Missing Links: LLM-Augmentation of Event Triggers of State Changes in the OpenPI Dataset

Kyeongmin Rim, James Pustejovsky
Brandeis University
Department of Computer Science
{krim,jamesp}@brandeis.edu

Abstract

Effective computational understanding of procedural text requires modeling not just the state changes that occur (entity transformations), but also the specific actions that cause them (event triggers). A lack of datasets that explicitly link these two primary information sources has hindered progress in theory-oriented research and applications of NLP. This paper presents two primary contributions: (i) a new silver-standard dataset where event trigger annotations are added to existing state-change data on task-oriented procedural text, enabling both theoretical investigation and practical benchmarking; and (ii) inverse annotation, a framework for recovering missing linguistic annotations from existing semantic annotations—which we apply to recover event triggers from OpenPI's state-change outcomes. We provide detailed pipeline analysis including error modes and quality filtering, and validate the dataset through comprehensive baseline evaluation of diverse trigger detection systems. Our work delivers both a reusable methodological framework applicable to other annotation recovery tasks and a new benchmark resource for modeling the relationship between linguistic actions and their semantic outcomes in procedural domains.

Keywords: event semantics, event trigger, inverse annotation, state tracking, procedural text, data augmentation, dataset creation

1. Introduction

Procedural text—including instructions, recipes, protocols, and manuals—constitutes a primary mode of human knowledge transfer. Understanding these texts computationally requires identifying not just *what* entities are mentioned, but crucially *what happens* to them: the events that transform ingredients into meals, reagents into compounds, or components into functioning systems. Event trigger detection, the task of identifying the words or phrases that express these actions, is fundamental to building computational models of procedural knowledge.

To build such models, research must be grounded in clear theoretical frameworks of event structure. Recent work has moved toward explicit, compositional models. Yamakata et al. (2020) model procedural sequences via Recipe Flow Graphs, while Tandon et al. (2020) introduced OpenPI, modeling event effects as state-change tuples (entity, attribute, before-state, after-state). However, these state-centric models lack explicit annotation of the event triggers that cause state changes. Rim et al. (2023) addressed this gap with Process-Oriented Event Model (POEM), which explicitly represents both actions (event triggers) and their effects (state transformations) via Coreference under Transformation (CuT).

Despite these theoretical advances, most large-scale procedural datasets follow an implicit event model, prioritizing annotation scalability by documenting action *outcomes* rather than the actions themselves. Datasets like OpenPI and NHK-recipe-data (Toyooka et al., 2025) lack formal annotation of event triggers. While useful for benchmarking state tracking, these implicit models cannot support research into compositional event semantics, verb grounding in physical transformations, or learning of causal event schemas.

This reveals a gap: theoretical models like POEM provide the framework for joint event-state modeling, but no large-scale dataset instantiates it with "dense-labeled" annotations of both triggers and

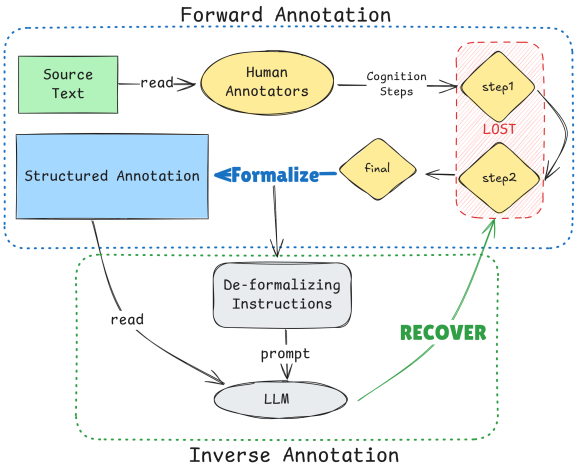

Figure 1: Inverse annotation reverses the traditional annotation direction. While forward annotation (blue) proceeds from text to structured outcomes while discarding intermediate cognitive artifacts, inverse annotation (green) uses LLM to recover the discarded linguistic layer from existing semantic annotations.

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 11992–12006
11–16 May 2026.

state changes. Ideally, human experts would add event trigger annotations to existing state-change datasets like OpenPI, but this task is prohibitively expensive extensively.

To address this challenge, we propose inverse annotation, a framework that recovers missing linguistic annotations from existing annotations (Section 3). We apply it to OpenPI, creating a silver-standard dataset that bridges theoretical models of explicit event structure and empirical data.

