Evaluating Spatiotemporal Consistency in Automatically Generated Sewing Instructions

Luisa Geiger[*+], **Mareike Hartmann**[+], **Michael Sullivan**[+], **Alexander Koller**[+]
[*]Congree Language Technologies GmbH
lgeiger@congree.com
[+]Department of Language Science and Technology, Saarland University
{mareikeh, msullivan, koller}@coli.uni-saarland.de

Abstract

In this paper, we propose a novel, automatic tree-based evaluation metric for LLM-generated step-by-step assembly instructions, that more accurately reflects spatiotemporal aspects of construction than traditional metrics such as BLEU and BERT similarity scores. We apply our proposed metric to the domain of sewing instructions, and show that our metric better correlates with manually-annotated error counts as well as human quality ratings, demonstrating our metric's superiority for evaluating the spatiotemporal soundness of sewing instructions. Further experiments show that our metric is more robust than traditional approaches against artificially-constructed counterfactual examples that are specifically constructed to confound metrics that rely on textual similarity.

1 Introduction

Creating consistent, high quality instructions in any domain can be a difficult process. Much research has investigated the ability of Large Language Models (LLMs) to generate such instructions, most commonly in the domain of cooking recipes (e.g. Li et al., 2024; Salvador et al., 2019).

High quality instructions make it as easy as possible for the reader to connect the textual instructions to the required actions in the physical world. For the instructions to make sense with regard to the physical world, the LLM that is generating them needs an understanding of the current world state. This necessitates spatial (where objects are located) and temporal (when objects change their location) awareness: although LLMs have continuously improved on a wide variety of benchmarks, they still have difficulties with consistency in the area of spatiotemporal reasoning (Aghzal et al., 2025).

In particular, the generation of textual sewing instructions is a multimodal reasoning problem that necessitates spatiotemporal awareness of multiple objects and their surroundings, as well as knowledge of the current world state: specifically, the current assembly state of the garment at each step of the instructions.

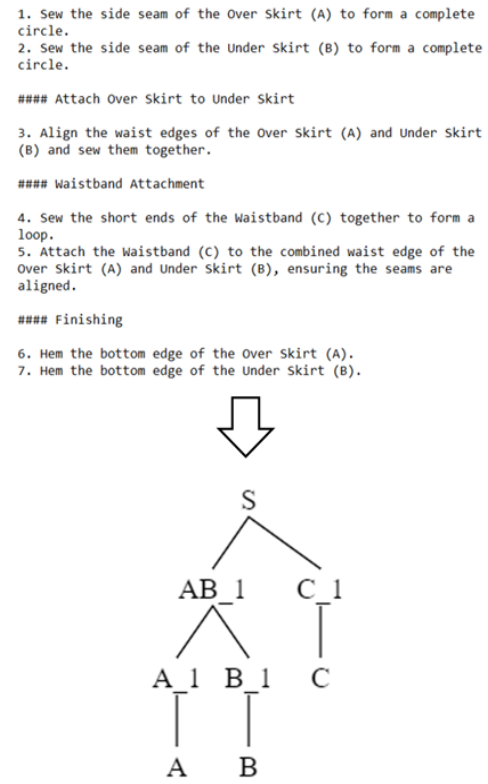

Figure 1: Illustration of our tree-based evaluation metric. Instructions are automatically converted a tree that encodes the order and components of each step. These automatically-extracted trees are then compared to gold trees to yield a score.

This task is similar to other instruction generation tasks such as cooking recipe generation and Minecraft instruction generation (e.g. Narayan-Chen et al., 2019), in that the generation task consists of generating procedural textual instructions, including explicit or implicit representations/descriptions of the current world state.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 18508–18525
November 4-9, 2025

However, generating sewing instructions differs from these more widely-researched tasks, due to the more complex operations that need to be performed.

The complexity of this process lends to sensitivity to even slight errors: incorrect use of terminology or ambiguous explanations of assembly operations can lead to catastrophic misunderstandings between writer and the reader. While this can also be the case for other instruction generation tasks, the increased complexity involved in the physical manipulation of fabric pieces leads to a far more spatiotemporally sensitive process than (for example) stacking blocks.

Since the quality of sewing instructions heavily depends on their ability to capture the correct way to assemble the pieces, evaluating generated sewing instructions in meaningful way is not a trivial task. Common evaluation metrics such as BLEU (Papineni et al., 2002) and BERT-Score (Zhang et al., 2020) only focus on superficial similarities between the generated instructions and a given set of gold reference instructions (see Sections 2 and 5).

In this paper, we introduce an automatic, tree-based evaluation metric (Section 4), that mitigates the shortcomings of traditional evaluation metrics with respect to the task of instruction generation. We compare our evaluation metric to BLEU, ROUGE-L (Lin, 2004), and BERT-Score on sewing instructions generated via a range of prompting strategies, demonstrating the versatility and robustness of our approach with respect to varying methods of instruction generation (Section 5).

We show that our metric better correlates with manually-annotated error counts in the model-generated instructions, indicating that our approach better reflects spatiotemporal correctness than competing metrics.

To further highlight the sensitivity of our approach to correctness, we construct an artificial dataset in which the steps of the model-generated instructions are randomly permuted: while our metric is highly sensitive to these nonsensical instructions, the traditional metrics fail to meaningfully reflect any difference between the original and permuted instructions.

Additionally, we find that our metric is weakly positively correlated with subjective human ratings, while all other, traditional similarity measures are negatively correlated.

All prompts, generated instructions, input images, code files, and data required to replicate these experiments are available on GitHub[1].

2 Related Work

Multimodal Reasoning. Liu et al. (2024b) evaluate a range of Multimodal LLMs (MLLMs) across a variety of web tasks, and find that many models do not exceed random-chance performance on action prediction and grounding tasks, speaking to these models' limited reasoning and grounding abilities—both critical skills for sewing instruction generation (as discussed in Section 1). Although GPT-4V (Achiam et al., 2023) and Claude (Anthropic, 2024) outperform their open-source counterparts by a notable margin, GPT-4V—the best-performing model they evaluate—only achieves an average score of 64.6 out of 100, indicating that there is still much room for improvement, even among closed-source MLLMs.

Ji et al. (2022) investigate human and LLM abstract visual reasoning abilities through the use of tangram puzzles (Hawkins et al., 2020): both models that they evaluate—CLIP (Radford et al., 2021) and ViLT (Kim et al., 2021)—demonstrate limited abstract reasoning abilities in comparison to human participants. However, Ji et al. (2022) find that explicit descriptions and color-coded highlights of various tangram components improve performance for both humans and MLLMs, suggesting that similar knowledge-augmentation can be leveraged to improve model-generated sewing instructions.

