Blip Copilot: a smart conversational assistant

Evandro Fonseca, Tayane Soares, Dyovana Baptista,
Rogers Damas and **Lucas Avanço**

Blip

evandro.fonseca, tayane.soares, dyovana.baptista,
rogers, lucas.avanco
{@blip.ai}

Abstract

This paper describes Blip Copilot plugin, an AI-based assistant that provides quick and smart suggested answers for an enriched conversational experience.

1 Introduction

Typically, customer service chats in large and mid-sized companies can face high demand, causing attendants to become overwhelmed and resulting in delays in responding to customers. Considering this overload and current advances in the development of language models in Natural Language Processing (NLP) (Brown et al., 2020), in this paper we present Blip Copilot. Blip Copilot is an assistant that optimizes customer service on Blip Desk[1]. With this extension, attendants can access personalized answer suggestions provided by a language model that takes into account a knowledge base built for a specific context or domain. All copilot suggestions are generated considering the conversational context (thread of messages in a chat) and the knowledge base provided during setup. Our approach is fully supervised. It is up to attendant to decide select, discard or edit the provided suggestions. To generate more accurate responses, we use NLP techniques to process, extract and find relevant information regarding current topic conversation. It will be more described in subsequent sections.

2 Architecture

Our architecture is language and LLM model agnostic. In our experiments for the Portuguese language, we tested two models: PaLM-2(Anil et al., 2023) and GPT-3.5-turbo(Brown et al., 2020). However, GPT-3.5-turbo has presented more accurate responses than PaLM-2. When developing applications that incorporate LLM into their pipeline, it is important to consider the limitations of these models. The texts generated by these tools may be biased, and their answers will not always be assertive (Bender et al., 2021). Therefore, it is always necessary to always validate the output of applications that use LLMs. Given these limitations, Blip Copilot combines the use of Retrieval Augmented Generation (RAG) (Lewis et al., 2020) with the prompt provided to the LLM to improve the accuracy of suggested responses. In this way, suggested responses are generated considering the context of the conversation in the customer service chat and the context retrieved from the specific domain knowledge base provided during plugin configuration. Before using Copilot, it is necessary to configure the knowledge base and other specific features, such as the company name, service demands, and other information[2]. Throughout this process, each entry of knowledge base is embedded and stored in a vector database. Once the setup process is complete, copilot is ready to use. Figure 1 shows the Copilot pipeline. When the attendant (user) calls Blip Copilot, an embedding vector of the last n[3] messages are extracted. The next step is to look for the most similar instance previously built in the setup stage. For this, we use cosine similarity(Rahutomo et al., 2012). Finally, we build a LLM call using the conversational context, the retrieved knowledge base instance, and the entire customer setup information.

3 Interface

In order to provide an easy setup, we turn available a web interface, which consists in allow specific configurations, such as: brand name, service

[1] Blip Desk is a customer service tool that allows a chatbot to redirect (overflow) a user's conversation to a human attendant on different channels. https://help.blip.ai/hc/pt-br/articles/4474416681495-Visão-geral-do-Blip-Desk

[2] see section 3 –Interface

[3] the window of messages is a configuration parameter

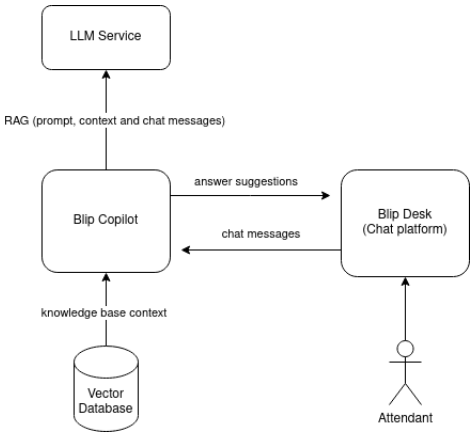

Figure 1: Copilot architecture

demands, profile, additional rules(observations) among others. Our interface is distributed in four tabs: Basic setup, knowledge database upload, greeting messages setup and advanced configurations.

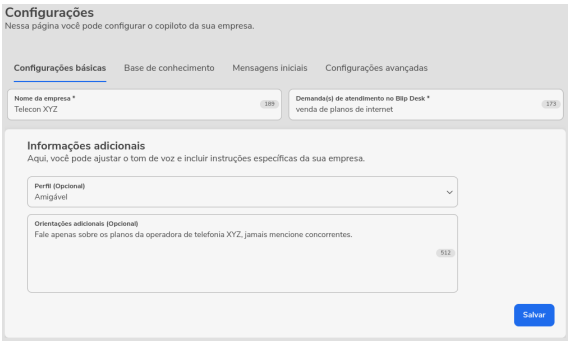

Figure 2: Basic setup

In figure 2 there specific configurations that addresses the copilot behavior. Each presented field is very important because the model follows a specific prompt which considers user data to provide the suggested answers. In the mentioned image, we can see that this copilot represents a Telecom company, acts in Sales of internet plans, should provide responses using a friendly discourse and never talk about other Telecom providers.

In Figure 3 the database upload interface is presented. The knowledge base is very useful to address information about business rules or specific products and its restrictions, for example. Regarding data structure, Blip Copilot accepts two file formats(txt and tsv), each line represents an instance. Moreover it is possible to provide some contextual information, complying with the following pattern: “Topic | Description”. Ex:

- Basic Plan | The Basic Plan is an option for basic internet needs. With speeds of up to 10 Mbps ...

