MINERVA UNIVERSITY


CAPSTONE: CLASS OF 2025

# LINKS: Generate linguistically grounded mnemonic devices for English vocabulary learning with reasoning, multilingual LLMs

TRA MY NGUYEN

submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computational Sciences

Capstone Committee
Dr. Patrick Watson
Dr. Philip Sterne

April 8, 2025

## Executive Summary

Note: Due to limited compute, some experiments conducted are small-scale and need more data for robust validation and conclusion. However, the codebase is reproducible and scalable when there is more compute.

Tags: computational linguistics, natural language processing, large language model, language education, english as a foreign language, vocabulary acquisition, synthetic data generation.

## Contents

# LINKS: Generate linguistically grounded mnemonic devices for English vocabulary learning with reasoning, multilingual LLMs

**My (Chiffon) Nguyen**
College of Computational Sciences
Minerva University
chiffonng@uni.minerva.edu

## Abstract

To acquire advanced vocabulary, English learners often use mnemonic devices, memorable associations linking a new concept to learned concepts to improve memory and recall. Reviewing the literature on mnemonic techniques, we characterize good mnemonics as **linguistically grounded**, which better link to the target vocabulary, improving long-term retention and linguistic knowledge, especially at advanced levels (CEFR B2+). We investigate whether Large Language Models can consistently help write such effective mnemonics, with three different settings: in-context learning, and reasoning distillation. Concretely, we first measured different prompting strategies with a frontier reasoning model, DeepSeek-R1 , and generated **LINKS**, a synthetic dataset of 2000 triplets of *reasoning trace, mnemonic, and example sentence* for 2000 vocabulary useful for TOEFL iBT [1], IELTS Academic [2], and SAT[3]. Second, using a subset of **LINKS**, we distilled linguistic reasoning from the *teacher model* to the *student model*, Gemma-3-1b-it [4] , with online reinforcement learning. The trained, quantized model can be served with a local application such as OpenWebUI (interface) and Ollama (command-line).

Preliminary evaluation shows

The project examplifies that carefully designed NLP systems can generate resources for language learning, either in classroom settings or in self-study.[5]

## 1 Introduction

Vocabulary acquisition challenges many English language learners, particularly at upper intermediate to advanced levels where abstract and academic vocabulary predominates. Mnemonics, cognitive tools that help learners create associations between new vocabulary and familiar concepts, serve as valuable tools for enhancing retention and recall. The deeper learners elaborate the connection between the mnemonic and the target vocabulary, the longer and better they can recall the term. However, creating such effective mnemonics demands both linguistic expertise and creative effort, presenting a significant barrier for most learners. Large Language Models (LLMs) have demonstrated capabilities as knowledge bases and creative text generators, suggesting their potential for automated mnemonic generation.

Previous work explored automated mnemonic generation through computational methods using the **keyword method**, which involves 1) generateing simpler keywords that together sound or look like the target vocabulary and 2) creating memorable explanations that include the vocabulary, the keywords, amd its meaning (Atkinson and Raugh, 1975). Savva et al. (2014) and Özbal et al. (2014) generated keywords of phonetic and orthographic similarities in the native language for foreign language vocabulary, across multiple languages. Lee and Lan (2023) extended this work and utilized LLMs to produce phonetically similar keywords and visual cues and Lee et al. (2024) prompted LLMs to generate multiple mnemonic candidates and evaluate them based on imageability and coherence. Most recently, Balepur et al. (2024) fine-tuned LLaMA-2-70B on 800 crowd-sourced mnemonics and aligned outputs with learner preferences and learning outcomes.

Although the keyword method is commonly used and empirically validated in classroom and laboratory contexts (Atkinson and Raugh, 1975, Pressley et al., 1982), it may lead to longer retrieval time (van Hell and Mahn, 1997) and be inadequate for fairly abstract vocabulary (Campos et al., 2003, Campos et al., 2011). Such methods typically ne-

[1] Internet-based Test of English as a Foreign Language
[2] International English Language Testing System
[3] Scholastic Aptitude Test
[4] https://huggingface.co/collections/google/gemma-3-release-67c6c6f89c4f76621268bb6d
[5] https://github.com/chiffonng/mnemonic-gen

glect other rich linguistic knowledge embedded in LLMs that could provide diverse mnemonic strategies beyond simple keyword associations. Second, previous works passively deliver generated mnemonics to learners. While SMART (Balepur et al., 2024) was further trained on learners' preferences, these preferences were aggregated, potentially missing alignment with individual learning styles. Language learners who use self-created mnemonics retain vocabulary more effectively and for longer duration (Madan, 2021).

