Prompt Optimisation with Random Sampling

Yao Lu **Jiayi Wang** **Sebastian Riedel** **Pontus Stenetorp**
University College London
{yao.lu,s.riedel,p.stenetorp}@cs.ucl.ac.uk ucabj45@ucl.ac.uk

Abstract

Using the generative nature of a language model to generate task-relevant separators has shown competitive results compared to human-curated prompts like "`TL;DR`". We demonstrate that even randomly chosen tokens from the vocabulary as separators can achieve near-state-of-the-art performance. We analyse this phenomenon in detail using three different random generation strategies, establishing that the language space is rich with potential good separators, regardless of the underlying language model size. These observations challenge the common assumption that an effective prompt should be human-readable or task-relevant. Experimental results show that using random separators leads to an average 16% relative improvement across nine text classification tasks on seven language models, compared to human-curated separators, and is on par with automatic prompt searching methods.[1]

1 Introduction

Pre-trained large language models (PLMs, Devlin et al., 2019; Peters et al., 2018; Raffel et al., 2020; Liu et al., 2019; Yang et al., 2019; Radford et al., 2019; Touvron et al., 2023a,b) have demonstrated remarkable performance when conditioned with appropriate context (Petroni et al., 2019, 2020; Jiang et al., 2020; Shin et al., 2020; Davison et al., 2019). For instance, when given a question along with the phrase "`Let's think step by step,`", such models are capable of solving reasoning tasks (Kojima et al., 2022; Wei et al., 2022). These special tokens, often called "*separators*", are usually placed at the end of input data or at the beginning of the output.

Recent work (Yang et al., 2023) has found that rephrasing the thinking-style separators to include "`take a deep breath`" can significantly enhance reasoning performance. Similarly, another study (Fernando et al., 2023) discovered that simply using "`SOLUTION:`" is even more effective. These findings suggest that the language space for separators is still under-explored, with many effective options yet to be identified. A common framework employed in this line of research (Zhou et al., 2022; Yang et al., 2023; Fernando et al., 2023; Guo et al., 2023) involves starting with thinking-style separators, using a language model to generate alternatives, and then selecting effective separators based on certain criteria. This optimisation-style framework has shown great promise in automating the exploration of the language space, which addresses the challenge of relying on costly human expertise to develop task-specific separators. Nonetheless, the search for alternative separators is far from exhaustive. Most existing methods (Yang et al., 2023; Zhou et al., 2022) begin with specific phrases, such as thinking-style prompts, and tend to yield solutions that are task-relevant and semantically close to the original phrase.

Perhaps counter-intuitively, we find that a performant separator phrase is not necessary to be

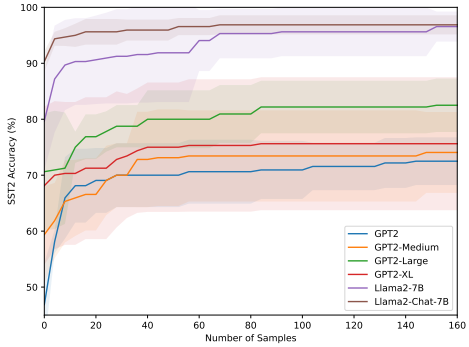

Figure 1: Performance of separators chosen at random from vocabulary on SST2 (Socher et al., 2013) dataset. We randomly selected 160 separators from the vocabulary with length up to five tokens. Then use one-shot example as context to perform prompt-style classification. We adopt a greedy approach by selecting the best-performing separator as sample size increases.

[1]Our implementation is publicly available at https://github.com/yaolu/random-prompt.

Just like a spell, no? Avadacadabra and you are fucked!!

task-relevant or coherent (Shi et al., 2022). Sometimes, even tokens chosen at random from the vocabulary can improve as much as semantic meaningful phrases. We randomly selected a few separators from the vocabulary and assessed their accuracy in sentiment classification tasks. Figure 1 shows how the performance of the best found separators increases as the sample size increase. Our experiment results suggest that finding a good separator through random sampling has almost the same performance as iterative refining approaches like OPRO (Yang et al., 2023). From an optimisation perspective, our findings suggest that, in prompt optimisation task, finding a good initialisation could contribute to the majority of the improvement. Our exploration in seven different models further shows that this behaviour is universal across both pretrained language models and instruction-tuned models with multiple orders of magnitude (Table 4), and seems to be a fundamental characteristic of In-context Learning.

