CombiSearch: Improving Dialogue State Tracking Through Pool Construction and Combination Scoring with Automated Error Analysis

Anonymous ACL submission

Abstract

In dialogue state tracking (DST), in-context learning (ICL) has shown promising performance, which typically involves training a retriever to select useful in-context examples. However, existing studies pay little attention to the quality of the data used for training the retriever, even though this quality dictates the performance upper bound of the retriever. Moreover, the lack of fine-grained error analyses hinders our understanding of where current ICL methods fall short in DST. To enhance the data for retriever training, we present CombiSearch, a method for scoring in-context examples based on their combinatorial contribution to DST performance. We also develop an error analysis framework that categorizes DST errors into seven specific types. Our experiments for DST demonstrate that using high-scoring examples from CombiSearch achieves a 13% absolute improvement over examples scored by the state-of-the-art (SOTA) method in a theoretical setting with a perfect retriever; this reveals that retriever training in prior work relied on suboptimal data. Furthermore, a retriever trained on CombiSearch-scored examples also outperforms the SOTA method. Through a fine-grained error analysis, we identify error types that are addressed and unaddressed by CombiSearch and discuss promising directions for future work.[1]

1 Introduction

Dialogue state tracking (DST) tracks user intentions and informed slot values throughout a conversation based on dialogue history and recent utterances. For instance, given a user utterance "I want to book a guest house for four people", the model may extract the dialogue state "[<hotel-type: guest house>, <hotel-book people: 4>]". Recently, in-context learning (ICL) methods have demonstrated superior performance without extensive fine-tuning, which is especially useful when the schema of dialogue states changes or when these methods are applied to unseen domains (Feng et al., 2023; King and Flanigan, 2023; Hu et al., 2022).

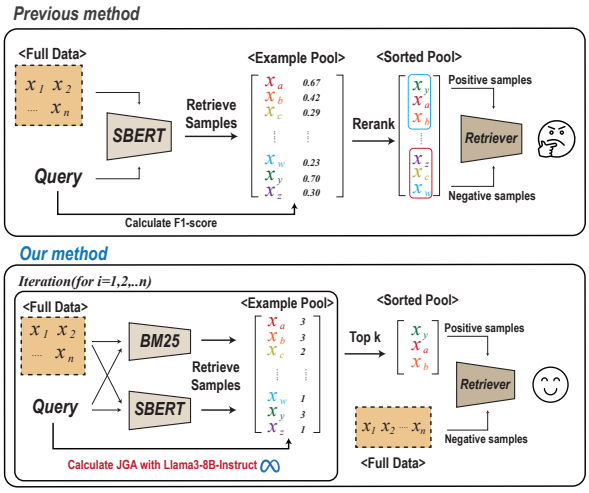

Figure 1: Comparison between previous method and our method.

ICL methods typically involve preparing a dataset for training a retriever to select in-context examples. Most approaches in DST score individual examples based on the similarity of their ground-truth dialogue states to that of the query (King and Flanigan, 2023; Hu et al., 2022). However, this scoring method fails to capture the combinatorial effect of in-context examples, and the scores of examples are not directly aligned with their contributions to the final DST performance. Moreover, there is a lack of analysis regarding the performance of high-scoring examples in DST *without retriever training*. Understanding this performance—beyond the results derived from trained retrievers, as seen in most studies—is crucial, as it dictates the upper bound and headroom for trained retrievers. This analysis can foster advancements in DST by enabling us to select high-

[1]We will release our data and source code upon the publication of the paper.

potential scoring methods and develop techniques that effectively train retrievers on the scored examples.

To that end, we present a new scoring method, **CombiSearch**, consisting of two main steps. First, it selects a diverse pool of example candidates based on both lexical and semantic similarity to the input query, with the goal of enhancing the diversity of examples to capture various factors that influence DST performance, such as dialogue history and utterances. Second, we score the examples in this pool in a combinatorial manner by repeatedly sampling k examples and predicting the dialogue state of the query using them; if the prediction is correct, we increment the scores of these examples by 1. This simple method, inspired by prior studies on combination search (Nguyen and Wong, 2023; Chang and Jia, 2023), effectively measures the contribution of each example when used in tandem with other examples, in a way that is directly aligned with the final DST performance. In addition to CombiSearch, we build and release an error analysis framework that categorizes DST prediction errors into seven types. This framework can identify weaknesses in large language models (LLMs) and the root causes of each error, thereby providing deeper insights into areas of improvement for both scoring methods and LLM inference.