Our contributions can be summarized as follows. (**1**) We demonstrate that LLMs can generate **linguistically grounded mnemonics**, which emphasizes the importance of linguistic features in creating effective mnemonics, through reasoning and creative writing. (**2**) We present **LINKS**, a synthetic dataset of 2000 triplets of *reasoning trace, mnemonic, and example sentence* for 2000 vocabulary. They can be integrated in a spaced repetition system (SRS) or language learning applications for vocabulary acquisition with better retrieval. (**3**) We distill the reasoning capabilities of a frontier, reasoning LLM, into a smaller model, Gemma-3-1b-it , using (DeepSeek-AI et al., 2025). The trained model **LINKS**, can be served locally, enabling users to generate mnemonics without relying on external APIs or internet connectivity.

## 2 Background

We assume a background on LLMs, including their transformer-based architecture (Section C.2), in-context learning (Section 3), and reinforcement learning (full preliminaries are provided in Section C). We briefly review the literature on mnemonic devices for vocabulary learning and the use of LLMs in linguistic tasks.

### 2.1 Mnemonic devices for vocabulary learning

### 2.2 LLMs: linguistic competence and creativity

## 3 In-context learning performance

### 3.1 Experimental setup

We systematically compared various in-context learning approaches to understand how different prompting techniques affect mnemonic generation. **??** illustrates the percentage of linguistically grounded mnemonics generated by different prompt formulations.

We used `curator` (Bespoke Lab, 2025) with `litellm` orchestration layer to interact with LLM APIs, simpify API calls, manage rate limits, and handle retries.

### 3.2 Results

We observed significant variation in the quality and linguistic grounding of generated mnemonics based solely on prompt formulation. Four distinct prompting strategies were evaluated (see details in section A) Vanilla Reasoning LLMs tend to overthink (Xu et al., 2025) Good practices: provide decomposed instructions, structured output format, demonstration examples (Mishra et al., 2022), and clarify definitions of linguistic features (Yin et al., 2023)

compress task definition (Yin et al., 2023),

## 4 Knowledge and reasoning distillation

We present **LINKS**, a pipeline that distills linguistic knowledge and reasoning through a teacher-student framework. This approach generates linguistically grounded mnemonics with reasoning traces and exemplifying sentences for vocabulary learning.

### 4.1 Data construction

To generate high-quality linguistically grounded mnemonics, we first created a comprehensive training dataset. Following best practices in synthetic data generation with LLMs (Long et al., 2024, Open Thoughts Team, 2025), we designed a data construction pipeline with several key components.

**Vocabulary collection** We collected 5,000 distinct vocabulary words from four complementary sources: English as a foreign language tests (TOEFL iBT, IELTS Academic), standardized tests (SAT, GRE), CEFR levels C1 and C2 word lists, and the Oxford Dictionary of Philosophy. We selected these sources to ensure coverage of academic and abstract vocabulary that would benefit from mnemonic devices. After deduplication and fuzzy-matching decontamination, we refined our dataset to 2,000 distinct vocabulary words for post-training.

**Prompt design** Based on the findings from section 3, we crafted system and user prompts that encouraged linguistically grounded reasoning. Our system prompt instructed the model to analyze potential linguistic features before generating a mnemonic. We used structured output format with

Table 1: Examples of feature categories for English words.

| feature | description | example |
|---|---|---|
| **phonetics** | vocab's sound patterns | *apparent* sounds like "a bare Asian." |
| **orthography** | written/spelling patterns | *abet* looks like "a + bet." |
| **morphology** | structure including free and bound morphemes | *aggrandize* = a + grand + –ize, to mean to make grander. |
| **etymology** | vocab origin and history | *adumbrate* comes from Latin ad- (to, on) + umbra (shade) + ate, to mean foreshadow or outline. |
| **semantics** | vocab meaning and relationships | *confound* has similar meaning and history with 'confuse'. |

designated sections for reasoning, mnemonic, and example, allowing for clearer evaluation and potential future extraction of specific components.

**Teacher model selection** We selected DeepSeek-R1 (670B parameters) (DeepSeek-AI et al., 2025) for its advanced reasoning capabilities and extensive knowledge as the teacher model.

**Dataset generation** Using the designed prompts and vocabulary list, we generated the **LINKS**, a dataset of approximately 2,000 entries, each containing: (1) a detailed reasoning trace exploring multiple linguistic features of the target vocabulary, (2) a concise mnemonic device leveraging the most salient linguistic connection, and, (3) an exemplifying sentence demonstrating proper usage.

**Quality control** To ensure the quality of the generated mnemonics, we implemented a multi-step validation process. We first filtered out any entries that did not meet our structured output format or contained incomplete reasoning traces. We then performed a manual review of a random sample of 200 entries to assess the linguistic grounding and coherence of the mnemonics. This review process involved checking for clear connections between the vocabulary and the mnemonic, as well as ensuring that the example sentence accurately reflected the vocabulary's meaning.

### 4.2 Training and inference

After generating the synthetic dataset, we implemented a distillation process to transfer this linguistic reasoning capability to a smaller, easier-to-deploy model.

