and each subcomponent can be set as either simple rules (e.g. regex), heuristics or more complex machine learning models, depending on the use case.

The recent advances in LLMs architectures and alignment techniques [15], coupled with the rise in popularity of frameworks such as LLama-Index and Langchain [4, 12, 17], have made LLMs a particularly attractive choice when designing a conversational system

[1] https://openai.com/chatgpt/

for the first time, particularly when the team lacks prior experience with such technologies or when resources are limited.

Agentic LLMs patterns like tools usage [7] can be leveraged to quickly create fully autonomous systems that are able to plan, execute actions such as API and database calls [18], and interact with both users and backend systems directly without human intervention. Even though, especially given the success of ChatGPT, it might be tempting to think that such a system can be easily designed and deployed at scale, these systems usually suffer from problems like hallucinations [18, 27], and issues around safety and privacy.

Hallucinations can be mitigated via finetuning, guardrails [20] and prompt engineering [23]. Moreover, modern libraries [4, 12] make it extremely easy to abstract away the specific LLM used, allowing to use interchangeably open and closed source models and seamlessly switch depending on needs. Finally, in terms of safety there are several guardrails [20] techniques which could be employed to mitigate potential issues.

2.2 Applications to Customer Support

Some applications in customer support [30] focus on tools that ease issue resolution for human agents without direct interaction with users. However, more recent studies [13] explore the development of chatbots that can engage in conversations with users, moving towards autonomous chatbots that handle interactions without human intervention. As mentioned in previous sections, recent breakthroughs in the field can be leveraged to enhance these customer service agents.

One example of this is the implementation of knowledge-based systems [14] or the development of Retrieval-Augmented Generation (RAG) engines [24] for question answering. However, these systems are more suited to tasks involving FAQ-style questions. RAG systems typically rely on a corpus of information from which they retrieve relevant data before generating a response. In dynamic environments like food delivery apps, where real-time information such as courier position, order status, and ETA estimations are dynamic, a RAG system might struggle.

Moreover, for customer service, especially in scenarios involving real-time data, accuracy and precision are paramount, and hallucinations can be problematic. In food delivery apps, for instance, customers expect exact details about their orders. While RAG systems are effective in generating human-like responses, they might not always provide the precise, up-to-date information required, potentially causing significant legal and reputational issues for companies [25].

To address these challenges, some approaches adopt a more defensive strategy, such as intent recognition systems. In [19] they explore different machine learning algorithms for intent classification, showing their effectiveness in understanding user queries.

Despite these advancements, there is still a gap in deploying large-scale, safe customer service agents using LLMs. To the best of our knowledge, implementations that balance customer satisfaction and cost-effectiveness using LLMs in a large-scale environment are scarce. Real-time data integration without hallucinations remains a challenge, making this a valuable area for research and development in multi-sided marketplaces.

3 METHODOLOGY

We adopted a hybrid approach that integrates traditional task-oriented components with modern LLM capabilities. Our primary goal was to design a robust, scalable, and low-risk modular solution that can be iteratively improved as new technologies mature. Here, we outline the components and methodology used in our system.

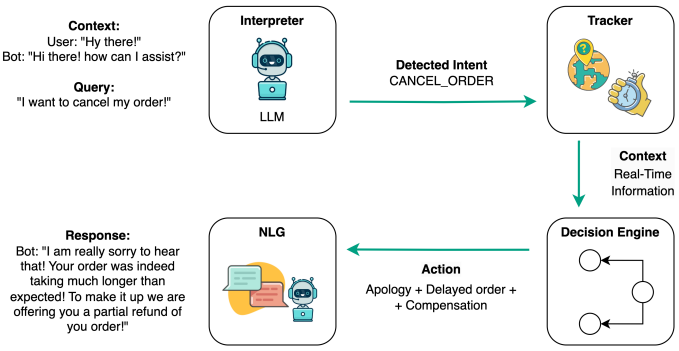

Figure 1: Glovo conversational system.

