Meta-prompting Optimized Retrieval-Augmented Generation

João Rodrigues(✉) and António Branco

NLX—Natural Language and Speech Group, Department of Informatics, Faculdade de Ciências (FCUL), University of Lisbon, Campo Grande, 1749-016 Lisboa, Portugal
{jarodrigues,antonio.branco}@fc.ul.pt

Abstract. Retrieval-augmented generation resorts to content retrieved from external sources in order to leverage the performance of large language models in downstream tasks. The excessive volume of retrieved content, the possible dispersion of its parts, or their out of focus range may happen nevertheless to eventually have a detrimental rather than an incremental effect. To mitigate this issue and improve retrieval-augmented generation, we propose a method to refine the retrieved content before it is included in the prompt by resorting to meta-prompting optimization. Put to empirical test with the demanding multi-hop question answering task from the StrategyQA dataset, the evaluation results indicate that this method outperforms a similar retrieval-augmented system but without this method by over 30%.

Keywords: RAG · Retrieval-Augmented Generation · Prompt Optimization · Large Language Models · Meta-prompting · Multi-hop QA

1 Introduction

Pre-trained Large Language Models (LLMs) [22,32] are known for their hallucinations [12] and for their further limitations regarding truthfulness [17]. To tackle these issues, remediation techniques have been explored such as, for instance, fine-tuning [10], prompt-engineering [18] or Retrieval-Augmented Generation [15], initiated by Houlsby et al. [9] among several others.

1.1 Retrieval-Augmented Generation

Focusing on Retrieval-Augmented Generation (RAG), this approach seeks to enhance truthfulness and curb hallucinations by expanding the initial prompt, which contains the initial query, with additional content retrieved from sources that are external to the LLM. Such additional content is obtained with the help of an auxiliary Retrieval Model where the retrieval model may be a simple Jacquard model or a vector database that extracts relevant content from external sources and pass it on to a Large Language Model that generates an appropriate

M. F. Santos et al. (Eds.): EPIA 2024, LNAI 14969, pp. 203–214, 2025.
https://doi.org/10.1007/978-3-031-73503-5_17

response given the original query and the extracted content. If this external content is unstructured text, it may be of different lengths, such as sentences, paragraphs or full documents, among others.

By feeding LLM's knowledge, and curbing its whim, with further knowledge from external sources, more accurate answers are likely to be provided.

Compared with other techniques, such as fine-tuning or prompt-engineering, RAG key advantage is the ease with which newer, up-to-date content is taken advantage of, as this does not require the costly compute of re-training neural networks (as in fine-tuning) or the costly human labour for the creation of further manually designed prompts (as in prompt-engineering). To be sure, all these techniques can nevertheless be mixed and function together.

1.2 Prompt Optimization

Usually, the pieces of content retrieved may be from heterogeneous sources and they tend to lack a connecting thread. They may also be redundant or may be of very high volume. These, among other aspects, may end up having a detrimental effect and eventually jeopardizing the generation task, rather than enhancing it.

To mitigate this problem, we present a method that consists of adding an intermediate step between the retrieval of the external content and the entering of the expanded prompt into the LLM to finally obtain the response to the initial query. Aiming at improving the performance of this generation-LLM, this intermediate step seeks to obtain a refined version of the external knowledge.

This refinement is accomplished by means of an auxiliary transformation-LLM that is entered with a prompt containing the pieces of retrieved contents, preceded by an instruction with the request for the sought refinement.

For example, if several pages of Wikipedia are retrieved as possible relevant content, the transformation-LLM processes this content and may generate a summary or remove unnecessary information from that original content.

Turning to this refinement instruction, this is obtained by an automatic procedure that is preliminary to running the RAG system made of transformation- and generation-LLMs, and it is undertaken by yet a third LLM.

Inspired in Yang et al. [35], in this procedure a meta-prompt is used as input to this third, optimizer-LLM for this to iteratively generate new tentative instructions, score them, and retain, in the meta-prompt itself, a list with the top k ones that induce better performance for the RAG system. By the end of this optimization process, the best scoring instruction in this list is the one retained to be used in the refinement step of the retrieved contents with the transformation-LLM.