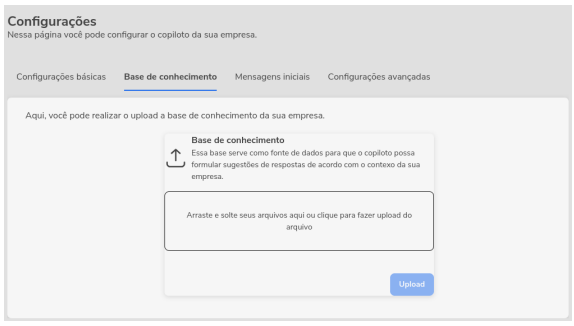

Figure 3: Knowledge base upload interface

Figure 4: Greeting messages setup

In Figure 4 we present the greeting messages setup interface. The greeting messages are provided as a first suggestion. It is useful to introduce a chat.

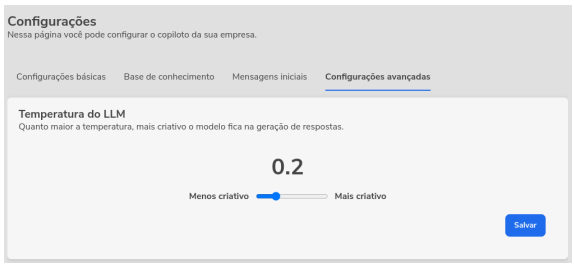

Figure 5: Advanced settings

The Figure 5 refers to model temperature. This parameter influences its "creativity" or randomness and is widely used by many LLM models. However, when we use high temperatures there is a risk of model "hallucinating", that is, high temperatures increase the randomness of the model and can generate answers with low assertiveness or that are wrong.

Finally, we show two cases and their outputs. In Figure 6, it is clear that the customer needs a second copy of his bill, and in Figure 7 the customer asks

about available internet plans. Here it is clear that copilot makes use of customer knowledge base to retrieve the correct plan names and values.

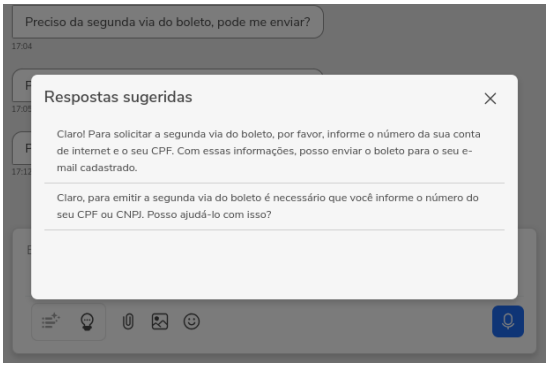

Figure 6: Suggested answers - invoice

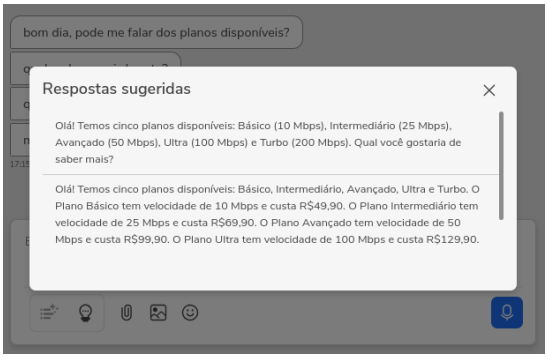

Figure 7: Suggested answers - internet plans

4 Experiments and Results

In order to show the relevance of Blip Copilot use, we have conducted an experiment involving five participants. Basically we compared each attendant with itself, considering the number of closed tickets and the average time of each service with and without Blip Copilot. For this experiments two window time were considered: September 2023 (without copilot) and November 2023 (with copilot). We skipped October because we believe there is a learning curve in refining the knowledge base to produce more accurate responses. Thus, each attendant had one month to learning how to use copilot and to obtain a better performance of its use.

Attendant	Closed Tickets Without Copilot	Closed Tickets Using Copilot	Avg. Time Without Copilot	Avg. Time With Copilot
attendant 1	34	773	01:00:12	00:15:54
attendant 2	568	628	00:36:10	00:17:28
attendant 3	875	823	00:15:00	00:08:31
attendant 4	651	782	00:22:11	00:11:41
attendant 5	612	719	00:28:40	00:14:33

Table 1: Results

As a result, it is possible to see that there is a significant improvement in average service time when Copilot is used. All attendants considerably reduced their time, and except for Attendant 3, all other participants increased the amount of their closed tickets.

5 Conclusion

In this paper, we presented Blip Copilot, a smart conversational assistant that considers the chat context, the brand/customer database, knowledge base, and custom parameters to suggest accurate responses. We also showed our Copilot architecture and how this chat assistant can improve the day-to-day of many brands/businesses, reducing the service time and maximizing the attendants efficiency. Also, the Copilot smooths the onboarding process of new attendants. As further work, we intend to add a feedback system so that Copilot can learn from user feedback and become more accurate over time. We also want to integrate our Copilot with Blip Desk mobile version[4].

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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*, 33:9459–9474.

Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. 2012. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1.

[4] A Blip Desk version for smartphones