Learning Complementary Action Pairs from Automotive Repair Manuals

Anonymous ACL submission

Abstract

Learning complementary action pairs is a fundamental prerequisite for procedural understanding in automotive repair. Repair manuals describe mechanical procedures as short, imperative instructions that implicitly encode inverse action relations, such as assembly–disassembly or tighten–loosen, without explicit semantic links. Recovering these relations is challenging due to highly repetitive documentation, concise phrasing, and a strong dependence on contextual cues. Although expert-defined heuristics can resolve a subset of complementary action pairs, such rules generalize poorly to linguistic variation and achieve only partial coverage in practice. We study the problem of learning complementary action pairs from real-world automotive repair manuals. We construct a dataset of complementary repair step pairs using automatic alignment followed by manual verification, and investigate learning-based formulations based on contrastive representation learning and generative modeling. Experimental results show that learned models capture complementary action relations beyond surface-level similarity and generalize more robustly than rule-based approaches under realistic industrial constraints.

1 Introduction

Modern industrial domains increasingly rely on large-scale procedural documentation to support maintenance, repair, and quality assurance workflows. In the automotive domain, repair manuals play a central role by specifying how vehicle elements are disassembled and reassembled during maintenance operations. Automatically learning such manuals has significant practical impact, ranging from intelligent repair assistance to automated consistency checking and documentation reuse.

A fundamental challenge is that repair manuals encode much of their procedural structure implicitly. Complementary mechanical actions are often authored as separate instructions without explicit links, even though they describe inverse operations applied to the same components. Human experts recognize these relations naturally, but they are difficult to recover automatically due to the highly repetitive text, concise phrasing, and strong dependence on contextual cues.

Identifying complementary action pairs is a prerequisite for higher-level procedural learning. Such relations reveal latent symmetry in repair workflows and provide structural constraints that are essential for downstream tasks such as process alignment and automatic repair manual generation. However, existing approaches struggle to robustly identify these relations in real-world industrial documentation.

In this work, we study the problem of automatically identifying complementary mechanical actions from real-world automotive repair manuals. We formulate this task as a locally constrained disambiguation problem and investigate how expert-informed rules and learned representations can be combined to recover implicit procedural structure. Although a substantial subset of complementary action pairs can be resolved using expert-defined rules, such heuristics achieve only partial coverage in practice. In our data, rule-based alignment accounts for approximately 64% of valid complementary pairs, leaving a large fraction of semantically valid relations unresolved. This gap highlights the need for learning-based approaches that can generalize across lexical variation and implicit procedural contexts.

Contributions. This paper makes the following contributions:

1. We investigate learning-based approaches for modeling complementary actions, including contrastive representation learning and generative formulations, and demonstrate their

effectiveness on a challenging industrial repair dataset.

2. We release a dataset of complementary action pairs derived from real-world automotive repair manuals, constructed through automatic alignment and manual verification.[1]

2 Related Work

Complementary action learning lies at the intersection of procedural text understanding, industrial information extraction, and semantic modeling of action relations. Prior work has studied how actions, entities, and state changes can be inferred from instructional text, primarily focusing on well-structured narratives or explicitly annotated benchmarks. In contrast, industrial repair manuals encode procedural structure and inverse action relations implicitly, posing challenges for both rule-based and learning-based approaches. This section reviews related work along four dimensions: procedural text understanding, industrial information extraction, contrastive representation learning for semantic matching, and generative modeling of procedural actions.

Procedural Text Understanding. Procedural text understanding has been widely studied as a problem of modeling actions, entities, and state changes across sequences of instructions. Dalvi et al. (Dalvi et al., 2018) introduce a benchmark for tracking entity state changes in process descriptions, while Gupta and Durrett (Gupta and Durrett, 2019) leverage transformer architectures for improved entity tracking. Subsequent work moves beyond linear narratives by explicitly modeling procedural structure. Das et al. (Das et al., 2018) construct dynamic knowledge graphs from text using machine reading comprehension, and Huang et al. (Huang et al., 2021) propose reasoning over entity–action–location graphs to capture richer procedural dependencies. While effective on curated benchmarks such as cooking recipes or scientific processes, these approaches typically assume explicit state transitions or clean narrative structure, and are less suited to industrial documentation where complementary relations are implicit and procedural steps are highly repetitive.