This meta-prompt contains a meta-instruction and a list of tentative instructions that is aimed at being updated during this process with new instructions that induce better RAG performance. After a new tentative instruction is generated, its contribution to approximate the gold output to the initial query is scored, and the list of tentative instructions in the meta-prompt is possibly updated so that it retains the top-performing ones so far. This is iterated, and

an optimization trajectory is hence accomplished to eventually find the new refinement instruction that maximizes the success of the RAG system.

***

In this paper, we propose a method for RAG to be enhanced with the refinement of the retrieved content, a refinement that is optimized by resorting to iterative meta-prompting. This is a novel method that can be combined with previous approaches aimed at enhancing RAG.

We report on the experiments performed to put this method to the test. This approach is extrinsically evaluated by being embedded in a demanding question-answering downstream task. Its performance demonstrates that it is an effective method to enhance RAG by improving by 30% the performance of a baseline RAG without this method, and that it can be combined with other previous state of the art methods for RAG enhancement proposed in the literature.

The remainder of this paper is structured as follows: Sect. 2 discusses related work; Sect. 3 describes the method proposed in this study; Sect. 4 reports on the models and dataset resorted to; Sect. 5 presents the experiments undertaken and their evaluation, and discusses the results obtained; and finally, Sect. 6 closes this paper with concluding remarks.

2 Related Work

Prompt optimization has gained traction as an effective mechanism for enhancing LLMs in several downstream tasks [1,14].

The earliest approaches in prompt optimization sought to directly optimize the prompt embedding space, such as prefix-tuning [16] or OptiPrompt [37]. These aimed at optimizing a sequence of continuous task-specific vectors applied to the prompt to leverage downstream tasks.

More recent studies have introduced further techniques to enhance prompts, such as chain-of-thought [34] and tree-of-thoughts [36]. The former involves extending prompts with a few manually written chain of thought demonstrations as examples, which results in improved performance across various tasks, including arithmetic, commonsense and symbolic reasoning. The latter builds upon the chain-of-thought by considering multiple reasoning paths, self-evaluating choices, and by making global decisions by looking ahead or backtracking when necessary.

Other methods for optimizing prompts include searching through a pool of prompt candidates generated by an LLM, employing principled planning algorithms based on Monte Carlo tree search [33], or applying iterative local edit operations at a syntactic phrase-level split within the prompts [21].

Further proposals encompass EvoPrompt [7], which uses evolutionary operators over a prompt population for optimization, while Sabbatella et al. employs Bayesian Optimization within a prompt search space [26], reinforcement learning to rewrite prompts [13] or a prompt optimization that integrates human-design feedback rules to suggest improvements automatically [5].

Recently, Yang et al. [35] introduced OPRO, leveraging LLMs as optimizers through meta-prompts, which are natural language descriptions that guide prompt optimization. It was applied to optimize prompts by retrieving and re-ranking top-K relevant instructions with respect to an initial instruction, and by appending them to the global task description.

In contrast, to enhance RAG, we propose a method to optimize the prompt that differs from the previous proposals in the literature.

A prompt for RAG includes a query and the content retrieved from external sources on the basis of that query. It may contain also an instruction about how to handle the query or how the retrieved content should be used by the generation-LLM to answer it. Related work for RAG enhanced with prompt optimization has concentrated on optimizing the instruction and/or the query. Differently from previous approaches, our method focuses instead on optimizing the version of the retrieved content that is included in the prompt entered into the generation-LLM. Hence, rather than being an approach alternative to previous ones, it is a new one that is complementary to them and may be combined.

3 Method

The objective of our method is to enhance the RAG performance of a generation-LLM by means of the improvement of its input prompt, which is made of a query introduced by the user and of pieces of content retrieved from external sources on the basis of that query. Before it is entered into the generation-LLM, this prompt is improved by means of a refinement of the retrieved content, performed by a transformation-LLM.

Table 1. Meta-prompt - An example of a meta-prompt: in black, the top paragraph with the meta-instruction actually used in the experiments; below, in green, the list of top performing instructions so far, and the respective scores.