3.1 System Architecture

The architecture of our dialogue management system consists of the following key components:

- **Natural Language Understanding (NLU):** The NLU unit interprets user inputs to extract intents and sentiments, enabling the system to understand the user's goals and emotional state.
- **Tracker:** Maintains the context of the conversation and tracks real-time information such as rider location and ETA, ensuring the dialogue remains relevant and accurate.
- **Decision Engine:** Determines the next action based on the current state and context, using a combination of rules and heuristics.
- **Natural Language Generation (NLG):** The NLG unit generates human-like responses from action templates, ensuring variability and appropriateness in the system's replies.

3.2 Natural Language Understanding (NLU)

The NLU unit is responsible for extracting relevant information from the user's query. In the initial iteration of our system, this component focuses on extracting the user intent and sentiment.

3.2.1 Intent Classification. Within the NLU unit, the intent classification task involves categorizing a user query into one of several predefined intents. It is the most critical step of our system and serves as a proxy for the system's overall effectiveness, as the behavior of the other components is deterministic by design. In Table 1 we display a few intents and sample utterances for our use case. Sample utterances are also passed as context in the LLMs prompt (Fig. 4) for few-shot classification. Figure 2 shows the distribution of intents from our curated dataset. We see that the most common customer intents are to greet and to enquire on the status of the

order being delivered. The intents presented are only a subset of the 20 used in our production system.

Table 1: Selected Intents with associated sample utterances.

Intents	Sample Utterances
GREETING	- hello, there! - Hola!
DISAPPOINTMENT	- such a bad service! - I am disappointed!
ORDER_STATUS	- Entonces mi pedido no va a llegar? - Where is my order?
RIDER_LOCATION	- the rider location is wrong - the rider is not moving,
CANCELLATION_REQUEST	- I want to cancel my order! - cancela mi pedido
REDIRECT_TO_AGENT	- I want to speak to an agent - I want to speak to a human
GOODBYE	- bye! - close the chat

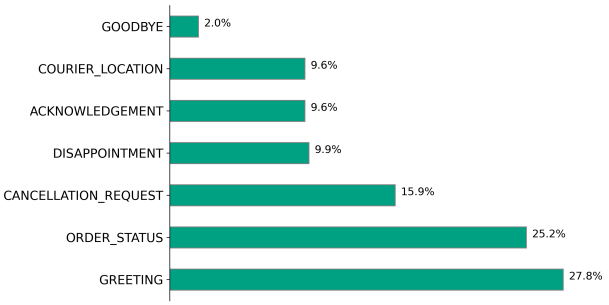

Figure 2: Distribution of Intents from curated dataset.

3.2.2 Dataset Curation. In order to evaluate different intent classifiers, we adopted an efficient pipeline inspired by BERTTopic and Llama2 [8] to expedite the data labelling process. Our method involves the following steps:

(1) **Automatic Clustering:** We start by clustering historical query data, using SentenceBERT as an embedding model [21]. This step groups similar queries together, forming clusters that represent different intents.
(2) **Sampling and Labeling:** From each cluster, we sample a subset of queries. We then utilize an LLM to assign one of the predefined intents to each cluster. If the LLM is unsure about a particular query, it labels the samples from the cluster as "UNKNOWN".
(3) **Human Verification:** We split the dataset in train and test, and we asked human agents to manually confirm or update all the pre-labelled data in the test set, while only reviewing the "UNKNOWN" from the training set.

This process enabled us to collect a dataset of approximately 7,000 samples within a single day. The dataset is structured in the following format: *context* | *query* $\rightarrow$ *intent*. The clustering approach was adopted to minimise costs during labelling iterations but in practice the cluster size could be set to 1 and we could get an LLM to label every single sample independently.

By leveraging this semi-automated approach, we ensured that our dataset is both comprehensive and accurately labeled, providing a solid foundation for training and evaluating our system.

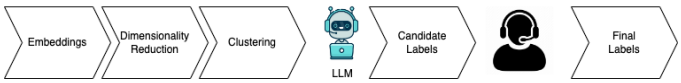

Figure 3: Dataset curation - semi-automated labelling pipeline with LLMs.