Rule-Based and Hybrid Information Extraction in Industrial Text. Rule-based information extraction remains prevalent in industrial NLP due to its interpretability, precision, and robustness when expert knowledge can be encoded explicitly. Chiticariu et al. (Chiticariu et al., 2013) argue that rule-based systems continue to play a central role in enterprise applications, particularly under limited supervision. Hybrid approaches that combine expert-defined rules with statistical or neural models have been explored to improve generalization while retaining reliability. Such methods are especially relevant for technical documentation, where domain-specific terminology and structural regularities can be exploited by heuristics, but linguistic variation and implicit semantics limit purely rule-driven solutions. Our work follows this hybrid paradigm by using expert-informed alignment to resolve clear complementary action pairs while seperate semantically challenging cases for learning-based modeling.

Contrastive Representation Learning for Semantic Matching. Contrastive representation learning has shown strong performance on semantic matching and retrieval tasks, particularly with transformer-based encoders. InfoNCE-style objectives (van den Oord et al., 2018) enable models to learn fine-grained alignment from paired data using in-batch negatives. Sentence-level contrastive learning has been widely applied to paraphrase identification, cross-lingual matching, and retrieval. However, the prior work by Søgaard et al. (Søgaard et al., 2021) assumes independently sampled text pairs and evaluates under random instance-level splits. In contrast, our work explicitly studies leakage-aware contrastive learning under bucket-level isolation, reflecting realistic structural constraints in procedural documentation.

Generative Modeling of Procedural Text. Generative approaches offer an alternative perspective on procedural understanding by modeling instructions as sequences to be produced rather than retrieved. Early neural models focus on domains such as cooking recipes. Kiddon et al. (Kiddon et al., 2016) propose neural checklist models to ensure global coherence during generation, while Dhingra et al. (Dhingra et al., 2018) explicitly model action-induced state changes. With the advent of large pretrained language models, generative procedural modeling has gained renewed interest. H.Lee et al. (H. Lee et al., 2020) demonstrate that pretrained GPT-style models capture substantial procedural knowledge when fine-tuned for recipe gener-

[1]The dataset will be made publicly available upon acceptance of the paper.

ation. Despite their fluency, purely generative models may suffer from hallucination or inconsistency, particularly in domain-specific or low-resource settings.

Retrieval-Augmented and Script-Based Generation. To mitigate the limitations of purely generative models, several works integrate retrieval or script knowledge into generation. Nishimura et al. (Nishimura et al., 2019) retrieve relevant instructional steps from large corpora and condition generation on retrieved exemplars, improving factual accuracy. Related, script learning and event prediction research models procedural knowledge as partially ordered sequences of actions. Sakaguchi et al. (Sakaguchi et al., 2021) show that pretrained transformers can generate plausible scripts for everyday tasks from minimal prompts. Although these approaches capture regularities in action sequences, they do not explicitly address inverse or complementary action relations.

Complementary Actions and Inverse Relations. Learning complementary or inverse actions has received comparatively limited direct attention. The existing work ClarET (Zhou et al., 2022) treats such relations implicitly through next-event prediction or script completion. Recent study by Zhu et al. (Zhu et al., 2022) combining contrastive and generative objectives have shown promise for modeling complex event relations, but remain sensitive to model capacity and pretraining quality. Our work complements this literature by systematically comparing contrastive retrieval and generative formulations to identify complementary actions under realistic industrial constraints, highlighting their respective strengths and limitations.

3 Dataset Construction

This section describes the construction of a dataset for learning complementary action pairs from real-world automotive repair manuals. Beyond collecting paired repair steps, our goal is to preserve the procedural structure inherent in industrial documentation, which plays a central role in defining valid complementary relations. To this end, we explicitly model repair processes as structural units and use them to organize repair steps into coherent contextual groups. This design enables both faithful representation of real repair workflows and leakage-aware evaluation of learning-based models under realistic constraints.

3.1 Source Repair Manuals

Our dataset is derived from automotive repair manuals provided by an original equipment manufacturer (OEM) and used in real industrial settings. The manuals describe vehicle maintenance and repair procedures as structured textual instructions. They are organized hierarchically into repair tasks, repair processes, and repair steps, reflecting different levels of procedural granularity.

A repair task specifies a high-level maintenance objective, such as replacing or servicing a major vehicle component. Each task is decomposed into multiple repair processes that describe coherent procedural phases, for example disassembly and assembly. Repair steps form the most fine-grained level and consist of short imperative instructions describing concrete mechanical actions applied to vehicle components. In this work, we operate exclusively at the level of individual repair steps, which naturally give rise to complementary action relations.