I have some prompts along with their corresponding scores. The prompts are arranged in ascending order based on their scores, where higher scores indicate better quality. Together with relevant information extracted from a database, these prompts are given as input to a large language model in order to optimize the provided relevant information. Several techniques may help the optimization, such as re-ranking paragraphs, cleaning, filtering and summarization. Write your new prompt taking into account the previous ones and aiming to achieve a higher score.

prompt:
Summarize the main idea of the previous text.
score:
3.0

prompt:
Summarize the main points in 30 words or less.
score:
3.0

And before a first query is accepted to put the RAG system to use, the prompt to be used with the transformation-LLM for refinement purposes is optimized. This prompt includes a refinement instruction and the pieces of retrieved content to be refined. It is optimized by means of the optimization of this instruction through iterative meta-prompting.

This meta-prompting optimization is undertaken by an optimizer-LLM that is entered with a (meta-)prompt that includes a (meta-)instruction and a list of tentative refinement instructions and respective performance scores. These scores are obtained by running the RAG with the tentative refinement instruction through a sample of training examples and evaluating the output against the respective gold responses.

Focusing on the optimization phase, a meta-prompt is used that contains both the description of the optimization problem and the history with previous best solutions for the instruction. Such meta-prompt is iteratively entered into the optimizer-LLM, and at each iteration that history is possibly updated with generated instructions if these support better performance for the task at stake in the generation phase. The instruction selected out of this optimization process is the best scoring one in the history obtained as this iteration is over.

An example of a meta-prompt is in Table 1, and a detailed description of this optimization via iterative meta-prompting is presented in Algorithm 1.

Algorithm 1. Optimization with meta-prompting

Input: Dataset D with n examples, each containing a query q, retrieved contents c and the answer a; meta-prompt $metaP$ with the description of the optimization task and with a list of instructions and respective scores
Output: List of scored instructions and the best scoring instruction
while optimizing prompt **do**
    Enter meta-prompt to optimizer-LLM
    Generate new instructions I
    Select a random subset E of examples e from D
    for each instruction I_j **do**
        for each example e_k in E **do**
            Assemble prompt $TransP$ from I_j and contents c_k
            Enter $TransP$ to transformation-LLM
            Generate transformed contents tc
            Assemble prompt $TaskP$ from query q_k and tc
            Input $TaskP$ to generation-LLM
            Generate answer and evaluate it against gold a_k
        end for
        Compute $I_j score$
    end for
    Update $metaP$ by replacing its worst scoring instruction by I_j and $I_j score$ if this is better scored
end while

4 Dataset and Models

To empirically assess the performance gains of the proposed method, it was integrated into an RAG for question-answering whose performance provides for its extrinsic evaluation.

4.1 Task and Dataset

Multi-hop question answering requires taking into account disparate pieces of content to get at the answer for a query, which constitutes a most demanding scenario for the task of question answering.

We resorted to a most complex benchmark for multi-hop question-answering available in the literature, the StrategyQA dataset [6,8,11,19], which contains 2,780 queries, each associated with related content made of paragraphs and the respective *yes* or *no* answer. Based on Wikipedia content, this dataset covers a range of diverse topics and the task consists in, given a query, to provide an accurate answer to it together with the passages retrieved from Wikipedia with the most relevant content to get at that answer—Table 2 displays an example.

Table 2. StrategyQA - An example from the StrategyQA, with a query, three of the most relevant pieces of content, and the respective answer.

Query	Could $1 for each 2009 eclipse buy a copy of TIME magazine in 2020?
Content #1	It set out to tell the news through people, and for many decades through the late 1960s, the magazine's cover depicted a single person. [...] Raymond Fielding also noted that Larsen was "originally circulation manager and then general manager of Time, later publisher of Life, for many years president of Time Inc., and in the long history of the corporation the most influential and important figure after Luce"
Content #2	Total eclipses are rare because the timing of the new moon within the eclipse season needs to be more exact for an alignment between the observer (on Earth) and the centers of the Sun and Moon. [...] because totality exists only along a narrow path on the Earth's surface traced by the Moon's full shadow or umbra.
Content #3	At least two lunar eclipses and as many as five occur every year, although total lunar eclipses are significantly less common. If the date and time of an eclipse is known, the occurrences of upcoming eclipses are predictable using an eclipse cycle, like the saros.
Answer	Yes