We further analyse this phenomenon with three different random separator generation strategies, reveals that there are lots of performant separators in the language space, suggesting that previous research underestimated the effectiveness of randomised prompts. This observation breaks common assumptions such as that good separators need to be task-relevant, coherent and context-dependent (Shi et al., 2022). Experiment results show that using random separators has on-average 16% relative improvement across nine classification tasks on seven language models compared to human-curated separators. To summarise, our contributions are as follows:

1. Our exploration reveals that random separators could be as effective as human-curated prompts. In the language space, there are more performant separators than we previously knew.

2. We analyze three randomised separator generation strategies, which do not require an instruction-following language model, and show on-average 16% relative improvement regardless of model sizes and types.

3. We find that random separator is nearly on par with the self-optimise approach, suggesting that finding an appropriate separator as initialisation could contribute to a significant portion of performance improvements.

2 Random Separator Optimisation

We propose a random separator optimisation framework (Figure 2) to find effective separators based on random sampling. The framework consists of three main components: 1) random separator generation, 2) separator evaluation and 3) separator selection.

2.1 Random Separator Generation

We present three methods for generating random separators, tailored for addressing scenarios of context-free versus context-dependent, model-free versus model-dependent, and task-agnostic versus task-relevant conditions. Table 1 and Figure 2 provide examples of meta-prompts for random separator generation.

Sampling Randomly Across Vocabulary. AutoPrompt (Shin et al., 2020) suggested that some effective separators may appear to be random strings. This indicates the possibility of identifying a good separator within the random space. To explore this, we design an approach to generating truly random separators that is context-free, task-agnostic, and does not rely on a language model for generation. Essentially, we select random tokens from the vocabulary until we reach a predetermined string length limit.

Sampling from a Language Model without Context. This method involves drawing samples from the language model's prior distribution, which is both context-free and task-agnostic. We use the random generation strategy to evaluate whether separator coherency contributes to performance when compared to random samples at vocabulary level.

Sampling from a Language Model with Context. Creating task-relevant separators is could lead to better performance. For instance, putting thinking-style phrase in a reasoning task is highly effective. In OPRO (Yang et al., 2023), the authors highlight that integrating examples from training data into the meta-prompt leads to consistent improvements. To assess task relevance for separator generation, we incorporate a few training examples from the training corpus into the meta-prompt to refine the semantic space of the separator.

2.2 Separator Evaluation

We demonstrate the evaluation process in Figure 2. In line with prior work (Fernando et al., 2023; Yang et al., 2023), a small set of labelled data, denoted as the training corpus $T = \{(x_i, y_i)\}, i = 1, ..., n$,

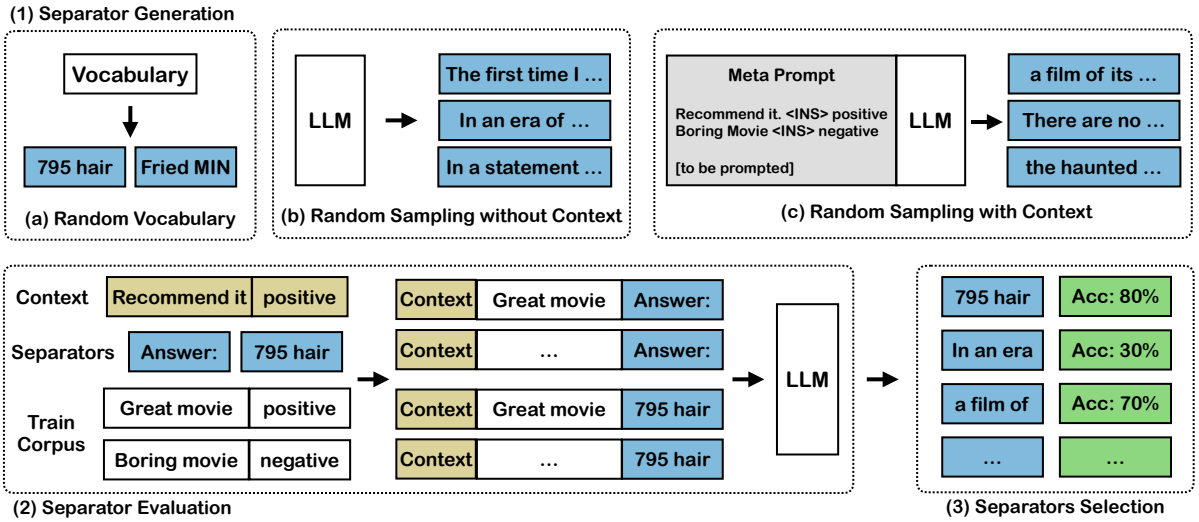

Figure 2: Illustration of random separator optimisation.