**Student model selection** We selected Gemma-3-1b-it as our student model due to its balance of performance, size, and deployment flexibility. This instruction-tuned variant of Google's Gemma-3 (1 billion parameters) (Team et al., 2025) offers several advantages: (1) demonstrated instruction-following abilities, (2) multilingual capabilities for potential cross-lingual applications, and (3) compact size enabling deployment on consumer hardware including Apple Silicon.

**Group Relative Policy Optimization (GRPO)** We employed GRPO (DeepSeek-AI et al., 2025) to distill the reasoning capabilities of the teacher model into the student model. The GRPO process consists of three main steps: (1) generating multiple candidate outputs for each input, (2) scoring these outputs using a reward model(s), and (3) updating the student model's policy based on the scores. GRPO technical details are included in Section C.4.1 and our used configuration is provided in Section C.4.1.

We defined three reward functions that encode basic characteristics of effective mnemonics: (1) adherence to the structured format with reasoning, mnemonic, and example, (2) explicit incorporation of linguistic features in Table 1, and (3) usage of the target vocabulary in the mnemonic, penalizing bad mnemonics such as acronyms. These reward functions operate directly on model outputs, assigning scalar scores based on how well the generation satisfies each criterion. The scores are then combined using weighted summation, with higher weights assigned to criteria 1 and 2. We generated two candidate outputs per training example to enable reinforcement from comparisons. Training

was performed on a single NVIDIA H100 GPU approximately 4 hours.

**Low-Rank Adaptation (LoRA)** We trained Gemma-3-1b-it using GRPO wrapped in LoRA layers (Section C.3.2) to reduce the number of trainable parameters and rank-stabilized LoRA that maintains stability for adapters with higher ranks. Full LoRA configuration is provided in Section D.2.

**Model quantization and serving** To enable efficient deployment on consumer hardware, we quantized the final model using 4-bit quantization with the NormalFloat (NF4) data type (Dettmers et al., 2023). This significantly reduced the model size while maintaining performance quality. The quantized model can be served with a local application such as OpenWebUI (interface) and Ollama (command-line), making it accessible for language learners without requiring continuous internet connectivity or sharing potentially sensitive language learning data with third-party services.

## 5 Evaluation

### 5.1 Experimental setup

**LLM-as-a-judge for 1-5 Likert ratings**

**Pairwise preference using double-blind annotation**

**Interactive side-by-side comparison**

### 5.2 Results

## 6 Discussion

## 7 Related work

## 8 Conclusion

## 9 Limitations

## Acknowledgements

We would like to express my gratitude to my capstone committee, Dr. Patrick Watson and Dr. Philip Sterne, for their guidance and support throughout this project. I also want to thank my classmates and friends for their encouragement and feedback during the development of this project.

## Ethics Statement

This project was conducted in accordance with the ethical guidelines of Minerva University. The dataset used for training and evaluation was generated using an LLM, and a **subset** of generated mnemonics were reviewed for quality and appropriateness. We acknowledge the potential biases present in the training data and the need for continuous monitoring and improvement of the model's outputs. The final model is designed to be deployed locally, ensuring user privacy and data security.

The generated mnemonics are intended to be used as a supplementary tool for language learners and should **not** replace other language learning methods. We recommend that users critically evaluate the generated mnemonics and adapt them to their individual learning styles and preferences. We welcome feedback on any aspect of the paper to help improve the model's performance and address any ethical concerns that may arise.

## References

Richard C. Atkinson and Michael R. Raugh. 1975. An application of the mnemonic keyword method to the acquisition of a Russian vocabulary. *Journal of Experimental Psychology: Human Learning and Memory*, 1(2):126–133.

Nishant Balepur, Matthew Shu, Alexander Hoyle, Alison Robey, Shi Feng, Seraphina Goldfarb-Tarrant, and Jordan Lee Boyd-Graber. 2024. A SMART Mnemonic Sounds like "Glue Tonic": Mixing LLMs with Student Feedback to Make Mnemonic Learning Stick. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14202–14225, Miami, Florida, USA. Association for Computational Linguistics.

Bespoke Lab. 2025. Bespoke Lab Curator. bespokelabs.ai.

Alfredo Campos, Estefanía Camino, and María José Pérez-Fabello. 2011. Using the Keyword Mnemonics Method Among Adult Learners. *Educational Gerontology*, 37(4):327–335.

Alfredo Campos, María Angeles González, and Angeles Amor. 2003. Limitations of the mnemonic-keyword method. *The Journal of General Psychology*, 130(4):399–413.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui

Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. Technical Report arXiv:2501.12948, DeepSeek-AI.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Stefan Hackmann, Haniyeh Mahmoudian, Mark Steadman, and Michael Schmidt. 2024. Word Importance Explains How Prompts Affect Language Model Outputs.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Jaewook Lee and Andrew Lan. 2023. SmartPhone: Exploring Keyword Mnemonic with Auto-generated Verbal and Visual Cues. In *Artificial Intelligence in Education: 24th International Conference, AIED 2023, Tokyo, Japan, July 3–7, 2023, Proceedings*, pages 16–27, Berlin, Heidelberg. Springer-Verlag.