Textual Spatiotemporal Reasoning and Planning. Although the findings of Lyu et al. (2020) indicate that fine-tuning and careful data-engineering can improve the common-sense reasoning abilities of then-SoTA LMs—i.e. BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), and GPT-2 (Radford et al., 2019)—to near-human performance, Aghzal et al. (2025) suggest that the capabilities of LLMs are limited when it comes to long-term planning and spatial reasoning. Specifically, Aghzal et al. (2025) find that in-context learning (ICL) does not help GPT-4 avoid obstacles—although ICL does improve the model's ability to reach the goal—in textual grid-navigation environments. While chain-of-thought (CoT; Wei et al., 2022) prompting does improve obstacle-avoidance abilities, these abilities still degrade as obstacle number and distance to the goal increase.

[1] https://github.com/coli-saar/generatingsewinginstructions

Of particular concern to the topic at hand are Wu et al.'s (2024) findings that LLM's reasoning performance degrades substantially when they are presented with counterfactual instances: reasoning problems in which well-known rules are altered, such as chess puzzles in which the piece types' valid moves are modified. This indicates that LLMs may struggle to accomplish a task such as sewing-instruction generation—examples of which are unlikely to occur frequently in the models' training data.

Cooking Recipe Generation. Given the similarities between the two tasks, it stands to reason that many findings in the area of cooking-recipe generation—a far more well-researched task—should translate to sewing-instruction generation.

Salvador et al. (2019) propose an approach for generating cooking recipes under which the model first predicts ingredients before generating a recipe. The authors find that their method generates higher quality recipes—and improves in ingredient prediction—over previous baselines, and creates more compellingly written recipes than retrieval methods, according to human judgment. Similarly, (Chandu et al., 2019) improve over baseline cooking-generation approaches via a storyboarding approach that imposes hierarchical structure on the model's step-by-step instruction-generation reasoning process.

On the other hand, Liu et al. (2024a) use Retrieval Augmented Generation (RAG) to address the common problem of hallucination in cooking-recipe generation with text generation.

Evaluation Metrics. Evaluation metrics for LLM-generated text broadly fall into one of three categories (Celikyilmaz et al., 2021): (i) human-centric, which focus on manual evaluation of generated text; (ii) untrained and automatic, which typically compute the similarity between the generated text and a reference text (e.g. Papineni et al., 2002); and (iii) machine-learned, which also compute the similarity between the generated text and a reference text, but via a neural model (e.g. Zhang et al., 2020).

Although—by definition—it most accurately reflects human judgment, manual text evaluation is very time-consuming and expensive, hindering its employment at scale. On the other hand, automatic evaluation metrics rely purely on surface-level similarity with the reference text, which is not always applicable for instruction-generation tasks: for example, task-oriented text generation allows for a large degree of diversity in the generated texts, in which case the usefulness of such similarity metrics is limited.

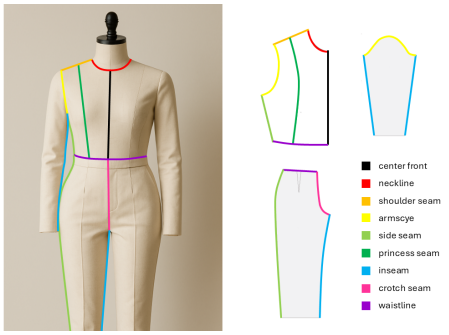

Figure 2: Example of a sewing pattern (right; edges colored in this figure to illustrate attachment location) and a mannequin wearing the completed garment built from the pattern (left). (Left-hand image generated by ChatGPT; front bodice pattern from https://www.moodfabrics.com/blog/; sleeve and front pant leg patterns from https://sewguide.com/princess-seams/)

Furthermore, Ostmeier et al. (2024) note that common textual-similarity evaluation metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) lack the ability to measure factual correctness, which is critical for evaluating a wide variety of tasks.

To address this deficiency in the medical domain, the authors introduce the GREEN metric, a quantitative metric derived from LLM-as-judge evaluation (Ostmeier et al., 2024, use GPT-4). Although they find that GREEN scores are highly correlated with human expert evaluations for radiology reports, the critical weakness of this approach lies in its reliance on an LLM to evaluate factual correctness: in a domain where the model lacks factual knowledge—such as sewing-instruction generation—it is not feasible to rely on an LLM as a critical component of the evaluation metric.

3 Task and Dataset

In this section, we provide a brief overview of sewing patterns in general (Section 3.1), and the particular sewing-pattern dataset that we employ in this work (Section 3.2).

3.1 Sewing Patterns

Sewing patterns are templates used to guide the construction of a garment while sewing. These paper patterns are placed on top of fabric, in order

Instruction: With back pant legs (right sides of fabric facing) sew down the center back seam.

Figure 3: Example of sewing instructions in our dataset.

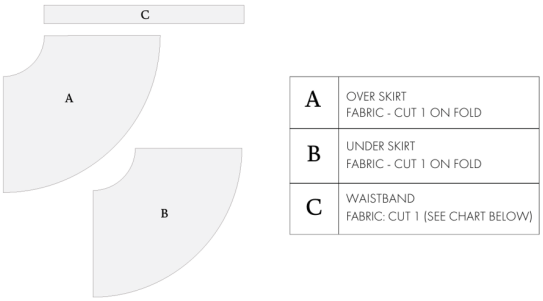

Figure 4: Example of a sewing pattern (left), with corresponding descriptions of each of its component pieces (right).

to cut out the individual pieces: the fabric pieces are then assembled into a garment (see Figure 2).

Since garments are typically symmetrical, many patterns are made to show only the left or the right half—for example, Figure 2 shows pattern pieces for the left half of the bodice, the left sleeve, and the left front pant leg. To create a symmetrical garment, those pieces are either cut multiple times or "on the fold": i.e. by folding the fabric in half and placing the paper pattern piece near the folded edge so that the pattern cutter cuts through two layers of fabric, resulting in a perfectly symmetrical piece when it is unfolded (see Figure 7 in the Appendix). For a bodice piece such as that in Figure 2, it is common to place the center front on the folded edge of the fabric, so that the center front acts as the mirror axis and the resulting piece covers both the left and the right front side of the torso.

3.2 Dataset

The website moodfabrics.com[2] features free sewing patterns and instructions. The instructions for each pattern describe the different steps required to go from the pattern pieces to the finished garment. Each step is illustrated with a picture of the current state of the process (see Figure 3).

To construct our dataset, we sampled 22 patterns spanning five garment types: five shirts, five dresses, five pants, five skirts, and two jumpsuits. Each of these patterns contains an overview picture and a description of each piece (see Figure 4).

The pattern pieces in the overview are labeled from A to Z, allowing one to refer to the individual pieces when writing instructions.

[2] https://www.moodfabrics.com/blog/

In order to obtain symmetrical pieces for the left and right half of the body, pieces on the overview image have to be cut out of fabric twice (as discussed in Section 3.1). In such cases, the pattern piece description gives the instructions to cut a "mirrored pair" of the piece X: the pattern cutter cuts a left (Xl) and a right (Xr) piece out of the fabric. When the pattern piece descriptions mention cutting multiples of a piece without mirroring them, we refer to these pieces as X1, X2, X3, etc.

4 Tree-Based Evaluation Metric

As discussed in Sections 1 and 2—and further demonstrated below in Section 5—traditional evaluation metrics used for text generation are poor measures of spatiotemporal consistency. To remedy this deficiency, we introduce a tree-based evaluation metric that is designed to reflect the order of and constituent pieces involved in assembly steps.

