



Universiteit
Leiden
The Netherlands

Improving RAG for personalization with author features and contrastive examples

Yazan, M.; Verberne, S.; Situmeang, F.; Hauff, C.; Macdonald, C.; Jannach, D.; ... ; Tonello, N.

Citation

Yazan, M., Verberne, S., & Situmeang, F. (2025). Improving RAG for personalization with author features and contrastive examples. *Lecture Notes In Computer Science*, 408-416.
doi:10.1007/978-3-031-88714-7_40

Version: Publisher's Version

License: [Licensed under Article 25fa Copyright Act/Law \(Amendment Taverne\)](#)

Downloaded from: <https://hdl.handle.net/1887/4295590>

Note: To cite this publication please use the final published version (if applicable).



Improving RAG for Personalization with Author Features and Contrastive Examples

Mert Yazan^{1,2} , Suzan Verberne² , and Frederik Situmeang¹ 

¹ Amsterdam University of Applied Sciences, Fraijlemaborg 133, 1102 CV Amsterdam, Netherlands

`{m.yazan,f.b.i.situmeang}@hva.nl`

² Leiden University, Einsteinweg 55, 2333 CC Leiden, Netherlands

`{m.yazan,s.verberne}@liacs.leidenuniv.nl`

Abstract. Personalization with retrieval-augmented generation (RAG) often fails to capture fine-grained features of authors, making it hard to identify their unique traits. To enrich the RAG context, we propose providing Large Language Models (LLMs) with author-specific features, such as average sentiment polarity and frequently used words, in addition to past samples from the author's profile. We introduce a new feature called *Contrastive Examples*: documents from other authors are retrieved to help LLM identify what makes an author's style unique in comparison to others. Our experiments show that adding a couple of sentences about the named entities, dependency patterns, and words a person uses frequently significantly improves personalized text generation. Combining features with contrastive examples boosts the performance further, achieving a relative 15% improvement over baseline RAG while outperforming the benchmarks. Our results show the value of fine-grained features for better personalization, while opening a new research dimension for including contrastive examples as a complement with RAG. We release our code publicly (<https://github.com/myazann/AP-Bots/tree/main>).

Keywords: Personalization · Retrieval Augmented Generation · Personalized Text Generation · Contrastive Examples

1 Introduction

Personalized text generation has many application areas, from recommendation [25] to personalized content creation [1, 15]. Its implementation with LLMs falls into two groups: via fine-tuning [18, 19, 27], or via Retrieval-Augmented Generation (RAG) [13, 15, 24]. In fine-tuning-based approaches, the language model is individually adapted for each user [19], which remains computationally costly even with parameter-efficient fine-tuning (PEFT) [7]. RAG-based personalization works by retrieving samples from the author profile, then instructing the LLM to generate personalized content given the samples [15]. RAG eliminates

the need to fine-tune for each author and outperforms PEFT-based methods when the amount of user data is scarce [16, 26].

The key challenge with RAG is to capture fine-grained author features: retrieving separate, standalone samples from the author’s history is not enough to recognize global patterns [18]. When presented with previous documents of the author, LLMs tend to memorize certain words and n-grams that are frequently used, instead of understanding the author’s style [26]. To address this problem, we derive features from the author’s profile. Features enhance the traditional RAG approach by providing global context about the author, nudging the LLM to do more than repeating words and n-grams. We experiment with different types of features, such as the average sentiment polarity, or the words and named entities the person uses the most. In addition, we introduce a new feature called *Contrastive Examples (CE)* which retrieves samples from other authors to better highlight the differences between individual styles. Figure 1 gives an overview of our framework.

Our results show that adding only a handful of author features alongside samples from the author’s profile significantly improves personalization. Furthermore, retrieving contrastive examples helps LLM better understand an author’s style. A combination of *CE* and a couple of author features improve the RAG performance by up to **15%**. Additionally, our approach does not add any computational overhead and can easily be implemented and deployed. We contribute to the literature by showing that: (a) LLMs can personalize better when presented with author features, (b) contrastive examples helps LLMs distinguish authors, and (c) retrieving contrastive examples can be a tool to differentiate between relevant and irrelevant contexts.