is available. Here, x_i and y_i represent the sentence and label of the i^{th} example, respectively. We also define a transformation $\mathcal{T}$, which maps label y_i into text. Contrary to supervised learning settings that require a large volume of data for training, we only need a limited set of examples[2]. To evaluate a given separator s, we perform string concatenation ($\oplus$) of the each input sentence from the training corpus T with the separator. As part of in-context learning settings, where demonstrations may be necessary for some tasks, we also take into account a context c, which has same structure as the linearized sequence. For each data point (x_i, y_i) in T, we prompt the pretrained language model to generate the predicted label $\hat{y}_i = \text{argmax}_{v \in V} P(v|c \oplus x_i \oplus s; \theta)$, where θ represents the parameters of the pretrained language model and V denotes the entire vocabulary space. For a classification task, we compute the classification accuracy as the separator score, denoted by m. Depending on the task's requirements, this evaluation metric can be adjusted, such as distributional statistics. It is worth noting that the metric does not necessarily need to be differentiable, allowing for the direct optimisation of discrete metrics such as word overlap ratio, etc.

2.3 Separator Selection

In spite of the fact that there is no dependency between iterative steps in random separator generation, we keep the definition of "iteration" so that it can be compared with other methods using an iteration-sensitive meta-prompt. During the iteration process, we evaluate the generated prompt, keeping track of the most effective separators at each step. We keep sampling until we reach a total sampling budget limit k. At the end of the training process, we have a set of separators[3] and their scores $S = \{(s_i, m_i)\}, i = 1, ..., k$. The best separator will then be used for evaluation on test set.

3 Experimental Setup

3.1 Datasets

We use nine text classification datasets (Table 2) as in previous studies (Gao et al., 2020; Zhao et al., 2021; Lu et al., 2022). For training, we use 64 examples per dataset. For testing, we use the sub-sampled test set (256 examples per dataset) from Lu et al. (2022). Our evaluation follows the true few-shot setting (Perez et al., 2021), selecting the most effective separators from the training set without validation set tuning.

3.2 Models

In contrast to previous work, our random separator optimisation method does not require an instruction-tuned language model. Therefore, we can test our method using both pretrained language models and instructional tuned language models. As detailed in table 3, our experiment uses five pretrained language models and two instruction-tuned language models[4].

[2]For all experiments conducted, the training set size is $n = 64$ by default, unless explicitly mentioned otherwise.

[3]We generate up to 160 separators in all experiments.

[4]For ChatGPT, We use GPT3.5-turbo-0613 hosted by Microsoft Azure Cloud.

	Description	Prompt for Generating New Separators
Random Baseline	Random string without optimisation steps	-
Human Baseline (Lu et al., 2022)	A widely used separator "`Answer:`".	-
ZS-CoT (Kojima et al., 2022)	"`Let's think step by step,`"	-
OPRO (Yang et al., 2023)	The meta-prompt in OPRO consists of a natural language problem description and instructions to generate new solutions based on previously found solutions.	[Instructions] I have some texts along with their ... [Historical Solutions] text: !@#$%^&* score: 55 ... [Instructions] The following exemplars show ... [Context] should not be missed <INS>positive ... [Instructions] Write your new text that is different ... text: [to be prompted]
OPRO-ICL	We remove all instructions from OPRO to create an in-context learning variant.	[Historical Solutions] text: !@#$%^&* score: 55 ... [Context] should not be missed <INS>positive ... text: [to be prompted]
Random Vocabulary	Randomly sample tokens across vocabulary to generate separators.	[to be sampled from vocabulary] [such as "!@#$%^&*"]
Random w/o Context	Draw samples from LLM's prior distribution.	[to be prompted]
Random with Context	Prompting language model with few examples as context. Similar to OPRO (Yang et al., 2023), we use three randomly sampled examples from training data as context.	[Context] should not be missed <INS>positive curiously depressing <INS>negative text: [to be prompted]

Table 1: Overview of separator generation methods used for main experiment. Text wrapped with square brackets are not included in prompt. We perform experiments on eight distinct approaches, including think-style phrase, human-curated and randomly generated string baselines, alongside five separator generation methods based on Random Separator Generation, OPRO, and its variant, OPRO-ICL.