To estimate the maximum utility of scoring methods without retrieval errors, we examined the *theoretical* upper bound performance of CombiSearch in DST. Specifically, for each input query, we scored example candidates using CombiSearch (accessing the ground-truth dialogue state of the query) and directly used the top k examples for ICL without retriever training (hence "theoretical"). CombiSearch achieved a 13% absolute improvement over the scoring method employed by the SOTA DST model (also with access to the ground truth), suggesting that the current DST models rely on suboptimal data for retriever training. Additionally, CombiSearch outperformed another combination search method based on greedy search (Liu et al., 2024) by 12% while reducing computational costs by 80%. In a more practical setting, where a retriever is trained on scored examples, the retriever trained using CombiSearch scores also outperformed the SOTA model.

Our contributions are threefold:

1. We present **CombiSearch**, a method for scoring in-context examples based on their combinatorial contributions to DST performance.

2. We assess the ideal condition performance of scoring methods and demonstrate that CombiSearch scores significantly improve the upper bound performance of DST.

3. We release an automated error analysis tool that categorizes DST prediction errors into fine-grained types, offering deeper insights into the behavior of LLMs in DST and areas of improvement.

2 Problem Setting

2.1 Dialogue State Tracking

DST aims to predict the dialogue state that represents the user's intentions and goals by processing the current interaction between the agent and the user, along with the conversation history. Let A^t and U^t represent the agent's and user's utterances at turn t, respectively, and this exchange is denoted by (A^t, U^t). The dialogue history up to turn t is represented as $C^t = [(A^1, U^1), (A^2, U^2), \ldots, (A^t, U^t)]$. The objective of DST is to predict the dialogue state y^t based on C^t, where y^t consists of a set of slot-value pairs: $y^t = \{(s^1, v^1), (s^2, v^2), \ldots, (s^n, v^n)\}$.

Following the approach of Hu et al. (2022), we predict the *state change* at the t-th turn. Specifically, the model takes the dialogue state up to turn $t-1$ (i.e., y^{t-1}) and the agent and user utterances at turn t (i.e., A^t, U^t) as input, represented as $x^t = [y^{t-1}, (A^t, U^t)]$. It then predicts the dialogue state change Δy^t, where $\Delta y^t = \{+(s^i, v^i), \ldots, -(s^j, v^j)\}$, with "+" indicating the addition of a slot-value pair and "-" indicating its removal. The updated dialogue state at turn t is computed as $y^t = y^{t-1} \cup \Delta y^t$, combining the predicted state change with the previous state.

2.2 In-Context Learning for DST

LLMs perform in-context learning (ICL) by receiving a set of K labeled examples retrieved for a test query x_{test}. The i-th example consists of a query x_i and its corresponding dialogue state change Δy_i as $(x_i, \Delta y_i)$. The LLM then predicts the dialogue state change for the test input x_{test}, conditioned on both these examples and x_{test}, as $\Delta \hat{y}_{test} = \text{LLM}\left(\{(x_i, \Delta y_i)\}_{i=1}^{K}, x_{test}\right)$.

The selection of these K examples is crucial, as their quality and relevance significantly impact the LLM's performance. The most common approach

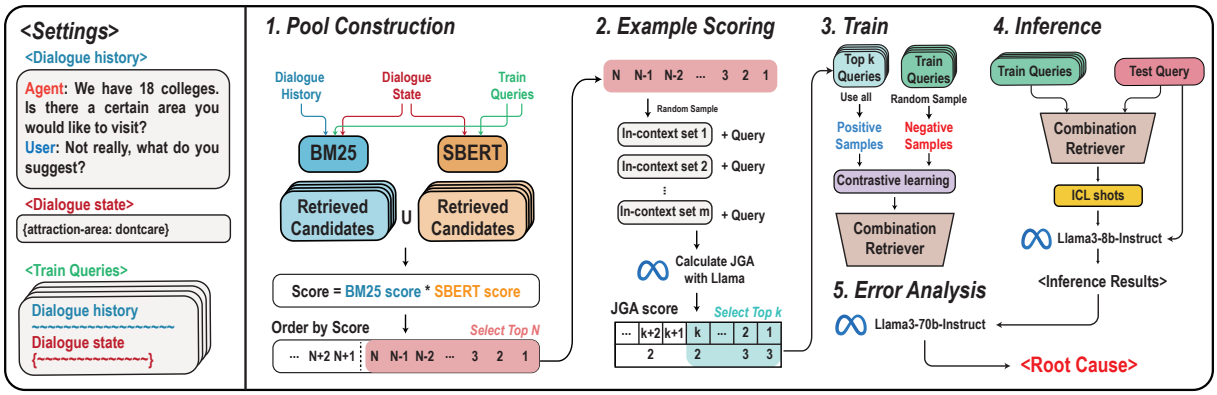

Figure 2: Overall structure of CoDST.

is to select the K examples most similar to x_{test} (e.g., based on dialogue states) from the training set. Similarity between x_{test} and x_i is typically computed using embedding-based or statistical measures.

