# BEYOND TOKENS IN LANGUAGE MODELS: INTERPRETING ACTIVATIONS THROUGH TEXT GENRE CHUNKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Understanding Large Language Models (LLMs) is key to ensure their safe and beneficial deployment. This task is complicated by the difficulty of interpretability of LLM structures, and the inability to have all their outputs human-evaluated. In this paper, we present the first step towards a predictive framework, where the genre of a text used to prompt an LLM, is predicted based on its activations. We consider two LLMs, `Tinyllama-15M` and `Mistral-7B` finding that the text type can be extracted with a significantly higher accuracy from the latter. With 6 text genres, we achieve an F1-Score of around $80\%$ with linear probes, serving as a proof of concept showing that text genres can be extracted linearly from LLMs.

## 1 INTRODUCTION

As language models continue to improve in performance and are applied in a growing number of sectors, it is becoming more important to understand how these models work and monitor their outputs. However, the current paradigm of transformer interpretability focuses almost exclusively on predicting single tokens outputs, with some limited research looking at a specific component of text such as truthfulness or emotion.

We set the ground for looking at language model outputs on the scale of clusters of tokens as interpretable natural language separations. By looking at larger blocks of text, we aim to make it easier to interpret the highest-level of predictions that a language model is making, and to make it easier to monitor large scale language models outputs.

We achieve this by building a small dataset of diverse language outputs, and make a machine- and human-curated natural language labelling of different chunks of these texts. We then show that these text chunks are interpretable and easy to classify using simple linear probes in even small models, and thus that these outputs may be relatively natural units of study.

## 2 RELATED WORK

Most mechanistic interpretability research focuses on understanding the effects neuron activations have on single token activations (Geva et al., 2021; Chan et al., 2022; Conmy et al., 2023; nostalgebraist, 2020; Belrose et al., 2023; Olsson et al., 2022). While this is an important and natural bottom-up lens through which to understand models, it stands that a more top-down approach may be underexplored.

Recent work on investigating language model activations and Representation Engineering (Zou et al., 2023; Burns et al., 2022; Turner et al., 2023; Gurnee & Tegmark, 2023; Meng et al., 2022), suggests that it is possible to extract human-understandable concepts with their activations. These results support the hypothesis that a top-down approach to language model interpretability like the method described in this paper is tractable.

In addition, research on modularity and activation sparsity in language models (Liu et al., 2023; Pochinkov & Schoots, 2023; Zhang et al., 2022; 2023b; Pfeiffer et al., 2023), as well as the achievement of high pruning ratios for task-specific models, (Xu et al., 2022) suggest there may be separable ways of looking at different components of text.

Research on image models, including interpretability research (Voss et al., 2021; Olah et al., 2017; Mordvintsev et al., 2018), and research into the machine unlearning of specific classes and concepts (Foster et al., 2023; Bourtoule et al., 2021; Nguyen et al., 2022), highlights that it is possible to understand how models activate not only on the specific pixel values, but also on larger scale concepts. Our work tries to lay the groundwork for finding these larger scale concepts in text.

There exists a large variety of research into using language models for text embedding (Muennighoff et al., 2022) and classification (Minaee et al., 2021; Howard & Ruder, 2018) on various datasets and tasks, including news categorisation (Zhang et al., 2015), sentiment analysis (Maas et al., 2011), question answering (Rajpurkar et al., 2016), semantic relatedness (Marelli et al., 2014), and retrieval augmentation (Thakur et al., 2021; Lin et al., 2022). However, these are focused on using language models to perform specific tasks, rather than understanding how a language model performs tasks.

Relatedly, there is research on topic analysis of single texts (De Paoli, 2023) and large corpuses of text (Gauthier & Wallace, 2022), as well as research into qualitative deductive coding, (Zhang et al., 2023a; Gebreegziabher et al., 2023; Rietz & Maedche, 2021). The research is often narrow in scope, such as finding aggregate themes of outputs from specific online communities (Tai et al., 2023; Chew et al., 2023), classifying types of questions (Xiao et al., 2023), or extracting and reformatting knowledge (Shi et al., 2023). Our research, instead of topics, aims to probe into the working of language models to understand on a high-level, how different multi-token scale components are represented in the activations, and thus investigate text genres (Drew, 2023) and block components.

