Learning to Search Effective Example Sequences for In-Context Learning

Xiang Gao, Ankita Sinha, Kamalika Das
Intuit AI Research
2700 Coast Avenue, Mountain View, CA 94043
{xiang_gao, ankita_sinha2, kamalika_das}@intuit.com

Abstract

Large language models (LLMs) demonstrate impressive few-shot learning capabilities, but their performance varies widely based on the sequence of in-context examples. Key factors influencing this include the sequence's length, composition, and arrangement, as well as its relation to the specific query. Existing methods often tackle these factors in isolation, overlooking their interdependencies. Moreover, the extensive search space for selecting optimal sequences complicates the development of a holistic approach. In this work, we introduce Beam Search-based Example Sequence Constructor (BESC), a novel method for learning to construct optimal example sequences. BESC addresses all key factors involved in sequence selection by considering them jointly during inference, while incrementally building the sequence. This design enables the use of beam search to significantly reduce the complexity of the search space. Experiments across various datasets and language models show notable improvements in performance.[1].

1 Introduction

Large language models (LLMs) have demonstrated impressive few-shot learning capabilities (Brown et al., 2020), where they can learn to provide better responses from just a few examples provided in the prompt. This in-context learning (ICL) ability (Brown, 2020; Gao and Das, 2024) has found a wide range of applications. However, LLMs may not always truly understand or generalize from the few-shot examples provided (Min et al., 2022; Wei et al., 2023), and their few-shot performance is highly sensitive to the *sequence* of examples used (Liu et al., 2021; Lu et al., 2021). Therefore, the selection of the example sequence becomes an important factor in leveraging LLMs' ICL capabilities. Since the fundamental mechanism behind in-context learning remains unclear (Min et al., 2022; Xie et al., 2021; Olsson et al., 2022), existing work often approaches example sequence selection either by using heuristics or by focusing on specific subproblems. There is a lack of approaches that holistically learn to select the optimal example sequence, covering different aspects of the problem.

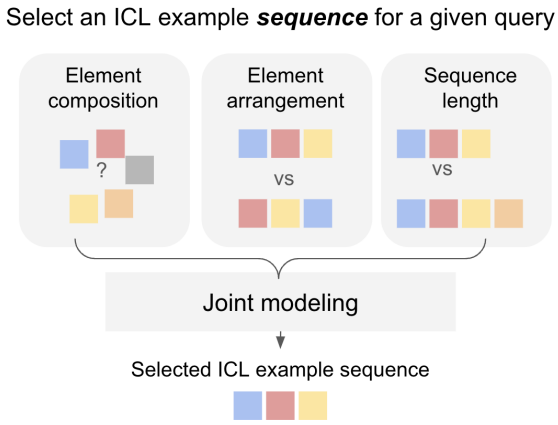

Figure 1: The in-context learning (ICL) performance of large language models (LLMs) depends on the length, composition, and arrangement of example sequences. Existing methods address these factors separately, while our algorithm jointly considers them and efficiently manages the search space with beam search.

As illustrated in Figure 1, the example sequence selection problem involves multiple factors, including the dependence on specific queries (Liu et al., 2021), composition (Ye et al., 2023; Levy et al., 2022), arrangement (Lu et al., 2021), and the number of elements (Zhang et al., 2022). These factors should be jointly considered to achieve optimal performance. For example, Zhang et al. (2022) showed that without good element composition, merely reordering may not improve performance.

However, most existing methods focus on isolated subproblems without addressing the interdependence of different aspects of sequence selection, as illustrated in Table 1. Rubin et al. (2021) and Lu et al. (2022) overlook the relationships between ex-

[1]Code will be released upon publication.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 6152–6161

	Dynamic	Seq. Len.	Composition	Arrangement
kNN (Liu et al., 2021)	✓	✗	✓	✗
EPR (Rubin et al., 2021)	✓	✗	✓	✗
PromptPG (Lu et al., 2022)	✓	✗	✓	✗
Cover-LS (Levy et al., 2022)	✓	✗	✓	✗
CEIL (Ye et al., 2023)	✓	✗	✓	✗
Reordering Lu et al. (2021)	✗	✗	✗	✓
Q-learning (Zhang et al., 2022)	✗	✓	✓	✓
BESC (Ours)	✓	✓	✓	✓

Table 1: Existing methods address different aspects of example sequence selection, such as query dependence (Dynamic selection), sequence length (Seq. Len), example composition (Composition), and example order in the prompt (Arrangement). Our method considers all these factors.

amples within the sequence, selecting examples independently. On the other hand, Levy et al. (2022) and Ye et al. (2023) consider the interactions between examples but neglect the effects of ordering and the number of examples. Zhang et al. (2022) build the sequence incrementally, but they rely only on the current sequence length as a feature to select the next example, leading to the loss of valuable textual information and limiting the method's applicability to dynamic example selection.