Jaewook Lee, Hunter McNichols, and Andrew Lan. 2024. Exploring Automated Keyword Mnemonics Generation with Large Language Models via Overgenerate-and-Rank. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5521–5542, Miami, Florida, USA. Association for Computational Linguistics.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand. Association for Computational Linguistics.

Christopher R. Madan. 2021. Exploring word memorability: How well do different word properties explain item free-recall probability? *Psychonomic Bulletin & Review*, 28(2):583–595.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022. Reframing Instructional Prompts to GPTk's Language. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 589–612, Dublin, Ireland. Association for Computational Linguistics.

Open Thoughts Team. 2025. Open Thoughts.

Gözde Özbal, Daniele Pighin, and Carlo Strapparava. 2014. Automation and Evaluation of the Keyword Method for Second Language Learning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 352–357, Baltimore, Maryland. Association for Computational Linguistics.

Michael Pressley, Joel R. Levin, and Harold D. Delaney. 1982. The Mnemonic Keyword Method. *Review of Educational Research*, 52(1):61–91.

Manolis Savva, Angel X. Chang, Christopher D. Manning, and Pat Hanrahan. 2014. TransPhoner: Automated mnemonic keyword generation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3725–3734, Toronto Ontario Canada. ACM.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar,

Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, C. J. Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju-yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. 2025. Gemma 3 Technical Report. Technical Report arXiv:2503.19786, Google DeepMind.

Janet G. van Hell and Andrea Candia Mahn. 1997. Keyword Mnemonics Versus Rote Rehearsal: Learning Concrete and Abstract Foreign Words by Experienced and Inexperienced Learners. *Language Learning*, 47(3):507–546.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of Draft: Thinking Faster by Writing Less.

Fan Yin, Jesse Vig, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2023. Did You Read the Instructions? Rethinking the Effectiveness of Task Definitions in Instruction Learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3063–3079, Toronto, Canada. Association for Computational Linguistics.

# A Prompt usage

All of the prompts include

**Vanilla vs. Alternative Phrasing** When comparing "Generate a mnemonic to help learn and remember the meaning of English vocabulary and how it is written: {term}" against "Generate a memory cue to help learn and remember the meaning of English vocabulary and how it is written: {term}", we observed substantial differences in output quality. This highlights the word importance effect noted by Hackmann et al. (2024), where specific terms like "mnemonic" may have acquired pre-training biases that associate them primarily with acronyms or keyword methods rather than broader linguistic strategies.

**Structured Prompting** We found improved performance with structured prompts that explicitly request linguistic analysis: "Generate a linguistically grounded mnemonic to help me learn and remember the meaning of English vocabulary and how it is written: {term}. Think in short traces and stop when you have a good linguistic connection. You must use that linguistic feature to form a mnemonic for the word." This approach yielded a higher percentage of mnemonics with clear linguistic association.

**Chain-of-Thought (CoT) Prompting** Incorporating chain-of-thought reasoning by providing both the instruction and examples of step-by-step linguistic analysis significantly improved the quality of generated mnemonics. Our implementation used 10 human-written examples, each demonstrating the process of finding linguistic association of the vocabulary before constructing a mnemonic.

**Concise Reasoning Traces** Inspired by Xu et al. (2025), we experimented with prompting models to generate minimal reasoning steps. For example: "Generate a mnemonic for {term}. Think step by step, but keep a minimum draft for each thinking step." This approach balanced comprehensive linguistic analysis with efficiency, preventing models from overthinking and/or elaborating on irrelevant aspects.

# B Annotation details

## B.1 Double-blind annotation

# C Technical preliminaries

## C.1 In-context learning

**Chain-of-Thought (CoT) prompting** (Wei et al., 2023) is a prompting technique that encourages LLMs to generate intermediate reasoning steps before arriving at a final answer. This approach has been shown to improve performance on complex tasks by guiding the model through a structured thought process.

## C.2 Neural Language Models and Transformer Architecture

Neural language models are probabilistic frameworks that assign probabilities to sequences of words or subword units, known as tokens. A token is the smallest unit of text that the model processes, which can be as granular as individual characters, subwords, or entire words, depending on the tokenization strategy employed.

Given a sequence of tokens $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, a language model estimates the joint probability $P(\mathbf{x})$ by factorizing it into conditional probabilities:

$$P(\mathbf{x}) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_{t-1})$$

At each time step $t$, the model predicts the next token $x_t$ based on the preceding sequence $(x_1, x_2, \ldots, x_{t-1})$. This autoregressive approach enables the generation of coherent text by sequentially predicting subsequent tokens.