In the original repair manuals, the repair steps are not independent, but organized under repair processes, each of which represents a coherent procedural phase. In this work, we formalize each repair process as a *bucket*, identified by a unique bucket identifier. All repair steps belonging to the same repair process are assigned the same bucket ID, and therefore complementary action pairs are modeled and evaluated within this process-level context. This formulation preserves the procedural structure of real repair manuals, while enabling structured learning and leakage-aware evaluation of complementary actions.

3.2 Construction of Complementary Action Pairs

We construct a dataset of pairs of complementary repair steps by combining automatic alignment with manual verification. From parsed repair manuals, candidate complementary steps are identified using expert-defined heuristics that exploit structural and textual regularities in the documentation.

Automatic alignment is driven by a domain-specific action dictionary encoding known pairs of inverse mechanical actions (e.g., remove–install or loosen–tighten). Repair step texts are lightly normalized to abstract away surface-level variation while preserving references to affected components. Candidate pairs are then generated by matching steps that express inverse actions applied to the

same components within compatible procedural contexts.

To handle linguistic variation, action terms are removed to obtain an action skeleton, and matching is performed using character-level string similarity with a conservative threshold. Low-impact modifiers that do not change the underlying mechanical operation are allowed, while location- or role-specific terms are treated as hard constraints to prevent spurious matches.

The automatically aligned candidate pairs are subsequently reviewed to ensure correctness. Manual verification resolves ambiguous cases and filters out remaining noise. The final dataset consists exclusively of high-confidence one-to-one complementary repair step pairs, with all identifiers anonymized for public release. Although expert-defined alignment rules achieve high precision, their coverage is inherently limited. In a quantitative analysis of the complete dataset, the rule-based pipeline successfully resolves approximately 64% of complementary action pairs, failing primarily in cases involving lexical divergence, implicit component references, or context-dependent formulations. These unresolved instances motivate the use of learning-based models to generalize beyond fixed action dictionaries and surface-level matching heuristics. To provide a high-level overview of the experimental pipeline, we include a process diagram in Figure 1. It summarizes the data flow from raw repair manuals through rule-based alignment and manual verification, culminating in the generation of high-confidence complementary action pairs. These pairs serve as input to two distinct experimental branches: a contrastive retrieval formulation and a generative modeling formulation.

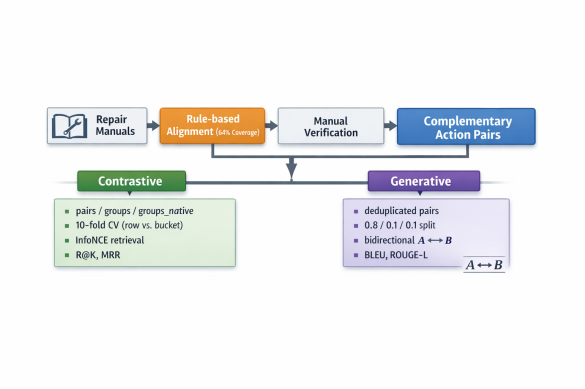

Figure 1: Overview of the experimental pipeline. Complementary action pairs are extracted via rule-based alignment and manual verification, then evaluated through contrastive and generative learning formulations.

4 Action-Centric Contrastive Learning Experiments

This section evaluates contrastive retrieval models for the task of correctly matching semantically corresponding text pairs. We focus on how imposing different levels of structural constraints on the data affects this matching objective. Specifically, we compare a pair-level setting, where contrastive learning is performed on deduplicated text pairs without bucket identifiers, with a bucket-aware setting, where text pairs are organized under bucket-level structure to enable structured contrastive matching. By contrasting these two settings with an otherwise identical objective, our experiments isolate the impact of bucket-level constraints on contrastive retrieval performance.

4.1 Training Data Preparation

The data preparation pipeline is designed to support two contrastive learning settings that differ in whether bucket-level structure is imposed on text pairs. Both settings start from the same raw dataset and share an initial deduplication step, after which the processing diverges depending on the experimental design.

We first remove all exact duplicate text pairs from the dataset. The resulting deduplicated dataset can be directly used for *pair-level* contrastive learning, where each text pair is treated as an independent instance and no higher-level structural assumptions are made. This setting does not require additional data transformation beyond deduplication.

For the *bucket-aware* contrastive setting, additional processing is required to ensure that all text pairs are associated with a bucket identifier, enabling structured definitions of positive and negative pairs. While a subset of text pairs in the original data is naturally grouped through existing bucket annotations, many pairs exist as isolated instances. To enable systematic bucket-aware experimentation, we therefore reorganize singleton buckets through a controlled re-bucketing procedure.