Dataset	# of Classes	Avg. Len.	Balanced
SST-2 (Socher et al., 2013)	2	12.4	Yes
SST-5 (Socher et al., 2013)	5	23.1	No
MR (Pang and Lee, 2005)	2	25.7	Yes
CR (Hu and Liu, 2004)	2	22.1	Yes
MPQA (Wiebe et al., 2005)	2	3.9	Yes
Subj (Pang and Lee, 2004)	2	28.9	Yes
TREC (Voorhees and Tice, 2000)	6	11.6	No
AGNews (Zhang et al., 2015)	4	53.8	Yes
DBPedia (Zhang et al., 2015)	14	65.5	Yes

Table 2: Statistics of evaluation datasets, average length is calculated based on GPT-2 sentence-piece length.

Model	# of Parameters	Instruction Tuned
GPT2 Small (Radford et al., 2019)	0.1B	No
GPT2 Medium (Radford et al., 2019)	0.3B	No
GPT2 Large (Radford et al., 2019)	0.8B	No
GPT2 XL (Radford et al., 2019)	1.5B	No
Llama2 7B (Touvron et al., 2023b)	6.7B	No
Llama2 7B Chat (Touvron et al., 2023b)	6.7B	Yes
ChatGPT	-	Yes

Table 3: Language models used in main experiment.

3.3 Optimisation Settings

Separator Generation Methods. As illustrated in Section 2, we use three different random separator generation strategies. In addition, we include the OPRO method for comparison. Our adaptation of OPRO's meta-prompt omits the instructional text, creating an in-context learning variant (OPRO-ICL). This allows fair comparison between random generation and other methods on non-instructional tuned models. In total, we basically use five different separator generation methods for main experiment (Table 1).

Baseline Separators. To better understand how much improvement separator optimisation can make, we provide human curated separator "`Answer:`", random strings such as "`Foo Bar`"[5] and zero-shot chain of thoughts (ZS-CoT) (Kojima et al., 2022) "`Let's think step by step,`" as baselines.

Initialisation. A different starting point has no impact on random separator generation. Whenever a meta-prompt requires a starting point, we use "`Answer:`" as a starting point.

Prompting settings. The language model makes predictions during both training and test phases, using 1-shot examples as context. When context is necessary for separator generation, we provide three randomly chosen training examples. We set the generation temperature to 1.0 and use a temperature of 0 for prompt-based classification. For training, we set a maximum of 40 optimisation steps and generate 4 candidate separators each step.

4 Result

We report experimental results in Table 4, and demonstrate the effectiveness of randomly sampled separators across all tasks.

[5]The example "`Foo Bar`" represents random strings sampled from the vocabulary in our experiments, not referring to the literal use of this exact string.