The Transformer architecture underpins many state-of-the-art language models due to its efficiency and capability to model long-range dependencies. It utilizes self-attention mechanisms to weigh the relevance of each token in a sequence relative to others, regardless of their positions. The architecture comprises stacked layers, each including multi-head self-attention and position-wise feed-forward networks, facilitating parallelization and effective learning of complex patterns in data.

### C.2.1 Tokenizer

A tokenizer is a preprocessing tool that converts raw text into tokens, aligning the text with the LM's vocabulary. Tokenizers can employ various strategies, such as word-based, character-based, or

subword-based tokenization, each with distinct advantages and use cases.

Byte Pair Encoding (BPE) is a subword tokenization algorithm that operates on the byte representation of text, enabling consistent handling of various scripts and special characters. It iteratively merges the most frequent pairs of adjacent bytes to form subword units, constructing a vocabulary that efficiently represents the training corpus. This method allows the tokenizer to decompose rare words into meaningful subword components, enhancing the model's capacity to process diverse and unseen terms.

For instance, the word "preposterous" might be tokenized into subwords like "pre", "poster", and "ous," facilitating the model's understanding and generation of these subwords in novel contexts. This subword granularity enables the model to generalize across morphologically complex words and out-of-vocabulary words, enhancing its robustness and vocabulary coverage. However, not all subwords are valid morphemes, which can limit the model's ability to capture morphological structure accurately. For instance, `tiktoken` (OpenAI's tokenizer)[6] recognizes "ephemeral" as a single subword rather than three morphemes ("ept", "hemera", "-al"), because the affixes are not explicitly segmented, and 'epheremal' is a rare word so BPE better learns it as a single token.

### C.3 Family of Fine-Tuning Methods

Fine-tuning is the process of adapting a pre-trained model to a specific task T or domain D by updating its parameters on a target dataset $\mathcal{D}$. This process is crucial for leveraging pre-trained models' knowledge and enhancing their performance on downstream tasks.

There are several approaches to fine-tuning, which can be categorized by: 1. the availability of labeled data (supervised vs unsupervised fine-tuning), 2. the extent of parameter updates (full-parameter vs parameter-efficient fine-tuning), and 3. task. We focus on supervised fine-tuning, which involves minimizing a task-specific loss function over a labeled dataset.

#### C.3.1 Supervised Fine-Tuning (SFT)

SFT involves adapting a pre-trained model to a target task by minimizing a task-specific loss function over a labeled dataset. For a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ is the input and $\mathbf{y}^{(i)}$ is the target output, the objective is to minimize:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$$

where $f(\mathbf{x}; \theta)$ represents the model's output with parameters $\theta$, and $\ell$ is the loss function, typically cross-entropy loss.

**Instruction-tuning** is a specialized form of SFT where models are trained on datasets comprising instruction-response pairs. This approach enables models to generalize across various tasks described by natural language instructions, enhancing their ability to follow diverse prompts. Formally, an instruction-tuning dataset consists of pairs $\{(\mathbf{I}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ or triplets $\{(\mathbf{I}^{(i)}, \mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, where $\mathbf{I}^{(i)}$ denotes the instruction, $\mathbf{x}^{(i)}$ is the optional input, and $\mathbf{y}^{(i)}$ is the desired output. The training objective is to minimize the loss:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{I}^{(i)}, \mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$$

where $f$ represents the model parameterized by $\theta$, and $\ell$ is the loss function measuring the discrepancy between the model's prediction and the target output.

#### C.3.2 Parameter-Efficient Fine-Tuning

Full-parameter fine-tuning updates *all* parameters of a pre-trained model on the target dataset, which can be computationally expensive and memory-intensive for large models. Parameter-efficient fine-tuning (PEFT) methods adjust only a subset of the parameters, reducing computational and storage requirements while maintaining performance (Xu et al., 2023).

The most common PEFT method is Low-Rank Adaptation (LoRA), and its variants. They are used in the training process as a wrapper around the model's weights, allowing for efficient updates without modifying the entire model. This approach is particularly useful for large models, where full fine-tuning may be impractical due to resource constraints.

**Low-Rank Adaptation (LoRA)** decomposes the weight updates into low-rank matrices, reducing the number of trainable parameters (Hu et al., 2021). Specifically, for a weight matrix $W \in \mathbb{R}^{d \times k}$, LoRA introduces two low-rank matrices $A \in \mathbb{R}^{d \times r}$ and

[6] https://platform.openai.com/tokenizer

$B \in \mathbb{R}^{r \times k}$, where $0 < r \ll \min(d, k)$. The adapted weight is:

$$W' = W + \alpha \cdot AB$$

Here, $\alpha$ is a scaling factor that controls the contribution of the low-rank adaptation. The rank $r$ determines the capacity of the adaptation, balancing between expressiveness and efficiency.