You are a professional customer support agent.

You are helping a customer with a question about their order status.

Your task is very simple. Given a user message, you need to classify the intent of the message based on the following intents to sample utterances:

Sample intent: utterances:
{intent_map}

Based on this current conversation:
{context}

You need to classify the intent of this new message written by the customer in the conversation: **{question}**

The answer must be one of the following intents: **{intents}** Unless the intent is "None of the above" in which case you should return UNKNOWN_INTENT

Figure 4: Interpreter Prompt.

3.3 Tracker

Effective customer support requires maintaining an updated context around the customer's inquiry. The tracker is a back-end component responsible for providing this real-time context to both the decision engine, which utilizes this information to make decisions, and the NLG unit, which conveys some of this information to the user.

Some of the key pieces of information managed by the tracker include:

- **Original ETA provided to the customer**: This helps in managing customer expectations and addressing any discrepancies that might arise if the estimated time of arrival changes.
- **Current ETA remaining estimate**: Providing an updated ETA helps keep the customer informed about any changes in the delivery schedule, enhancing transparency.
- **Store saturation**: Store saturation levels help understand potential delays.
- **Bundling information**: Knowing if an order is part of a bundle helps understand potential delays.

Additionally, the tracker supplies the interpreter with the message history, which can aid in accurately interpreting the user's intent.

3.4 Decision Engine

Given the metadata extracted from the NLU and the Tracker information, the rule-based decision engine selects a macro action. Macro actions are collections of action templates that are subsequently translated into messages by the NLG unit.

Figure 5 illustrates the process followed by the Decision Engine. A decision tree is selected based on the detected intent by the Interpreter. This tree is traversed, making decisions based on the data provided by the Tracker. Ultimately, a macro action is selected and passed to the NLG unit, along with the Tracker's data, to generate the message sent to the customer. Once the action is taken, the system continues to listen for a new customer intent or concludes the conversation.

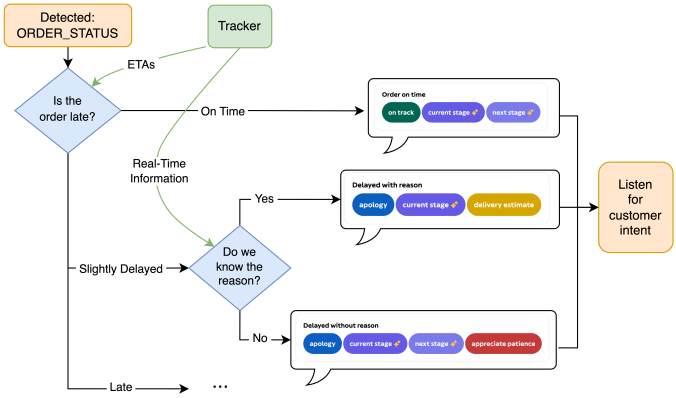

Figure 5: Decision Engine Flow example.

3.5 Natural Language Generation (NLG)

The NLG module is responsible for generating the messages to be sent to customers, based on the macro actions selected by the Decision Engine. Macro actions comprise a set of action response templates, each associated with a pool of text variations designed to convey the same core message. The NLG unit randomly selects a variant from each action response template and combines these selections to generate the final message. An illustrative example of response variations for the macro action "Late order with reason" is provided in Figure 6. It is important to note that the NLG unit uses the information provided by the Tracker to construct the final text.

While LLMs were used to generate text variations per action template, human reviewers ensure the text maintains acceptable quality and aligns with company values, tone, and standards. This step is crucial to prevent hallucinations, which could negatively impact performance and brand reputation.