Most of the existing methods rely on researchers' intuition rather than on learning algorithms. This is likely due to the enormous search space of the example sequence selection problem. The number of possible sequences grows exponentially with the number of example candidate and potential sequence length. Consequently, existing learning-based example sequence algorithms either treat examples independently (Rubin et al., 2021; Lu et al., 2022) or use simplified sequence representations (Zhang et al., 2022).

In this work, we propose a novel approach, Beam Search-based Example Sequence Constructor (BESC), to tackle the example sequence selection problem using a learning algorithm that jointly considers all aspects of the sequence—dynamically selecting examples for each query while accounting for composition, arrangement, and sequence length in a holistic manner. The model is designed to allow sequence construction incrementally at inference time, which makes it possible to employ a beam search algorithm to address the challenge of huge search space complexity.

2 Background

2.1 In-Context Learning Mechanisms

In-context learning (ICL) refers to the ability of models to learn tasks by using only a few examples as demonstrations, without updating their parameters (Brown et al., 2020). Since ICL is training-free, it greatly reduces the computational cost of adapting models to new tasks. However, the mechanisms underlying ICL remain unclear.

While Min et al. (2022) suggest that certain language models primarily rely on semantic prior knowledge triggered by the examples, Wei et al. (2023) show that large models can override semantic priors when presented with in-context exemplars that contradict these priors. This suggests that both semantic priors and input–label mappings play important roles in ICL. Additionally, Olsson et al. (2022) argue that the ICL capabilities of large language models are rooted in "induction heads," which are transformer (Vaswani et al., 2017) circuits formed during pretraining that can copy patterns from ICL examples. On the other hand, Xie et al. (2021) showed that not only transformers but also LSTM models (Hochreiter, 1997) can exhibit ICL behavior.

The unclear nature of ICL mechanisms, along with its dependence on model scale and architecture, underscores the potential for learning-based approaches to optimize the selection of example sequences accordingly.

2.2 Example Sequence Selection

Suppose we have a set of examples for a given task, e_i, as a candidate pool, where each example is a pair consisting of an input and a label, $e_i = (x_i, y_i)$. The example sequence selection problem involves selecting a sequence, $E = [e_1, e_2, \ldots, e_k]$, to include in the prompt in order to improve the ICL generation quality. The selection may depend on the query x^{query} (dynamic examples) or may be independent of it (static examples). Example sequence selection is typically studied across three main aspects: the composition of the examples, their arrangement, and the length of the sequence. A comparison of existing methods with respect to

