EXPLORING DEMONSTRATION ENSEMBLING FOR IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) operates by showing language models (LMs) examples of input-output pairs for desired tasks, *i.e.,* demonstrations. The standard approach for ICL is to prompt the LM with concatenated demonstrations followed by the test input. This approach suffers from some issues. First, concatenation offers almost no control over the contribution of each demo to the model prediction. This can be sub-optimal when some demonstrations are not very relevant to the test example. Second, due to the input length limit of transformer models, it can be infeasible to fit many examples into the context, especially when dealing with long-input tasks. In this work, we explore **D**emonstration **Ense**mbling (DENSE) as an alternative to simple concatenation. DENSE predicts outputs using subsets (*i.e.*, buckets) of the demonstrations and then combines the output probabilities resulting from each subset to produce the final prediction. We study different ensembling methods using GPT-j (Wang & Komatsuzaki, 2021) and experiment on 7 different language tasks. Our experiments show max ensembling to outperform concatenation by an average of 3.8 points.

1 INTRODUCTION

Large-scale language model (LM) pre-training on large corpora is currently dominating the NLP scene. One impressive aspect of such large language models (LLMs) is their capability to do in-context learning (Brown et al., 2020) by conditioning the model on a few examples (*i.e.,* demonstrations) of the desired task and then asking the LM to predict the label for a given input.

The standard approach for feeding in-context demonstrations (demos, *for short*) to the LM is by *concatenating* the task examples (Brown et al., 2020; Min et al., 2022c; Lu et al., 2022). While simple, concatenation suffers from a few drawbacks. First, it provides no control over each demo's contribution to the model's output, which is left to the attention weights to decide. Second, the concatenation of demos can easily use up the context window of transformer-based models, especially when we have access to many demonstrations or when dealing with lengthy inputs. Lastly, it has been shown that LLMs are sensitive to the ordering of the demonstrations Zhao et al. (2021); Lu et al. (2022), and a long chain of concatenated demos can indeed exacerbate this problem.

In this work, we explore an alternative to the concatenation approach, which is to prompt the model with demonstrations in an ensembling approach. In particular, we partition the examples into non-empty subsets or buckets and then combine the predictions obtained from each bucket to obtain the final prediction. We investigate three different ensembling methods to combine the predictions from different buckets including a clustering-based approach to partition the examples. Experiments on 7 different language tasks show that ensembling can outperform the standard concatenation approach.

2 RELATED WORK

This work is related to work that aims to improve few-shot learning with LLMs (Min et al., 2022b; Rubin et al., 2022; Lu et al., 2022). For instance, Perez et al. (2021) try to find optimal prompts using techniques such as cross-validation and minimum description length. Min et al. (2022a) applied demonstration ensembling for text classification in a limited setting. This paper, on the other hand, explores the more generalized ensembling setting with different bucket sizes and different types of

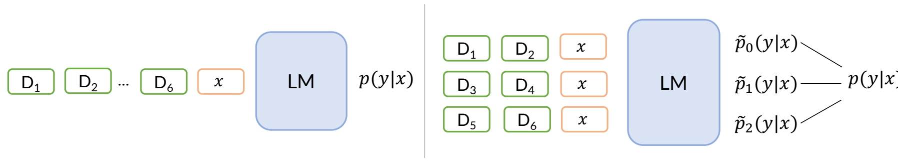

Figure 1: In-context learning with six demonstrations. **Left:** The standard concat-based approach for feeding the examples (Brown et al., 2020). **Right:** Ensembling with three buckets of size two each. For a given label y, the probability $\tilde{p}_i(y|x)$ is computed using the i-th bucket. All probabilities are ensembled to give the final probability $p(y|x)$.

tasks. Wang et al. (2022) explore rationale-augmented ensembles, where different in-context demonstrations are augmented with LM-generated rationales. Different from our work, the ensembling is done over the rationales in the examples, while we ensemble the examples themselves. Qin & Eisner (2021) trained mixtures of soft prompts for knowledge extraction from language models. Khalifa et al. (2022) explored demonstration ensembling to in-context learn to rerank document paths for multi-hop QA.

3 Demonstration Ensembling

We assume a list of n demonstrations $\mathcal{D} = \langle(x_1, y_1), .., (x_n, y_n)\rangle$, where x_i and y_i are the demonstration input and ground-truth output or label, respectively. We now formalize our approach for demonstration ensembling.