2 Background and Related Work

RAG for personalization is initially implemented by retrieving similar documents from the user’s profile, and instructing the LLM to personalize given the documents [15]. Later work focuses on different parts of the RAG pipeline, such as optimizing the retriever [13, 14] or summarizing and synthesizing the documents to improve the reading comprehension of the LLM [10]. Alhafni et al. [1] experiment with fine-grained features, such as lexical and syntactic attributes. The crucial difference with our method is that they train smaller models ($\leq 1\text{B}$) by prepending the attribute vectors to the inputs, instead of RAG. In addition, they do not include features such as the most commonly used words of the author.

Even though personalization by its nature is user-specific, finetuning with multiple users before personalizing to one can aid in learning shared features [18, 27]. Tan et al. [18] introduce Personalized Pieces, where clusters of users share PEFT parameters, and HYDRA framework proposes an approach to obtain the shared general knowledge between all authors [27]. What makes our approach unique is we focus on the differences between authors by retrieving contrastive examples, whereas previous work focused on capturing the similarities.

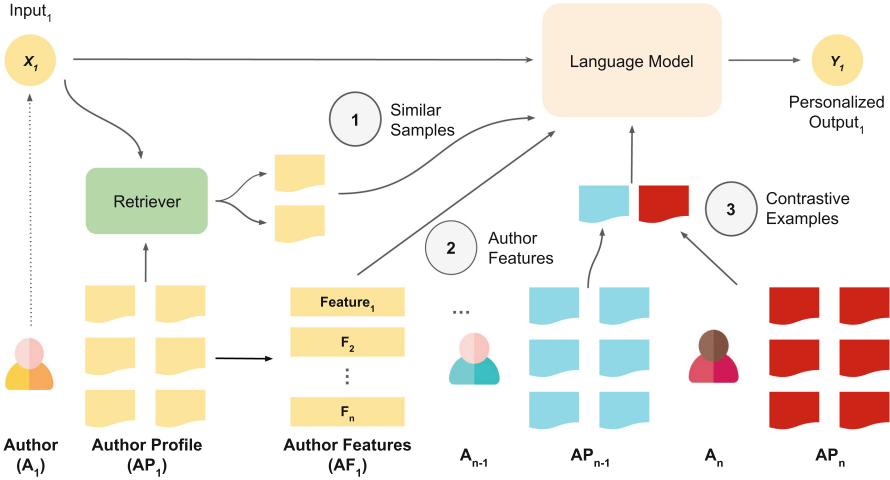


Fig. 1. An overview of our approach. The language model receives three inputs: (1) most similar samples from the author profile, given the author’s input, (2) features derived from the author profile, (3) contrastive examples gathered from other users. Input (1) denotes the baseline RAG approach, while (2) and (3) are our additions to enrich RAG context.

The novelty of our idea is not limited to personalized text generation, but can be seen as a new paradigm for retrieval in RAG. Traditionally, RAG is based on retrieving the most relevant documents, and how contrastive examples would work is an open question. A contrastive chain-of-thought approach with both valid and invalid demonstrations performs better than regular chain-of-thought prompts [3] and shows that LLMs are capable of making use of contrastive examples. In the context of RAG, retrieving random documents can help with question-answering, while high-scored documents that are not directly relevant hurts the performance [4]. Since contrastive examples do not fit into either of these categories, their impact on RAG remains unclear. Contrastive learning has previously been studied in Information Retrieval (IR) for training retrievers to differentiate between relevant and irrelevant documents [9]. In contrast, our method incorporates both types of documents directly into the LLM’s prompt, without modifying the retriever.