New buckets are generated to replicate the size distribution of the native buckets. The number of text pairs assigned to each bucket is randomly drawn from 2 to 6, following the distribution ob-

served in the native data. In addition, the proportion of newly constructed buckets is globally constrained to align with that of the native dataset, ensuring consistency between the original and reconstructed bucket populations.

To align the internal semantic characteristics of newly constructed buckets with those of native buckets, we encode all text pairs using normalized sentence embeddings from a multilingual Sentence-Transformer based on paraphrase-multilingual-mpnet-base-v2. The intra-bucket semantic dispersion is calibrated based on the distance distribution observed within native buckets: text pairs are grouped such that the pairwise ℓ_2 distances between embeddings within the same bucket follow the same empirical range as those observed in the original bucket structure. This procedure ensures that newly constructed buckets are not only similar in size, but also comparable in semantic variability.

We emphasize that the `groups_native` setting differs from `groups` only in data filtering, where buckets labeled as `new` are excluded. The contrastive loss, the training procedure, and the model configuration remain identical in the two settings. As a result, performance differences between `groups` and `groups_native` can be attributed solely to changes in the structure of the data rather than to optimization or modeling artifacts.

4.2 Evaluation Protocol

To ensure a fair and controlled comparison between pair-level and bucket-aware contrastive learning, we adopt evaluation protocols that differ only in how structural constraints are enforced during data splitting, while keeping the training objective unchanged.

All contrastive experiments are evaluated using k-fold cross-validation with $k = 10$. In the pair-level setting, folds are constructed at the row level, treating each deduplicated text pair as an independent instance. In contrast, for the bucket-aware settings (`groups` and `groups_native`), folds are constructed at the group level based on the `bucket_id`, so that all text pairs associated with the same bucket are assigned to the same fold. We fix the test fold offset to one, ensuring that validation and test folds do not overlap in any run.

Final performance scores are obtained by aggregating results across the ten folds, allowing us to assess both average performance and variability in each experimental setting.

4.3 Models and Training Objective

We evaluated four encoder families for contrastive text matching: a multilingual BERT encoder, a German-specific BERT encoder, an MPNet-based Sentence-Transformer encoder, and a character-level Transformer trained from scratch. For readability, we refer to these models as *multilingual BERT*, *German BERT*, *MPNet-based encoder*, and *character-level encoder* in the remainder of this chapter.

Despite architectural differences, all encoders are trained and evaluated under a unified retrieval formulation. Multilingual BERT and German BERT use CLS pooling, the MPNet-based encoder uses sentence-level pooling as defined in the Sentence-Transformer framework, and the character-level encoder uses mean pooling over token representations. All encoders output ℓ_2-normalized vectors and use dot-product similarity, which is equivalent to cosine similarity under normalization. This ensures a consistent similarity scale across models.

We train all encoders with a symmetric InfoNCE objective using *in-batch negatives* only. For a mini-batch of N paired texts $\{(a_i, b_i)\}_{i=1}^{N}$, the encoder produces ℓ_2-normalized embeddings z_i^a and z_i^b, and we define the temperature-scaled similarity

$$s_{ij} = \frac{z_i^a \cdot z_j^b}{\tau}, \tag{1}$$

where τ is a temperature hyperparameter. The overall loss is the average of the two directional objectives:

$$\mathcal{L} = \frac{1}{2}\left(\mathcal{L}_{ab} + \mathcal{L}_{ba}\right), \tag{2}$$

with

$$\begin{aligned} \mathcal{L}_{ab} &= -\frac{1}{N}\sum_{i=1}^{N}\left(s_{ii} - \log\sum_{j=1}^{N}\exp(s_{ij})\right), \\ \mathcal{L}_{ba} &= -\frac{1}{N}\sum_{i=1}^{N}\left(s_{ii} - \log\sum_{j=1}^{N}\exp(s_{ji})\right). \end{aligned} \tag{3}$$

All BERT and Sentence-Transformer encoders are fine-tuned end-to-end under this objective, whereas the character-level encoder is trained from scratch. Optimization is performed with AdamW using a fixed random seed. For contrastive learning, we do not apply early stopping; instead, we

train for a fixed number of epochs and select the best checkpoint by validation Mean Reciprocal Rank (MRR). The pair-level setting is trained for 20 epochs with batch size 64, while the bucket-aware settings (`groups` and `groups_native`) are trained for 30 epochs with batch size 128. Unless stated otherwise, all settings share the following hyperparameters: learning rate 1×10^{-5}, weight decay 0.01, maximum sequence length 256, gradient clipping at 1.0, and mixed-precision training when using CUDA. The full training configuration is reported in Table 3 in the Appendix A.1, Table 4 in the Appendix A.2 and Table 5 in the Appendix A.3.