We first annotate each pattern with sets of gold trees (see e.g. Figure 5), each of which encodes a valid assembly order for the garment in question (see Section 4.1). We then employ an automatic tree-extraction procedure over the model-generated instructions, which constructs trees encoding the order in which the pieces of the garment are assembled together in the instructions (see Section 4.2). These automatically-extracted trees are compared to their corresponding patterns' gold trees to yield the tree score (see Section 4.3), a measure of the spatiotemporal consistency of the generated instructions.

4.1 Gold Instruction Annotation

As discussed above, our tree-based evaluation metric requires the manual annotation of gold trees for each pattern in the dataset. As each garment

Garment Type	Annotated Patterns	Mean Trees per Pattern
Shirt/Top	5	5,017.2
Dress	5	417.2
Pants	5	1,685.2
Skirt	5	3.6
Jumpsuit	2	991.0
Total	**22**	**1,709.0**

Table 1: Overview of the annotated patterns in our dataset.

permits multiple possible assembly orders to arrive at the same end product (see Figure 9 in the Appendix), each pattern is annotated with a set of gold trees, each of which corresponds to a valid order of assembly for the garment.

There is a large number of gold trees required for each pattern (see Table 1): in practice, we annotate each pattern with a set of Context-Free Grammar (CFG) rules—which then generate the set of gold trees for the pattern—in order to semi-automate the annotation process.

4.1.1 Gold Tree Structure

For each valid assembly order, we construct a corresponding tree t (see Figure 5). Each constituent piece in the garment is represented by a leaf node in t, and each intermediate component is represented by a node higher in the tree. For each assembly step that joins two pieces α and β, there is a node $\alpha\beta$ and edges $\alpha\beta \rightarrow \alpha$, $\alpha\beta \rightarrow \beta$ in t: the completed garment is represented by the root node S.

Each node in each gold tree is only permitted at most two daughter nodes. This restriction to unary- (see Section 4.1.2) and binary-branching trees is imposed to facilitate the predicted tree extraction procedure (see Section 4.2): we impose this constraint on the gold trees to permit one-to-one comparison with the predicted trees.

We further require that the constituent piece names within the node names be in alphabetical order with respect to the root node labels, regardless of the daughter node to which each root node belongs: for example, A $\leftarrow$ AB $\rightarrow$ B and AC $\leftarrow$ ABCD $\rightarrow$ BD are valid subtrees, while A $\leftarrow$ BA $\rightarrow$ B and AC $\leftarrow$ ACBD $\rightarrow$ BD are not.

Lowercase labels "l" and "r" indicating directionality (see Section 3) stay with the label that they modify, and only serve as tie-breakers to decide the ordering between two labels with the same capital letter piece label. For example, ABl $\leftarrow$ ABlBr $\rightarrow$ Br is a valid subtree, while ABl $\leftarrow$ ABBlr $\rightarrow$ Br is not.

These constraints are again intended to permit one-to-one comparison with the predicted trees, in which parent node labels are automatically derived from their daughter nodes (see Section 4.2).

4.1.2 Self-Attachment

Aside from attaching two pieces to each other, sewing instructions often require attaching a piece to *itself*—for example, a sleeve can be assembled by rolling up a rectangular piece of fabric, and sewing the two connected edges together—including repeated self-attachment in more complex garments.

To account for this phenomenon, we append a subscriped, integer-valued self-attachment counter n to the label of each node α in each assembly tree t: α_n. In cases where the assembly requires the attachment of α_n to itself, we include a node α_{n+1} and an edge $\alpha_{n+1} \rightarrow \alpha_n$ in t. Note that for the sake of representational simplicity, we often omit the self-attachment counter when $n = 0$ in this work.

When two pieces are attached to one another, the parent node inherits the larger self-attachment number of its daughter nodes, resulting in subtrees of the form: $\alpha_n \leftarrow \alpha\beta_{max(n,m)} \rightarrow \beta_m$. As a consequence, each parent node label contains only one self-attachment counter, appended to the far-right edge of the node label: for example, $AB_2 \leftarrow ABCD_3 \rightarrow CD_3$ and $A_1 \leftarrow ABC_1 \rightarrow BC_0$ are valid subtrees, while $AB_2 \leftarrow AB_2CD_3 \rightarrow CD_3$ and $A_1 \leftarrow A_1BC_0 \rightarrow BC_0$ are not.

We again impose this constraint to facilitate comparison with the predicted-instruction-derived trees (see Section 4.2). Although this self-attachment-inheritance procedure does result in parent node labels that forget information regarding the self-attachment levels of their daughter nodes, this does not impair evaluation, as this information is still encoded within the respective subtrees that they dominate.

4.2 Prediction Postprocessing

For each pattern P in the dataset, the model in question is prompted to generate corresponding instructions I: in order to facilitate the tree construction procedure described in this section, we prompt the model to combine no more than two pieces in one step in I. We then employ a Piece-Extraction LLM, which is prompted to extract the piece labels mentioned in each step of I.

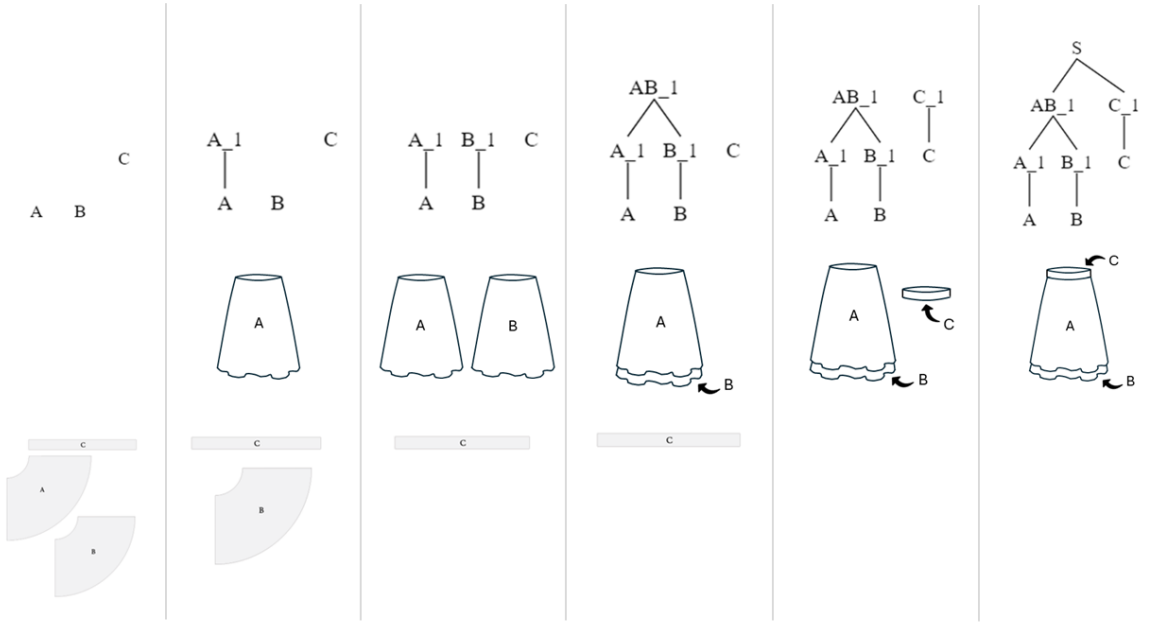

Figure 5: Step-by-step illustration of the assembly of a skirt (middle), and the corresponding tree (top). The bottom row indicates the remaining pattern pieces at each step.