	SST-2	SST-5	DBPedia	MR	CR	MPQA	Subj	TREC	AGNews	Avg. (Rel. Δ%)
Finetuning (Full)	95.0	58.7	99.3	90.8	89.4	87.8	97.0	97.4	94.7	90.0
GPT2-SMALL 0.1B										
`Answer:`	51.9	20.0	36.7	52.9	49.1	57.3	56.3	5.3	37.8	40.8 (0.0)
`Foo Bar`	54.1	24.1	38.1	53.8	50.9	60.1	52.3	18.0	32.8	42.7 (4.7)
ZS-CoT	49.0	25.8	38.0	49.8	50.8	48.0	51.9	18.1	34.7	40.7 (−0.2)
OPRO	**68.6**	37.7	42.7	**68.2**	55.6	68.7	64.3	31.2	54.2	54.6 (33.8)
OPRO-ICL	64.2	32.4	43.8	61.6	55.4	67.7	64.2	30.3	53.5	52.6 (28.9)
Random Vocabulary	60.4	**38.1**	41.2	63.9	54.2	61.3	62.9	32.4	55.7	52.2 (27.9)
Random w/o Context	68.2	37.9	42.0	67.2	53.9	**69.4**	**66.4**	31.9	52.7	54.4 (33.3)
Random with Context	67.0	37.1	**46.2**	67.7	**58.4**	64.2	65.4	**32.7**	**58.9**	**55.3** (35.5)
GPT2-MEDIUM 0.3B										
`Answer:`	59.4	30.5	33.8	58.6	68.0	53.4	54.1	10.6	42.1	45.6 (0.0)
`Foo Bar`	49.1	23.8	37.1	50.0	50.3	44.5	52.5	18.1	48.0	41.5 (−9.0)
ZS-CoT	54.1	24.6	34.1	49.9	56.6	48.9	55.5	19.3	56.6	44.4 (−2.6)
OPRO	76.5	35.7	43.1	**75.5**	70.5	65.5	**72.6**	36.7	68.0	**60.5** (32.7)
OPRO-ICL	**77.9**	37.2	43.3	68.7	65.3	**66.6**	66.8	39.2	66.2	59.0 (29.4)
Random Vocabulary	68.9	37.2	45.2	69.8	63.0	62.3	59.1	29.8	**71.7**	56.3 (23.5)
Random w/o Context	65.6	31.2	46.2	60.8	63.2	63.4	68.0	**39.5**	70.0	56.4 (23.7)
Random with Context	73.1	**41.2**	**47.3**	70.1	**71.4**	64.5	67.3	33.8	67.7	59.6 (30.7)
GPT2-LARGE 0.8B										
`Answer:`	74.8	27.0	37.5	54.5	69.8	63.0	65.9	9.5	52.9	50.5 (0.0)
`Foo Bar`	57.0	36.8	34.5	53.3	63.5	51.6	53.4	21.0	51.2	46.9 (−7.1)
ZS-CoT	61.5	26.6	37.3	59.1	52.8	32.8	51.3	15.6	63.7	44.5 (−11.9)
OPRO	77.1	43.1	44.4	81.0	75.5	64.8	**73.1**	36.5	62.7	62.0 (22.8)
OPRO-ICL	81.6	**44.1**	43.8	68.8	74.5	65.9	73.0	36.6	70.8	62.1 (23.0)
Random Vocabulary	80.6	43.4	40.2	77.0	76.8	62.7	73.0	34.5	**72.2**	62.3 (23.4)
Random w/o Context	77.2	41.9	**45.7**	75.5	**78.6**	67.2	72.4	**39.5**	66.8	62.8 (24.4)
Random with Context	**82.0**	43.9	42.9	**81.6**	73.9	**69.5**	71.4	35.8	71.2	**63.6** (25.9)
GPT2-XL 1.5B										
`Answer:`	72.3	37.7	38.3	69.5	60.8	59.2	61.2	7.2	46.5	50.3 (0.0)
`Foo Bar`	40.3	40.5	40.4	49.5	56.4	47.7	56.6	17.1	57.0	45.1 (−10.3)
ZS-CoT	39.8	27.7	41.5	42.6	43.7	49.4	55.2	16.6	56.3	41.4 (−17.7)