LoRA introduces $2dr$ trainable parameters (size of $A$ and $B$), which is significantly smaller than the original $dk$ parameters. This reduction in parameters enables efficient fine-tuning of large models on limited hardware. In practice, LoRA is applied to specific modules of the model, such as attention and feed-forward layers, to balance performance and efficiency.

**Rank-Stabilized LoRA (rsLoRA)** modifies the scaling factor in LoRA to improve performance across different ranks. The standard scaling factor $\gamma_r = \alpha/r$ can slow learning for higher ranks. rsLoRA proposes adjusting the scaling factor to $\gamma_r = \alpha/\sqrt{r}$, enhancing fine-tuning performance without increasing inference costs.

### C.4 Reinforcement Learning (RL)

#### C.4.1 Group Relative Policy Optimization (GRPO)

Intuively, GRPO generates multiple responses for a given prompt, scores them using reward models, calculates the average reward of the group, and then compares each response's score to the average to determine which are better or worse. The model then updates its policy to favor high-reward responses.

Formally,

## D Training details

### D.1 Environment setup

The training was conducted, alternately, on a NVIDIA Tesla T4 GPU provided for free by Google Cloud (through Google Colab, Kaggle Notebook, or Google Cloud's Deep Learning Virtual Machine image) and a H100 NVIDIA GPU server with RunPod [7] (paid by the author, for detailed costs, see Appendix F). The T4 GPU has 16GB of memory, while the H100 GPU has 80GB of memory. The T4 GPU was used for initial experiments and supervised fine-tuning, while the H100 GPU was employed for more extensive training runs and reinforcement learning. The T4 GPU was used for initial experiments and supervised fine-tuning, while the H100 GPU was employed for more extensive training runs and GRPO (Appendix D.3).

The training environment was set up using these libraries developed by HuggingFace: `bitsandbytes` library for quantization, `peft` for parameter-efficient fine-tuning, `transformers` for model management. The training process was executed using the `trl` library, which provides tools for pre-training and post-training with transformers (including online RL). `unsloth` was used to reduce memory usage on single-GPU environment. `accelerate` and `deepspeed` were used to manage distributed training across multiple GPUs, while `vllm` was used for fast inference and serving of the trained model, especially during online RL.

The base student model used is Google's Gemma-3-1b-it , an open-weight 1-billion parameter Transformer-based decoder-only text-to-text model pre-trained to work well on general-purpose tasks in multiple languages, and fine-tuned to increase instruction following capabilities. To save memory, a 4-bit quantized version of the model was used, which reduces the model size and speeds up inference without significantly sacrificing performance.

### D.2 LoRA configuration

To reduce computational overhead, we employed LoRA (Hu et al., 2021) and rank-Stabilized LoRA (rsLoRA) scaling. The LoRA configuration parameters were set as follows: rank $r = 8$, scaling factor $\alpha_{\text{LoRA}} = 16$, and dropout rate of 0. These configurations were applied to both the attention and feed-forward layers.

The rank $r$ determines the dimensionality of the low-rank adaptation matrices, controlling the number of trainable parameters introduced during fine-tuning. A higher rank allows the model to capture more complex adaptations but increases computational complexity. The scaling factor $\alpha_{\text{LoRA}}$ modulates the impact of the low-rank updates on the original weights, effectively controlling the contribution of the adaptation matrices to the final model parameters. Setting the dropout rate to 0 indicates that no dropout regularization was applied during the LoRA updates, allowing all connections to be utilized during training.

[7] https://www.runpod.io/

### D.3 GRPO configuration

We used a batch size of 16, a learning rate of $2 \times 10^{-5}$, and a weight decay of 0.05. The training process was monitored using the validation set, and early stopping was applied to prevent overfitting. The training process was conducted over 3 epochs, with a total of 2000 training examples. The model was trained using the paged AdamW optimizer, which is a variant of the AdamW optimizer designed to handle large models efficiently. The training process was distributed across multiple GPUs using the `accelerate` library, which allows for efficient parallelization and memory management.

## E Documentation of previous iterations

### E.1 Fine-tune OpenAI

### E.2 Fine-tune Gemma-3-4b-it

For supervised fine-tuning (SFT), we utilized the `trl` library with the following hyperparameters: batch size $b = 16$, number of epochs eps $= 4$, learning rate $\alpha = 2 \times 10^{-5}$, weight decay $\lambda = 0.05$, and a cosine annealing learning rate scheduler with restarts.