Note that the instruction-generating model often does not mention intermediate pieces in the instructions, but rather single components of those pieces: for example "attach the pocket (A) to the front piece of the skirt (B)", when the front piece has already been attached to the rest of the skirt in prior steps. Naively extracting the pieces mentioned in such a step would result in a subtree of the form A $\leftarrow$ AB $\rightarrow$ B, when the desired outcome is of the form A $\leftarrow$ Aβ $\rightarrow$ β, where β denotes the skirt to which B is attached.

To account for this problem, we employ a rule-based procedure that replaces each mentioned piece label B with the label β of the larger, intermediate piece to which B is attached (if applicable).

From the pieces extracted from each step s_k of I, we derive a subtree t_k of the form $t_k = \alpha_n \leftarrow \alpha\beta_{max(n,m)} \rightarrow \beta_m$ if two pieces α_n, β_m were extracted from s_k, or $t_k = \alpha_{n+1} \rightarrow \alpha_n$ if one piece α_n is extracted. These extracted subtrees are constructed to conform to the assembly-tree constraints described in Section 4.1.

Finally, we derive a tree[3] $T(I)$ from the disjoint union of the extracted subtrees t_k by gluing together nodes with the same label.

This pipeline is illustrated in Table 2.

[3] $T(I)$ may not necessarily be a tree, but rather a forest: for example, when the model forgets to attach a piece to the final garment.

4.3 Tree Score

Let P be some pattern in the dataset, with a set of associated gold trees $G(P)$ (see Section 4.1). For a given model-generated instruction I for P, we compute its *tree score* with respect to P ($S_P(I)$) as the maximum F_1 score between $T(I)$ and each of the gold trees $g \in G(P)$ (Equation 1). As each $g \in G(P)$ is represents a valid assembly order for P, we return the maximum F_1 score in order to account for the multitude of possible valid assembly orders for a given pattern.

$$S_P(I) = \max_{g \in G(P)} F_1(T(I), g) \quad (1)$$

We compute the F_1 score with respect to subtrees: for each non-leaf node $\alpha \in T(I)$ and $\beta \in g$, we extract the subtrees $T(I)_\alpha$, g_β spanning α/β and their immediate respective daughter nodes. We then compute the exact-match F_1 score between $\{T(I)_\alpha\}_{\alpha \in parents(T(I))}$ and $\{g_\beta\}_{\beta \in parents(g)}$ to yield $F_1(T(I), g)$.

5 Experiments

We evaluated our proposed tree-based evaluation metric against three existing metrics—BLEU, ROUGE-L, and BERT-Score—on a set of sewing instructions generated from a variety of prompting strategies (Section 5.1). Specifically, we compared

Instructions	Extracted Pieces	Extracted Subtree
1. Align the Over Skirt (A) and Under Skirt (B) at the side seams. Sew the left side seam of the Over Skirt (A) to the left side seam of the Under Skirt (B).	1: [A, B]	$AB \rightarrow A \quad B$
2. Sew the right side seam of the Over Skirt (A) to the right side seam of the Under Skirt (B).	2: [A, B]	$AB_1 \rightarrow AB$
3. Align the Waistband (C) with the top edge of the joined Over Skirt (A) and Under Skirt (B). Sew the Waistband (C) to the top edge, ensuring the seams are evenly distributed.	3: [C, A, B]	$S \rightarrow AB_1 \quad C$
4. Fold the Waistband (C) over to the inside of the skirt, enclosing the raw edge. Stitch in place to secure.	4: []	
5. Hem the bottom edge of the Over Skirt (A) and Under Skirt (B) to the desired length.	5: []	

Table 2: Subtree extraction, side-by-side comparison.

our metric to the existing approaches with respect to correlation with error count (Section 5.2) and sensitivity to random permutations (Section 5.3).

5.1 Data Generation

We employed GPT-4V (model = gpt4o, seed = 1, temperature = 0.0, max_tokens = 3000) to generate step-by-step instructions given an image overview and a description of each of the 22 patterns in our dataset (see e.g. Figure 4), and as the Piece Extraction model for our evaluation metric (see Section 4.2). To assess the robustness of our evaluation metric, we evaluated four prompting strategies:

- **Baseline:** A zero-shot prompted model. The model is instructed to explicitly mention all connecting seams—including those that occur twice on mirrored pieces (e.g. sleeves)—in order to accurately reflect the assembly process. As required by our evaluation metric, the model is instructed to include at most one connecting seam in each step. For the same reason, we require that the instructions explicitly mention the labels of each piece that is included in a given step.

- **Few-Shot:** A few-shot prompted model that is presented with three example instructions.

- **Generalized Instructions:** A model that is few-shot prompted with archetypical examples of instructions of each of the five garment types (without images), to serve as a guide for instruction generation.

- **Intermediate Representations:** A model that is prompted after each generated step to generate an intermediate representation of the current state of the garment assembly process: the used pieces, the current intermediate garment(s), and the unused pieces in the assembly process.

Example prompts for these strategies are located in Figures 12-14 in the Appendix.

Instruction Generation. We evaluated all possible combinations of these prompting strategies: few-shot prompting, few-shot prompting with generalized instructions, etc. Including the baseline approach, this results in eight possible combinations across the 22 patterns in our dataset, yielding 176 generated sets of instructions.

To enable an accurate, one-to-one comparison to BLEU, ROUGE-L, and BERT-Score, we handcrafted gold instructions for each of the patterns in our dataset, that align with the style and constraints imposed by the tree-based evaluation metric: i.e. always mentioning piece labels, connecting at most one seam per step, etc.

5.2 Error Reflection

We first compare our metric to existing approaches with respect to their capacity to reflect errors of spatiotemporal reasoning in generated instructions.

5.2.1 Experimental Setup

For each of the 176 model-generated instructions, we computed BLEU, ROUGE-L, and BERT-Score with respect to the gold text, and computed our tree-score between the extracted tree and the annotated gold trees (see Section 4).

We manually annotated the model-generated instructions for errors, counting errors across four

Metric	Corr.	p
BLEU	-0.111	.1432
ROUGE-L	-0.179	< .05
BERT-Score	-0.080	.2915
Tree Score (ours)	**-0.599**	**< .001**

Table 3: Pearson's correlation of evaluation metrics with the number of errors per number of steps (df = 174).