To provide for the evaluation of the proposed method, and isolate the accrued performance induced by it, thus disregarding possible fluctuation or loss of performance due to the retrieval process, only the gold pieces of content from a test set should be taken into account. Since the answers are not provided in the original test set of StrategyQA, a new test set for the present evaluation exercise had to be built. Accordingly, we divided the original training set into two parts: a new test set with 490 of the original training examples, which matches the size of the original test set, and a new training set containing a subset with 1800 such examples. The resulting train and test sets have an average query length of 9.6 words, and 2.33 contents (paragraphs) per query and are almost balanced. The training set contains 834 yes answers and 966 no answers (46.32%/53.68%). The test set contains 237 yes answers and 253 no answers (48.40%/51.60%).

4.2 Models

Two Transformer-based language models with 70 Billion parameters were used, a pre-trained Llama-2-70b and an instruct model Llama-2-70b-chat fine-tuned for dialogue use [31]. These models were trained and fine-tuned with a context length of 4k tokens over 2 Trillion tokens on a mix of publicly available data.

In general, the default Llama2 model hyper-parameters were applied and no hyper-parameters search bound was performed. All language models use a temperature value of 1.0, a maximum of 64 generation tokens for the new instructions, a maximum of 128 generation tokens for the refined content, and a maximum of 64 generation tokens for the response to the task. The optimization run was performed for two days on two NVIDIA A100 40 GB GPUs.

All software and versioning along with hyper-parameters are fully described in the source code of these experiments.[1]

4.3 Evaluation Procedure and Metrics

Based on empirical experimentation, we arrived at a meta-prompt, presented in Table 1, that indicates the aim of the optimization problem and includes a starting example instruction.[2]

The instruction optimization was iterated over 100 steps. At each step, 3 instructions were generated, each such instruction was evaluated on a random sample of 6 training examples, and the meta-prompt was eventually updated to containing the 8 top scoring queries so far. When this iteration was concluded, the best scoring instruction was retained as the optimized instruction.

We compare against the same generation-LLM using test queries and associated pieces of content, that is without the later being refined by the transformation-LLM under the instruction that was optimized by the optimizer-LLM.

For the StrategyQA task, a Boolean answer is expected. Accuracy is thus the metric used for evaluating the match between the answer output by the system and the gold answer in the data set. Accuracy score is given by the proportion of correct answers, and a generated response was counted as correct if the gold answer was found in the exact beginning of it. The response underwent minimal normalization, with just lowercasing.

As for the instrumental process of instruction optimization, it is worth recalling that the evaluation is performed over sample examples from the training set. For a tentatively generated instruction, a correct answer to it counted 1 point; an incorrect answer, in turn, counted 0.5 points if it was nevertheless in a Boolean format, or counted 0 points otherwise. The maximum possible score was thus 6 points, given each tentative instruction was evaluated against 6 sampled queries as indicated above.

5 Results and Discussion

In this section, we report on the evaluation exercise undertaken to assess the proposed method and discuss its results, summarized in Table 3

[1] For the sake of reproducibility, data and code are available at https://github.com/nlx-group/rag-meta-prompt.

[2] The starting instruction is “Clean and organize the previous text.”.

5.1 Experiments

All in all, six experiments were undertaken, two resorting to the model Llama-2-70b, developed with a pre-training regime only, and four resorting to the model Llama-2-70b-chat, which resulted from further fine-tuning it with dialogue data.

Table 3. Evaluation - From the total 490 test set examples, the number of correct answers is presented and the respective accuracy.

Model	Method	Accuracy
Llama-2-70b	query	17 (3.46 %)
Llama-2-70b	query+contents	33 (6.73 %)
Llama-2-70b-chat	query	81 (16.53 %)
Llama-2-70b-chat	query+contents (plain RAG)	128 (26.12 %)
Llama-2-70b-chat	refined query+contents (ours)	**170 (34.69 %)**
Llama-2-70b-chat	ref. query+contents no iteration	127 (25.92 %)

Both these models were used in two evaluation scenarios. In one of these scenarios—noted as **query** in Table 3—, the response to the query entered was provided by the LLM alone, with no further content from external sources being entered. In the other scenario, in turn,—noted as **query+contents**—, further content from external sources was included in the prompt as well. The performance scores for these two scenarios with the two models are displayed in the top four rows of Table 3.