## 3 METHOD

The goal of this work is to investigate how well different classifiers manage to predict the genre of a text chunk, given the residual stream this text generates in a pre-trained model. To this end, we create a dataset and train various classifiers as prediction models. We describe the process and our methods in the following subsections.

### 3.1 DATASET CREATION

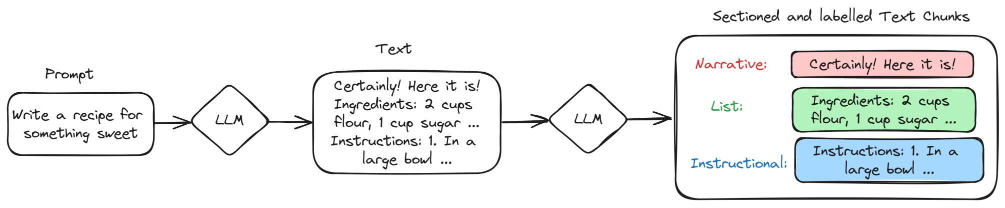
Prompt
Write a recipe for something sweet
LLM
Text
Certainly! Here it is! Ingredients: 2 cups flour, 1 cup sugar ... Instructions: 1. In a large bowl ...
LLM
Sectioned and labelled Text Chunks
Narrative:
Certainly! Here it is!
List:
Ingredients: 2 cups flour, 1 cup sugar ...
Instructional:
Instructions: 1. In a large bowl ...

Figure 1: Generation of the labelled dataset.

As shown in Figure 1, we build our dataset by creating a list of prompts, that are constructed such that they can elicit varied texts from a list of genres and topics. We use them together with `Mistral-7B-Instruct-v2.0` (Albert Q. Jiang, 2023) to generate texts, by calling each prompt multiple times. These generated texts are then passed on to `GPT-4`(OpenAI et al., 2023), which is asked to section the text into chunks according to genres and subsequently label it with the selected genre and a fitting topic. The genres we consider are: instructional, list, narrative, title, explanatory, dialogue, code. The labelling is reviewed by humans and mislabelled text sections are removed. The prompts and examples of the labelled text are shown in the appendix. In total, we pass a total of 23 prompts to Mistral to generate 784 text samples, which is then split into 3929 distinct text chunks.

The primary motivation for using an open weights model like `Mistral-7B-Instruct-v2.0` to generate the texts, is that it allows us to access its activations. This lays the groundwork to do future work where we predict the sequence of labels based on the activations produced by the prompt.

### 3.2 TRAINING PROCEDURE

We work with transformer architecture language models (Vaswani et al., 2017), specifically with the Meta's decoder-only causal `TinyLlama-15M` model (Touvron et al., 2023; Karpathy, 2023) trained with 15 million parameters on TinyStories Eldan & Li (2023), and `Mistral 7B` (Albert Q. Jiang, 2023) with 7–billion parameters. The models are accessed via the Hugging Face transformer library (Taylor et al., 2022) and run using PyTorch (Paszke et al., 2019). We prompt the LLMs using text chunks from our labelled dataset, and we study the residual stream activations at the output of each attention and MLP layer, as described in (Radford et al., 2019). Specifically, a text chunk produces activations $a_i^j$ in layer $j$ of the LLM for token $t_i$.

The activations $a_i^j$ serve as input for the probe models we train. They are trained on the activations to predict the correct genre for each labelled text chunk. We employ a range of commonly used classifiers as our probe models. We train a deep learning model and several shallow learning algorithms Pedregosa et al. (2011): logistic regression, linear discriminant analysis, ridge classifier and SGDClassifier. See Appendix A for more model details. A sketch of the training procedure is shown on the left of Figure 3.

For evaluating our predictor, we use the Macro F1-Score metric (Margherita Grandini, 2020) to give the same weight to all classes, even in an unbalanced dataset, including both recall and precision.

### 3.3 Dimensional Reduction

To study how the model might represent different aspects of text, we use a text embedding model, `UAE-Large-V1` Li & Li (2023) to encode the split chunks of text we have generated. For dimensionality reduction and analysis, we use PHATE Moon et al. (2017), and show the labelling for both the generation prompt (the task it was trained for), and the labelled genre.