The batch size $b$ defines the number of training examples processed simultaneously during each forward and backward pass. A batch size of 16 balances computational efficiency and gradient estimation accuracy. Training for 4 epochs (eps $= 4$) means the model will see the training data a total of four times, which ensures sufficient exposure to the training data without risking overfitting. The learning rate $\alpha$ controls the step size for weight updates; a value of $2 \times 10^{-5}$ is typical for fine-tuning large language models, facilitating gradual convergence. Weight decay $\lambda$ serves as a regularization term, penalizing large weights to prevent overfitting. The cosine annealing scheduler adjusts the learning rate following a cosine decay pattern, periodically restarting to allow the model to escape local minima and potentially achieve better generalization, compared to linear decay.

## F Costs

## G Minerva Appendix: LOs, HCs, AI Statement

### G.1 LOs

1. **CS110-codeReadability** The codebase exemplifies best practices in Python programming, adhering strictly to PEP conventions. Each module is documented with detailed Google-style docstrings and descriptive inline comments to ensure that the logic behind functions and classes is transparent to collaborators and future users. By utilizing tools like Ruff for linting and formatting, and mypy for type-checking, the codebase achieves a consistent style and minimizes errors. Additionally, the inclusion of pre-commit hooks ensures that these standards are maintained across all contributions, fostering a robust and maintainable codebase.

2. **CS162-communication** The documentation for strives to adhere to industry standards, by including an informative README, clear issue tracking, and documented pull request. Each module has a module-level docstring and function/class-level docstrings, to ensure that the code is understandable to users and collaborators. Linting and formatting tools, including Ruff and mypy, are employed to enforce consistent and error-free code presentation, to facilitate readability and maintainability for the author and future collaborators (if any).

3. **CS156-MLCode** The machine learning pipeline was designed in Python to be both functional and comprehensible. The code integrates data processing, model fine-tuning, and hyperparameter tuning into a seamless pipeline, with each step working and documented. Model evaluation are explicitly defined in appendix Appendix D, ensuring replicability.

4. **CS156-MLExplanation** Currently, most details and explanations are defined in section Appendix C and appendix Appendix D. The final paper will provide more high-level diagrams of the machine learning techniques used in fine-tuning the Gemma-2 model. The supporting diagrams will visualize key processes, such as hyperparameter tuning and model evaluation. It will ensure that the methodology and results are accessible to a less specialized audience.

5. **CS162-separationofconcerns** The codebase is organized into distinct Python modules, each focused on a specific task such as data processing, mnemonic processing, and model fine-tuning. This separation of concerns aligns with best practices in software design, ensuring that each function is highly cohesive and performs a single well-defined responsibility. By maintaining modularity, the codebase facilitates easier debugging, testing, and future scaling, contributing to its long-term maintainability and effectiveness

### G.2 Capstone LOs

6. **navigation**
7. **outcomeanalysis**
8. **curation**
9. **qualitydeliverables**

### G.3 HCs

10. **gapanalysis** Section Appendix 1 identifies critical limitations in existing mnemonic generation approaches: (1) overreliance on the keyword method, which fails for abstract vocabulary lacking concrete referents, and (2) neglect of the rich linguistic knowledge embedded in LLMs that could enable more diverse mnemonic strategies beyond simple keyword associations. Prior work has also passively delivered mnemonics to learners rather than leveraging individual learning preferences, despite research showing self-created mnemonics enhance retention. These identified gaps motivate our development of a linguistic knowledge mining approach from LLMs.

11. **hypothesisdevelopment** The research questions in Section Appendix 1 establish testable hypotheses regarding LLMs as linguistic knowledge bases for mnemonic generation. We hypothesize that fine-tuning LLMs on linguistically annotated examples will improve mnemonic quality across semantic relevance, diversity, and helpfulness dimensions. This hypothesis is predicated on the theoretical assumption that LLMs encode significant linguistic knowledge during pre-training that can be accessed through targeted fine-tuning. The hypothesis is operationalized through specific evaluation metrics described in Section Appendix 5, ensuring empirical testability.

12. **optimization** Our approach optimizes model performance through systematic hyperparameter tuning with the objective function minimizing validation loss. We employ parameter-efficient fine-tuning techniques (QLoRA with rsLoRA scaling) that significantly reduce computational requirements while maintaining performance. The hyperparameter space exploration includes learning rate, batch size, LoRA rank, and scaling factors, with population-based training enabling efficient identification of optimal configurations. This iterative optimization process is crucial for extracting maximal linguistic knowledge from the models while preventing overfitting to the training data.

13. **audience** The project addresses dual audiences: (1) NLP researchers investigating LLMs' linguistic capabilities, for whom we provide detailed technical methodologies and evaluation metrics; and (2) educational technology developers seeking practical approaches to vocabulary learning assistance, for whom we demonstrate application potential and implementation strategies. The Background section introduces key concepts with sufficient depth for computational linguists while remaining accessible to educational technology practitioners. Technical content is balanced with practical implications for vocabulary acquisition, ensuring relevance to both research and application-oriented readers.