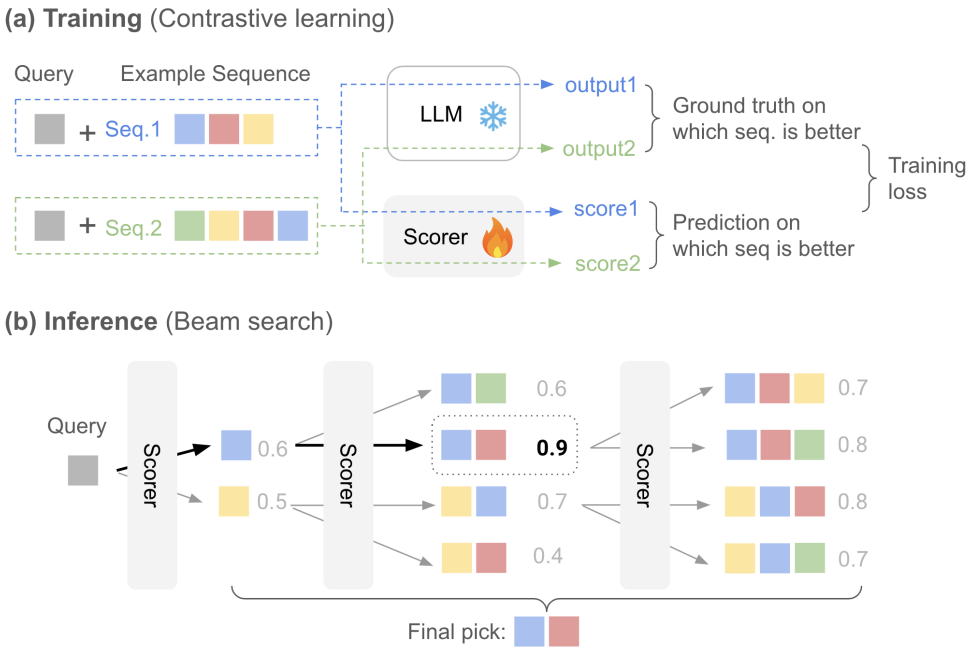

Figure 2: An overview of our BESC approach. (a) We train a scoring model to predict the effectiveness of an example sequence for a given query, using contrastive learning to compare LLM performance between sequences. (b) During inference, the example sequence is constructed incrementally with beam search, where the trained scorer ranks candidates and prunes nodes.

these factors is shown in Table 1.

Static examples In this setting, the goal is to select E for a given task, and this selection does not depend on the query x^{query}. In practice, researchers use a set of labeled queries as a validation set to determine E based on specific metrics evaluated on this set (Lu et al., 2021; Zhang et al., 2022).

Dynamic examples In this setting, the goal is to dynamically select a sequence of ICL examples, E, for each given query x^{query}. The dynamical selection is expected to improve performance compared to using a static sequence, as different queries—even within the same task—may require different sets of skills and information demonstrated by varying examples. Several works have explored this approach (Liu et al., 2021; An et al., 2023; Rubin et al., 2021; Lu et al., 2022; Levy et al., 2022; Ye et al., 2023).

Example composition This line of work focuses on the composition of examples. Liu et al. (2021) proposed selecting the top k examples, e_i, that are semantically similar to x^{query}. An et al. (2023) suggested selecting examples that demonstrate a similar set of "skills" required to solve x^{query}. Beyond similarity, diversity is another important factor in determining example composition. Levy et al. (2022) proposed selecting diverse examples that cover different aspects needed to solve x^{query}. Ye et al. (2023) jointly considered both similarity and diversity in example selection. Additionally, Zhang et al. (2022) and Lu et al. (2022) employed reinforcement learning to optimize example selection.

Example arrangement Lu et al. (2021) demonstrated that the order in which examples are presented in the prompt significantly impacts LLM performance. They proposed using several entropy-based metrics, measured on a validation set, to determine the optimal order. These metrics are grounded in the intuition that the model's predictions should not be overly confident or excessively unbalanced.

Sequence length LLMs may struggle to effectively utilize long contexts (Liu et al., 2024), and ICL performance does not necessarily improve as more examples are provided. Therefore, sequence length is another important factor to consider in example sequence selection. Zhang et al. (2022) proposed including an "early termination" action when constructing the example sequence to control its length and prevent performance degradation.

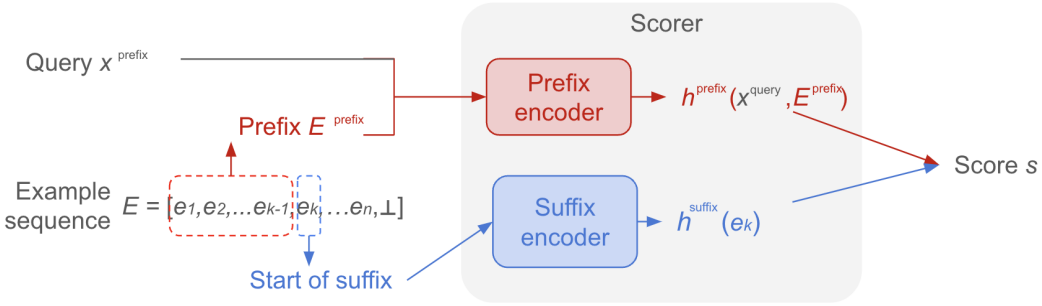

Figure 3: An illustration of the dual-encoder architecture of the scoring model. This design enables the model to rank the next element for a given query and prefix, allowing incremental sequence construction with beam search during inference.