Metric	Baseline	Shuffled	Δ
BLEU	0.127	0.126	-0.001
ROUGE-L	0.377	0.322	-0.055
BERT-Score	0.881	0.878	-0.003
Tree Score (ours)	0.512	0.272	**-0.240**

Table 4: Mean evaluation scores across the baseline and randomly-permuted instructions.

categories: (i) incorrect assembly operations, (ii) missing assembly operations, (iii) incorrect order of assembly operations, and (iv) conflicting information/incorrect use of terminology. We then computed the correlation between error count and score for each of the four evaluation metrics.

5.2.2 Results

There is a significant, moderate negative correlation between tree score and number of errors (see Table 3)—substantially lower than that observed for BLEU, ROUGE-L, and BERT-Score. In fact, BLEU and BERT-Score are not significantly correlated with error rate at all: these metrics are entirely unable to detect such errors.

These results clearly demonstrate our evaluation metric's superior ability to reflect errors of spatiotemporal reasoning.

5.3 Robustness to Permutation

Next, we evaluated the robustness of the four evaluation metrics with respect to artificially-constructed counterfactual examples that are designed to disrupt spatiotemporal correctness.

5.3.1 Experimental Setup

In this experiment, we randomly permuted the steps of each of the 22 instructions generated by the baseline prompting approach (see Section 5.1): this constructs a set of instructions that is stylistically and lexically similar to the gold instructions, but entirely unexecutable due to the nonsensical ordering of the instruction steps.

We then computed the BLEU, ROUGE-L, BERT-Score, and tree score for each of the permuted instructions, and compared these scores to each of the metrics respective scores for the original, unpermuted baseline-approach instructions.

5.3.2 Results

The results of this experiment (Table 4) clearly demonstrate that our metric is more sensitive to errors of spatiotemporal reasoning than existing evaluation metrics. The average tree score drops by almost 50% from the baseline to the permuted examples, reflecting the decrease in correctness of the artificially-constructed instructions.

In comparison, ROUGE-L score decreases by only 15%, while BLEU and BERT-Score fail to decrease by any meaningful amount: these metrics entirely fail to capture any difference in terms of correctness between the actual model-generated instructions and the randomly-permuted instructions.

5.4 Alignment with Human Judgment

To evaluate the degree to which our tree score metric aligns with human judgment, we recruited 20 participants with experience in sewing garments to rate the model-generated instructions of Section 5.1. All human evaluations were conducted on the LingoTurk platform (Pusse et al., 2016).

5.4.1 Experimental Setup

Data Each participant was asked to evaluate 4 sets of sewing instructions since a brief pilot study showed that 4 sets can be completed in about 30 minutes. We collected ratings 110 sets of instructions, a subset of the dataset that we used in previous experiments (spanning all 22 patterns).

Task We provided the participants with the overview image/description of the pattern pieces—the same context available to the generation model—and the model-generated instructions.

We employed a step-by-step evaluation: participants were presented with one step of the instructions at a time, and gave individual ratings for each step. This method reduces the mental load on the participants and results in more detailed ratings, enabling a more fine-grained analysis.

At each step, the participants answered the following questions with a rating on a 5-point Likert Scale:

S1. Does this step make sense with regard to the picture?

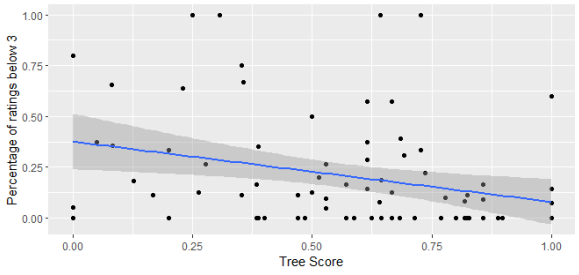

Figure 6: Proportion of below-3-ratings for S3, plotted against tree score.

S2. Does this step make sense with regard to the pattern piece descriptions?

S3. Does this step make sense with regard to the previous step?

S4. Is this step clear enough to follow?

S5. Is the terminology in this step used correctly (e.g. seam names)?

Once the ratings were provided, the next step was displayed on the screen—the previous steps always remained visible. An example of the participant interface is given in Figure 8 in the Appendix.

After rating each step, the participants were asked to give two ratings (also on a 5-point Likert Scale) of the instructions as a whole:

I1. Is it possible to assemble the pieces following the given instructions?

I2. This pattern was designed to be a <garment type>. Do you think the assembly steps lead to the correct outcome?

Data Aggregation For each step-level question, we aggregated the step-by-step ratings into three instruction-level summarized ratings ($5 \times 3 = 15$ summarized ratings per instruction): the mean rating across steps, the proportion of ratings above 3, and the proportion of ratings below 3.

5.4.2 Results

Mean step-level ratings from each rating category are weakly, positively correlated with tree score (Table 5), although only the correlation with S3 is significant ($r(78) = 0.23$, $p < .05$). In addition, there is a significant, weak negative correlation between tree score and the proportion below 3 for S3 ($r(78) = -0.29$, $p < .01$; see Figure 6).

For the traditional evaluation metrics, we observe a negatively correlated relationship with human judgment (see Table 3) suggesting that these measures do not align with the perceived quality of the instructions.

	S1	S2	S3	S4	S5	I1	I2
Tree (ours)	0.14	0.12	**0.23**	0.15	0.12	0.05	0.16
BLEU	**-0.26**	**-0.29**	-0.18	-0.19	-0.22	-0.18	-0.07
ROUGE-L	-0.04	-0.12	-0.02	-0.02	-0.12	-0.11	0.03
BERT	-0.12	-0.20	-0.10	-0.08	-0.13	-0.14	0.00

Table 5: Pearson's correlation between evaluation scores, and mean step-level (left; S1-5) and instruction-level (right; I1-2) human ratings. Significant values ($p < .05$) are indicated in bold.

Under the assumption that human judgment is most likely to consider spatiotemporal consistency at the step (rather than instruction) level, the correlation of S3 with tree score indicates that our metric reflects human assumptions regarding spatiotemporal consistencies to a higher degree than traditional similarity scores.

6 Conclusion

We introduced a novel, tree-based evaluation metric that is designed to more accurately reflect spatiotemporal correctness in generated instructions (see Section 4). This metric affords the flexibility required to capture multiple orders of assembly, while reflecting the spatial and temporal aspects of reasoning required for instruction generation.

In the domain of sewing instructions, we showed that our metric better (negatively) correlates with error counts than existing evaluation metrics (see Section 5), demonstrating this metrics superior ability to reflect *correctness*—rather than simple textual similarity. Further experiments show that our metric is far more robust to artificially-constructed instructions specifically designed to confound metrics that rely purely on similarity.

In addition, the results of a human evaluation study indicate that human perception of coherence in step-by-step sewing instructions is positively correlated with our proposed evaluation metric, but not with traditional similarity measures.

These results indicate that our tree score is a more meaningful metric for tasks such as sewing-instruction generation, in which step-by-step correctness and consistency is far more important than stylistic and lexical resemblance.

Limitations

Reflection of Attachment Method. In order to represent the exactly how the pattern pieces have to be connected to form the finished garment, it

is important to know which edges of which pieces connect to which other edges on which other pieces. The approach that is introduced in this work only represents which pieces are attached to which other pieces and when.