3 Methodology

Data. We used 3 datasets offered by the LaMP (When Large Language Models Meet Personalization) benchmark [15]: LaMP-4, LaMP-5, and LaMP-7. Since we are focusing on text generation, we excluded the datasets with classification tasks, as well as LaMP-6 because it wasn’t accessible. We chose the user-based splits for evaluation. The task in LaMP-4 is to generate a title in the style of the news editor based on a news article given as the input. The author profile

consists of previous article-title pairs. The task in LaMP-5 is to generate a title, given the abstract as the input, and the author profile consists of the previous abstract-title pairs of the scholar. Finally, in LaMP-7, the input is a tweet and the task is to paraphrase it given the author’s past tweets.

Models. Given its success in the LaMP benchmark [15,26], we used a pre-trained Contriever [8] as the retriever. The average length of the samples differs between datasets, where LaMP-5U has longer documents [15]. Therefore, we retrieved 50 documents from author profiles for LaMP-4 and 7, and 7 documents for LaMP-5. Previous research demonstrated that it is not necessary to use that many documents [24,26], but we see that it is possible to maximize the RAG performance by using more, allowing us to better understand the contribution of our features.

We started our experiments with OpenChat [22], given its success on LaMP [26]. Then we experimented with LLaMA-3.1 [5] and Gemma-2 [20]. We didn’t find drastic variations between models, but Gemma-2 27B performed the best overall. Because of space limitations, we only present its results in Sect. 4. We didn’t change the default generation parameters on HuggingFace, except for the *temperature* and *max_new_tokens* (maximum number of generation tokens). Given the shortness of the expected generations, we set *max_new_tokens* to 128 tokens, and *temperature* is set to 0.7. We noticed negligible changes in the final result with different generation parameters, so we decided to use the default ones.

We constructed the prompts by first giving a role to the LLM (*you are a news editor*), and explaining the types of inputs it will receive. We put similar documents from the author’s past as the first input. Then we include features and contrastive examples (see Fig. 1). Similar past documents are always included as input, since they form the baseline RAG. Finally, we instruct the model to generate the proper text (e.g. title) given the features it received. The prompt structure is consistent for each dataset, deviating only on the role and the instruction. We didn’t see an improvement using Chain-of-Thought [23] or similar variants, while noticing an exponential increase in inference times. Therefore, we didn’t use such prompting methods. Prompts are available in our public repository.

Features. Our features can be categorized into two: (1) *Author Features*, derived from the author’s profile and (2) *Contrastive Examples*, which are samples coming from other authors.

Author Features: We define the following features:

- **SP:** Average sentiment polarity ($SP \in [-1, 1]$)
- **SUBJ:** Average subjectivity ($SUBJ \in [0, 1]$)
- **SMOG:** SMOG Index¹

¹ SMOG is a measure of readability and typically has a value between 1–12, estimating the years of education needed to understand a piece of writing.

- **ADVU**: Adverb usage percentage on average ($ADVU \in [0, 100]$)
- **ADJU**: Adjective usage percentage on average ($ADJU \in [0, 100]$)
- **PU**: Pronoun usage percentage on average ($PU \in [0, 100]$)
- **NEF**: Most frequently used named entities ($[ne_0, \dots, ne_9]$)
- **DPF**: Most frequently used dependency patterns ($[dp_0, \dots, dp_9]$)
- **WF**: Most frequently used words ($[w_0, \dots, w_9]$)

We incorporated the features into the prompt by converting their definition into the following sentence format: $\{feat_def\}$ for the writer is $\{feat_value\}$. We use TextBlob [11] to obtain the subjectivity and sentiment polarity scores, and textstat [21] for the SMOG index. We use NLTK [2] for calculating *ADVU*, *ADJU*, *PU* and *WF*. Finally, we use spaCy [6] to get named entities and dependency patterns. The averages are calculated by taking the mean of the feature values for all the samples in the author profile. We choose the top 10 most frequent elements for frequency features, since our experiments showed that more than 10 does not change the performance.