4.4 Metrics

Contrastive retrieval performance is evaluated using ranking-based metrics that directly reflect the correctness of text pair matching. We report Recall@1, Recall@5 and Recall@10, which measures the proportion of queries for which the correct matching text is ranked first, and Mean Reciprocal Rank (MRR), which captures the average inverse rank of the first correct match across all queries and is sensitive to ranking quality at the top of the list. All metrics are computed on held-out evaluation splits for each fold and then aggregated across folds to obtain mean performance and standard deviation.

4.5 Results and Analysis

It is important to emphasize that the contrastive models are not merely replicating the behavior of the rule-based alignment. Rather, they are trained on the full dataset of verified pairs, including the 36% of cases that cannot be resolved by expert-defined heuristics. The strong retrieval performance under bucket-level isolation indicates that learned representations capture complementary relations beyond fixed action dictionaries. Table 1 summarizes the main contrastive retrieval results under different training settings, where each setting reports the best-performing encoder, and all scores are averaged over 10-fold cross-validation. The complete results are recorded in the Table 6 in Appendix B.

The results show a clear effect of structural constraints on contrastive text matching. In the pair-level setting, where no bucket-level constraint is imposed, retrieval performance is very strong, with the best model achieving an MRR of 0.94. This indicates that, when structural overlap between semantically related text pairs is unconstrained, contrastive learning can effectively exploit local semantic regularities to identify correct matches.

Training Setting	Encoder	R@1	MRR
pairs	MPNet-based encoder	0.91	0.94
groups	German BERT	0.68	0.77
groups-native	MPNet-based encoder	0.74	0.83

Table 1: Main contrastive retrieval results under different training settings. Each row reports the best-performing encoder for the corresponding setting. All results are averaged across 10-fold cross-validation.

When bucket-level structure is enforced, retrieval performance drops substantially. Under the bucket-aware setting, the best MRR decreases to 0.77, corresponding to an absolute reduction of more than 0.17 compared to the pair-level baseline. This gap highlights the increased difficulty of matching text pairs once models are required to generalize across bucket boundaries rather than relying on semantically related pairs appearing in both training and evaluation data.

Restricting evaluation to native buckets leads to a partial recovery in performance, with the best MRR increasing to 0.83. This improvement suggests that newly constructed buckets introduce additional distributional challenges beyond those present in the original data. Nevertheless, performance under this setting remains well below the pair-level results, indicating that the dominant source of the performance gap is the removal of bucket-level overlap rather than artifacts introduced during bucket construction.

5 Generative Experiments

This chapter investigates a generative formulation of complementary action learning. Unlike contrastive retrieval, which implicitly assesses semantic alignment through representation similarity, generative modeling requires complementary relations to be encoded explicitly enough to support conditional sequence generation. We therefore position generative experiments not as a primary replacement for retrieval-based matching, but as a diagnostic probe into whether complementary repair actions can be modeled as structured, directional mappings.

This perspective is particularly relevant given the limitations of rule-based alignment. While expert-defined heuristics resolve approximately 64% of complementary action pairs in our data, the remain-

ing cases involve lexical divergence, implicit component references, or context-dependent phrasing. Generative modeling provides a means to evaluate whether learning-based methods capture these relations beyond surface similarity, by testing whether one repair step can be generated directly from its complementary counterpart.

5.1 Experimental Setup

All generative experiments are conducted on the deduplicated text-pair dataset. Data are split at the row level into training, validation, and test sets using a ratio of 0.8/0.1/0.1, without cross-validation. Models are trained under a bidirectional generation scheme, where both directions (A→B and B→A) are optimized jointly. Each direction follows a standard sequence-to-sequence formulation, and losses from both directions are summed during training.

We evaluated five multilingual sequence-to-sequence models covering different architectures and model capacities: mBART-large-50-many-to-many-mmt, FLAN-T5-large, FLAN-T5-base, mT5-large, and mT5-base. This selection enables the analysis of both architectural differences and capacity effects under a unified training objective.

Across all experiments, we used a maximum generation length of 128 tokens, greedy decoding with beam size 1, a batch size of 16, and train for up to 20 epochs with a learning rate of 1×10^{-5}. Optimization is performed using AdamW with weight decay 0.01 and mixed-precision training. Early stopping is applied based on validation performance with a patience of two epochs. The best checkpoint is selected according to validation using ROUGE-L.