External, Non-parametric Content Improves Generation. As expected, and in line with results in the literature, the retrieval-augmented generation (26.12%) outperforms the plain generation based solely on the query (16.53%).

Fine-Tuning Improves Generation. Also as expected, and by a very large margin, better performance scores were obtained with Llama-2-70b-chat, which had been fine-tuned on dialogue tasks, namely 16.53% against 3.46%, with the query only, and 26.12% against 6.73%, with the query and external content.

Retrieved Content Refinement via Meta-prompting Optimization Improves RAG—The Proposed Method is Effective. The model Llama-2-70b-chat was thus retained and two further evaluation scenarios were considered.

A scenario with the application of the proposed method—noted as **refined query+contents**—, where the external content was refined with the help of an instruction optimized with meta-prompting.

The performance scores indicate that, with 34.69% accuracy, our proposed method of enhanced RAG outperforms plain RAG, with 26.12%, thus contributing for a large improvement of over 8.5% points, that represents here an improvement rate of almost 33%.

Retrieved Content Refinement via "Brute Force" Optimization Does Not Improve RAG. A sixth scenario—noted as **refined query+contents no iter**—was also considered. Here the external content was refined as in the proposed method, but the instruction was refined under an alternative way that dispensed with iterative meta-prompting.

All in all, 300 tentative instructions are generated during all optimization steps—recall that we had 100 iteration steps with 3 tentative instructions generated per step with meta-prompting optimization. To dispense with this iterative meta-prompting, the same number of 300 new tentative instructions were generated at once, in a "brute force" fashion. By the end of this process, all tentative instructions were scored with the same scoring function as in the proposed method, and the top instruction was evaluated on the test set.

This "brute force" optimization approach, scoring 25.92%, is outperformed not only by the proposed method of meta-prompting optimization, with 34.69%, but even also by the baseline, plain RAG, with 26.12%.

Statistical Significance. To assess the statistical significance of the improvements by our method, we employed the unpaired t-test.[3] We evaluated the baseline system with three seeds and did the same for the meta-prompting optimized system. Both samples are independent and one may assume the samples are normally distributed. Applying the unpaired t-test, a two-tailed P value equal to 0.0004 was obtained, which is considered statistically significant.

5.2 Examining the Tentative Instructions

Table 4 presents the top generated prompts. The best scoring prompt (last row), with 5.5 (out of a maximum of 6), was obtained at iteration step 46 (out of 300 steps in total), and a good prompt (first row) can be obtained with only 28 steps.

When reading the best prompt (last row), one realizes that it aims to improve the task through the summarization of the retrieved contents, considering their broader context, and identifying the main theme or message. It appears thus like a reasonable prompt a human might have thoughtfully arrived at if aiming at improving the performance of the task.

Table 4. Top meta-prompting optimized instructions scoring 5 or higher, with respective scores and iteration steps at which they were obtained, ordered top to bottom, with the top-scoring, retained instruction in the last row.

Generated instruction	Score	Iter. step
Summarize the previous text in 2-3 sentences, while also considering the broader context, the author's intent, the potential implications of the information, and also identify the main theme or message and its significance, and also analyze the impact of the information on the reader.	5	65
Summarize the previous text in 1 sentence, while also considering the broader context, the author's intent, the potential implications of the information, and also identify the main theme or message and its significance, and also analyze the impact of the information on the reader, and also provide recommendations for further	5	72
Summarize the previous text in 2-3 sentences, while also considering the broader context, the author's intent and the potential implications of the information, and also identify the main theme or message.	**5.5**	46

[3] The unpaired t-test evaluates if there exists a statistically significant distinction between the means of two independent samples by comparing them.

It is reasonable to assume that the meta-prompt iteration in subsequent steps used this query and its score in its search for further tentative instructions, with the generated instructions in three subsequent steps (58, 65 and 72) being some derivation of it (second, fourth and fifth rows).

When taking a look at the entire set of generated queries, a high fluctuation of the evaluation scores can be observed along the iteration steps. This is likely due to some interim, generated instructions happening to perform poorly.