1.1. Our Contributions

First, we introduce OpenPI-A, a **silver-standard dataset** of procedural text spanning multiple topics, where event trigger annotations are added to steps that already have state-change annotations. This enables (i) theoretical investigation of event-state relationships and validation of models like POEM and CuT, and (ii) benchmarking of NLP systems. We establish baselines with diverse approaches from statistical to neural methods.

Second, we introduce **inverse annotation** and apply it to OpenPI. We provide detailed pipeline analysis, including error modes, quality filtering strategies, and model selection criteria, revealing that generation success rate and content quality are uncorrelated.

2. Related Work

Our work intersects three research areas: procedural text understanding, event extraction methodologies, and dataset creation with language models.

2.1. Procedural Text Understanding

Procedural text understanding aims to extract structured representations of actions, entities, and their relationships from instructional text. Wu et al. (2023) build large-scale training data from minimal supervision for action condition inference, motivating our silver-standard generation pipeline. Du et al. (2024) demonstrate that models struggle with complex procedural structures, validating our focus on foundational trigger detection.

Entity state tracking has been studied extensively. Dalvi et al. (2018) (ProPara) and Zhang et al. (2023) introduce benchmarks for tracking and reasoning about state changes. Lal et al. (2025) propose CAT-BENCH, a cooking-domain benchmark testing LLM understanding of temporal and causal dependencies—relevant to modeling trigger-state relationships.

Kazeminejad et al. (2021) uses a VerbNet-based resource to annotate entity state changes in ProPara, supporting our choice of VerbNet as a baseline. Tu et al. (2024) extend POEM with GLAMR, augmenting AMR with subevent structure from Generative Lexicon and VerbNet, providing richer representations of event decomposition and argument property changes. Nandy et al. (2024) propose order-based pre-training strategies for procedural text, complementary to our focus on explicit event-state linkages.

2.2. Event Trigger Detection

Bottom-up, data-centric approaches rely on large training resources. The ACE corpus (Doddington et al., 2004) established early benchmarks with 5,300 events. Wang et al. (2020) introduced MAVEN, with 118,732 event mentions across 168 event types from Wikipedia, which trains our transformer baseline. Top-down approaches use abstract event schemas to guide detection. Jin and Ji (2024) employ schema-based data augmentation for trigger detection—directly relevant to our inverse annotation framework.

Rule-based methods like EVITA (Saurí et al., 2005) use linguistic rules and TimeML (Pustejovsky et al., 2005) for news text. We adopt EVITA to assess domain transfer to procedural instructions. Frame-semantic approaches leverage FrameNet (Baker et al., 1998) and VerbNet (Kipper et al., 2008). Brown et al. (2019) and Gung and Palmer (2021) develop a VerbNet parser, which we employ as our frame-semantic baseline.

2.3. Dataset Creation with Language Models

Recent work has explored using LLMs for data augmentation and annotation. Cho et al. (2022) use GPT-3 in a generate-then-select paradigm for entity linking in WikiHow, an approach analogous to our silver-standard generation pipeline.

A growing body of work leverages *inverse generation*—reversing the typical information extraction direction—to create synthetic training data. Josifoski et al. (2023) introduce SynthIE, which prompts LLMs to generate plausible input text from target output structures, synthesizing training examples for closed IE task. Ma et al. (2024) propose STAR, using structure-to-text generation to synthesize training data for low-resource event and relation extraction. Xu et al. (2024) survey these approaches, noting that LLMs excel at inverse generation because it aligns with their pretraining objective.

However, these inverse generation methods generate *natural language text* as their target output for data augmentation. Our inverse annotation framework differs fundamentally: while methodologically similar in reversing the annotation direction, we generate *structured linguistic annotations* rather than text. This structure-to-structure transformation enables annotation recovery rather than data

augmentation, addressing the distinct challenge of completing partially-annotated datasets.

3. Inverse Annotation for a Silver-Standard Dataset

Many annotation tasks produce datasets that record semantic outcomes while omitting the linguistic expressions that realize them. A human annotator performs a complex cognitive step—reading text, identifying relevant phenomena, and understanding their semantic function—but the annotation schema only captures the final interpretation (e.g., a semantic label, a structured outcome). The explicit link between the linguistic form and its semantic function is not preserved in the dataset.