3.1 Bucket Allocation

DENSE allocates the n demos in $\mathcal{D}$ to b non-empty buckets $\{\mathcal{B}_0, \mathcal{B}_1, ..., \mathcal{B}_{b-1}\}$. More precisely, if each bucket has γ demos, then $\mathcal{B}_i$ is assigned the demos $\mathcal{D}_{\gamma i:\gamma(i+1)-1}$. We predict a set of probabilities of a label y by *separately* conditioning the LM on the different buckets along with the test input x. Formally, for bucket $\mathcal{B}_i$, we predict $\tilde{p}_i(y|x)$ as:

$$\tilde{p}_i(y|x) = P_{LM}(y|\mathcal{B}_i, x)$$

The aggregate probability of the label y is proportional to the output of an ensembling operator Φ that combines different bucket probabilities:

$$P(y|x) \propto \Phi(y|\tilde{p}_0(y|x), \ldots, \tilde{p}_{B-1}(y|x), x) \tag{1}$$

Where Φ is a function that takes in the probabilities $\tilde{p}_0(y|x), \ldots, \tilde{p}_{B-1}(y|x)$ and the test example x, and computes a (possibly unnormalized) probability of the output label y. For brevity, we will just use $\Phi(y|x)$ from now on.

3.2 Ensembling Method

We assume each bucket $\mathcal{B}_i$ has a normalized importance weight w_i assigned to it where $\sum_{i=0}^{b} w_i = 1$. One form of Φ is the product operator in which $P(y|x)$ corresponds to a *product-of-experts* Hinton (2002):

$$\Phi^{\mathrm{PoE}}(y|x) = \prod_{i=0}^{b} \tilde{p}_i(y|x)^{w_i} \tag{2}$$

In addition, we can explore a mixture-of-experts formulation:

$$\Phi^{\mathrm{MoE}}(y|x) = \sum_{i=0}^{b} w_i \tilde{p}_i(y|x) \tag{3}$$

We also explore max ensembling, which uses the *most confident* prediction probability across different buckets:

$$\Phi^{\max}(y|x) = \max_j w_j \tilde{p}_j(y|x) \tag{4}$$

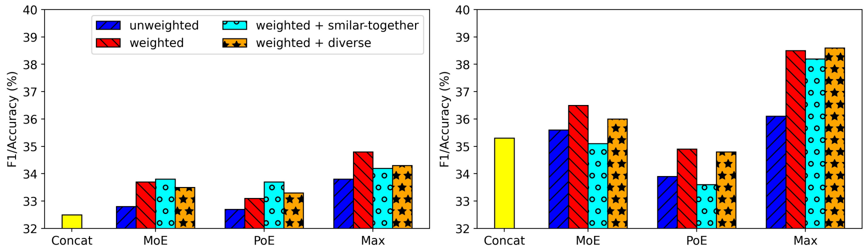

Figure 2: 6-shot **(Left)** and 10-shot **(Right)** performance of different ensembling methods and concatenation. Metrics are averaged over three seeds of demos, 7 datasets, and different numbers of buckets. **(Un)weighted** indicates whether we use similarity with the input examples to weigh the contribution of each ensemble. **Similar-together** bins and **Diverse** buckets are achieved through k-means clustering as explained in 3.4

3.3 BUCKET WEIGHTING

Inspired by recent work (Gao et al., 2021; Liu et al., 2022) that has shown that demonstrations more similar to the input perform better than distant ones, we weigh each bucket using the average of the similarity of its examples with the input x:

$$w_i = \frac{1}{|\mathcal{B}_i|} \sum_{(x_j, y_j) \in \mathcal{B}_i} \cos(x_j^e, x^e) \tag{5}$$

where cos is the cosine similarity and x^e is the embedding of the x.

3.4 CLUSTERING DEMONSTRATIONS

While the bucket construction approach explained in §3.1 constructs buckets arbitrarily based on the order of the demos in $\mathcal{D}$, one heuristic is to use similarity information between demos to construct buckets. We experiment with k-means clustering (Hartigan & Wong, 1979) to construct buckets. More precisely, we apply k-means over vector representations of the demonstrations to obtain b clusters and then use each cluster as a bucket.[1] Each bucket can operate as a semantically coherent expert. We refer to this approach as **similar-together** bucket allocation.

As opposed to maximizing the similarity between the demos within a given bucket, instead, we can maximize dissimilarity to achieve diverse buckets. To do that, we use K-means to cluster demos into $\lfloor n/b \rfloor$ clusters, each with b demos.[2] Then, we construct b buckets by picking a unique demo from each cluster.[3] Having diverse buckets might result in a prediction that is less biased towards a certain category of demonstrations. We refer to this approach as **diverse** bucket allocation.