Contrastive Examples (CE): Contrastive examples can be considered hard negatives, as we choose authors that are the least similar to the current author. We calculate the distances between authors using the inputs (see the Data paragraph for the inputs of each dataset) with Contriever. Alternatively, we tried looking at the similarity between author profiles based on the average embeddings. We didn’t find a significant difference, therefore we proceeded with using the inputs. After the least similar authors are identified, samples from their profiles are chosen randomly. The number of samples coming from other authors were fixed: 1 sample per each author for LaMP-5, and 3 samples for LaMP-4 and LaMP-7. We use fewer samples compared to how many we retrieve from the current author’s profile because of context window limitations and saturated performance with fewer samples. In LaMP-5, using too many contrastive examples even hurts the performance. We suspect the length of the samples in that dataset causes noise and decreases the LLMs’ attention to the different parts of the context.

4 Results

Our baseline is a RAG experiment without any author features. Then, we test each feature individually on validation sets to understand how much they contribute to the baseline. Figure 2 shows that some features improve the baseline for every dataset, while others hurt. *CE* emerges as a robust feature since it offers at least a 1% increase regardless of the dataset. Retrieving samples from 5 different authors performs worse than 3, possibly due to increased noise. LaMP-7 benefits the most from the features, *DPF* itself can increase Rouge-L by more than 10%. Since LaMP-7 is about tweets, which are highly personal and informal, dependency patterns better highlight individual’s styles compared to others. Compared to tweets, news articles are more formal and contain domain-specific keywords. Therefore, *WF* becomes a more suitable feature for LaMP-4 instead of *DPF*.

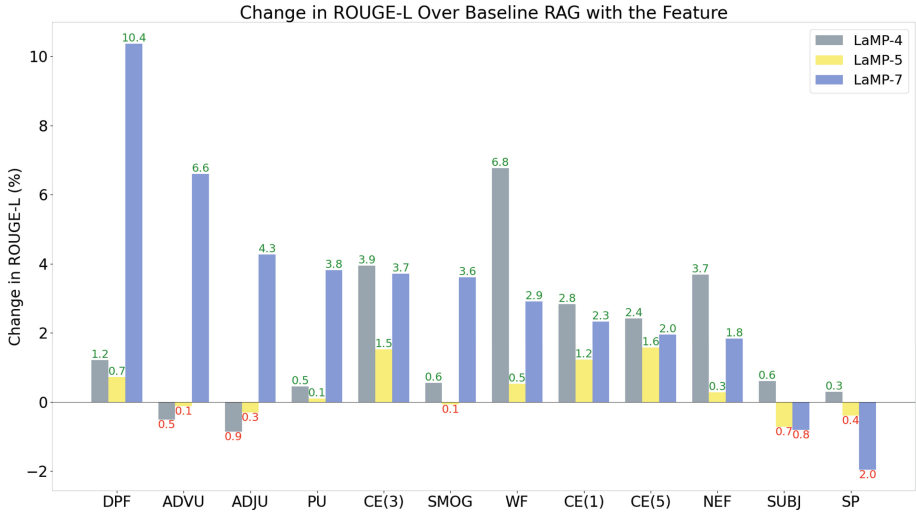


Fig. 2. The improvement each feature provides for Rouge-L on validation sets when included on top of baseline RAG. In this case, features are not combined but used individually. A negative change signifies the feature hurting the baseline. Values inside parentheses show the number of contrastive users.

Performance gains in LaMP-5 are more limited. Authors in LaMP-5 represent scholars, each working in a very distinct and highly specific academic domain. Also, documents in LaMP-5 are much longer than other datasets. Therefore, retrieved samples contain more information compared to other datasets, making it hard for author features to contribute. In comparison, contrastive examples highlight the vast differences between specialized academic domains, becoming the best performing feature in LaMP-5.