We introduce inverse annotation, a general framework for recovering these missing links. The core premise is that existing outcome-based annotations already encode semantic information about the underlying linguistic content; by inverting the annotation process, we can use this information to reverse-engineer likely linguistic realizations. Specifically, instead of the traditional forward direction (text $\rightarrow$ annotation outcome), inverse annotation proceeds backward: given a known semantic outcome, we use Large Language Models to generate the likely linguistic expressions that would have produced it. This inverse process allows us to bootstrap datasets that make explicit connections between linguistic form and semantic function extensively, without requiring expensive dual-layer manual annotation.

We demonstrate this framework by addressing a critical gap in procedural text understanding. OpenPI serves as an ideal testbed: we provide known state-change outcomes to an LLM to reverse-engineer event triggers. By applying inverse annotation to OpenPI's state changes, we add event trigger annotations to steps that already have state-change annotations, creating a silver-standard dataset extensively without manual effort.

In practice, given an OpenPI state-change tuple such as (*dough*, *shape*, *whole*, *cut into oblongs*), the LLM generates a candidate trigger word "cut" along with a plausible sentence, which is then validated against the original procedural step and converted into a character-level span annotation. We walk through one complete instance below.

Worked Example Consider a procedural step from the document *"Make Pigs in the Blanket with Bisquick"*: the original sentence is *"Cut the dough into 4 x 3-inch oblong shapes."* The OpenPI annotation for this step records the state change (dough, shape, whole $\rightarrow$ oblongs) but does not annotate which word expresses the action.

Step 1: LLM Generation. The state-change tuple is formatted into a prompt (Appendix C) and sent to the LLM, which returns five candidate triggers with example sentences in JSON format. The top candidate is `{"event": "cut", "example": "Cut the dough into oblong shapes."}`.

Step 2: Semantic Filtering. The quality filter computes two scores: a `trigger_score` (GloVe similarity between "cut" and the original sentence tokens) of 1.0, and an `example_score` (Sentence-BERT similarity between the generated sentence and the original) of 0.92. The combined score ($1.0 \times 0.92 = 0.92$) exceeds the given threshold.

Step 3: Span Alignment. The span converter searches for "cut" in the original sentence *"Cut the dough into 4 x 3-inch oblong shapes."* and finds a substring match (case-insensitive) at character offsets (0, 3).

Result: The silver-standard annotation is the span "Cut" at positions (0, 3) in the original sentence, for the step containing the state change (dough, shape, whole $\rightarrow$ cut into oblongs).

3.1. Source Data: OpenPI

The OpenPI dataset contains 810 procedural documents from WikiHow with 4,050 steps and 31,473 entity state-change annotations. The dataset spans six domains: Food and Entertaining, Home and Garden, Hobbies and Crafts, Sports and Fitness, Cars & Other Vehicles, and Health. Each annotation records an (entity, attribute, before-state, after-state) tuple capturing transformations across attributes including location, temperature, composition, and physical properties. We use the train split (644 documents, 3,216 steps) as our source. Note that the Health domain (77 documents) appears exclusively in the test split, likely due to an oversight in the original data partitioning.

3.2. Silver-standard Annotation Pipeline

Our data generation process involves three key stages.

Pre-generation Filtering The pipeline begins by filtering documents to ensure they contain concrete, actionable steps. The filter removes documents where the goal of the task, as described in the title, does not necessarily involve multi-step procedural tasks or imply meaningful state changes of the participants. Examples of such filtered titles include *"Cut a Lemon Half with a Knot"* (Food and Entertaining), *"Defeat Gi Nattak in Final Fantasy VII"* (Hobbies and Crafts), and *"Fumblerooski"* (Sports and Fitness). We use spaCy (Honnibal et al., 2023) to first identify the main verb of the document title (task goal), and then apply a hand-crafted per-topic lexical filters to identify overly general actions or

skills. This filter deemed 86.54% of the documents are concrete.

LLM-Based Trigger Generation For each step in concrete documents (2783 out of total 3216 steps), we use local instruction-following LLMs to generate a natural language sentence and identify the primary action verb. We used seven models of various sizes from gemma, qwen and gpt-oss family. The model is prompted with the known state changes annotations but nothing else from OpenPI and instructed to generate a JSON object containing 5 likely event triggers and an example sentence for each trigger. The full prompt template is provided in Appendix C. The reliability of this stage is measured by the **Generation Success Rate**, the percentage of prompts for which the model returns a well-formed, parsable JSON output.