3 Method

We present **CombiSearch**, a scoring method for identifying effective examples for each query. As illustrated in Figure 2, CombiSearch consists of (1) constructing a diverse, high-quality pool of candidate examples, and (2) scoring the examples combinatorially by repeatedly sampling K examples from the pool and updating their scores based on their success in DST. In the following subsections, we first describe the CombiSearch algorithm and then present a retriever training method on examples scored by CombiSearch.

3.1 CombiSearch

Pool Construction (Figure 2-1) Since it is impractical to perform combinatorial search on a large number of examples, we first construct a pool of high-quality examples that is of a manageable size. The diversity of examples is crucial here, because in DST, dialogue context and dialogue state interact in complex ways, and errors often arise when these complex interactions are not informed by in-context examples. For instance, overlooking the relationship within the dialogue context can lead to failure in co-reference resolution. Since no single example can capture all these interactions, it is essential to construct a diverse pool of examples that reflect various interaction patterns, ensuring that as many effective examples as possible can be considered and recognized in the subsequent scoring step based on combination search.

To create a candidate pool E consisting of examples that are diverse yet relevant to the query, we calculate the similarity between the query and examples by considering both dialogue context and dialogue state. Specifically, we combine BM25 and SBERT (Robertson and Zaragoza, 2009; Reimers and Gurevych, 2019) for retrieval, as they complement each other; BM25 captures lexical matches, while SBERT focuses on contextual meaning. The examples retrieved through this combination exhibit similarities with the query across various aspects, including dialogue context and nuances in utterances. From each retriever, we select the top-N examples and rerank them by multiplying BM25's tf-idf scores with SBERT's cosine similarity scores. Finally, we retrain the top-N examples to form the final pool E.

Example Scoring (Figure 2-2) After constructing E with N candidate examples for each query, we proceed to score these examples. Scoring each example independently is not ideal because the effectiveness of an example is influenced by the other examples present together in the prompt; for instance, a high-quality example may become redundant when used alongside a similar one. Hence, our primary objective is to ensure that the highest-scoring examples for a given query form a synergistic combination that leads to successful DST. This objective naturally leads us to a combinatorial search for example scoring. However, it is impractical to evaluate all possible combinations of examples to determine which lead to successful DST.

Instead, we assume that an example is a "key team player" when the "teams" it belongs to frequently achieve successful DST. CombiSearch leverages this intuition (Algorithm 1) by repeat-

Algorithm 1 Pseudo-code for **CombiSearch**

Input: D (Train Dataset), Q(Query Dataset)
Output: $D_{\text{CombiSearch}}$
for $(x_{test}, y_{test}) \in Q$ **do**
  (bm25_ids, bm25_scores) $\leftarrow$ BM25($(x_{test}, y_{test}), D$)
  (sbert_ids, sbert_scores) $\leftarrow$ SBERT($(x_{test}, y_{test}), D$)
  union_ids $\leftarrow$ bm25_ids $\cup$ sbert_ids
  combined_scores $\leftarrow$ $\{u$: bm25_scores$[u]$ $\times$ sbert_scores$[u] \mid u \in$ union_ids$\}$
  $E \leftarrow$ topk(combined_scores, B)
  for $i = 1 \rightarrow M$ **do**
    $Prompt \leftarrow$ random_select(C, k)
    $JGA \leftarrow$ LLM$(Prompt, x_{test})$
    update(final_scores, $Prompt, JGA$)
  end for
  $Prompt_{\text{final}} \leftarrow$ topk(final_scores, k)
  $D_{\text{CombiSearch}} \leftarrow (x_{test}, y_{test}, E)$
end for

edly sampling k examples and measuring the Joint Goal Accuracy (JGA), the most common metric in DST. If the JGA is 1, indicating a correct prediction, we increment the score of each of the k examples by 1; if the JGA is 0, the scores remain unchanged. By repeating this process M times, we accumulate scores for each example based on its contribution to correct predictions. This result in a final dataset $D_{\text{CombiSearch}}$, where each query is paired with its corresponding candidate pool E, sorted by each example's score.