5.3 Examining the Responses

To gain insight about where our method outperformed the plain RAG baseline, we examined the first 10 instances where our method correctly provided the answer while the baseline failed. Among these, in six cases, the baseline provided a verbose response and might have failed the exact-match evaluation criterion used.[4] In the remaining four cases, the baseline either answered incorrectly, responded with a query, or failed to provide an answer.

Conversely, we reviewed the first 10 instances where our method failed to provide the correct answer while the baseline succeeded. We observed that our method exhibited a verbose response behavior in five cases that eventually arrived at the correct answer but failed the exact-match evaluation criterion. In two other cases, our method gave a verbose response without providing an answer, while in two remainder cases, it provided an incorrect response. Finally, in one instance, our method did not provide any response.

Both methods seem thus to be similarly penalized by the evaluation criterion for not providing straight answers when the correct answer may happen to be included in the verbose response.

6 Conclusion

While providing a method that effectively enhances RAG, our proposal paves the way for future research, such as the exploration of optimal hyper-parameters, refining content retrieved without gold content, scaling up with larger models, exploring further evaluation functions, or tackling other downstream tasks.

It is worth noting that the evaluation with exact matching is a binary task, and achieving an exact match with a task demanding a more complex string match still needs to be studied, questioning the need for additional training, a different meta-prompt, or a different approach.

It will be interesting also to study the interaction of our proposed method with the Portuguese language [2,20] with the existing family of LLMs [24,29,30] and multi-modal LLMs [27] as also with other tasks such as argument mining [25], exploring data spuriousness and others [3,4,23,28].

[4] An example of a verbose response: [query] Was Superhero fiction invented in the digital format? [response] The answer is no; superhero fiction did not originate in digital format. Superheroes have their roots in pulp magazines, comic strips, and comic books, which were all print media formats before the advent of digital technology.

Acknowledgements. This research was partially supported by: PORTULAN CLARIN - Research Infrastructure for the Science and Technology of Language, funded by Lisboa 2020, Alentejo 2020 and FCT (PINFRA/22117/2016); ACCELERAT.AI - Multilingual Intelligent Contact Centers, funded by IAPMEI (C625734525-00462629);

References

1. Aarohi Srivastava, A.R., et al.: Quantifying and extrapolating the capabilities of language models. Trans. Mach. Learn. Res. (2023)
2. Branco, A., Mendes, A., Quaresma, P., et al.: Infrastructure for the science and technology of language PORTULAN CLARIN. In: Proceedings of the 1st International Workshop on Language Technology Platforms (2020)
3. Branco, A., Rodrigues, J., Salawa, M., et al.: Comparative probing of lexical semantics theories for cognitive plausibility and technological usefulness. In: Proceedings of the 28th COLING (2020)
4. Branco, R., Branco, A., António Rodrigues, J., Silva, J.R.: Shortcutted commonsense: Data spuriousness in deep learning of commonsense reasoning. In: Proceedings of the 2021 Conference on EMNLP (2021)
5. Chen, Y., Arkin, J., Hao, Y., Zhang, Y., Roy, N., Fan, C.: Prompt optimization in multi-step tasks (PROMST): integrating human feedback and preference alignment. arXiv preprint arXiv:2402.08702 (2024)
6. Geva, M., Khashabi, D., Segal, E., Khot, T., Roth, D., Berant, J.: Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. Trans. ACL **9**, 346–361 (2021)
7. Guo, Q., et al.: Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. arXiv preprint arXiv:2309.08532 (2023)
8. Ho, X., Duong Nguyen, A.K., Sugawara, S., Aizawa, A.: Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In: Proceedings of the 28th COLING. International Committee on Computational Linguistics, December 2020
9. Houlsby, N., et al.: Parameter-efficient transfer learning for NLP. In: International Conference on Machine Learning, pp. 2790–2799. PMLR (2019)
10. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the ACL, pp. 328–339 (2018)
11. Inoue, N., Stenetorp, P., Inui, K.: R4C: a benchmark for evaluating RC systems to get the right answer for the right reason. In: Proceedings of the 58th Annual Meeting of the ACL, pp. 6740–6750 (2020)
12. Ji, Z., Lee, N., Frieske, R., Yu, T., et al.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12), 1–38 (2023)
13. Kong, W., Amba Hombaiah, S., Zhang, M., Mei, Q., Bendersky, M.: Prewrite: prompt rewriting with reinforcement learning. arXiv e-prints pp. arXiv–2401 (2024)
14. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on EMNLP, pp. 3045–3059 (2021)
15. Lewis, P., Perez, E., Piktus, A., Petroni, F., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv. NIPS **33**, 9459–9474 (2020)
16. Li, X.L., Liang, P.: Prefix-tuning: optimizing continuous prompts for generation. In: Proceedings of the 59th Annual Meeting of the ACL, pp. 4582–4597 (2021)