Post-generation Filtering and Conversion The final stage filters the generated content for quality and converts the proposed trigger word into a text span annotation. For quality filtering, we implemented a semantic scoring mechanism combining two similarity metrics: a `trigger_score` and an `example_score`. The `example_score` measures similarity between the generated sentence and the original source text using Sentence-BERT (Reimers and Gurevych, 2019). The `trigger_score` is computed as an averaged pairwise GloVe (Pennington et al., 2014) cosine similarity between the generated trigger and the original sentence tokens whose lemma matches the trigger word. We multiply these two scores to obtain a final quality score between 0 and 1. After applying a quality threshold to the semantic score, we convert the generated trigger word into a character-level span in the original sentence using cascading strategies: exact match, lemma match, semantic match, and partial match. Only examples that exceed the quality score threshold and have a valid span are retained as silver-grade. The final yield is measured by the **Conversion Rate**.

3.3. Discussion and Error Analysis

This section discusses the challenges and limitations encountered in each stage of the pipeline, along with error analysis that informed our design decisions.

Concreteness Filtering The heuristic-based filtering approach is conservative and may occasionally filter out valid documents with ambiguously worded titles. We accept this trade-off to improve the quality of the input to the generation stage, prioritizing precision over recall in document selection.

LLM-Based Generation This stage is sensitive to model-specific instruction-following failures. Our analysis revealed two primary failure modes: The first, which we term *Chain-of-Thought Contamination*, was common in models like 'gpt-oss-20b', which externalized their reasoning process into the output. This violated the "JSON only" constraint and led to a low generation success rate of approximately 70%. A second class of errors involved *Syntactic Failures*, where models failed to produce well-formed JSON. For example, 'qwen3-14b' often omitted necessary characters or generated incomplete structures, resulting in a moderately high failure rate of around 16%.

These errors highlight several challenges in structured data generation from LLMs: (1) output unpredictability, where even capable models fail inconsistently on seemingly identical inputs; (2) resource-intensive prompt optimization, requiring tedious trial-and-error iterations to identify phrasing that reliably produces valid outputs; and (3) the observation that simply using larger models does not solve these fundamental reliability issues, necessitating robust post-processing for error handling and careful model selection to ensure pipeline reliability.

Semantic Filtering and Span Annotation Conversion The semantic scoring depends on the quality of the embeddings and can be noisy. The span annotation can fail if the generated sentence deviates too significantly from the original, making it impossible to find a confident match for the trigger. This limitation is reflected in the conversion rates, where even high-quality generators like 'qwen3-14b' achieve only 15% conversion at the strictest threshold (0.80), indicating that the majority of generated examples are filtered out as insufficiently similar to the original text.

3.3.1. Model Reliability vs. Content Quality

We evaluated seven LLMs based on generation success rate and conversion success rate. The results reveal a crucial distinction between model reliability (ability to follow formatting rules) and content quality (semantic relevance). Specifically, we found no direct correlation between a model's ability to produce valid JSON (Generation Success Rate) and the quality of the content it produced (Conversion Rate). For instance, some models demonstrated high reliability but low content quality; 'gemma-2b' had a near-perfect 99.8% generation success rate but a very low 15% conversion rate, indicating it reliably produced irrelevant content. Conversely, other models showed low reliability but high quality. 'gpt-oss-20b', for example, had a low 70% success rate due to chain-of-thought contamination, but its successful generations had a high

Model	Successful Gen.		Conversion@0.5		Conversion@0.6		Conversion@0.7		Conversion@0.8	
	Rate (%)	# Ex.	Rate (%)	# Ex.	Rate (%)	# Ex.	Rate (%)	# Ex.	Rate (%)	# Ex.
qwen3-14b	84.53	2267	59.33	1345	44.82	1016	29.82	676	15.13	**343**
gemma-3n-4b	99.78	2676	59.72	1598	44.96	1203	28.70	768	14.28	**382**
gemma-3-12b	100.00	2682	56.04	1503	42.28	1134	26.32	706	13.24	355
gpt-oss-20b	70.28	1885	54.32	1024	39.31	741	24.56	463	11.94	225
gemma-7b	99.96	2681	49.72	1333	37.71	1011	23.54	631	11.86	318
qwen2.5-14b	99.96	2681	53.11	1424	38.12	1022	23.13	620	10.52	282
gemma-2b	99.81	2677	15.02	402	9.82	263	5.01	134	2.47	66