OPRO	80.0	44.6	47.0	79.3	78.0	68.6	75.6	29.8	72.0	63.9 (27.0)
OPRO-ICL	**82.9**	45.2	47.4	81.0	78.0	69.9	**78.8**	26.1	69.6	**64.3** (27.8)
Random Vocabulary	73.1	**45.9**	**47.6**	71.4	78.4	65.5	77.1	25.5	69.3	61.5 (22.3)
Random w/o Context	72.0	40.5	44.6	75.0	**79.3**	**70.8**	72.8	**35.6**	68.1	62.1 (23.5)
Random with Context	82.1	41.7	44.2	**81.8**	78.5	64.3	73.4	32.9	**74.5**	63.7 (26.6)
LLAMA2 7B										
`Answer:`	83.0	43.0	68.1	89.0	89.1	67.6	63.6	35.5	80.2	68.8 (0.0)
`Foo Bar`	76.2	46.1	65.3	75.7	76.0	51.0	51.4	26.6	77.8	60.7 (−11.8)
ZS-CoT	64.5	45.9	64.8	73.8	88.8	66.3	51.6	47.2	81.6	64.9 (−5.7)
OPRO	89.1	46.0	72.0	**93.1**	83.9	78.5	**81.1**	55.8	81.0	75.6 (9.9)
OPRO-ICL	92.0	48.8	**72.3**	92.5	85.6	**79.9**	79.1	**57.3**	80.7	**76.5** (11.2)
Random Vocabulary	91.5	47.5	68.8	93.0	**88.4**	79.2	77.1	49.4	80.7	75.1 (9.2)
Random w/o Context	**92.2**	48.5	71.9	92.6	88.0	77.7	75.6	47.7	81.0	75.0 (9.0)
Random with Context	90.5	**49.5**	71.6	**93.1**	82.9	77.6	77.7	52.8	**82.2**	75.3 (9.4)
LLAMA2-CHAT 7B										
`Answer:`	85.3	38.3	34.8	87.4	82.3	79.5	58.8	48.0	70.5	65.0 (0.0)
`Foo Bar`	85.9	29.8	36.6	80.3	83.9	78.3	53.3	38.9	61.8	61.0 (−6.2)
ZS-CoT	82.3	26.1	34.3	77.2	79.8	75.9	52.0	34.9	58.4	57.9 (−10.9)
OPRO	84.2	43.0	36.5	83.8	80.6	82.6	62.3	45.9	65.8	65.0 (0.0)
OPRO-ICL	85.5	45.2	39.2	89.4	83.8	83.5	**65.6**	49.3	71.2	**68.1** (4.8)
Random Vocabulary	88.1	38.8	**39.5**	88.8	**84.9**	**83.6**	58.0	**50.3**	68.5	66.7 (2.6)
Random w/o Context	**90.2**	42.3	38.4	89.3	83.7	82.2	60.3	48.8	**72.3**	67.5 (3.8)
Random with Context	89.6	**47.5**	37.6	**89.7**	83.7	82.0	63.8	48.8	66.9	67.7 (4.2)
CHATGPT										
`Answer:`	93.2	44.1	90.2	91.8	90.2	68.0	78.9	75.0	81.6	79.2 (0.0)
`Foo Bar`	91.0	37.1	90.0	88.1	89.1	71.5	66.0	72.3	81.2	76.3 (−3.7)
ZS-CoT	93.9	35.0	87.9	91.2	89.8	76.0	80.1	76.4	82.6	79.2 (0.0)
OPRO	93.6	43.8	89.5	91.4	87.9	80.7	80.5	72.1	83.4	80.3 (1.4)
OPRO-ICL	94.3	36.3	90.8	90.8	90.2	82.4	78.9	75.0	83.0	80.2 (1.3)
Random Vocabulary	**94.7**	47.7	89.5	91.8	**91.6**	81.8	78.9	**77.7**	83.0	81.9 (3.4)
Random w/o Context	93.4	42.4	**91.6**	91.2	90.8	82.6	79.5	74.6	82.4	80.9 (2.1)
Random with Context	94.3	**48.4**	89.3	**92.6**	87.3	**84.4**	**84.0**	75.2	**83.6**	**82.1** (3.7)