3 Method

We propose Beam Search-based Example Sequence Constructor (BESC) to address the challenges of holistically multiple aspects of example sequence selection and the huge search space complexity.

To learn how to construct the optimal example sequence, we train a model to score example sequences for a given query. The scoring model $s(E, x^{\text{query}})$ is trained to predict the relative LLM generation quality given the query x^{query} and example sequence E. Using this scoring model as a guiding signal, we employ a beam search algorithm to navigate the vast sequence space at inference time. Figure 2 provides an illustration of the training and inference stages of our approach.

3.1 Model Design

We model the sequence using a prefix-suffix decomposition, enabling incremental construction during inference. At each step, the query and the current prefix $E^{\text{prefix}} = [e_1, e_2, \ldots, e_{k-1}]$ are used to select the next element e_k, or the head of the suffix. This element is appended to the prefix, and the process repeats until a special termination element, $\perp$, is selected, marking the end of the sequence.

To enable this prefix-suffix selection, we design a dual-encoder architecture (Figure 3). The model generates an embedding for the prefix, $h^{\text{prefix}}(x^{\text{query}}, E^{\text{prefix}})$, and another for the suffix, $h^{\text{next}}(e_k)$. The score s is calculated as the dot product of these two embeddings:

$$s(E, x^{\text{query}}) = h^{\text{prefix}}(x^{\text{query}}, E^{\text{prefix}}) \cdot h^{\text{suffix}}(e_k)$$

Since the remaining elements $[e_{k+1}, \ldots, e_l]$ are unknown at inference time, the suffix encoder only considers the head of the suffix, e_k. This forces the model to learn to predict the effectiveness of a sequence without having access to all its elements. However, the model can also score a complete sequence by setting the suffix to the termination element $\perp$, with the prefix comprising the actual sequence elements $[e_1, e_2, \ldots, e_l]$.

The dot product formulation transforms this ranking task into a nearest-neighbor search in the embedding space, which can be efficiently performed using a vector store with precomputed embeddings (see Section 3.3).

Algorithm 1: Training BESCscorer

Requirement:
$\{e_i^{\text{cand}}\}$ the example candidate pool.
$\{e_i^{\text{query}}\}$ the set of labeled queries.
L the maximum length of example sequence.
Training:
// Illustrated with batch size 1 in 1 epoch
for $e_i^{query} = (x_i^{query}, y_i^{query})$ *in* $\{e_i^{query}\}$ **do**
  // Sample a prefix sequence
  $l^{\text{prefix}} \leftarrow$ Rand. int. $0 \leq l^{\text{prefix}} \leq L$
  $E^{\text{prefix}} \leftarrow l^{\text{prefix}}$ rand elem from $\{e_i^{\text{cand}}\}$
  // Sample two suffix sequences
  for $j \in \{1, 2\}$ **do**
    $l_j \leftarrow$ Rand int $1 \leq l^{\text{prefix}} + l_j^{\text{suffix}} \leq L$
    $E_j^{\text{suffix}} \leftarrow l_j^{\text{suffix}}$rand elem from $\{e_i^{\text{cand}}\}$
    $E_j \leftarrow E^{\text{prefix}} + E_j^{\text{suffix}}$
    $g_j \leftarrow \text{LLM}(x_i^{\text{query}}, E_j)$
    $a_j \leftarrow$ Quality of g_j given y_i^{query}
    $s_j \leftarrow s_\theta(x_i^{\text{query}}, E_j)$
  // Which is a better example sequence for x_i^{query}? The ground truth:
  $q \leftarrow \begin{cases} [1,0] & \text{if } a_1 > a_2 \\ [0,1] & \text{otherwise} \end{cases}$
  // and the prediction:
  $p \leftarrow \text{Softmax}([s_1, s_2])$
  // Loss and optimization
  $L \leftarrow \text{CrossEntropyLoss}(p, q)$
  Update θ given $\nabla_\theta L$
return s_θ

3.2 Model Training

We train the scoring model to rank example sequences using a contrastive learning approach, as outlined in Algorithm 1.