3.6 Human-AI Interaction

We adopted a defensive UX design strategy [26] for integrating our system with the product. This strategy anticipates any inaccuracies that may occur during user interactions with our AI-based product, addressing these issues proactively by directing user behavior, preventing misuse, and managing errors effectively. Inspired by

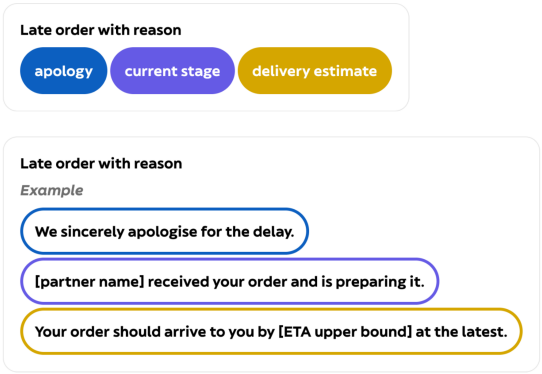

Figure 6: Action response templates.

guidelines from [1] [16], we implemented the following design patterns to ensure a quality Human-AI interaction:

- **Setting the right expectations:** Clearly informing the user that they are interacting with an automated bot rather than a human. As illustrated in Figure 7, this can be achieved by using a distinct bot profile picture and an introductory message from the bot.
- **Enabling user supervision of automation:** Providing users with an option to complete their task when the system does not work as intended by always displaying a "Redirect to human agent" button or by detecting the "REDIRECT_TO_AGENT" intent.
- **Facilitating user feedback:** Allowing users to provide feedback at the end of the interaction through the display of a star-rating system to indicate their level of satisfaction.
- **Avoiding hallucinations:** Our system mitigates the risk of generating misleading or erroneous responses by design.

Figure 7: Design for agent's introduction to user.

4 EXPERIMENTS

The evaluation of our system is divided into two parts. First, we use offline evaluations of the intent classification model as a proxy for the effectiveness of the whole system, allowing us to iterate fast on this critical sub-component. Subsequently, to accurately measure the business impact of the whole chatbot system, we run AB tests

Table 2: Evaluation of intent classification models.

Model	Accuracy	Precision	Recall	F1
gpt-4	**0.85**	**0.92**	**0.85**	**0.88**
gpt-4-turbo	0.83	0.90	0.83	0.86
gpt-4o	0.76	0.86	0.76	0.80
Llama3-8B-Instruct	0.75	0.85	0.75	0.74
gpt-3.5-turbo	0.73	0.85	0.73	0.78
Mistral-7B-Instruct-v0.2	0.39	0.86	0.39	0.51
Llama-2-7b-chat	0.17	0.47	0.17	0.21

that randomise exposure to interactions with the chatbot, and track impact on key operational metrics. In the following section, we report results of one of our offline evaluation studies. We also provide an overview of the results of a recent AB tests in the Online Experiments section.

4.1 Offline Experiments

We evaluate several open and proprietary LLMs on the few-shot intent classification task based on accuracy, precision, recall and f1 score on our test dataset.

The results, presented in Table 2 align in terms of relative rankings with publicly available benchmarks from the chatbot arena [6], except for GPT-4o being outperformed by both GPT-4 and GPT-4-Turbo in our experiments. GPT-4 and GPT-4-Turbo significantly outperformed the other models tested. Due to its cost-effectiveness, being one-third the cost of GPT-4 (see Table 3), and comparable performance, we selected GPT-4-Turbo for our online experiments. This choice is further justified by the faster insights it provides into the potential benefits of deploying such a system globally. Additionally, using GPT-4-Turbo, allows for quicker deployment without the need to set up a GPU cluster for serving model predictions, unlike other open-source models.

Smaller, fine-tuned models have demonstrated the capability to outperform larger models on specific tasks [9, 29]. In the future, once the benefits of the system are fully validated, we plan to fine-tune a smaller model to reduce the costs associated with the intent classification module and improve performances.

4.2 Online Experiments

To measure the real-world impact of our chatbot, we conducted A/B tests where users were randomly assigned to either interact with the chatbot or with human agents. Key operational metrics, such as the rate of conversation escalations, cancellation rates, and customer satisfaction scores (CSAT), were tracked.