However, the same applies to the textual sewing instructions: the gold instructions as well as the generated instructions assume that the reader has background knowledge in sewing and do not explicitly mention the edges of the pieces in many cases. This makes it impossible to reliably extract information about the edges of the pieces as this information might not be present in the textual instructions to begin with.

Dependence of the Evaluation Metric on Input Format. Further, the tree-based evaluation only works as intended when the generated instructions follow specific constraints that allow the tree extraction algorithm to function. This naturally limits the number of scenarios where our tree-based evaluation approach is practical.

Garment Complexity. In addition, our tree-based evaluation is limited to the evaluation of simple garments. In order to be able to evaluate more complex garments, the tree-based evaluation framework must be expanded (for example to accommodate patterns that include multiple left and right copies of the same piece). We leave the investigation of the extension of our metric to accommodate more complex sewing patterns—as well as other instruction generation tasks—to future work.

Self-Attachment. Another limitation concerning the tree-based evaluation is that the property of an intermediate product is always represented by the same node, regardless of how it was assembled: this is true for most cases (most shirts, skirts, dresses) but not all (see Appendix A).

Node Naming. Errors propagating up through the tree builds a further limitation, as the mistakes that are made earlier in the assembly process affect the tree more heavily. This is due to the fact that these trees are constructed bottom-up, and the names of the later nodes are defined by the previous assembly steps.

Although it makes sense to punish early mistakes more heavily—because later steps cannot be followed when the earlier steps produced an incorrect intermediate garment—the current tree-based evaluation metric assigns a similar score to instructions with frequent early mistakes, regardless of whether the following steps are correct.

Domain We show that our proposed evaluation approach outperforms traditional evaluation metrics in the presented setting, which however is restricted to the very specific task of sewing instruction generation. While we believe this CFG-based tree evaluation framework to be a generalizable approach that could be translated to other tasks where textual instructions need to be evaluated with respect to real world 3D applications, such as furniture building, puzzle games or Minecraft, the ability to generalize our approach to these domains remains to be explored in future work.

Conversely, as our CFG-based evaluation metric delivers a structured representation of a physical assembly strategy, future work could investigate how training on these representation might improve the performance of language models in tasks requiring complex spatiotemporal understanding.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2025. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *Preprint*, arXiv:2310.03249.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1:1.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2021. Evaluation of text generation: A survey. *Preprint*, arXiv:2006.14799.

Khyathi Chandu, Eric Nyberg, and Alan W Black. 2019. Storyboarding of recipes: Grounded contextual generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6040–6046, Florence, Italy. Association for Computational Linguistics.

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

Robert D. Hawkins, Michael C. Frank, and Noah D. Goodman. 2020. Characterizing the dynamics of learning in repeated reference games. *Cognitive Science*, 44(6):e12845.

Anya Ji, Noriyuki Kojima, Noah Rush, Alane Suhr, Wai Keen Vong, Robert D. Hawkins, and Yoav Artzi. 2022. Abstract visual reasoning with tangram shapes. *Preprint*, arXiv:2211.16492.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. *Preprint*, arXiv:2102.03334.

Wenyan Li, Jiaang Li, Rita Ramos, Raphael Tang, and Desmond Elliott. 2024. Understanding retrieval robustness for retrieval-augmented image captioning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9285–9299, Bangkok, Thailand. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Guoshan Liu, Hailong Yin, Bin Zhu, Jingjing Chen, Chong-Wah Ngo, and Yu-Gang Jiang. 2024a. Retrieval augmented recipe generation. *Preprint*, arXiv:2411.08715.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024b. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *Preprint*, arXiv:2404.05955.

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

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with wikihow. *CoRR*, abs/2009.07690.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in Minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415, Florence, Italy. Association for Computational Linguistics.

Sophie Ostmeier, Justin Xu, Zhihong Chen, Maya Varma, Louis Blankemeier, Christian Bluethgen, Arne Edward Michalson Md, Michael Moseley, Curtis Langlotz, Akshay S Chaudhari, and Jean-Benoit Delbrouck. 2024. GREEN: Generative radiology report evaluation and error notation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 374–390, Miami, Florida, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Florian Pusse, Asad Sayeed, and Vera Demberg. 2016. LingoTurk: managing crowdsourced tasks for psycholinguistics. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 57–61, San Diego, California. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Amaia Salvador, Michal Drozdzal, Xavier Giro-i Nieto, and Adriana Romero. 2019. Inverse cooking: Recipe generation from food images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, Mexico City, Mexico. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

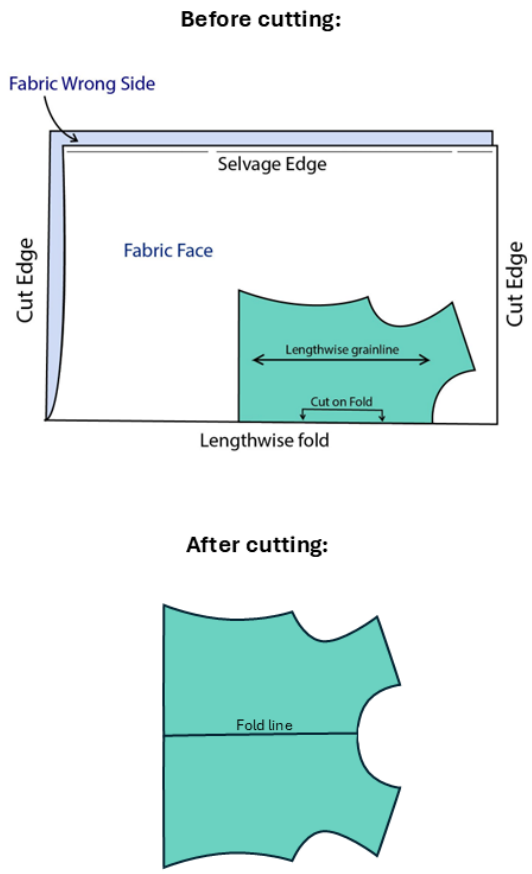

Figure 7: Illustration of cutting a pattern piece "on the fold" (source: https://i.pinimg.com/originals/61/d9/2f/61d92fede2df0c2a159caac9d96bbafb.jpg).

A Appendix

A.1 Cutting on the fold

Figure 7 illustrates the principle of cutting pattern pieces on the fold of the fabric. The paper pattern only shows half of the piece and is placed on the folded edge of the fabric before cutting. Like this, two layers of fabric are cut at the same time to reveal the full piece once the fabric is unfolded again. This is commonly done to easily obtain a symmetrical piece (here a front bodice).

A.2 Human Evaluation Experiment

Figure 8 shows the experiment interface with the overview of the pattern pieces in the top left corner, the description of the pattern pieces in the top right corner, the step-by-step instructions in the bottom right corner and the evaluation questions in the bottom left corner.