Table 1: Model Generation Success Rates and Conversion Rates at Different Quality Thresholds, ranked by highest threshold conversion rate. The table progresses from generation (Successful Gen.) through increasingly strict quality thresholds (0.5 to 0.8). "Rate" is the percentage of prompts that produced valid output (for Successful Gen.) or the percentage of successfully generated examples that were converted into silver-standard data (for thresholds). "# Ex." is the absolute number of examples at each stage. The top two models at threshold 0.8 were selected to create the final silver-standard dataset via ensemble union.

54% conversion rate. 'qwen3-14b' showed a similar pattern, with an 84% success rate (failing due to malformed JSON) but a 59% conversion rate. A third group of non-reasoning models, including the 'gemma' family (excluding 'gemma-2b') and 'qwen2.5-14b', proved to be the most dependable, achieving near-100% success rates in generation.

3.3.2. Model Ranking by Content Quality

Ranking by conversion rate showed 'gemma-3n-4b' was best for generating quantity at lower quality thresholds, whereas 'qwen3-14b' excelled at producing top-quality examples at the strictest thresholds. Notably, the small 4B-parameter 'gemma-3n-4b' nearly matched the large 'qwen3-14b"s top-tier conversion rate (14.28% vs. 15.13%), suggesting model size alone does not determine generation quality.

3.4. Final Dataset Selection and Dataset Statistics

Table 1 summarizes model performance across all OpenPI domains. Our final silver-standard dataset contains **515 examples**, created by taking the union of annotations from the two top-performing models to merge the complementary strengths of the high-volume 'gemma-3n-4b' and the high-precision 'qwen3-14b'. When both models annotated the same sentence, we selected the annotation with the maximum score. The dataset is composed of 172 examples (33.4%) agreed upon by both models, 191 (37.1%) unique to 'gemma-3n-4b', and 152 (29.5%) unique to 'qwen3-14b'. Within the 172 commonly annotated examples, the inter-model agreement was high, with a 95.3% exact span match and 4.7% partial overlap, validating the quality of the silver-standard annotations.

The dataset spans multiple procedural domains, with sentences that are typically imperative ("Combine the ingredients"), averaging 8-12 tokens in length. Trigger spans are predominantly verbs (98%), with occasional multi-word triggers ("fold in," "bring to a boil").

3.5. Human Validation

To assess the quality of the silver-standard annotations and demonstrate the cognitive simplicity of verification under inverse annotation, we conducted a human validation study. Two graduate student annotators independently verified a stratified sample of 452 annotation units drawn from 101 steps, proportional to the topic distribution of the full dataset.

Each annotation unit presents one entity group within a step: all attribute changes for a single entity, the full sentence with clickable tokens, and the silver trigger pre-highlighted. Note that while the silver generation pipeline processes one state-change tuple at a time, the validation groups all tuples sharing the same entity string (by exact string match, without normalization) to provide the annotator with full context for the trigger decision. This per-entity design avoids both the ambiguity of per-step presentation (where multi-event sentences conflate triggers) and the redundancy of per-transformation presentation (where multiple attributes of the same entity repeat the same trigger decision). For each unit, the annotator either confirms the silver trigger, selects a different span, or trash the unit as unannotatable (e.g., negation sentences, nonsensical transformations).

Table 2 summarizes the results. Both annotators completed the task in under 25 minutes (median 1.6s and 1.3s per unit), confirming that verification under inverse annotation is a cognitively simpler task compared to forward trigger annotation. Silver acceptance rates are consistent across annotators (75.6% and 76.4%). Inter-annotator agreement on the accept-or-correct decision yields a Cohen's κ of 0.857, and among the 439 units confirmed by both annotators, exact span match is 92.5%. Both non-acceptance and the span disagreements concentrate in the same linguistic categories: phrasal verbs ("hold" $\rightarrow$ "hold down"), light verb construc-