## 4 Results

### 4.1 Dimensional Reduction

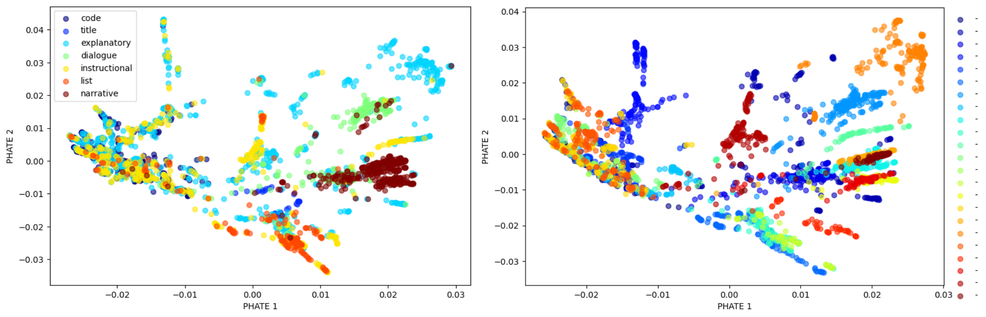

Figure 2: A PHATE dimensional reduction for each of the different clusters of text analysed. We see that there is some correspondence with clusters to the labelled genre (left), as well as with the generating prompts (right). See Appendix B.1 for all prompts

By comparing the plots, in Figure 2 that the different prompts often lead to similar genres of text output, but also that there is a great variation depending on the prompt. Neither classification fully explains the clusters seen in the dimensional reduction, but we see the relatively high correspondence between text genres with various clusters in embedding space as a promising signal to the encoding of these themes as being captured within the models.

### 4.2 Labelling Accuracy

In the right panel of Figure 3 we see the F1-score performance for various probe models predicting the genre based on the residual activations from `Tinyllama-15M` and `Mistral-7B`. The axes show the fractional layer depth at which the probes are trained at the outputs of the MLP and Attention layers, as well the F1-score that probe achieves.

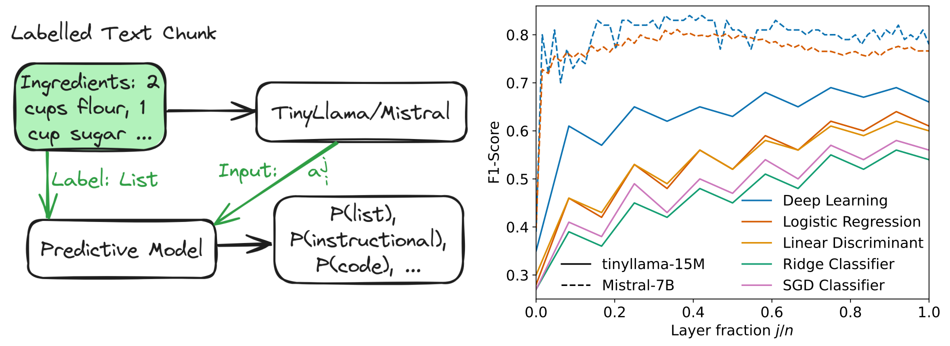

Figure 3: Training procedure (left)) and F1-score performance as a function of layer fraction (right) of prediction models on the task of predicting the genre of a text section. The activation $a_i^j$ of token $t_i$ in the $j$-th layer out of a total of $n$ layers has been extracted from `Tinyllama-15M` and `Mistral-7B`.

We see that the Mistral model is relatively easy to probe, with various methods reaching an F1-score of around 0.8, with probing being slightly better at central layers, in line with results from Meng et al. (2022) which showed higher-level concepts being represented in middle layers.

Interestingly, despite being a weak model not exposed to text beyond TinyStories, the TinyLlama model achieves a respectable score of up to 0.69 with a deep learning model. On the other hand, the linear techniques perform substantially worse at most layers, with scores increasing based on layer depth up to the best score of 0.64. As a general feature, we can see from the graphs that the odd-numbered attention layer outputs perform better than the even-numbered MLP layer outputs, although this effect dissipates in later layers of `Mistral-7B`.