3.2 Retriever Training

We train the retriever using $D_{\text{CombiSearch}}$ to select examples that enhance DST performance (Figure 2-3). Following the convention of using embedding-based retrievers (King and Flanigan, 2023; Hu et al., 2022), we use two encoders: $E_e(\cdot)$ for encoding examples and $E_x(\cdot)$ for encoding queries. InfoNCE is used as a loss function (van den Oord et al., 2019). The objective is to maximize the bien-coder score between the input query and its most effective examples. For each test input x_{test}, we use the 10 highest-scoring examples from $D_{\text{CombiSearch}}$ as positive examples P. For each positive example e^+, we designate the other B positive examples within the batch as its negative examples $(e_1^-, e_2^-, \ldots, e_B^-)$. The loss is calculated as follows:

$$L(x, P, N) = \sum_{e^+ \in P} -\log\left(\frac{\exp(\text{sim}(x, e^+))}{\sum_{e' \in \{N \cup \{e^+\}\}} \exp(\text{sim}(x, e'))}\right) \quad (1)$$

During inference (Figure 2-4), we use the trained retriever and BM25 just as we did during data generation. This helps retrieve lexically relevant examples to query that SBERT alone might miss. For each query, we retrieve N examples for each retriever and then multiply the scores from SBERT and BM25, and select the top k examples with the highest product scores for inference.

4 Error Analysis Framework

Before we jump to experiments, we introduce a comprehensive error analysis tool that categorizes DST prediction errors into seven types, allowing us to better understand the underlying causes of mistakes made by LLMs and inform directions for improving the methods.

The Figure 3 shows our classification of error types (error types are marked in boxes). Recall that the predicted dialogue state in the current turn t is $y^t = \Delta y^t \cup y^{t-1}$. Hence, at the top level, we classify errors based on two sources: (1) Misprediction of slot-value pairs from the utterances in the current turn Δy^t (*delta*), or (2) Propagation of mispredictions from the past turns y^{t-1} (*error propagation*). These two sources share the same set of specific error types.

At the second level, we classify errors by comparing slot-value pairs between the ground truth dialogue state and the prediction: (1) a slot-value pair in the ground truth is absent in the prediction (*miss*), and (2) a slot-value pairs in the prediction is absent in the ground truth (*hallucination*). These two categories have their own set of error types, constituting seven specific error types as follows.

For the *miss* category, an error may arise because the prediction (1) misses an entire slot-value pair (*miss-total*), (2) misses a special token "delete" (*miss-delete*), (3) misses "don't care", which indicates the user's openness to any options (*miss-dont care*), or (4) misses a slot by confusing it with another slot (*miss-confuse*).

In the *hallucination* category, errors may arise when the prediction (1) hallucinates a slot-value pair that is not informed by the dialogue (*hallucination-total*), (2) hallucinates a value with a correct slot that does not match the dialogue (*hallucination-value*), or (3) incorrectly modifies the previous dialogue state (*hallucination-overwrite*).

This categorization is systematic and identifies various sources of errors intuitively. Additionally, it elucidates the weaknesses of LLMs concerning the different capabilities required for DST, including utterance comprehension, hallucination detection,