5.2 Training Objective

Each generative model is optimized with a combined objective consisting of a bidirectional sequence-to-sequence loss and a contrastive regularization term. For a mini-batch of paired texts $\{(a_i, b_i)\}_{i=1}^{N}$, the generative component is defined as the sum of the token-level cross-entropy losses in both directions:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{CE}}(A \to B) + \mathcal{L}_{\text{CE}}(B \to A). \quad (4)$$

To explicitly regularize semantic consistency, we introduce a contrastive term computed on decoded sequences rather than teacher-forced representations. Let $\hat{b}_i = \text{generate}(a_i)$ and $\hat{a}_i = \text{generate}(b_i)$ denote the generated outputs. Generated texts and gold references are embedded using a fixed multilingual BERT encoder with CLS pooling and ℓ_2 normalization. A symmetric InfoNCE loss with in-batch negatives is then applied:

$$\mathcal{L}_{\text{ctr}} = \mathcal{L}_{\text{InfoNCE}}(B, \hat{B}) + \mathcal{L}_{\text{InfoNCE}}(A, \hat{A}). \quad (5)$$

The overall objective is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{gen}} + \lambda \mathcal{L}_{\text{ctr}}, \quad (6)$$

with $\lambda = 0.5$ for all experiments.

5.3 Evaluation Metrics

We report BLEU and ROUGE-L on the validation set as surface-level measures of generation fidelity. BLEU evaluates n-gram precision against reference outputs, whereas ROUGE-L assesses global sequence similarity via the longest common subsequence. While these metrics do not fully capture procedural correctness, the high lexical overlap expected between complementary repair steps makes them reasonable first-order proxies for conditional generation quality.

5.4 Results and Analysis

Table 2 summarizes generative performance on the validation set. Among all evaluated models, mBART-large-50 achieves the strongest results across both BLEU and ROUGE-L, followed by FLAN-T5-large and mT5-large. This ranking is consistent with differences in pretraining strategies and model capacity The complete results of the generative experiments are shown in Table 7 of Appendix C.

A pronounced performance gap is observed between large and base variants within the same model families. FLAN-T5-base performs substantially worse than its large counterpart, indicating that reduced capacity limits the ability to model fine-grained complementary relations. More strikingly, mT5-base exhibits near-degenerate behavior, yielding almost zero BLEU and ROUGE-L scores. This failure suggests that bidirectional generation combined with contrastive regularization exceeds the representational capacity of smaller multilingual models. In contrast, larger pretrained models reliably capture complementary relations through generation, indicating that generative formulations are viable only under sufficient model capacity. Compared to contrastive retrieval, which remains stable across model sizes, generative modeling is therefore more sensitive to architectural and pretraining choices.

Model	BLEU	ROUGE-L
mBART-large-50	57.99	0.7751
FLAN-T5-large	56.07	0.7608
mT5-large	55.16	0.7462
FLAN-T5-base	50.57	0.7337
mT5-base	0.40	0.0550

Table 2: Generative performance on the validation set for five multilingual seq2seq models trained under the bidirectional generation objective (A→B and B→A) with contrastive regularization ($\lambda = 0.5$, $\tau = 0.05$). All models use the same training configuration (max length 128, batch size 16, learning rate 1×10^{-5}) and greedy decoding (beam=1). The reported scores correspond to the best checkpoint selected by validation ROUGE-L.

6 Conclusion

This work studies the problem of learning complementary action pairs from automotive repair manuals, where procedural structure is implicit and inverse relations between actions are rarely stated explicitly. Starting from expert-informed alignment heuristics, we demonstrate that rule-based approaches, while precise, achieve only partial coverage, resolving approximately 64% of complementary action pairs in practice.

To address this gap, we investigate learning-based formulations under realistic industrial constraints. Through leakage-aware contrastive experiments, we find that the unconstrained text-pair setting yields the best empirical performance. When introducing additional bucket-level constraints to enforce stricter isolation, performance decreases, although the overall trends remain consistent and the model still learns useful representations. This suggests that, particularly in small, domain-specific datasets, overly restrictive structural constraints may reduce the amount of informative cross-instance signal available for representation learning. Taken together, our results indicate a trade-off between evaluation rigor and achievable performance: stronger constraints improve the reliability of the protocol but can limit the model's ability to leverage fine-grained textual regularities.