## 5 Discussion and Conclusions

Overall, we find the results promising, and they preliminarily support the hypothesis that models represent high-level multi-token text structures in relatively easy to identify ways. Starting out with a total set of 6 possible genres, we found that we can achieve a high F1-score for all deep-learning and shallow learning probe models. Additionally, our embedding analysis show promising results for how models represent concepts.

We saw that the performance of our probe method can be increased by choosing a model such as Deep Learning, which had consistently the highest and most stable F1-scores for the examples we studied. Interestingly, the shallow learning probe models were not far behind, indicating that most of the signals are likely linearly represented in the model activations. In particular, logistic regression lead to F1 scores of 0.61 and 0.81 from `Tinyllama-15M` and `Mistral-7B` respectively. In this case, the performance for `Tinyllama-15M` is surprisingly high, considering the model is quite small and was trained exclusively on TinyStories, which lacks genres such as code.

### 5.1 Future Work and Broader Impacts

So far, our results indicate that even with a small dataset and a few relatively overlapping text genre categories, we are able to probe relatively accurately into the model. Future work could investigate scaling up the number of text categories, possibly to include text topic of discussion or other properties of text, and to find better ways of incorporating the overlap of multiple text genres.

Future work could try to use these findings and probes to find longer timescale predictions on which genres are likely to emerge in the future of the residual stream, improving our ability to verify the trustworthiness of models stating their intended future actions.

## REFERENCES

Arthur Mensch Chris Bamford Devendra Singh Chaplot Diego de las Casas Florian Bressand Gianna Lengyel Guillaume Lample Lucile Saulnier Lélio Renard Lavaud Marie-Anne Lachaux Pierre Stock Teven Le Scao Thibaut Lavril Thomas Wang Timothée Lacroix William El Sayed Albert Q. Jiang, Alexandre Sablayrolles. Mistral 7b. 2023.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 141–159. IEEE, 2021. doi: 10.1109/SP40001.2021.00019. URL `https://doi.org/10.1109/SP40001.2021.00019`.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: a method for rigorously testing interpretability hypotheses. AI Alignment Forum, 2022. URL `https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN`.

Robert Chew, John Bollenbacher, Michael Wenger, Jessica Speer, and Annice Kim. Llm-assisted content analysis: Using large language models to support deductive coding. *arXiv preprint arXiv:2306.14924*, 2023.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *CoRR*, abs/2304.14997, 2023. doi: 10.48550/arXiv.2304.14997. URL `https://doi.org/10.48550/arXiv.2304.14997`.

Stefano De Paoli. Performing an inductive thematic analysis of semi-structured interviews with a large language model: An exploration and provocation on the limits of the approach. *Social Science Computer Review*, pp. 08944393231220483, 2023.

C. Drew. 18 text types (with examples) – writing styles explained. Helpful Professor, September 2023. URL `https://helpfulprofessor.com/text-types/`.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL `https://github.com/Lightning-AI/lightning`.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. *CoRR*, abs/2308.07707, 2023. doi: 10.48550/ARXIV.2308.07707. URL `https://doi.org/10.48550/arXiv.2308.07707`.

Robert P Gauthier and James R Wallace. The computational thematic analysis toolkit. *Proceedings of the ACM on Human-Computer Interaction*, 6(GROUP):1–15, 2022.

Simret Araya Gebreegziabher, Zheng Zhang, Xiaohang Tang, Yihao Meng, Elena L Glassman, and Toby Jia-Jun Li. Patat: Human-ai collaborative qualitative coding with explainable interactive rule synthesis. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.446. URL `https://doi.org/10.18653/v1/2021.emnlp-main.446`.

Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2018. URL `https://api.semanticscholar.org/CorpusID:40100965`.

Andrej Karpathy. llama2.c. `https://github.com/karpathy/llama2.c`, 2023. Using llama models trained on tinyllama.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Xianming Li and Jing Li. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature, 2022.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22137–22176. PMLR, 2023. URL `https://proceedings.mlr.press/v202/liu23am.html`.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 1–8, 2014.