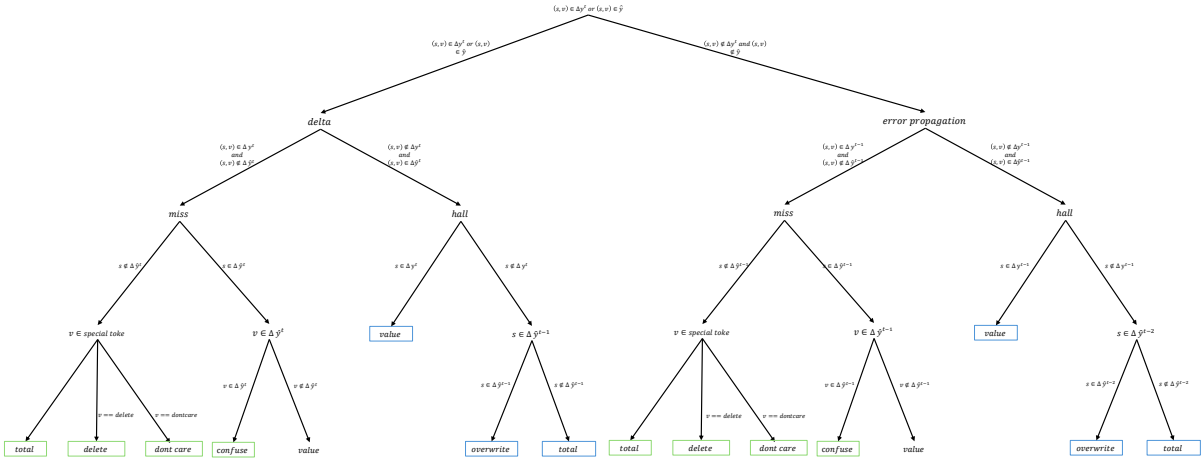

Figure 3: Error tree to divide error type by slot and value

and the identification of key slots. We release the framework as a Python library to facilitate its easy integration in future studies.

5 Experiments

In this section, we present four experiments. First, we analyze the weaknesses of modern LLMs in DST using our error analysis framework (§5.2). Next, we examine the utility of CombiSearch in both a theoretical setting without retrieval errors, to estimate the upper bound performance of scored examples (§5.3) and in a practical setting with trained retrievers (§5.4). Lastly, we conduct an ablation study (§5.5). We begin with a description of the common experimental settings applicable to all experiments. Experiment-specific settings are detailed in their respective subsections.

5.1 Common Experiment Settings

For all experiments, we use the MultiWOZ datasets, which consists of approximately 10,000 multi-domain task-oriented dialogues. Following the experimental settings of prior studies, we used five domains (hotel, attraction, taxi, train, restaurant) for the training data, and the validation and test data were also limited to these domains. We trained our models using MultiWOZ 2.1 (Eric et al., 2020) and evaluated the final performance using MultiWOZ 2.4 (Ye et al., 2022). To measure performance, we primarily use Joint Goal Accuracy (JGA), a common metric in DST across all experiments. To count as correct, the predicted dialogue state at turn t must match the ground-truth dialogue state exactly, including all slot names and values.

If not specified otherwise, we use the default settings: $N = 100$ for the candidate pool size, $k = 10$ for the number of examples in the prompt, and $M = 3$ for the number of times each example is evaluated. Lastly, we used the Llama3-8B-Instruct and Llama3-70B-Instruct (Dubey et al., 2024). Due to the large size and slower inference speed of the Llama 70B model, we applied 4-bit quantization using GPTQ to optimize its performance.

5.2 Experiment 1. Error Analysis of Modern LLMs

To understand the weaknesses of modern LLMs in DST, we used our error analysis framework and evaluated the zero-shot and 10-shot DST performance of four models: RefPyDST (King and Flanigan, 2023), GPT-3.5-turbo, LLaMA-3-8B, and 4-bit quantized LLaMA-3-70B (Dubey et al., 2024). RefPyDST, the current SOTA model, uses prompts in Python format, while the other models employ natural language prompts. In the 10-shot setting, RefPyDST retrieves in-context examples using its own fine-tuned retriever. For the other models, we randomly select examples from the full training dataset.

Overall, *error propagation* errors were significantly more frequent than *delta* errors. In the zero-shot setting, *error propagation hallucination total* was most common, while in the random setting, *error propagation miss total* dominated. For RefPyDST, an exception was observed where format errors were the most frequent due to Python code formatting, with *hallucination total* being the next most prominent type of error.

In the 10-shot setting, *miss* errors outnumbered *hallucination* errors in all models except GPT. In zero-shot, hallucination errors were generally

Model	Scoring Method	1%	5%	100%
llama 8b	RefPyDST	58.0	62.7	69.7
llama 8b	Hybrid	60.4	68.0	75.9
llama 8b	**CombiSearch**	68.4	75.1	82.7
llama 70b	RefPyDST	63.8	66.4	67.9
llama 70b	Hybrid	60.2	62.3	67.6
llama 70b	**CombiSearch**	71.8	72.7	79.6

Table 1: Theoretical DST performance across scoring methods and backbone models. % indicates the percentage of the training set used for example selection.

more prevalent. Across all models, *miss total* and *hallucination total* errors—where the correct slot and value are either missed or incorrectly generated—were the most frequent. *Miss total* errors affected subsequent dialogue turns more extensively, especially for name-related slots, due to issues like user confirmation and flexible preferences not being properly interpreted. Meanwhile, *hallucination total* errors often stemmed from oversensitivity to system descriptions, incorrect co-reference resolution, or user question misinterpretation, notably affecting area-related slots.