For a given task, the training dataset is divided into an example candidate set, $\{e_i^{\text{cand}}\}$, and a query set, $\{e_i^{\text{query}}\}$. The maximum number of examples in the sequence is L, a hyperparameter. We assume access to an automatic evaluation metric, $a(g, y)$, that measures the quality of the LLM's generated output g against a reference label y[2].

For each query, we sample a pair of example sequences, E_1 and E_2, which share a prefix E^{prefix}, but their suffix may differ in length, composition, and/or arrangement. The prefix E^{prefix} could be empty, in which case the next element selected will be the first in the sequence.

The goal is to train the model to predict which of the two sequences will lead the LLM to generate a higher-quality output for the given query x_i^{query}. The ground truth ranking is determined by evaluating the LLM's output for each sequence E_j using the evaluation metric a.

We optimize the model by minimizing the cross-entropy loss between the ground truth ranking and the predicted probabilities. In this contrastive learning setup, the sequence with better output quality is treated as the positive sample, and the other as the negative sample.

3.3 Inference with Beam Search

The search space for example sequences is vast, on the order of $O(N^L)$, where N is the number of example candidates and L is the maximum sequence length[3]. An efficient search algorithm is therefore necessary.

Inspired by its success in speech understanding (Medress et al., 1977), natural language generation (Och et al., 1999), and recommendation systems (Gao et al., 2020), we employ beam search to construct the example sequence during inference. Beam search is a form of pruned breadth-first search where the breadth is limited by a parameter, b, called the beam width.

We incrementally extend the example sequence by appending promising examples to the partial sequence while pruning less promising ones. The algorithm is detailed in Algorithm 2.

Algorithm 2: BESC inference

Requirement:
x^{query} the query input.
$\{e_i^{\text{cand}}\}$ the example candidate pool.
L the maximum sequence length.
s_θ the trained scoring model
b the beam width
c the number of next elements to consider for each prefix
Trained model $s_\theta(E, x^{\text{query}})$ to rank candidates

Beam search:
$B^0 \leftarrow \{\emptyset\}$, initial beam
for $l \leftarrow 1$ *to* L **do**
  // Prune previous beam with scorer
  $B^{l-1}_{\text{kept}} \leftarrow$ Top b candidates in B^{l-1}
  // Branch and save in the new beam
  $B^l \leftarrow \emptyset$, current beam
  for $E_i \in B^{l-1}_{kept}$ **do**
    $e_j^{\text{next}} \leftarrow$ Top-c candidates given E_i
    for $j \leftarrow 1$ *to* c **do**
      Add seq $[E_i, e_j^{\text{next}}]$ to B^l

return E of the highest BESC score $s_\theta(x, E)$

At each step, we use the embedding $h^{\text{prefix}}(x^{\text{query}}, E^{\text{prefix}})$ to find the top c examples with the nearest embeddings, $h^{\text{suffix}}(e_i)$, that have not yet been added to E^{prefix}. This step can be efficiently implemented using a vector store with pre-computed $h^{\text{suffix}}(e_i)$ embeddings, using the dot product as the distance measure. This process generates $b \cdot c$ example sequences at each time step, which are pruned by the scoring model, retaining only the top b candidates as the new prefix for the next step.

Initially, E^{prefix} is empty, so the first step involves selecting the most promising starting example for the given query. We repeat the grow-prune cycle until a sequence is pruned, reaches the maximum length L, or the end action $\perp$ is selected.

Finally, we rank all complete sequences explored during beam search, which may vary in length, based on their score s, returning the top-ranked sequence as the final selection (see Figure 2).

3.4 Reduced Complexity

Pruning allows us to score only a small subset of all possible example sequences, which greatly reduces the size of the search space.

In addition, our dual-encoder architecture, combined with the dot product formulation and approximate k-nearest neighbor (k-NN) search (Andoni and Indyk, 2008), makes the process more efficient. The prefix encoder h^{prefix} is run only once per pre-

[2]The metric can also be reference-free, provided it yields a scalar quality score.

[3]The complexity to evaluate all possible sequences from length 1 to L is $\sum_{k=1}^{L} A(N,k) = \sum_{k=1}^{L} \frac{N!}{(N-k)!}$, which can be approximated as N^L when N is much larger than L.

fix, and the suffix encoder h^{suffix} is computed just once per candidate example, significantly lowering computational costs.