Giorgio Visani Margherita Grandini, Enrico Bagli. Metrics for multi-class classification: An overview. *arXiv preprint arXiv:2008.05756*, 2020.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40, 2021.

Kevin R Moon, David van Dijk, Zheng Wang, William Chen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Phate: a dimensionality reduction method for visualizing trajectory structures in high-dimensional biological data. *BioRxiv*, 120378, 2017.

Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL `https://arxiv.org/abs/2210.07316`.

Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *CoRR*, abs/2209.02299, 2022. doi: 10.48550/arXiv.2209.02299. URL `https://doi.org/10.48550/arXiv.2209.02299`.

nostalgebraist. interpreting gpt: the logit lens. *LessWrong*, 2020. URL `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Josh OpenAI, Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulic, and Edoardo Maria Ponti. Modular deep learning. *CoRR*, abs/2302.11529, 2023. doi: 10.48550/arXiv.2302.11529. URL `https://doi.org/10.48550/arXiv.2302.11529`.

Nicky Pochinkov and Nandi Schoots. Dissecting large language models. In *Socially Responsible Language Modelling Research*, 2023.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications. *OpenAI Blog https://openai.com/blog/better-language-models*, 1(2), 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Tim Rietz and Alexander Maedche. Cody: An ai-based system to semi-automate coding for qualitative research. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2021.

Breno Batista da Silva Shankar Rao Pandala. Lazy predict, 2022. URL `https://lazypredict.readthedocs.io/en/latest/`.

Yucheng Shi, Hehuan Ma, Wenliang Zhong, Gengchen Mai, Xiang Li, Tianming Liu, and Junzhou Huang. Chatgraph: Interpretable text classification by converting chatgpt knowledge to graphs. *arXiv preprint arXiv:2305.03513*, 2023.

Robert H Tai, Lillian R Bentley, Xin Xia, Jason M Sitt, Sarah C Fankhauser, Ana M Chicas-Mosier, and Barnas G Monteith. An examination of the use of large language models to aid analysis of textual data. *bioRxiv*, pp. 2023–07, 2023.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *CoRR*, abs/2211.09085, 2022. doi: 10.48550/arXiv.2211.09085. URL `https://doi.org/10.48550/arXiv.2211.09085`.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *CoRR*, abs/2308.10248, 2023. doi: 10.48550/arXiv.2308.10248. URL `https://doi.org/10.48550/arXiv.2308.10248`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chelsea Voss, Gabriel Goh, Nick Cammarata, Michael Petrov, Ludwig Schubert, and Chris Olah. Branch specialization. *Distill*, 2021. doi: 10.23915/distill.00024.008. https://distill.pub/2020/circuits/branch-specialization.

Ziang Xiao, Xingdi Yuan, Q Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. Supporting qualitative analysis with large language models: Combining codebook with gpt-3 for deductive coding. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*, pp. 75–78, 2023.

Runxin Xu, Fuli Luo, Chengyu Wang, Baobao Chang, Jun Huang, Songfang Huang, and Fei Huang. From dense to sparse: Contrastive pruning for better pre-trained language model compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11547–11555, 2022.

He Zhang, Chuhao Wu, Jingyi Xie, ChanMin Kim, and John M Carroll. Qualigpt: Gpt as an easy-to-use tool for qualitative coding. *arXiv preprint arXiv:2310.07061*, 2023a.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Neural Information Processing Systems*, 2015. URL `https://api.semanticscholar.org/CorpusID:368182`.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 877–890. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.71. URL `https://doi.org/10.18653/v1/2022.findings-acl.71`.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers, 2023b. URL `https://openreview.net/forum?id=XHuQacT6sa6`.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

## A Training Hyperparameters

We study the residual stream activations at the output of each attention and MLP layer, as described in (Radford et al., 2019). We train a deep learning algorithm and several shallow learning prediction models. To select these, we used the Lazy Predict library (Shankar Rao Pandala, 2022), which enables us to test several scikit-learn algorithms on the same dataset. We then chose those that run in a reasonable amount of time and have a reasonable performance. Our final choices which we use for our results are: logistic regression, linear discriminant analysis, ridge classifier and SGDClassifier.