A.3 Self-Attachment Limitation

As an example, Figure 10 shows two strategies for assembling a pair of pants. A pair of pants usually consists of 4 pieces, two front pieces (here Fl and Fr) and two back pieces (here Bl and Br). The picture shows the right front piece Fr. The right back piece Br looks the same, while the left pieces Fl and Bl are mirrored versions of Fr.

With those 4 pieces, we have the opportunity to either attach the left and right pieces to each other at the crotch seam forming one front piece (FlFr) and one back piece (BlBr) (see (1a)) or we attach the two left pieces to each other (BlFl) and the two right pieces (BrFr).

The first approach ($(1a) \rightarrow (2a)$) needs 3 more seams: the left side seam, the right side seam and the inseam (reaching from the bottom of one leg all the way to the bottom of the other leg; see Figure 10). These 3 seams can be executed in any order, leading to the following rules, where the first seam attaches the front and back pieces to each other while the second and third seams are self-attachment steps:

$$\text{BlBrFlFr} \rightarrow \text{BlBr} \quad \text{FlFr} \tag{2a}$$
$$\text{BlBrFlFr}_1 \rightarrow \text{BlBrFlFr} \tag{2b}$$
$$\text{BlBrFlFr}_2 \rightarrow \text{BlBrFlFr}_1 \tag{2c}$$

The second approach forms the left and right pieces into two tubes which are attached along the crotch seam in the next step to form the final pair of pants (see Figure 10). Adhering to the established constraints, this results in the following rules:

$$\text{BlFl}_1 \rightarrow \text{BlFl} \tag{3a}$$
$$\text{BrFr}_1 \rightarrow \text{BrFr} \tag{3b}$$
$$\text{BlBrFlFr}_1 \rightarrow \text{BlFl}_1 \quad \text{BrFr}_1 \tag{3c}$$

We can see that even though the number of seams and finished product are the same when comparing the two assembly strategies, the representation of the finished product is different.

Changing the constraints to $parentnode\_x \rightarrow child1\_ychild2\_z$ where x = y+z, conversely, would break the tree logic for the shirt example in Figure 9.

A.4 Instruction Generation Prompts

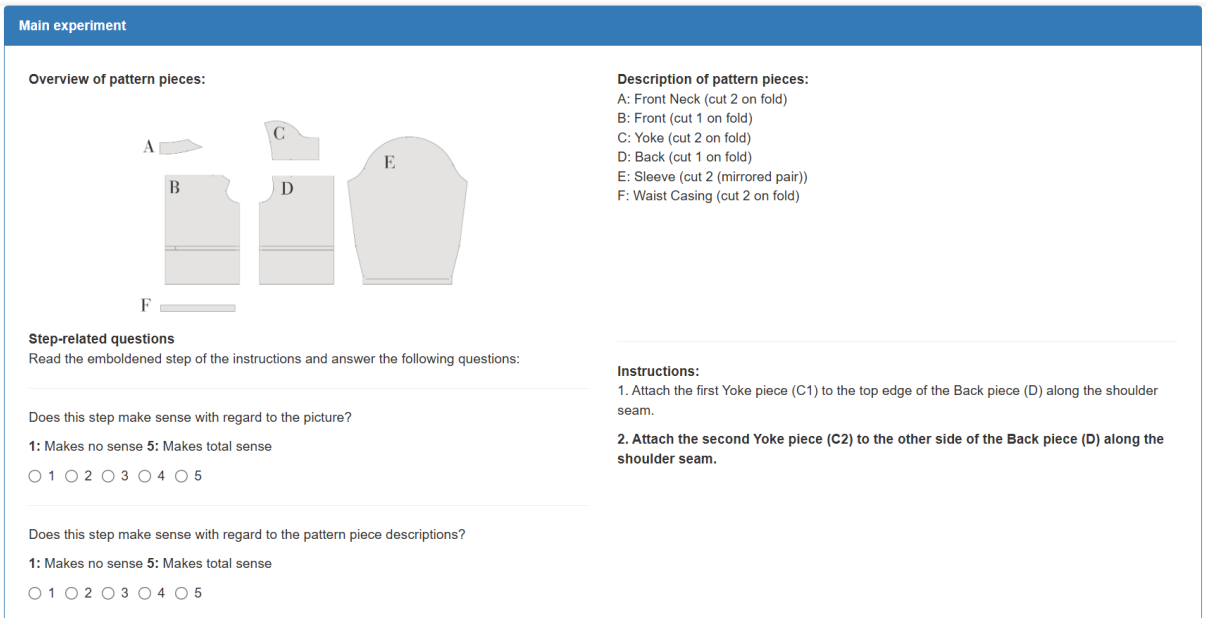

Figure 8: Example of the human-evaluation experiment participant interface.

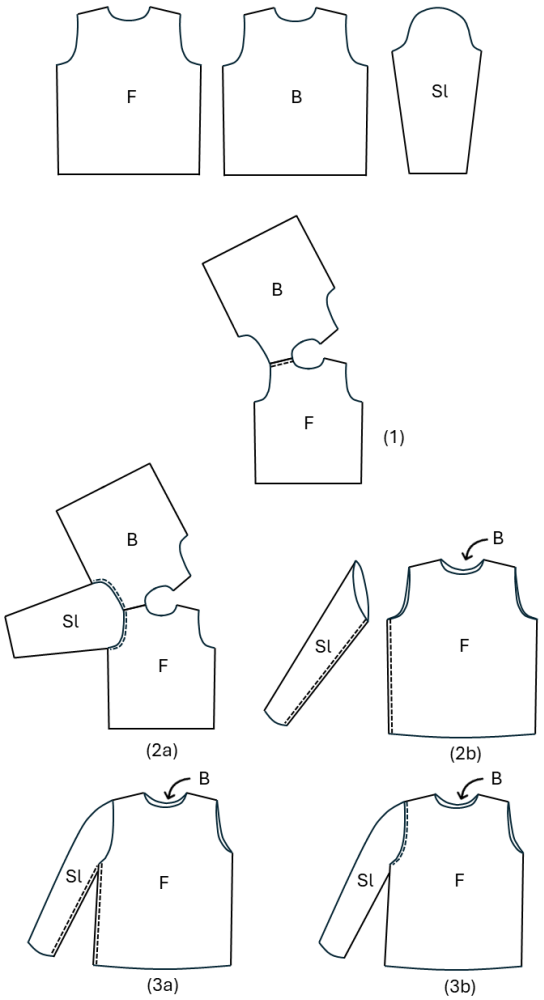

Figure 9: Two ways of assembling a shirt (a and b): the pieces F (front), B (back), and Sl (left sleeve) can be sewn together following either sequence of steps: $(1) \rightarrow (2a) \rightarrow (3a)$ or $(1) \rightarrow (2b) \rightarrow (3b)$.