Complementing this retrieval-based perspective, we explore generative modeling as a diagnostic probe for complementary action learning. Our experiments show that large pretrained sequence-to-sequence models can explicitly generate complementary repair steps when sufficient model capacity is available. However, generative formulations are highly sensitive to capacity and optimization stability, and may fail entirely in low-capacity regimes. In contrast, contrastive learning provides a more stable and scalable mechanism for modeling complementary relations across model sizes.

Taken together, our findings suggest that contrastive and generative approaches capture complementary aspects of procedural understanding. Contrastive learning offers robustness and generalization under realistic structural constraints, while generative modeling provides insight into whether complementary relations are encoded as explicit, directional mappings. This combined perspective supports practical applications such as intelligent repair assistance, automated consistency checking between disassembly and assembly procedures, and scalable reuse of industrial repair documentation.

7 Limitations

While our results demonstrate the effectiveness of learning-based approaches for modeling complementary actions in repair manuals, several limitations remain.

First, the dataset is derived from a single vehicle platform and documentation structure, which may limit generalization to other manufacturers, languages, or manual styles. Although the proposed methods are not language-specific, broader industrial or multilingual deployment would require further validation and potentially domain-specific preprocessing.

Second, the models depend on an initial rule-based candidate matching stage. Although learning substantially improves robustness beyond handcrafted rules, biases or errors introduced during candidate construction may still propagate into the learned representations. Fully data-driven discovery of complementary actions therefore remains an open challenge.

Third, the formulation emphasizes locally constrained disambiguation within structurally related procedural contexts rather than global retrieval across entire manuals. While this reflects common industrial authoring practices, it does not address cases where inverse actions occur in distant or weakly connected sections.

Finally, the generative formulation exhibits a strong dependence on model capacity and pretraining quality. Large pretrained sequence-to-sequence models perform reliably, whereas smaller models often show instability or degenerate generation,

suggesting that purely generative approaches may be unsuitable for low-capacity settings and motivating future work on parameter-efficient or hybrid contrastive–generative methods.

Acknowledgements

The authors acknowledge the use of generative AI tools to assist with language polishing and code debugging during the preparation of this manuscript. All generated content was carefully reviewed, verified, and edited by the authors, who take full responsibility for the correctness of the writing, methods, and results presented in this paper.

References

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of EMNLP*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *Proceedings of NAACL-HLT*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018. Building dynamic knowledge graphs from text using machine reading comprehension. In *International Conference on Learning Representations*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Aditya Gupta and Greg Durrett. 2019. Effective use of transformer networks for entity tracking. In *Proceedings of EMNLP-IJCNLP*.

Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, and Lav R. Varshney. 2020. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In *Companion Proceedings of the Web Conference 2020*, WWW '20. ACM.

Hao Huang, Xiubo Geng, Jian Pei, Guodong Long, and Daxin Jiang. 2021. Reasoning over entity-action-location graph for procedural text understanding. In *Proceedings of ACL*.

Chloe Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of EMNLP*.

Taichi Nishimura, Atsushi Hashimoto, and Shinsuke Mori. 2019. Procedural text generation from an image sequence. In *Proceedings of the 12th International Conference on Natural Language Generation*.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. Script generation with pre-trained language models. In *Proceedings of ACL*.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Yucheng Zhou, Tao Shen, Xiubo Geng, Guodong Long, and Daxin Jiang. 2022. Claret: Pre-training a correlation-aware context-to-event transformer for event-centric generation and classification.

Fangqi Zhu, Jun Gao, Changlong Yu, Wei Wang, Chen Xu, Xin Mu, Min Yang, and Ruifeng Xu. 2022. A generative approach for script event prediction via contrastive fine-tuning.

A Contrastive Training Configuration

This appendix lists the full training configuration used for contrastive retrieval. Values in Table 3 are defaults from `configs/base.yaml`; overrides for each experiment are shown in Table 4.

A.1 Common Settings

Setting	Value
Random seed	42
Device	auto
Optimizer	AdamW
Learning rate	1×10^{-5}
Weight decay	0.01
Epochs (default)	20
Batch size (default)	64
Max length	256
Grad clip norm	1.0
Log interval	10 steps
Mixed precision	True (CUDA only)
InfoNCE temperature	0.05
Eval metrics	R@1, R@5, R@10, MRR
Text columns	`aus_text`, `ein_text`
Bucket column	`bucket_id`
DataLoader workers	0

Table 3: Common hyperparameters used across contrastive experiments (from `configs/base.yaml`).