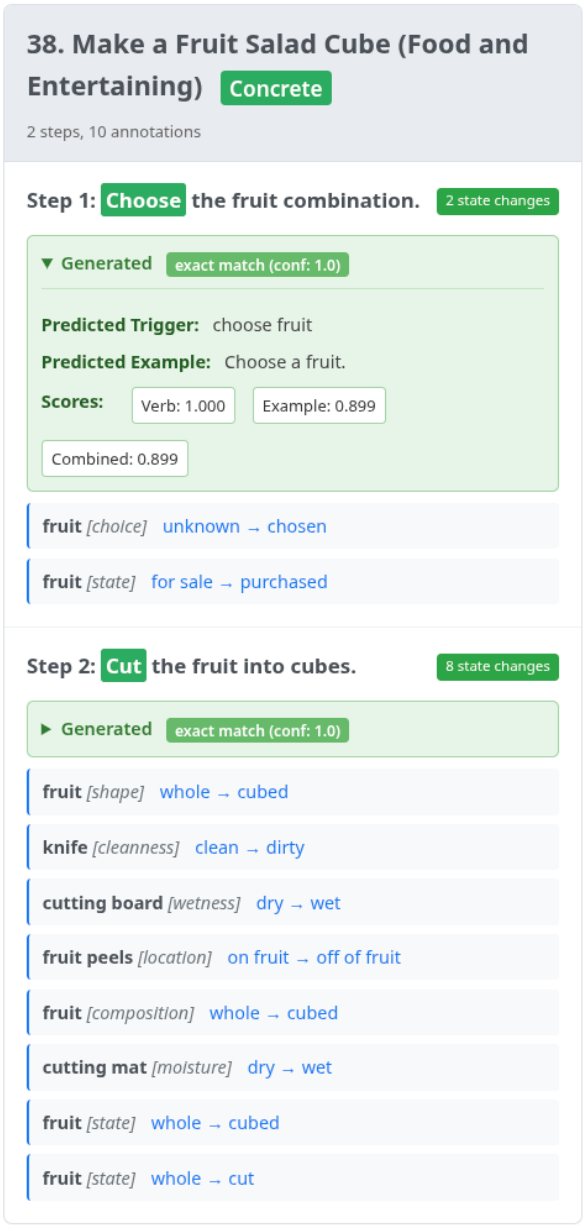

Figure 2: Silver-standard annotation example illustrating the three pipeline stages: (1) state-change tuples from OpenPI are input to the LLM, (2) candidate triggers are generated and scored via semantic filtering (Verb and Example scores shown for Step 1), and (3) validated triggers (green highlights) are aligned to character-level spans in the original procedural text.

tions ("bring" → "boil"), and coordinated predicates ("cut and push")—cases where the trigger boundary is genuinely ambiguous. When both annotators corrected, 89% selected the same new span, confirming these are systematic boundary adjustments rather than incorrect triggers. The trash rate is low (1.3–2.7%), with most trashed units concentrated in advisory or prohibitive sentences (e.g., "Resist the temptation to…") from the Sports domain.

	Ann. A	Ann. B
Trashed	6 (1.3%)	12 (2.7%)
Silver accepted	337 (75.6%)	336 (76.4%)
	Agreement	
Action agr.	98.2%	
Decision κ	0.857	
Exact span	92.5%	
Relaxed span	93.4%	

Table 2: Human validation results. Silver acceptance rate reflects units where the annotator confirmed the pipeline trigger without correction. Decision κ measures agreement on whether to accept or correct the silver annotation.

4. Baseline Evaluation

We benchmark a diverse range of event trigger detection systems—statistical, rule-based, frame-semantic, neural, and LLM—to validate the dataset and characterize task difficulty. Event trigger detection is formalized as a character-level span identification task; we focus on the *primary* event trigger when multiple events are present.

4.1. Evaluation Metrics

We evaluate system predictions using character-level span-based metrics that form a hierarchy of strictness. Our primary metric is **Exact Match (EM)**, where a prediction is correct if and only if the predicted span (i_{pred}, j_{pred}) exactly matches the reference span $(i_{reference}, j_{reference})$. We report standard Precision, Recall, and F1 scores based on exact matches.

To further analyze system performance, we supplement EM with two additional metrics that progressively relax the boundary alignment requirement. **Intersection-over-Union (IoU)** quantifies the degree of overlap between predicted and reference spans: IoU = $\frac{|\text{predicted} \cap \text{reference}|}{|\text{predicted} \cup \text{reference}|}$, ranging from 0 (no overlap) to 1 (perfect match). **Overlap Accuracy** adopts the most lenient criterion, measuring simply whether any character-level overlap exists between spans, regardless of alignment quality.