Our deep learning model is a MLP with 2 hidden layers. The input layer have a number of neurons equal to the $d\_model$ of the model to be studied. The output has one neuron per class, meaning 6 neurons in total, to which softmax is applied for the training only. The hidden layers have 512 neurons each. In every hidden layer we apply the 1D batch norm, the ReLU activation function and 20% dropout. The model is trained in PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019) using the Adam optimizer (Kingma & Ba, 2014) to minimize cross-entropy loss. We use early stopping with a patience of 3 and train the model with a batch size of 16. We divide our dataset in 50% for training set and 50% for test set.

We used shallow learning models from scikit-learn library Pedregosa et al. (2011) with vanilla hyperparameters, apart from $max\_iter$ which was fixed at 100000. We scaled our data by removing the mean and scaling to unit variance with $StandardScaler()$.

## B Dataset Generation

We generate our dataset by following these steps:

1. Manually create a list with prompts that can elicit varied texts from an LLM. The goal is that each element of the list can prompt an LLM to generate text pertaining to particular genres. The genre categories we consider are:

   { instructional, list, narrative, title, explanatory, dialogue, code }

   Additionally, the prompts should be formulated in an open-ended way, such that the same prompt can be used to prompt the same LLM multiple times and result in diverse texts. You can find the full list containing 23 prompts in section B.1.

2. Loop over the prompt list to generate texts using `Mistral-7B-Instruct-v2.0`. Each of these prompts is called 35 times, generating a total of 805 synthetic texts.

3. Ask GPT-4 to section each text from step 2. according to the genre categories. Additionally, it should Label each section with one word that describes a main theme of this section. Therefore, we will have two labels for each text section, a genre and a topic label. We use the following prompt to achieve this:

   "please return a json list that labels different chunks of the text above into genres:
   instructional, list, narrative, explanatory, dialogue, code
   Also add a coarse topic label. E.g: "cooking", "food", "weather", "function definition", "code business logic", "import statements", "
   Please escape characters such as "\n".
   Here is how you should format the output:
   [ {"genre": ..., "topic":..., "text": ...},
   {"genre": ..., "topic":..., "text": ...},
   ... ]"

   + synthetic text

You can find some examples how the sectioning and labelling looks like in section B.2

Due to memory constraints the dataset is reduced to 784 labeled text chunks, distributed over the genres as follows: 'code': 252, 'explanatory': 508, 'title': 40, 'narrative': 553, 'instructional': 316, 'dialogue': 210, 'list': 108.

The text chunks have varying lengths, so the number of tokens in each genre are: 'code': 24745, 'explanatory': 30269, 'title': 553, 'narrative': 40343, 'instructional': 34976, 'dialogue': 16368, 'list': 14932.

This dataset is divided into test and training sets containing equally many text chunks to folly utilise the available memory.

### B.1 PROMPTS

We use the following list of prompts designed to produce a mix of genres:

- "Write a fable for children.",
- "Write a short story about betrayal.",
- "Write a story about a young little puppy navigating the world.",
- "Instruct someone with anger management problems how to handle them.",
- "Write a recipe for a good cocktail.",
- "Write a recipe for a savory meal.",
- "Create a poem about war.",
- "Create a poem about friendship, love and trust.",
- "Write a sorting algorithm in your favorite coding language",
- "Be creative and write a code that implements a specific task.",
- "Make me a dialogue from a political debate program.",
- "Write me a dialogue for a play.",
- "Write a coding task and solve it.",
- "Provide a coding problem with solutions for an exam.",
- "Make a software engineering coding interview task and solve it.",
- "Use coding to solve a mathematical problem appropriate for undergraduates.",
- "Make and solve a physics problem using both analytical and numerical methods. Implement your solution in your favorite coding language.",
- "Make and solve an elementary school problem with code.",
- "Write me a recipe for something I can serve my guests.",
- "Write me a recipe for a traditional desert to serve my guests.",
- "How do I make a business?",
- "Pick a natural phenomenon and explain it in a short text.",
- "Pick a topic and explain it to an 8 year old."

### B.2 EXAMPLES

These examples have been slightly edited from their original format for the purposes of easier readability.