To reduce both miss and hallucination errors, addressing *delta* errors and ensuring accurate understanding of current dialogue turns are essential. We now incorporate both the dialogue state and conversation similarity in retrieval, and provide explicit instructions in the prompts for better information extraction.

5.3 Experiment 2. Theoretical Upper Bound without Retrieval Errors

This experiment aims to establish the *theoretical* upper bound performance of CombiSearch-scored examples. Specifically, we are interested in the DST performance when, for a test query, examples are scored by CombiSearch with access to the ground-truth dialogue state of the query, and the highest-scoring examples are directly chosen for ICL without a retrieval step. Although this setting is impractical—because in reality, we cannot access the ground-truth dialogue state of the query and trained retrievers may make errors—this theoretical performance establishes the upper bound for trained retrievers. Understanding this limit helps us gauge the potential improvement achievable through given retriever training data and assess whether current DST approaches are employing data with high enough potential.

Settings In this experiment, we evaluated the models on 10% of the test set due to time constraints. To minimize bias, we conducted three independent runs with random sampling from the test set and averaged the scores. For each test query, we constructed $D_{\text{CombiSearch}}$ using instances from the training set. To assess the impact of example diversity, we also explored constructing $D_{\text{CombiSearch}}$ using only 1% and 5% of the training data.

For the baselines, we first used RefPyDST, reproduced with Llama3-8B-Instruct (Dubey et al., 2024) because Codex has been deprecated. Additionally, we introduced a 'Hybrid' baseline, which scores examples based solely on the combined scores of BM25 and SBERT (as in the pool construction step in CombiSearch) without combinatorial example scoring.

Results As shown in Table 1, the JGA score of CombiSearch is 82.7% for the Llama 8B model (row 3) and 79.6% for the Llama 70B-4bit quantized model (row 6). This represents an approximately 13% improvement over the performance of RefPyDST for each model (rows 1 and 4). The higher upper bound signifies greater headroom for practical settings where retrievers are trained. It also indicates that the existing methods relied on suboptimal data for retriever training. Furthermore, CombiSearch demonstrates 8%–12% higher performance than Hybrid which uses the combined scores of BM25 and SBERT obtained during pool construction, without the combinatorial scoring (rows 2 and 5). This highlights the importance of aligning the scores of examples with DST performance in a combinatorial manner.

When comparing errors between CombiSearch and RefPyDST using our error analysis framework, we found that RefPyDST frequently produced predictions that did not conform to Python format. Excluding these errors, the error patterns were similar between the two methods, with the most common errors being *miss total* and *hall total*. In RefPyDST, the main cause of *miss total* errors was missing user confirmations (e.g., "yes . please book a seat on train tr6332 for me."), while the main cause of *hall total* errors resulted from incorrectly connecting information from recent conversations with dialogue history. In contrast, the main cause of *miss total* errors in our method was missing the agent's suggestions (e.g., "the place is a great nightclub and there are some museums in the area as well."). Overall, errors related to missing user preferences

Model	Scoring Method	1%	5%
llama 8b	RefPyDST	45.3	47.2
llama 8b	**CombiSearch**	51.3	52.3
llama 70b	RefPyDST	53.7	55.8
llama 70b	**CombiSearch**	58.1	56.5

Table 2: Practical DST performance with trained retrievers across scoring methods and backbone models. % indicates the percentage of the training set used for retriever training and for example selection during inference.

Model	Pool Construction	JGA(%)
llama 8b	BM25&SBERT(Ours)	**70.74**
llama 8b	Random	56.92
llama 8b	BM25	66.00
llama 8b	SBERT	67.74

Table 3: Theoretical DST performance of CombiSearch across candidate retrieval methods for pool construction.

Model	Combinatorial Search	JGA(%)
llama 8b	CombiSearch	**70.74**
llama 8b	Se2	58.80

Table 4: Theoretical DST performance of different methods for combinatorial search.

or confirmation utterances were reduced compared to RefPyDST.

Error Rectification We analyzed which error types were significantly rectified by CombiSearch (Figure 4). When examples were selected based only on their similarity to the query (i.e., using BM25 or SBERT scores only), the most frequent error types were *miss total* and *hall total*. However, CombiSearch significantly reduced these errors. In BM25 and SBERT, the primary cause of *miss total* errors was to miss user confirmations, and this was effectively mitigated by CombiSearch. This finding suggests that aligning example scores with direct feedback from DST performance helps identify examples that better comprehend user utterances.