Overall, features presented as numerical values (*SP*, *SUBJ*, *ADVU*, *ADJU*, *PU*) are less effective or even hurtful compared to ones presented as words (*CE*, *WF*, *NEF*, *DPF*). This may be due to the LLMs shortcoming of working with numerical values [17]. In our qualitative analysis, we noticed LLMs overly polarizing (e.g. too negative or too positive) their outputs in the presence of numerical features. They especially have a hard time adjusting to subjectivity and sentiment polarity, not being able to capture the nuances of scores that are closer to the mean. In fairness, it is hard to conceptualize the difference between a sentiment score of 0.29 and 0.4, especially given the personal differences in expression.

Next, we evaluate the effect of combining features. We chose the best combinations from the validation set results, and Table 1 shows their performance on test sets (for brevity, we present only the combinations that work best for

Table 1. Our best feature combinations compared to the current leaderboard for user-based test splits. * The method [12] is based on finetuning a FlanT5-XXL with PEFT.

	LaMP-4		LaMP-5		LaMP-7	
	Rouge-1	Rouge-L	Rouge-1	Rouge-L	Rouge-1	Rouge-L
Finetuned FlanT5-base [15]	0.186	0.171	0.450	0.409	0.528	0.475
Learning to Prompt* [12]	0.221	0.202	0.472	0.432	–	–
RAG	0.214	0.196	0.475	0.425	0.437	0.380
RAG + (WF, CE(3))	0.228	0.210	0.481	0.431	0.483	0.431
RAG + (DPF, CE(3))	0.226	0.207	0.483	0.432	0.496	0.437
RAG + (WF, DPF, CE(3))	0.221	0.202	0.481	0.430	0.501	0.441

each dataset). Combining features significantly² improves the performance. *CE* emerges as a great complementary feature, as it is present in all the best performing combinations. Compared with the leaderboard³, our approach surpasses the top methods in LaMP-4 and LaMP-5. Since the other methods are finetuning-based, we can say that RAG-based personalization combined with our features can eliminate the need to finetune for each author. Contrarily, finetuning comes out on top in LaMP-7. Still, our method reduces the gap between fine-tuning and RAG by boosting the baseline RAG performance by **15%** relatively (from 0.437 to 0.501). Since LaMP-7 benefitted the most from the features, we believe there is still room for improvement to improve the performance with more features.

5 Conclusion

Our research pushes the boundaries of personalization by enhancing the retrieval component of RAG with fine-grained author features and contrastive examples. We show that a single sentence providing which words an author utilizes the most combined with contrastive examples can improve the RAG performance by **15%**. Our research offers a new paradigm for RAG, where contrastive examples complement the retrieved samples and help LLM better identify the ground truth. For future work, more suitable ways to present numerical features can be explored. More advanced methods to mitigate the limitations of presenting numerical features to LLMs might be a promising direction for exploration. How to retrieve contrastive examples is an open question for the IR community, as well as extending the contrastive retrieval approach with other IR tasks.

² Paired t-tests between baseline RAG and the best performing methods with $\alpha = 0.05$ confirmed the statistical significance: LaMP-4: ($p = 2.7 \times 10^{-11}$), LaMP-5: ($p = 0.0014$), LaMP-7: ($p = 9.6 \times 10^{-32}$). Since test set predictions are not accessible, t-tests are conducted on the validation sets.

³ <https://lamp-benchmark.github.io/leaderboard>.