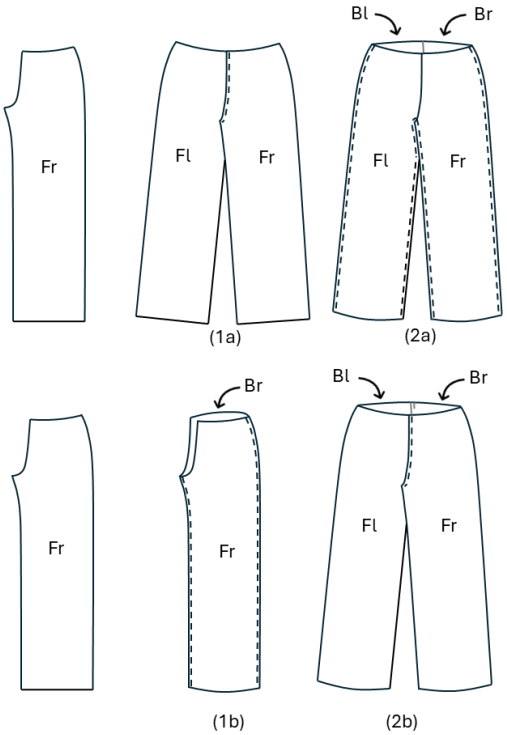

Figure 10: Two ways of assembling a pair of pants (a and b), Fl: left front piece, Fr: right front piece, Bl: left back piece, Br: right back piece, dotted lines indicate where seams are sewn

Extract the connecting seams from the sewing instructions below. Use the format [“X”, “Y”] where X and Y are two pattern piece labels that are connected to form a new piece. The output should be in json format, where the number of the step is the key and the value is a list of the piece labels of the pieces that are being connected.
Here is an example:
{1: [“Al”, “Bl”], 2: [“Cl”, “Cr”], 3: [“A1”, “A2”, “B1”], 4: [“El”, “Dl”, “Fl”], 5: [“Al”, “Bl”, “El”, “Dl”, “Fl”]}

A zipper should be treated like a regular connecting seam.

A seam that connects a piece to itself (for example a sleeve inseam) counts as a connecting seam. In this case, the list in the output contains only one label.
Example:
{1: [“Fl”], 2: [“Fr”]}

Note that not every step in the sewing instructions has to describe a connecting seam. Here is a list of steps that do not count as connecting seams:

1. Sewing hems
2. Finishing seams
3. Pressing
4. Sewing darts
5. Gathering fabric

If the step describes one of these actions, leave the list in the output empty.

Figure 11: Piece Extraction Prompt

Give me step by step instructions on how to sew the sewing pattern in the picture. Make sure to be as precise as possible. Put special emphasis on the assembly process meaning the seams that are needed to connect the pieces.
Make sure to include all the pattern pieces from the picture in your instructions using the following description of the pieces:

A: Collar (cut 2 on fold)
B: Front (cut 2 (mirrored pair))
C: Sleeve (cut 2 (mirrored pair))
D: Yoke (cut 2 on fold)
E: Back (cut 1 on fold)
F: Pocket (cut 2)

Each step in the instructions should explain one connecting seam.
For each step, make sure to include the piece labels from the description above while following the rules below.

Rule 1:
Naming convention for the pieces:
If there are two mirrored versions of one piece, refer to them as Xl (for the left piece) and Xr (for the right piece), substituting X with the respective piece label. If there are two or more versions of one piece and it's not indicated that they are mirrored, refer to them as X1, X2, X3 ..., substituting X with the respective piece label.

Rule 2:
Each step in the instructions can only describe a maximum of one connecting seam.
This also applies for seams that generally come in pairs, for example shoulder seams, side seams, sleeve seams etc. Whenever there are more than two pieces that need to be connected, the process should be split into multiple steps so that each seam is described in its own step. For example, if Al is connected to Bl in step 1, Ar and Br cannot also be connected in step 1, they have to be connected in the next step.

Rule 3:
Add the corresponding piece labels to each step.
Examples:
Incorrect: Attach the left sleeve to the left armhole of the bodice.
Correct: Attach the left sleeve (Il) to the left armhole of the bodice (Al and Bl).

Here is an example of what the instructions should look like:

Sewing Instructions

Upper Bodice Assembly

1. Taking the Front Center Upper panel (E), add in your running stitch between the notches using the longest stitch setting on your machine. Pull the stitching to gather between the notches.
2. Join the left Back Side Upper (Bl) and left Back Center Upper panel (Al) at the princess seam, being sure to clip after sewing any clipped seam allowance before pressing them open.
3. Repeat the same for the right Back Side Upper piece (Br) and the right Back Center Upper piece (Ar).

...

Figure 12: Baseline Prompt

Use the following resources on how to sew different garment types and adapt them to the pattern that's shown in the picture.

How to sew pants:

1. Sew the left front and right front pant together at the crotch seam.
2. Sew the left back and right back pant together at the crotch seam.
3. Attach back pockets to the back pants.
4. Attach front pockets to the front pants.
5. Attach the front pant to the back pant along the inseam.
6. Sew the left and right side seam of the pants.
7. Attach all waistband pieces until they form one circular waistband.
8. Attach the waistband to the pants.

How to sew a shirt:

1. Attach the yoke piece(s) to lower back piece to form the full back piece.
2. Attach the front piece to the back piece at the left and right shoulder seam.
3. Attach the front and back piece at the left and right side seam.
4. Sew the inseam of the left sleeve and the right sleeve.
5. Attach the cuff pieces to each other forming a left and a right cuff.
6. Attach the cuffs to the ends of the left and right sleeve.
7. Attach the left and right sleeve to the bodice at the armholes.
8. Sew the collar pieces together to form the full collar.
9. Attach the collar to the neckline of the shirt.
10. Sew the pocket(s) to the front of the shirt.

...

Figure 13: Generalized Instructions Prompt

... Now give step-by-step assembly instructions using all the pattern pieces from the list of pattern pieces. After each step, give a list of the used / connected pieces (taking into account all previous steps) and a list of the unused pieces up to this point in the process using this format:

-> used / connected pieces: (Dl, Dr), (El, Er)
-> unused pieces: A1, A2, Bl, Br, C

Here is an example of what the instructions should look like:

Sewing Instructions

Upper Bodice Assembly

1. Taking the Front Center Upper panel (E), add in your running stitch between the notches using the longest stitch setting on your machine. Pull the stitching to gather between the notches.
   -> used / connected pieces: (E)
   -> unused pieces: Al, Ar, Bl, Br, Cl, Cr, Dl, Dr, F, Gl, Gr, Hl, Hr, Il, Ir, Jl, Jr, K
2. Join the left Back Side Upper (Bl) and left Back Center Upper panel (Al) at the princess seam, being sure to clip after sewing any clipped seam allowance before pressing them open.
   -> used / connected pieces: (E), (Al, Bl)
   -> unused pieces: Ar, Br, Cl, Cr, Dl, Dr, F, Gl, Gr, Hl, Hr, Il, Ir, Jl, Jr, K
3. Repeat the same for the right Back Side Upper piece (Br) and the right Back Center Upper piece (Ar).
   -> used / connected pieces: (E), (Al, Bl), (Ar, Br)
   -> unused pieces: Cl, Cr, Dl, Dr, F, Gl, Gr, Hl, Hr, Il, Ir, Jl, Jr, K

...

Figure 14: Intermediate Representation Prompt