- **Reduction in Escalations:** The chatbot reduced the number of conversations escalated to a human agent by over 35% compared to the existing flow.
- **Customer Satisfaction:** There was no significant change in the CSAT scores, indicating that the chatbot didn't impact significantly user experience.
- **Cancellation Rates:** The rate of order cancellations remained unchanged, suggesting that the chatbot did not negatively impact user decisions.

4.3 Failure Modes

We followed a Conversation-Driven-Development [2], meaning that while we were aware that there would be many things we couldn't anticipate, we prioritised experimenting and iterating fast over feedback. Below, we summarize some of the failure modes encountered and the corresponding mitigation strategies:

Loops: In certain instances, conversations became trapped in a loop wherein users repeatedly asked the same question due to unsatisfactory responses. To address this, we implemented a detection mechanism to identify such situations and reroute them to an agent for resolution.

Intent confusion Queries like "*Donde esta?*" could be classified ambiguously both as ORDER_STATUS or RIDER_LOCATION if no context is provided. This could cause issues leading to a potential sub-optimal action and should be avoided wherever possible. To mitigate the issue, our Tracker back-end system not only passes the latest user query to the Intent Classifier but also includes a history of previous messages as context.

Multiple intents in a single query It is common to have a user mixing several intents in the same query, as an example, a user might ask "*Hello there! I am very frustrated that my order hasn't arrived yet, where is it? I want to cancel it now!*". In such a query we observe at the same time many of our intents (GREETING, ORDER_STATUS, CANCELLATION_REQUEST). To address this issue, setting clear expectations and refining prompts, as discussed in Section Human-AI Interaction, can guide the system in handling such situations effectively.

Context aggregation: Queries may be fragmented, with users typing and sending messages sequentially. Implementing specific UI or backend rules to discern when a user has finished typing, aggregate the message, and send it to the AI Interpreter is necessary to ensure accurate interpretation.

Hallucinations: Misspelling or made-up intents. This could be mitigated via prompt engineering and robust response parsing strategies.

Latency and Costs: When conducting initial experiments, it is crucial to weigh the trade-offs between using external APIs and investing in infrastructure to serve open-source models on a proprietary cluster. Given the significant performance gains demonstrated by GPT-4 family (Table 2), we opted to use the external API. To manage costs and improve efficiency, we implemented a semantic caching strategy similar to GPTCache [2]. This involved using SentenceBERT [21] multilingual embeddings from the Hugging Face library[3], which allowed us to significantly reduce the number of calls to the service.

[2]https://rasa.com/docs/rasa/conversation-driven-development
[3]https://huggingface.co/

Table 3: Approximate cost breakdown for different GPT models based on estimated monthly tokens volume.

Model	Monthly Cost
gpt-4	$60,000
gpt-4-turbo	$20,000
gpt-4o	$10,000
gpt-3.5-turbo	$1,000

5 CONCLUSIONS

Modern LLMs may still lack the maturity to create fully autonomous conversational systems. Customer-facing conversational systems are inherently complex and critically important for companies, as they interact directly with customers, making safety and reliability paramount concerns. Recent developments show promise for creating fully autonomous, reliable, and flexible conversational agents. For instance, Gao et al. [7] demonstrate that fine-tuning LLMs to produce an abstract, multi-step reasoning chain (Chain of Abstraction) allows these models to effectively use tools, aggregate results, and provide comprehensive responses.

As these technologies mature, our modular and extensible hybrid system proves to be successful in combining traditional task-oriented structures with LLMs in specific sub-modules such as intent classification, natural language generation, and dataset curation. The modularity of the system allows new innovations to be integrated quickly, moving towards increased autonomy as technology advances. We demonstrate that our hybrid workflow, coupled with a UI centered around Human-AI interaction design, can deliver significant business impact without substantial investments in computing, specialized skills, and dataset curation. These results are validated through both offline benchmarks on intent classification and real-world live experiments, showing that the system, even in its first iteration, can significantly reduce agents' workload while maintaining customer satisfaction.