Together, these three metrics—Exact Match (strictest), IoU (intermediate), and Overlap Accuracy (most lenient)—distinguish whether systems locate trigger regions versus precisely delineate boundaries. We emphasize exact-match F1 as our primary criterion, as downstream applications require precise boundaries.

4.2. Baseline Systems

The approaches range from simple statistical methods to large transformer models and few-shot prompted LLMs.

4.2.1. Statistical Co-occurrence (PMI)

Our simplest baseline uses Pointwise Mutual Information (Church and Hanks, 1990) to identify triggers by computing PMI scores between sentence words and the "before" and "after" state-change attributes (e.g., "whole" and "chopped"). Scores are calculated using pre-computed co-occurrence statistics from a subset of the Corpus of Contemporary American English (COCA) (Davies, 2008). This approach has privileged access to state-change annotations, making it not strictly comparable but useful for testing statistical association.

4.2.2. Rule-Based Event Extraction (EVITA)

We use EVITA (Events In Text Analyzer) (Saurí et al., 2005), a rule-based system from the Tarsqi Toolkit (TTK) (Verhagen and Pustejovsky, 2012) that implements TimeML event detection. Originally designed for news text, EVITA applies domain-agnostic linguistic patterns based on part-of-speech, syntax, and lexical resources. We convert the `EVENT` tags returned by EVITA into our required character-level span predictions to assess the generalizability of news-domain rules to procedural text.

4.2.3. Transformer-Based Baselines

We evaluate three baselines derived from fine-tuned transformer architectures, all trained on non-procedural domains, to assess cross-domain transfer and compare different training objectives and architectures.

Frame-Semantic Parsing (VerbNet Parser) Our first transformer-based approach uses a VerbNet parser (Gung and Palmer, 2021), which is built on a fine-tuned BERT model. The parser is trained to identify verb predicates and predict their VerbNet class. We repurpose this frame-semantic system for trigger detection by extracting the predicate spans it identifies. This baseline tests whether a model trained on a semantic parsing task, leveraging the rich structure of VerbNet, can be effectively transferred to trigger identification.

Sequence Labeling (RoBERTa) Our second approach is a standard RoBERTa-base transformer (Zhuang et al., 2021) fine-tuned on the MAVEN dataset (Wang et al., 2020) for sequence labeling (BIO tagging). This represents a more traditional approach to event extraction, learning trigger patterns from a large, general-domain corpus (Wikipedia). This baseline tests the effectiveness of cross-domain transfer from a large, non-procedural dataset. Full training and implementation details are provided in Appendix B.

Seq2Seq Generation (FLAN-T5) To contrast encoder-only and encoder-decoder architectures under identical training conditions, we fine-tune FLAN-T5-large (Chung et al., 2022) (780M parameters) on MAVEN using a text-to-text formulation: the model generates the input sentence with `<trigger>` markup tags inserted around event triggers. Unlike token classification, this formulation does not force the model to make a prediction for every token—the decoder may reproduce the input unchanged.

4.2.4. LLM-Based Few-Shot Prompting

Finally, we evaluate 10 large language models using few-shot prompting. The models, spanning the Gemma, Qwen, and GPT families, are instructed to act as expert linguists and return JSON-formatted predictions with exact text and character offsets. The prompt includes five diverse examples from our procedural domains to demonstrate the task. This approach tests in-context learning without fine-tuning, relying solely on the knowledge encoded in the pre-trained models.

4.3. Results and Discussion

Table 3 presents the performance of 14 trigger detection pipelines (a 15th, FLAN-T5, is discussed in the error analysis). For aggregate performance across all topics, we report F1, Precision, Recall, Overlap Accuracy, and IoU. For individual topics, we report F1 only to maintain table clarity.

We observe the following patterns in the results:

System Performance Spectrum The fine-tuned **RoBERTa** transformer and rule-based **EVITA** system achieve the highest exact match F1 scores (both 65.9% and 65.7% respectively), substantially outperforming all other approaches. RoBERTa, trained on the MAVEN dataset, demonstrates effective cross-domain transfer to procedural text. EVITA, designed for news text in the early 2000s using TimeML event extraction rules, achieves remarkable competitive performance through domain-agnostic linguistic patterns. At the other end of the spectrum, the PMI statistical baseline achieves F1=.144, as expected for a short-sighted statistical model lacking syntactic and semantic structures.