A.2 Experiment-Specific Overrides

Setting	Pairs	Groups	Groups-native
Dataset	`train_dedup_by_text_pairs.csv`	`rebucketed_singletons.csv`	`rebucketed_singletons.csv`
Split type	k-fold (row)	k-fold (group)	k-fold (group)
Group column	–	`bucket_id`	`bucket_id`
k (folds)	10	10	10
Fold index	0	0	0
Test fold offset	1	1	1
Epochs	20	30	30
Batch size	64	128	128
Temperature τ	0.07	0.05	0.05
Native-only filter	–	–	`bucket_id` not starting with "`new`"

Table 4: Experiment-specific configuration overrides (from `configs/pairs.yaml`, `configs/groups.yaml`, and `configs/groups_native.yaml`).

A.3 Encoder Configurations

Encoder	Backend	Model / Architecture	Pooling / Normalize
Multilingual BERT	HF	`bert-base-multilingual-cased`	CLS / L2-norm
German BERT	HF	`bert-base-german-cased`	CLS / L2-norm
ST-MPNet	ST	`paraphrase-multilingual-mpnet-base-v2`	ST / L2-norm
Char Transformer	Char	V=5000, d=512, h=8, L=6, FF=2048, drop=0.1	Mean / L2-norm

Table 5: Configurations of different encoders used in contrastive experiments (from `configs/encoders/*.yaml`).

B Contrastive Retrieval Results (10-fold mean±std)

Pairs (k=10, mean±std) Encoder	Recall@1	Recall@5	Recall@10	MRR
Multilingual BERT	0.9027 ± 0.0250	0.9863 ± 0.0115	0.9945 ± 0.0060	0.9388 ± 0.0166
German BERT	0.8979 ± 0.0206	0.9808 ± 0.0114	0.9925 ± 0.0048	0.9348 ± 0.0157
ST-MPNet	0.9089 ± 0.0210	0.9870 ± 0.0112	0.9966 ± 0.0063	0.9430 ± 0.0135
Char Transformer	0.8472 ± 0.0328	0.9548 ± 0.0157	0.9726 ± 0.0150	0.8942 ± 0.0233
Groups (k=10, mean±std) Encoder	Recall@1	Recall@5	Recall@10	MRR
Multilingual BERT	0.6820 ± 0.0204	0.8680 ± 0.0181	0.9326 ± 0.0172	0.7667 ± 0.0174
German BERT	0.6812 ± 0.0214	0.8672 ± 0.0199	0.9296 ± 0.0184	0.7672 ± 0.0186
ST-MPNet	0.6785 ± 0.0207	0.8698 ± 0.0203	0.9345 ± 0.0181	0.7662 ± 0.0174
Char Transformer	0.6656 ± 0.0267	0.8546 ± 0.0197	0.9203 ± 0.0197	0.7543 ± 0.0218
Groups-native (k=10, mean±std) Encoder	Recall@1	Recall@5	Recall@10	MRR
Multilingual BERT	0.7337 ± 0.0198	0.9599 ± 0.0265	0.9935 ± 0.0072	0.8259 ± 0.0161
German BERT	0.7330 ± 0.0243	0.9619 ± 0.0220	0.9946 ± 0.0073	0.8260 ± 0.0189
ST-MPNet	0.7384 ± 0.0221	0.9589 ± 0.0199	0.9935 ± 0.0053	0.8289 ± 0.0166
Char Transformer	0.7058 ± 0.0323	0.9403 ± 0.0248	0.9867 ± 0.0158	0.8040 ± 0.0244

Table 6: Contrastive retrieval results for Pairs, Groups, and Groups-native. All scores are mean±std over 10-fold cross-validation. Within each fold, the best checkpoint is selected by validation MRR; Recall@1/5/10 and MRR are reported on the validation split.

C Generative Results on the Validation Set (best checkpoint by ROUGE-L)

Model	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
facebook/mbart-large-50-many-to-many-mmt	57.9882	0.7851	0.6673	**0.7751**
google/flan-t5-base	50.5703	0.7497	0.6136	0.7337
google/flan-t5-large	56.0744	0.7714	0.6515	0.7608
google/mt5-base	0.3955	0.0559	0.0131	0.0550
google/mt5-large	55.1577	0.7615	0.6389	0.7462

Table 7: Generative results on the validation set for five seq2seq models trained with bidirectional generation and contrastive regularization ($\lambda = 0.5$, $\tau = 0.05$) on the deduplicated pair dataset (row split 0.8/0.1/0.1). The best checkpoint is selected by validation ROUGE-L; BLEU and ROUGE-1/2/L are reported.