References

1. Alhafni, B., Kulkarni, V., Kumar, D., Raheja, V.: Personalized text generation with fine-grained linguistic control. In: Deshpande, A., Hwang, E., Murahari, V., Park, J.S., Yang, D., Sabharwal, A., Narasimhan, K., Kalyan, A. (eds.) *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, pp. 88–101. Association for Computational Linguistics, St. Julians, Malta (2024). <https://aclanthology.org/2024.personalize-1.8>
2. Bird, S., Klein, E., Loper, E.: *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc. (2009)
3. Chia, Y.K., Chen, G., Luu, A.T., Poria, S., Bing, L.: Contrastive chain-of-thought prompting. *ArXiv abs/2311.09277* (2023). <https://api.semanticscholar.org/CorpusID:265221368>
4. Cuconasu, F., et al.: The power of noise: redefining retrieval for rag systems. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 719–729. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3626772.3657834>
5. Dubey, A., Jauhri, A., Pandey, A., et al.: The llama 3 herd of models (2024). <https://arxiv.org/abs/2407.21783>
6. Honnibal, M., Montani, I.: *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing* (2017). to appear
7. Houlisby, N., et al.: Parameter-efficient transfer learning for NLP. *ArXiv abs/1902.00751* (2019). <https://api.semanticscholar.org/CorpusID:59599816>
8. Izacard, G., et al.: Unsupervised dense information retrieval with contrastive learning (2022). <https://arxiv.org/abs/2112.09118>
9. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.emnlp-main.550>, <https://aclanthology.org/2020.emnlp-main.550>
10. Li, C., et al.: Teach LLMs to personalize – an approach inspired by writing education (2023). <https://arxiv.org/abs/2308.07968>
11. Loria, S.: <https://textblob.readthedocs.io>
12. Mineiro, P.: <https://github.com/pmineiro/lampstuff>
13. Mysore, S., Lu, Z., et al.: Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. *ArXiv abs/2311.09180* (2023). <https://api.semanticscholar.org/CorpusID:265213422>
14. Salemi, A., Kallumadi, S., Zamani, H.: Optimization methods for personalizing large language models through retrieval augmentation. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 752–762. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3626772.3657783>
15. Salemi, A., Mysore, S., Bendersky, M., Zamani, H.: LaMP: when large language models meet personalization. In: Ku, L.W., Martins, A., Srikumar, V. (eds.) *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392. Association for Computational Linguistics, Bangkok, Thailand (2024). <https://doi.org/10.18653/v1/2024.acl-long.399>

16. Salemi, A., Zamani, H.: Comparing retrieval-augmentation and parameter-efficient fine-tuning for privacy-preserving personalization of large language models. arXiv preprint [arXiv:2409.09510](https://arxiv.org/abs/2409.09510) (2024)
17. Schwartz, E., Choshen, L., Shtok, J., Doveh, S., Karlinsky, L., Arbel, A.: Numero-logic: Number encoding for enhanced LLMs' numerical reasoning (2024). <https://arxiv.org/abs/2404.00459>
18. Tan, Z., Liu, Z., Jiang, M.: Personalized pieces: Efficient personalized large language models through collaborative efforts (2024). <https://arxiv.org/abs/2406.10471>
19. Tan, Z., Zeng, Q., Tian, Y., Liu, Z., Yin, B., Jiang, M.: Democratizing large language models via personalized parameter-efficient fine-tuning (2024). <https://arxiv.org/abs/2402.04401>
20. Team, G., Riviere, M., Pathak, S., et al.: Gemma 2: Improving open language models at a practical size (2024). <https://arxiv.org/abs/2408.00118>
21. Textstat: <https://textstat.org>
22. Wang, G., Cheng, S., Zhan, X., Li, X., Song, S., Liu, Y.: OpenChat: Advancing open-source language models with mixed-quality data. ArXiv abs/2309.11235 (2023). <https://api.semanticscholar.org/CorpusID:262064307>
23. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2024)
24. Wu, B., Shi, Z., Rahmani, H.A., Ramineni, V., Yilmaz, E.: Understanding the role of user profile in the personalization of large language models (2024). <https://arxiv.org/abs/2406.17803>
25. Wu, L., et al.: A survey on large language models for recommendation. World Wide Web **27**(5) (2024). <https://doi.org/10.1007/s11280-024-01291-2>
26. Yazan, M., Verberne, S., Situmeang, F.: The impact of quantization on retrieval-augmented generation: An analysis of small LLMs (2024). <https://arxiv.org/abs/2406.10251>
27. Zhuang, Y., et al.: HYDRA: Model factorization framework for black-box LLM personalization (2024). <https://arxiv.org/abs/2406.02888>