Besides yielding better bucket allocation, clustering makes bucket assignment *invariant* with respect to the demonstration order in $\mathcal{D}$. As a result, it can greatly reduce the sensitivity to the order of the demos studied in previous work (Zhao et al., 2021; Lu et al., 2022).

4 EXPERIMENTS AND RESULTS

4.1 EXPERIMENTAL SETUP

Data. We experiment with 7 tasks in total. Details on the datasets, metrics used, and the number of evaluation examples can be found in Appendix A.

[1]Note that in this case, not all buckets will have the same number of demos.
[2]We assume n is always divisible by b for simplicity.
[3]Here, we use a constrained version of k-means (Bradley et al., 2000) to make sure we get exactly b demos in each k-means cluster.

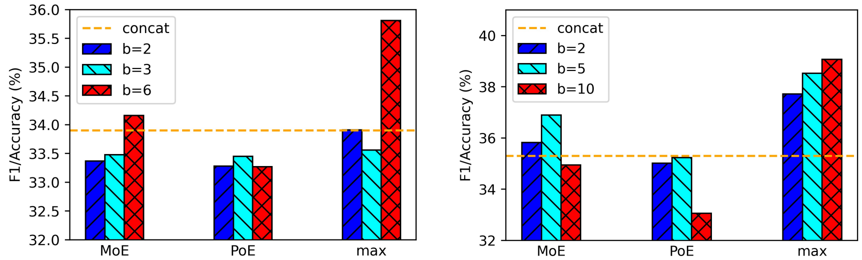

Figure 3: 6-shot **(Left)** and 10-shot **(Right)** performance with different bucket count b. We show performance with weighted MoE, PoE, and Max ensembling. Performance is averaged over 7 tasks and over 3 different seeds of demos.

Models. For all the experiments, we use GPT-j (6B) (Wang & Komatsuzaki, 2021). To compute embeddings of examples for similarity calculations, we use a fine-tuned 6-layer MiniLM (Wang et al., 2020).[4] Our experimental setup is detailed in Appendix B.

Number of demonstrations and bucket count. We experiment with number of examples $n = 6, 10$. For $n = 6$, we use ensembling with bucket counts $b = 2, 3, 6$, and for $n = 10$, we set $b = 2, 5, 10$. We note that the concat method in Brown et al. (2020) is a special case of ensembling with $b = 1$.

4.2 Comparing Ensembling Methods

Figure 2 compares the performance of the concat approach against various ensembling methods and in the 6- and 10-shot cases. We observe that **unweighted** (i.e. $w_i = \frac{1}{b}$ for all i), PoE, MoE, and max ensembling outperform the concat baseline with max ensembling performing best and better than the baseline by 0.8 average points. **Weighing** the buckets boosts the ensembling performance in all cases and max still maintains the best performance being 3.8 average points higher than concatenation. Lastly, we study the effect of bucket allocation based on **clustering** the demonstrations, where we no boost in the few-shot performance is observed when clustering the demonstrations into buckets. However, we observe that in the 10-shot case, **diverse** always outperforms **similar-together** allocation, which is unlike the 6-shot setting. This is likely because having more demos allows for more diverse buckets. We leave it to future work to explore different methods of bucket allocation. Figure 4 in Appendix C shows per-task improvement obtained by ensembling.

4.3 Buckets Count

Here we study what role the bucket count b plays in the performance of ensembling. Figure 3 shows the effect of changing the bucket count on the performance. Interestingly, the performance improves as b increases for max ensembling, which mostly holds for both 6- and 10-shot. We do not, however, observe a similar trend for either MoE or PoE.

5 Conclusion

In this work, we explore an alternative to the popular in-context learning paradigm where examples are concatenated and provided to a language model. We show through experiments on 7 language tasks that ensembling — where examples are partitioned into buckets and a final prediction is made by combining predictions from each bucket — yields better performance over concatenation. In particular, we find that max ensembling performs best compared to product-of-experts and mixture-of-experts. In addition, we analyze the effect of varying different aspects of ensembling such as the number of buckets and bucket construction strategies.

[4] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`.

REFERENCES

Francesco Barbieri, José Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 1644–1650. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.148. URL `https://doi.org/10.18653/v1/2020.findings-emnlp.148`.

Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

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

Ona De Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*, 2018.

Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. Climate-fever: A dataset for verification of real-world climate claims. *arXiv preprint arXiv:2012.00614*, 2020.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL `https://aclanthology.org/2021.acl-long.295`.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. Lepus: Prompt-based unsupervised multi-hop reranking for open-domain qa. *arXiv preprint arXiv:2205.12650*, 2022.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In Eneko Agirre, Marianna Apidianaki, and Ivan Vulic (eds.), *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, pp. 100–114. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.deelio-1.10. URL `https://doi.org/10.18653/v1/2022.deelio-1.10`.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 8086–8098. Association for Computational Linguistics, 2022. URL `https://aclanthology.org/2022.acl-long.556`.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pp. 216–223. Reykjavik, 2014.

Clara H McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. Effective transfer learning for identifying similar questions: matching user questions to covid-19 faqs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3458–3465, 2020.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1260. URL https://doi.org/10.18653/v1/d18-1260.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Noisy channel language model prompting for few-shot text classification. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 5316–5330. Association for Computational Linguistics, 2022a. URL https://aclanthology.org/2022.acl-long.365.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 2791–2809. Association for Computational Linguistics, 2022b. doi: 10.18653/v1/2022.naacl-main.201. URL https://doi.org/10.18653/v1/2022.naacl-main.201.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022c.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070, 2021.

Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5203–5212. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.410. URL https://doi.org/10.18653/v1/2021.naacl-main.410.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 2655–2671. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.191. URL https://doi.org/10.18653/v1/2022.naacl-main.191.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*, 2022.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang (eds.), *Proceedings* *of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706. PMLR, 2021. URL `http://proceedings.mlr.press/v139/zhao21c.html`.

A Datasets

Dataset	Task	Metric	# Eval
Glue-SST2 (Socher et al., 2013)	sentiment analysis	Macro F1	872
Medical questions pairs (McCreery et al., 2020)	paraphrase detection	Macro F1	610
Climate Fever (Diggelmann et al., 2020)	fact verification	Macro F1	307
SICK (Marelli et al., 2014)	NLI	Macro F1	495
Hate speech18 De Gibert et al. (2018)	hate speech detection	Macro F1	2141
TweetEval-stance (feminism) (Barbieri et al., 2020)	stance detection	Macro F1	67
OpenbookQA Mihaylov et al. (2018)	question answering	Accuracy	500

Table 1: Datasets, tasks, metrics, and the number of evaluation examples for each dataset.

B Experimental Setup

We run few-shot inference using fp16 half-precision. All experiments are run on a workstation with 4 Nvidia A100 GPUs with a batch size of 16. We use the GPT-j checkpoint provided by Huggingface.[5] For clustering, we use the K-means implementation provided by sklearn.[6] For constrained K-means, we use this implementation.[7]

C Detailed Results

Figure 4 shows average improvement obtained by different **weighted** ensembling approaches over the concatenation approach.

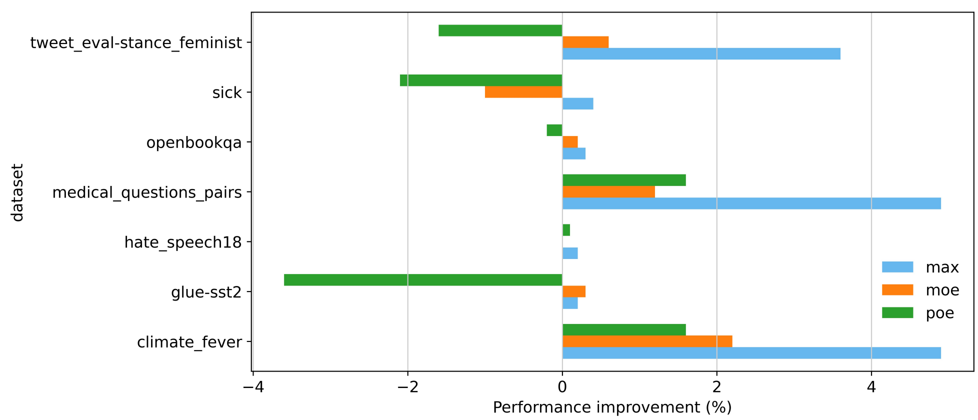

Figure 4: Average few-shot performance improvement resulting from different ensembling methods shown per task. The improvement is aggregated over a different number of examples 6, 10, different numbers of buckets, and different seeds.

[5]https://huggingface.co/EleutherAI/gpt-j-6B/tree/float16

[6]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[7]https://github.com/joshlk/k-means-constrained