ACKNOWLEDGEMENTS

The authors would like to thank Glovo for providing the data and the resources to carry out this work. The authors would also like to thank Ezequiel Smucler, Esteban Atlasovich, Preeti Grover, Sebastian Calderon, Raghavendra Sai, Gustavo Bonilla, Valentin Delbeke, Marc Aynés, Fidel Omolo and Patricia Sánchez for their support and contributions to this project.

REFERENCES

[1] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *CHI 2019*. ACM. https://www.microsoft.com/en-us/research/publication/guidelines-for-human-ai-interaction/

[2] Fu Bang. 2023. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. 212–218.

[3] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open Source Language Understanding and Dialogue Management. *CoRR* abs/1712.05181 (2017). arXiv:1712.05181 http://arxiv.org/abs/1712.05181

[4] Harrison Chase. 2022. *LangChain*. https://github.com/langchain-ai/langchain GitHub repository.

[5] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter* 19, 2 (2017), 25–35.

[6] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. arXiv:2403.04132 [cs.AI]

[7] Silin Gao, Jane Dwivedi-Yu, Ping Yu, Xiaoqing Ellen Tan, Ramakanth Pasunuru, Olga Golovneva, Koustuv Sinha, Asli Celikyilmaz, Antoine Bosselut, and Tianlu Wang. 2024. Efficient Tool Use with Chain-of-Abstraction Reasoning. *arXiv preprint arXiv:2401.17464* (2024).

[8] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794* (2022).

[9] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL]

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[11] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008* (2017).

[12] Jerry Liu. 2022. *LlamaIndex*. https://doi.org/10.5281/zenodo.1234

[13] Dylan Malvin, AHC Rangkuti, et al. 2022. WhatsApp chatbot customer service using natural language processing and support vector machine. *Int. J. Emerg. Technol. Adv. Eng.* 12, 3 (2022), 130136.

[14] Eric WT Ngai, Maggie CM Lee, Mei Luo, Patrick SL Chan, and Tenglu Liang. 2021. An intelligent knowledge-based chatbot for customer service. *Electronic Commerce Research and Applications* 50 (2021), 101098.

[15] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

[16] Google PAIR. 2019. People + AI Guidebook. https://pair.withgoogle.com/guidebook. Updated May 18, 2021.

[17] Keivalya Pandya and Mehfuza Holia. 2023. Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations. *arXiv preprint arXiv:2310.05421* (2023).

[18] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334* (2023).

[19] V Rajesh, B Perumal, Uppu Sai Ganesh, Vemulakonda Rajkumar, Divi Mani Kumar, and Manoranjan Kumar. 2023. Building Customer Support Chatbots With Intent Recognition. In *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*. IEEE, 1–5.

[20] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. *arXiv preprint arXiv:2310.10501* (2023).

[21] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[22] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* 36 (2024).

[23] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* (2023).

[24] Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024. Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering. *arXiv preprint arXiv:2404.17723* (2024).

[25] Maria Yagoda. [n. d.]. Airline held liable for its chatbot giving passenger bad advice - what this means for travellers. *BBC* ([n. d.]). https://www.bbc.com/travel/article/20240222-air-canada-chatbot-misinformation-what-travellers-should-know

[26] Ziyou Yan. 2023. Patterns for Building LLM-based Systems & Products. *eugeneyan.com* (Jul 2023). https://eugeneyan.com/writing/llm-patterns/

[27] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469* (2023).

[28] Shayan Zamanirad, Boualem Benatallah, Carlos Rodriguez, Mohammadali Yaghoubzadehfard, Sara Bouguelia, and Hayet Brabra. 2020. State machine based human-bot conversation model and services. In *Advanced Information Systems Engineering: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, Proceedings 32*. Springer, 199–214.

[29] Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. 2024. LoRA Land: 310 Fine-tuned LLMs that Rival GPT-4, A Technical Report. *arXiv preprint arXiv:2405.00732* (2024).

[30] Huaixiu Zheng, Yi-Chia Wang, and Piero Molino. 2018. COTA: Improving Uber Customer Care with NLP & Machine Learning. https://www.uber.com/en-ES/blog/cota/