For *hall total* errors, all methods exhibited similar causes, which seem to stem from the task setup of DST, where dialogue context is represented as a dialogue state rather (i.e., y^{t-1}) than as raw utterances. Another common cause was the overinterpretation of user utterances. However, such errors were significantly reduced by CombiSearch, likely because scoring examples multiple times in different combinations is helpful for choosing examples that minimize the risk of overinterpretation.

5.4 Experiment 3. Practical Performance with Trained Retrievers

In this experiment, we evaluate the practical performance of CombiSearch by training a retriever on $D_{\text{CombiSearch}}$ obtained from the training set of MultiWoZ 2.1.

Settings To obtain data for training a retriever (§3.2), we constructed $D_{\text{CombiSearch}}$ from instances in the training set of MultiWOZ 2.1. However, due to time and resource constraints, we opted not to use all instances in the training set as queries. Instead, we used a subset of the training set, exploring 1% and 5%.

Results As shown in Table 2, CombiSearch achieved approximately a 5% improvement over RefPyDST on the same amount of data used for retriever training. This gap was pronounced in smaller models and smaller data. This suggests that when training data is limited, CombiSearch has high utility.

According to the error analysis, our practical retriever exhibited a significant number of errors in the miss total category, similar to the upper bound results, followed by errors in the miss dontcare category. The primary causes of miss total errors were missing the user's confirmations and failing to capture the agent's suggestions. The miss dontcare errors were mainly due to the LLM frequently overlooking the user's flexible preferences. For SBERT trained on 1% of RefPyDST's data, format errors were more prevalent compared to our method, and, similar to the upper bound, miss total remained the most frequent error. The main cause was the inability to capture the user's flexible preferences, followed by the failure to detect the agent's suggestions.

5.5 Experiment 4. Ablation Studies

In this experiment, we conduct several ablation studies to verify the effectiveness of CombiSearch, specifically focusing on the methods used for pool construction and combinatorial scoring. For easy comparison with other methods that use a different number of examples, we employed 3-shot ICL in all the ablation studies below.

Pool Construction In the pool construction step of CombiSearch, we use both BM25 and SBERT to retrieve example candidates. To analyze the contribution of these retrievers, we explored different

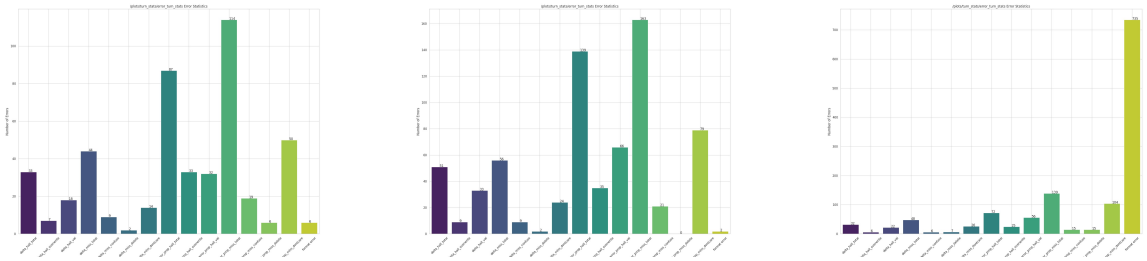

Figure 4: Statistics of error types made by different models in §5.3. From the left: CombiSearch, Hybrid, RefpyDST. Formatting error in RefPyDST is not included.

retrievers and measured theoretical DST performance.

As shown in Table 3, using both BM25 and SBERT (row 1) yielded a 14% higher JGA score than composing the pool with random examples (row 2). Additionally, using either BM25 or SBERT alone (rows 3 and 4) led to a performance drop of 3%-4% compared to the combined approach. This finding underscores the importance of example diversity within the pool. Employing both retrievers results in examples that capture a wider range of dialogue characteristics and enables the combinatorial scoring step to effectively consider and recognize useful examples.

Combinatorial Search In the combinatorial scoring step of CombiSearch, we repeatedly sample several examples and score them based on their contribution to correct predictions. An alterantive approach for combinatorial search is greedy search, which involves iteratively selecting the example with the highest contribution. To demonstrate the effectiveness of our method, we compared it with the greedy search-based method Se2 (Liu et al., 2024).