The overall complexity to score sequences is reduced from $O(N^L)$ to:

$$O(L \cdot b \cdot c) + O(N) + O(L \cdot b \cdot c \cdot \log N)$$

The first term corresponds to the computation of the prefix encoder for all prefix candidates explored. The second term accounts for computing the suffix encoder for all N suffix head candidates. The third term represents the complexity of the approximate k-NN search to find the top c examples at each step.

4 Experiments

4.1 Datasets

Given the wide application of few-shot learning with LLMs, prior works on example sequence selection have been evaluated across a range of tasks, including text classification (Zhang et al., 2015; Socher et al., 2013; Li and Roth, 2002; Hovy et al., 2001) and text generation (Wolfson et al., 2020; Li et al., 2020; Lu et al., 2022).

Following this, we consider a broad set of six datasets[4], including AGNews (Zhang et al., 2015), SST (Socher et al., 2013), and TREC (Li and Roth, 2002; Hovy et al., 2001) for text classification, and TabMWP (Lu et al., 2022), BREAK (Wolfson et al., 2020), and GSM8K (Cobbe et al., 2021) for text generation, as summarized in Table 3.

4.2 Language Models

We experiment with four language models of varying sizes and training paradigms: GPT-2 (Radford et al., 2019), GPT-neo (Black et al., 2021), GPT-3 (Brown, 2020), and GPT-3.5.

While prior studies have primarily focused on earlier models that lack instruction tuning (Ouyang et al., 2022), we include GPT-3.5 to examine the effect of example sequence selection on instruction-tuned models.

Our experiments cover both small and large models, spanning open-source models (GPT-2 and GPT-neo) and closed API models (GPT-3, GPT-3.5), as well as models that are pretrain-only and those with post-training (instruction-tuned) capabilities.

[4]GSM8K and BREAK are released under the MIT license, and TabMWP under the CC BY-NC-SA license.

4.3 Baselines

We compare our method with several existing example selection approaches, including kNN (Liu et al., 2021), EPR (Rubin et al., 2021), CEIL (Ye et al., 2023), and PromptPG (Lu et al., 2022), which focus on example composition. We also include Reordering (Lu et al., 2021), which prioritizes example arrangement, and a Q-learning-based method (Zhang et al., 2022), which considers sequence length, composition, and arrangement but overlooks query dependence and textual information. A comparison of the design features of these methods and ours is summarized in Table 1.

4.4 Implementation

4.4.1 Modeling

The two encoders in the model can be implemented in various ways. In our implementation, we ensure that the prediction depends on the textual information from the query and examples, while considering sequence length, composition, and arrangement.

For the prefix encoder, we use a hierarchical approach. We first encode x^{query} and each e_i using transformers initialized with a pretrained sentence BERT model (Reimers and Gurevych, 2019), then pass the resulting embeddings to a two-layer LSTM network to generate the final sequence representation. This hierarchical structure helps handle long sequences, especially when L is large.

The suffix encoder also uses transformer-based textual encoders initialized with sentence BERT. Both the transformer and LSTM parameters are fine-tuned during contrastive learning.

4.4.2 Training

For each task, we split the training data into 1,000 instances as the query set, with the remainder serving as the example candidate set. We use greedy sampling for LLMs generation.

To generate positive and negative samples for contrastive learning, we use $L = 7$ and apply uniform random sampling to generate the variables (e.g., l^{prefix}, l_j^{suffix}, E^{prefix}, and E_j^{suffix}). However, more effective sequences for contrastive learning may be obtained by either leveraging existing example selection methods or using the BESC scoring model trained in the previous epoch. Adding these samples to the training data may further improve the model. We leave the exploration of such strategies for future work.

	GPT-2			GPT-3		GPT-neo		GPT-3.5	
	AGNews	SST2	TREC	TabMWP	BREAK	SST5	BREAK	TREC	GSM8K
Random	0.55	0.66	0.41	0.65	0.04	0.31	0.02	0.55	0.87
Reordering (Lu et al., 2021)	0.63	0.68	0.33	-	-	-	-	0.59	0.91
kNN (Liu et al., 2021)	-	-	-	0.68	-	-	-	0.70	0.90
EPR (Rubin et al., 2021)	-	-	-	-	0.25	0.43	0.30	0.75	0.93
Q-learning (Zhang et al., 2022)	0.71	0.81	0.43	-	-	-	-	-	-
PromptPG (Lu et al., 2022)	-	-	-	0.71	-	-	-	-	-
CEIL (Ye et al., 2023)	-	-	-	-	-	0.47	0.34	-	-
BESC(ours)	**0.77**	**0.87**	**0.48**	**0.77**	**0.28**	**0.50**	**0.38**	**0.85**	**0.97**

Table 2: Comparison between the accuracy achieved by our method (BESC) and baseline methods.