17. Lin, S., Hilton, J., Evans, O.: TruthfulQA: measuring how models mimic human falsehoods. In: Proceedings of the 60th Annual Meeting of the ACL, pp. 3214–3252 (2022)
18. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput. Surv. **55**(9), 1–35 (2023)
19. Min, S., Wallace, E., Singh, S., Gardner, M., Hajishirzi, H., Zettlemoyer, L.: Compositional questions do not necessitate multi-hop reasoning. In: Proceedings of the 57th Annual Meeting of the ACL (2019)
20. Osório, T.F., et al.: PORTULAN ExtraGLUE datasets and models: Kick-starting a benchmark for the neural processing of Portuguese. In: Proceedings of the 17th Workshop on Building and Using Comparable Corpora (BUCC) (2024)
21. Prasad, A., Hase, P., Zhou, X., Bansal, M.: Grips: gradient-free, edit-based instruction search for prompting large language models. In: Proceedings of the 17th Conference of the EACL, pp. 3845–3864 (2023)
22. Raffel, C.: Shazeer: exploring the limits of transfer learning with a unified text-to-text transformer. Mach. Learn. Res. **21**, 5485–5551 (2020)
23. Rodrigues, J., Branco, R., Silva, J., Saedi, C., Branco, A.: Predicting brain activation with wordnet embeddings. In: Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing (2018)
24. Rodrigues, J., et al.: Advancing neural encoding of Portuguese with transformer Albertina PT. In: EPIA Conference on Artificial Intelligence (2023)
25. Rodrigues, J.A., Branco, A.: Transferring confluent knowledge to argument mining. In: Proceedings of the 29th COLING (2022)
26. Sabbatella, A., Ponti, A., Giordani, I., Candelieri, A., Archetti, F.: Prompt optimization in large language models. Mathematics **12**(6) (2024)
27. Santos, R., Branco, A., Silva, J.R.: Cost-effective language driven image editing with LX-DRIM. In: Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models (2022)
28. Santos, R., Rodrigues, J., Branco, A., Vaz, R.: Neural text categorization with transformers for learning Portuguese as a second language. In: Proceedings of the 20th EPIA (2021)
29. Santos, R., et al.: Fostering the ecosystem of open neural encoders for Portuguese with Albertina PT* family. In: Proceedings of the 3rd Meeting of the Special Interest Group on Under-resourced Languages (2024)
30. Santos, R., Silva, J.R., Gomes, L., Rodrigues, J., Branco, A.: Advancing generative AI for Portuguese with open decoder Gervásio PT*. In: Proceedings of the 3rd Meeting of the Special Interest Group on Under-resourced Languages (2024)
31. Touvron, H., Martin, L., Stone, K., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
32. Vaswani, A., et al.: Attention is all you need. Adv. NIPS **30** (2017)
33. Wang, X., et al.: PromptAgent: strategic planning with language models enables expert-level prompt optimization. arXiv preprint arXiv:2310.16427 (2023)
34. Wei, J., Wang, X., Schuurmans, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. NIPS **35** (2022)
35. Yang, C., et al.: Large language models as optimizers. arXiv preprint arXiv:2309.03409 (2023)
36. Yao, S., Yu, D., Zhao, J., et al.: Tree of thoughts: deliberate problem solving with large language models. Adv. NIPS **36** (2024)
37. Zhong, Z., Friedman, D., Chen, D.: Factual probing is [mask]: learning vs. learning to recall. In: Proceedings of the 2021 Conference of the NACL (2021)