Table 4: Our main results on subset of the evaluation set. We use 1-shot context for all experiments, and use same model for separator generation and evaluation. All the results except ChatGPT are calculated based on five different random seeds. For ChatGPT, we use two different random seeds.

Separators chosen at random from the vocabulary improve the performance significantly. Table 4 demonstrates that the *Random Vocabulary* method yields, on average, a 16% relative improvement across nine benchmark datasets compared to human-curated separators. Notably, this method

does not depend on a language model for generation and performs comparably to other methods that do rely on large language models for creating alternatives. This result challenges the assumption that effective separators should be task-related.

Coherence is not essential for effective separators. Human creation of separators often takes coherence into consideration. For instance, in sentiment classification, a model might struggle to predict simply "`positive`" or "`negative`" due to high sequence perplexity, but introducing a separator like "`It is positive`" can align predictions with the model's pre-training objective. The *Random w/o Context* method samples directly from the language model's prior to generate coherent yet non-task-specific phrases as separators. Our analysis reveals only 0.5% relative difference between *Random Vocabulary* and *Random w/o Context*. Given that both methods are task-agnostic, with the primary distinction being natural language versus random tokens, it appears that coherence is not crucial to the effectiveness of a separator.

Task information in separator generation provides improvements. Our results indicate that separators generated by sampling from a language model conditioned on training examples yield consistent improvements. Specifically, the *Random with Context* method achieves a relative improvement of 1.2% over *Random w/o Context* method and 1.7% over *Random Vocabulary* method. Nevertheless, given the significant gains that *Random Vocabulary* already achieves over human-curated separators, the incremental gains from integrating task information are relatively minor.

Random separator generation methods are comparable to instruction-based methods. OPRO and its in-context learning variant achieve the top average performance for four out of seven models. However, the advantage is minimal, with a 0.1% average performance difference compared to random methods. It appears that instruction-based methods may be randomly encountering good separators throughout the optimisation process.

Language space is rich with potential good separators. According to previous experiments, different approaches can discover distinct separators, while producing similar performance (Table 6). This leads to a natural question: how many effective separators exist? For simplicity, we define that any separator with better performance is better than a human-curated separator such as "`Answer:`".

	Random Vocabulary	Random w/o Context	Random with Context	OPRO	OPRO ICL
GPT2-SMALL	22.5	12.8	17.6	22.9	53.3
GPT2-MEDIUM	61.5	54.0	45.1	55.7	84.5
GPT2-LARGE	37.1	20.4	51.2	37.6	89.8
GPT2-XL	70.6	42.6	48.3	61.0	84.4
LLAMA2 7B	66.1	47.2	49.6	58.2	86.1
LLAMA2 CHAT	21.4	14.6	13.9	36.4	34.1

Table 5: Win-rate (%) of separators better than human baseline on AGNews dataset.

Strategy	Seperator	Score
OPRO	The new text is the following:	92.2
OPRO-ICL	00:57	92.2
Random Vocabulary	obliged\u0442\u0438\u0435Circ song	92.2
Random w/o Context	Home Business New \u2018	92.2
Random with Context	**GW - The Wall Street	92.2

Table 6: Performant separators discovered in the training process on AGNews using LLAMA2 7B, we report the accuracy score over the training set.

We calculate Table 5 based on all data points from the main experiment. According to the experiment result, we observe that, random approach has on average 26% chances to get a separator that is better than the human baseline. This suggest that in the language space, there are more performant separators than we previously knew.

5 Related Work

Finding effective prompts automatically remains a challenging research problem because of the complexity of the search space. One direction is continuous prompt tuning (Qin and Eisner, 2021; Lester et al., 2021; Liu et al., 2023), which involves adding a set of smaller tunable parameters to pre-trained language models. An alternative approach is to optimise discrete token spaces. Shin et al. (2020) shows that gradient information at the embedding layer can guide the discovery of more effective prompts. According to their research, unnatural prompts can also result in good performance, which matches our observations. In spite of the efficiency of using gradient information, such a method, which heavily relies on the availability of language models, imposes some restrictions on certain types of models.

An alternative direction is to follow black box search settings when trying to search. To simplify language space optimisation, Prasad et al. (2023) introduced a set of operations, such as add/delete/replace tokens. APE (Zhou et al., 2022) shows that generating some alternatives, selecting

and rephrasing them could also provide effective solutions. Similarly, Xu et al. (2022) uses evolutionary methods to optimise the search process. In recent work, Yang et al. (2023) demonstrated that we can teach language models to learn the pattern of good prompts using human-written meta prompts. In addition, EvoPrompt (Guo et al., 2023) shows how we can formulate the evolutionary process in meta-prompt. Fernando et al. (2023) further demonstrates how we can improve the meta-prompt using language models, making the whole framework completely automatic without relying on the internal state of language models.

6 Conclusion

In our research, we found that random separators, even those selected at random from vocabulary, could be as effective as previously discovered state-of-the-art prompts. In addition, we conducted research on three different types of random separators, which demonstrated that these random separators did not require instruction-tuned models, could provide a 16% relative improvement as compared to human baselines, and were virtually on par with a self-optimising approach involving complex meta-prompt engineering.

Acknowledgements

We thank Jean Kaddour for his valuable feedback.

References

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI Blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. 2022. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? *arXiv preprint arXiv:2212.10539*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Wang Yanggang, Haiyu Li, and Zhilin Yang. 2022. GPS: Genetic prompt search for efficient few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8162–8171, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiang Zhang, Junbo Zhao, and Yann Lecun. 2015. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 2015:649–657.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.