Dataset	Task	Train set
AGNews	Topic classification	44.3k
SST	Sentiment classification	67.3k
TREC	Question classification	5.5k
TabMWP	Mathmatical reasoning	38.4k
BREAK	Meaning representations	44.3k
GSM8K	Mathmatical reasoning	7.5k

Table 3: Datasets used in this work.

4.5 Inference

We perform beam search using $b = 5$ beams and a maximum of $c = 5$ candidates per beam.

5 Results and Discussion

In this section, we present our results, comparing our method to baselines and analyzing key components through ablation studies. We also examine the model's transferability across tasks.

5.1 Improved Performance over Baselines

We compare our method with the baseline approaches on the datasets they were originally tested on. The results, summarized in Table 2, show that our method consistently outperforms all baselines across six datasets and four language models.

We hypothesize that this improvement is due to the holistic consideration of all key factors involved in dynamic example sequence selection that our method incorporates.

To further investigate, we conduct ablation studies to evaluate the contribution of each key factor. All ablation studies are conducted on the TREC and GSM8K datasets using GPT-3.5 as the LLM.

5.2 Effects of Dynamic Examples

The first key factor we explore is the impact of dynamic example selection, where the example sequence is adapted to each query rather than being fixed for all queries (static examples). Previous works by Lu et al. (2021) and Zhang et al. (2022) focused on static examples, and we hypothesize that adapting the example sequence to the query can improve performance.

To evaluate this, we create a "static" version of BESC by removing x^{query} from the inputs of the prefix encoder, making the prediction (or next element selection) not dependent on the query. We train this static version of the scoring model and compare it to the original, query-dependent "dynamic" BESC.

	Dynamic	Static
TREC	0.85	0.79
GSM8K	0.97	0.92

Table 4: Ablation study on the effects of dynamic example sequence selection using two BESC variants. The "Static" version is trained without taking queries as input to the scoring model.

The results, summarized in Table 4, show that the dynamic version significantly outperforms the static version, highlighting the importance of selecting examples dynamically based on the specific query.

5.3 Effects of Automatic Sequence Length

Our method automatically determines the optimal sequence length ("Auto l") during inference.[5] To study the impact of this feature, we compare BESC with fixed-length versions ($l = 1$, $l = 3$, $l = 5$), where only candidates of the specified length are considered during inference.

The results, summarized in Table 5, show that automatically determining sequence length leads to further performance improvements compared to simply increasing the sequence length. We hypothesize that this is because the LLM may struggle to

[5]This is achieved by: 1) learning to selecting the termination token $\perp$, and 2) picking the best sequence from explored candidates of varying lengths during beam search. See Section 3

	Auto l	$l = 3$	$l = 5$	$l = 7$
TREC	0.85	0.80	0.83	0.84
GSM8K	0.97	0.91	0.95	0.95

Table 5: Ablation study on the effects of example sequence length. BESC, which automatically determines the sequence length ($1 \leq l \leq 7$), is compared with variants using fixed sequence lengths.

fully utilize longer contexts, especially when they include less useful or distracting information (Liu et al., 2024).

5.4 Effects of the Element Arrangement

The impact of element ordering has been discussed by Lu et al. (2021), but their work focused on the ordering of randomly selected elements. In contrast, we study the effects of ordering for carefully selected examples as part of our ablation study.

	BESC	Shuffled
TREC	0.85	0.84
GSM8K	0.97	0.95

Table 6: Ablation study on the effects of example arrangement in the ICL example sequence. "Shuffled" uses the same examples as BESC for each query, but in a shuffled order.

We randomly shuffle the example sequence generated by BESC for each query and present the experimental results in Table 6. The results show a decrease in performance after shuffling, indicating that element arrangement remains a relevant factor in dynamic example sequence selection.