#### B.2.1 EXAMPLE 1: COOKING

**Prompt:** Write me a recipe for a traditional desert to serve my guests.

- **"genre"**: narrative, **"topic"**: cooking, **"text"**: "Here's a recipe for a traditional American dessert that is sure to impress your guests: Apple Pie à la Mode."

- **"genre"**: list, **"topic"**: cooking, **"text"**: "Ingredients for the Apple Pie:
  - 7-8 cups sliced and peeled apples (about 6 large apples)
  - 1 tablespoon lemon juice
  - 3/4 cup white sugar
  - 1 teaspoon ground cinnamon
  - 1/2 teaspoon ground nutmeg
  - 1/4 teaspoon salt
  - 1 tablespoon unsalted butter
  - 1 recipe pie crust (homemade or store-bought)
  - 1 tablespoon milk",

- **"genre"**: list, **topic**: cooking, **"text"**: "For the Pie Crust:
  - 2 1/2 cups all-purpose flour
  - 1 teaspoon salt
  - 1 teaspoon sugar
  - 1 cup (2 sticks) cold unsalted butter, cut into small pieces
  - 1/4 to 1/2 cup ice water",
- **"genre"**: instructional, **"topic"**: cooking, **"text"**: "Instructions for the Apple Pie:
  1. Preheat the oven to 425°F (220°C).
  2. In a large bowl, combine sliced apples and lemon juice.
  3. In a small bowl, mix sugar, cinnamon, nutmeg, and salt.
  4. Add the sugar mixture to the apples and toss to coat.
  5. Transfer the apples to a large, deep-dish pie plate. Dot with butter.
  6. Roll out the pie crust and place it over the apples, trimming and crimping the edges.
  7. Brush the crust with milk.
  8. Bake for 15 minutes at 425°F.
  9. Then reduce the temperature to 350°F and bake for an additional 45 minutes, or until the crust is golden brown and the apples are tender.",
- **"genre"**: "instructional", **"topic"**: "cooking", **"text"**: Instructions for Serving à la Mode:
  1. Scoop a slice of warm apple pie onto each plate.
  2. Top each slice with a large scoop of vanilla ice cream.
  3. Drizzle a small amount of caramel or butterscotch sauce over the ice cream and apple pie (optional).
  4. Serve and enjoy"

### B.3 Example 2: Coding

**Prompt:** Write a coding task and solve it.

- **"genre"**: instructional, **"topic"**: function definition, **"text"**: "Create a function named "reverse_string" that takes a string as an argument and returns that same string in reverse order.
- **"genre"**: code, **"topic"**: function definition, **"text"**:

```python
def reverse_string(string):
# Your code here
```

- **"genre"**: "instructional", **"topic"**: "code business logic", **"text"**: "To solve this coding task, follow the steps below:",
- **"genre"**: "list", **"topic"**: "code business logic", **"text"**:
  "1. Define the function 'reverse_string' with the given signature.
  2. Write a 'return' statement to the empty string "". This is where we will store the reversed string.
  3. Use a loop or slicing to reverse the string.",
- **"genre"**: "explanatory", **"topic"**: "code business logic", **"text"**: "Here's an example of implementing the solution using slicing:",

- **"genre"**: "code", **"topic"**: "code business logic", **"text"**:

```python
def reverse_string(string):
    reversed_string = ''
    last_index = len(string) - 1

    while last_index >= 0:
        reversed_string += string[last_index]
        last_index -= 1
    return reversed_string
```

- **"genre"**: "explanatory", **"topic"**: "code business logic", **"text"**: "Using this reversing process, our function will correctly reverse the given input string and return it.",
- **"genre"**: "code", **"topic"**: "example usage", **"text"**:

```python
print(reverse_string('hello')) # outputs: 'olleh'
print(reverse_string('python')) # outputs: 'netyp'
print(reverse_string(''))      # outputs: ''
```

- **"genre"**: "explanatory", **"topic"**: "interview preparation", **"text"**: "Editor's note: This is a common coding challenge in interviews and exercise backgrounds. It's essential to familiarize yourself with these kinds of tasks and practice different approaches to enhance your problem-solving skills."