As shown in Table 4, CombiSearch outperformed Se2 by 12%. At the same time, CombiSearch is also more cost-efficient, requiring only 30 evaluations to construct $D_{\text{CombiSearch}}$ for a given query (i.e., 100 examples divided into 10 groups and evaluated 3 times), compared to the 150 evaluations needed by Se2 to select three examples.

Error Analysis We observed a notable behavior of the LLM that was not present in 10-shot ICL. The LLM often predicted all domain-related slots as "dontcare" in the first turn and then updated these predictions in subsequent turns. This suggests that, without sufficient examples, the LLM tends to overinterpret the user's not specifying certain slot values as indicating no preference. Other than this type of error, *miss total* errors were the most frequent across all models. The main cause was the failure to capture system suggestions.

6 Related Work

For related work, please refer to Appendix A.

7 Conclusion

We presented **CombiSearch**, a combinatorial method for scoring in-context examples in dialogue state tracking. We explored the theoretical upper bound of performance and demonstrated that CombiSearch-scored examples achieve a substantially higher JGA score than the SOTA model. In practical settings, the retriever trained on CombiSearch-scored examples also outperformed the SOTA model. Our error analysis tool sheds light on common error types made by LLMs. Our work highlights the importance of utilizing high-potential data for retriever training and identifies several areas of improvement in LLMs to reduce common errors.

8 Limitations

Our method enables the generation of training data for effective ICL example retrieval retrievers in DST. It also demonstrates high ideal performance, showing the upper bound for retriever training, which highlights the potential of our approach.

However, there are two issues. First, while it requires less computational cost than traditional in-context example selection methods, generating the data still incurs significant costs, with each example requiring 30 evaluations. Second, the performance of the practical retriever training fell short of the upper bound. This indicates the need for more detailed analysis of examples and a more refined training approach tailored to the data.

References

Ting-Yun Chang and Robin Jia. 2023. Data curation alone can stabilize in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8123–8144, Toronto, Canada. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian

Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards LLM-driven dialogue state tracking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 739–755, Singapore. Association for Computational Linguistics.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Brendan King and Jeffrey Flanigan. 2023. Diverse retrieval-augmented in-context learning for dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5570–5585, Toronto, Canada. Association for Computational Linguistics.

Haoyu Liu, Jianfeng Liu, Shaohan Huang, Yuefeng Zhan, Hao Sun, Weiwei Deng, Furu Wei, and Qi Zhang. 2024. se^2: Sequential example selection for in-context learning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5262–5284, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Tai Nguyen and Eric Wong. 2023. In-context example selection with influences. *Preprint*, arXiv:2302.11042.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK. Association for Computational Linguistics.

A Related Work

A.1 Dialogue State Tracking

Many works fine-tune a pretrained language model such as seq2seq models or causal language models to be suitable for DST (Wu et al., 2020; Hosseinin-Asl et al., 2020; Li et al., 2021; Su et al., 2021). There has been study that trained models to enhance cross-domain transferability(Wu et al., 2019). However, these approaches have the limitation that

the model needs to be retrained to apply it to different datasets or domains. To overcome these limitations, there have been studies aimed at addressing DST through in-context learning without the need for parameter updates(Hu et al., 2022; Chen et al., 2023; King & Flanigan, 2023). Specifically, we referenced the work of King & Flanigan (2023) the most. Their work addressed the DST problem with text-to-python and used ground-truth similarity. In our study, we went further by determining the upper bound of the LLM and devising a new scoring method. Although there has been study analyzing errors in DST(Hu et al., 2022), they did not focus on error analysis as the main analysis. In contrast, our research offers a more detailed categorization and in-depth analysis of the types of errors.

A.2 In-Context Example Selection

Choosing right method for selecting in-context examples is critical for effective ICL strategies. Previous work typically samples examples randomly, evaluates their significance, and prioritizes high-scoring examples during inference. Most approaches either select these high-scoring examples directly (Nguyen and Wong, 2023; Chang and Jia, 2023), or train retrievers using the scored examples as training dataset (Liu et al., 2024; King and Flanigan, 2023) . Prior work often assumes an example is significant if its presence improves dev-set performance. (Nguyen and Wong, 2023) measures the dev-set performance difference with and without each example. (Chang and Jia, 2023) uses average dev-set accuracy, or learns a linear regressor to estimate each example's impact on dev-set performance. (Liu et al., 2024) constructs a training dataset that captures sequential relationships between examples, while (King and Flanigan, 2023) trains a retriever to minimize the distance between examples that result in similar dialogue state changes.