5.5 Effects of Sequential Modeling

Zhang et al. (2022) observed performance improvement from example sequence selection in smaller models like GPT-2, but noted that this improvement diminishes with larger models such as GPT-3. They hypothesized that this reduction is due to the emerging capabilities of larger LLMs. However, other works, including ours, experimenting with GPT-3 (Lu et al., 2022; Rubin et al., 2021), show that example sequence selection can still provide significant performance improvements in larger models.

We suspect that the diminished improvement observed by Zhang et al. (2022) is due to their method using an oversimplified representation of the sequence, relying only on sequence length as the feature.

To test this, we implemented a variant of BESC where the prefix encoder only uses the prefix length as the feature, ignoring the textual information. The performance of this variant decreases significantly compared to the original BESC, highlighting the importance of more comprehensive sequential modeling.

5.6 Transferability

Similar to other learning-based methods (Lu et al., 2022; Zhang et al., 2022; Rubin et al., 2021), our approach requires task-specific training, which introduces a one-time additional cost compared to learning-free methods such as kNN (Liu et al., 2021). To mitigate this, we explore transfer learning by applying a pretrained BESC model to new tasks without additional task-specific training.

		BESC	
	kNN	Single-task	Pretrained
TREC	0.70	0.85	0.82
GSM8K	0.90	0.97	0.94

Table 7: Comparison of a learning-free method (kNN), the original BESC, and a pretrained version of BESC.

In a leave-one-out setting, we use five of the six datasets listed in Table 3 to train a "pretrained" BESC scoring model and test its performance on the remaining target task. The experimental results, with TREC and GSM8K as target tasks and GPT-3.5 as the LLM, are summarized in Table 7. The results show that while the pretrained BESC model performs worse than one trained specifically on the target task, it still outperforms the learning-free kNN method (Liu et al., 2021). This suggests that knowledge about optimal example sequence selection can transfer across tasks.

6 Conclusion

We introduce Beam Search-based Example Sequence Constructor (BESC) to address the example sequence selection problem. BESC incrementally constructs sequences during inference, using beam search to reduce search space complexity. Experiments across multiple datasets and language models show substantial performance improvements. Ablation studies highlight key contributions: dynamic example selection, automatic sequence length, element arrangement, and sequential modeling. We also explore BESC's potential in transfer learning. Lastly, we discuss limitations and propose future directions, including open-ended tasks and integration with prompting strategies.

7 Limitations

The datasets used in this work mainly consist of short text and focus on tasks with a unique ground truth. It is unclear how this method, or in-context learning (ICL) in general, performs on more open-ended tasks like dialogue (Zhang et al., 2020), role-play (Sadeq et al., 2024) or creative writing, which warrants further exploration.

Moreover, the effectiveness of ICL examples can540 vary with different prompting strategies, espacially those eliciting LLM reasoning abilities (Wei et al., 2022; Yao et al., 2024; Lightman et al., 2023). The interaction between example selection and such techniques remains an open question.

The proposed method could introduce bias in the generated content through the selection of examples, and like other large language model techniques, it could be misused for harmful content generation.

References

Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. 2023. Skill-based few-shot selection for in-context learning. *arXiv preprint arXiv:2305.14210*.

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58(2).

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

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. 2020. Deep retrieval: learning a retrievable structure for large-scale recommendations. *arXiv preprint arXiv:2007.07203*.

Xiang Gao and Kamalika Das. 2024. Customizing language model responses with contrastive in-context learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18039–18046.

S Hochreiter. 1997. Long short-term memory. *Neural Computation MIT-Press*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Itay Levy, Ben Bogin, and Jonathan Berant. 2022. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2020. Mtop: A comprehensive multilingual task-oriented semantic parsing benchmark. *arXiv preprint arXiv:2008.09335*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Mark F. Medress, Franklin S Cooper, Jim W. Forgie, CC Green, Dennis H. Klatt, Michael H. O'Malley, Edward P Neuburg, Allen Newell, DR Reddy, B Ritea, et al. 1977. Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, 9(3):307–316.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Nafis Sadeq, Zhouhang Xie, Byungkyu Kang, Prarit Lamba, Xiang Gao, and Julian McAuley. 2024. Mitigating hallucination in fictional character role-play. *arXiv preprint arXiv:2406.17260*.

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.

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

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.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pages 39818–39833. PMLR.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. *arXiv preprint arXiv:2211.04486*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.