14. **organization** The paper follows standard ACL formatting conventions, with a logical progression from theoretical foundations through methodology to empirical results. The structure facilitates efficient information extraction, with each section building upon previous content. Key contributions are identified early (Section Appendix 1) and systematically developed throughout subsequent sections. Technical details that might interrupt argumentative flow are relegated to appendices, maintaining narrative coherence while ensuring methodological transparency for reproduction purposes. This organization aligns with expectations of the computational linguistics community while supporting efficient knowledge transfer.

15. **algorithms** The fine-tuning methodology detailed in Section Appendix C.3 incorporates algorithmic innovations in parameter-efficient adaptation. Specifically, we implement QLoRA with rank-stabilized scaling (rsLoRA), modifying the standard scaling factor to improve performance across different ranks. This algorithm reduces memory requirements by quantizing the pre-trained model to 4-bit precision while allowing selective updates to low-rank adaptation matrices. Population-based training explores the hyperparameter space dynamically, pruning underperforming configurations and exploring promising regions to optimize validation performance.

16. **heuristics** Our approach employs several problem-solving heuristics to enhance mnemonic generation. The linguistic feature classification system provides a structured framework for identifying and leveraging different linguistic aspects in vocabulary. We use problem decomposition by separating mnemonic creation into linguistic analysis and creative association phases, enabling more systematic knowledge utilization. Visualization techniques in the form of embedding space projec-

tions help identify semantic relationships between vocabulary terms and potential mnemonic content. These heuristics guide both the model fine-tuning process and the subsequent evaluation methodology.

17. **sampling** The evaluation methodology employs stratified random sampling to ensure representation across linguistic feature categories and vocabulary complexity levels. For human evaluation, we randomly selected 50 test examples, stratified by linguistic feature type, to obtain unbiased assessments of mnemonic quality. This sampling strategy ensures balanced representation of different mnemonic types while maintaining statistical validity. Each example received multiple independent ratings to mitigate individual rater bias, with inter-rater reliability calculated using Gwet's AC2 to confirm assessment consistency.

18. **studyreplication** To facilitate replication, we provide comprehensive implementation details in Appendix Appendix D, including environment configurations, model parameters, and hyperparameter settings. The dataset construction process is documented in Section Appendix 4.1, with preprocessing steps explicitly specified. All code and datasets are made publicly available through GitHub and HuggingFace repositories, with standardized formats ensuring compatibility with common ML frameworks. This transparency ensures that other researchers can validate our findings and build upon our methodology.

19. **observationalstudy** Our human evaluation component constitutes an observational study assessing how learners perceive and utilize generated mnemonics. The Likert-scale ratings provide quantitative metrics, while qualitative feedback offers insights into specific features that enhance mnemonic effectiveness. Though limited in scale, this observational component validates computational metrics and provides preliminary evidence regarding learning impact. Future work will extend this observational approach through longitudinal studies tracking actual vocabulary retention over time.

20. **dataviz** We employ targeted data visualizations to communicate complex relationships between linguistic features and mnemonic effectiveness. Radar charts display the distribution of linguistic features across different model outputs, while heatmaps visualize correlation patterns between computational metrics and human evaluations. Embedding space projections illustrate semantic relationships between vocabulary terms and their mnemonics, providing intuitive visual confirmation of semantic relevance scores. These visualizations enhance interpretability of results while supporting our conclusions regarding the contribution of different linguistic features to mnemonic quality.

21. **significance** Statistical significance testing confirms the reliability of our comparative results between baseline and fine-tuned models. We employ paired statistical tests (Wilcoxon signed-rank) to account for vocabulary-specific variation when comparing model outputs on identical test sets. Effect size calculations quantify the practical significance of improvements, while confidence intervals provide transparency regarding the precision of our estimates. Multiple comparison corrections maintain statistical rigor when evaluating performance across different linguistic feature categories.

22. **shapingbehavior** The intended application of our approach shapes vocabulary learning behavior by encouraging deeper engagement with linguistic features. Rather than passive memorization, the generated mnemonics prompt learners to recognize morphological patterns, etymological connections, and semantic relationships, fostering more robust mental representations. By explicitly highlighting these linguistic features, the system promotes analytical processing of vocabulary, which research indicates enhances long-term retention. This behavior-shaping aspect represents a significant advantage over keyword-only approaches that rely on shallow phonetic associations.

23. **ethicalconsiderations** Our work addresses ethical considerations in educational technology deployment. We prioritize linguistic diversity by including vocabulary from various etymological backgrounds rather than focusing exclusively on Latin/Greek derivatives. The evaluation process incorporates feedback from learners with diverse linguistic profiles to ensure the approach benefits various learning styles. We acknowledge limitations regarding potential cultural specificity in some mnemonics and identify this as an area for future refinement. All human evaluation participants provided informed consent, and data collection followed established ethical guidelines for educational research.

### G.4 AI Statement