Metric Hierarchy Examining the three metrics together, from strict to lenient, reveals distinct system behaviors. **EVITA** achieves high overlap accuracy (.825) and IoU (.835) but lower EM-F1 (.657), successfully identifying trigger regions while struggling with exact boundaries. **RoBERTa** maintains strong performance across all metrics (Overlap=.781, IoU=.780, EM=.659), demonstrating balanced boundary precision. In contrast, **gpt-3.5-**

Pipeline	F1	P	R	Ovlp	IoU	F1	F1	F1	F1	F1
	All Topics					Cars	Food	Hobbies	Home	Sports
PMI	*.144*	*.144*	*.144*	*.144*	*.146*	.250	.116	.176	.167	.191
EVITA	.657	.548	**.822**	**.825**	**.835**	**.638**	.659	.642	**.708**	.589
VNP	.621	.522	.767	.784	.790	.571	.607	.619	.667	.595
RoBERTa	**.659**	.586	.752	.781	.780	.623	**.667**	**.659**	.676	**.625**
gemma-2b	*.015*	*.023*	*.011*	*.029*	*.043*	*.000*	*.029*	*.000*	*.014*	*.000*
gemma-3-12b	.516	.440	.624	.717	.728	.583	.491	.497	.642	.397
gemma-3n-4b	.413	.367	.472	.671	.662	.455	.358	.447	.473	.371
gemma-7b	.421	.350	.528	.671	.667	.500	.383	.387	.522	.336
gpt-3.5-turbo	.247	.218	.286	.732	.658	.281	.233	.239	.286	.225
gpt-5	.569	**.690**	.483	.501	.506	.618	.601	.564	.548	.500
gpt-5-mini	.622	.623	.621	.647	.659	.629	.639	.615	.641	.531
gpt-oss-20b	.517	.585	.464	.493	.501	.400	.542	.527	.533	.500
qwen2.5-14b	.466	.413	.536	.682	.682	.578	.399	.478	.553	.412
qwen3-14b	.524	.463	.604	.700	.711	.622	.492	.494	.628	.451

Table 3: Trigger detection performance across all pipelines. **Bold**: highest score per column; *Italic*: lowest score per column. Non-LLM baselines: PMI (statistical), EVITA (rule-based), VerbNetParser, RoBERTa (MAVEN-finetuned). LLMs use 10-shot examples across in-domain topics. For "All Topics", we report F1, Precision, Recall, Overlap Accuracy (Ovlp), and IoU. For individual topics, we report F1 only.

turbo exhibits large metric gaps (Overlap=.732, IoU=.658, EM=.247), finding relevant regions but producing poor boundary alignment. Unusually, **gpt-5** shows relatively low overlap (.501) despite higher EM (.569), suggesting highly conservative but precise predictions. For most LLMs, Overlap Accuracy substantially exceeds both IoU and F1, indicating that the primary challenge is not finding trigger regions but achieving exact span alignment.

Precision-Recall Trade-offs EVITA achieves the highest recall (.822) but lower precision (.548), suggesting it casts a wider net by identifying all TimeML event classes including states and aspectual triggers. In contrast, gpt-5 shows the highest precision (.690) but lower recall (.483), indicating conservative predictions. RoBERTa balances both metrics more effectively (P=.586, R=.752).

Large Language Model Performance We employ few-shot prompting as described in Section 4.2 (full prompt in Appendix D).

LLMs show varied performance. The best-performing LLM (gpt-5-mini, F1=.622) approaches but does not exceed the top non-LLM systems. Within model families, larger models generally perform better (gemma-3-12b > gemma-7b > gemma-3n-4b), and commercial models outperform open-weight alternatives (gpt-5-mini outperforms all local LLMs). However, the smallest model (gemma-2b) achieves near-zero performance (F1=.015), indicating insufficient capacity for this task. Overall, while few-shot prompting provides a viable approach without task-specific training data, it does not yet match the performance of fine-tuned transformers or well-engineered rule systems.

Topic-Level Variation Performance varies across procedural topics. EVITA achieves the highest F1 on Home and Garden (.708) and Cars & Other Vehicles (.638), while RoBERTa performs best on Food and Entertaining (.667), Hobbies and Crafts (.659), and Sports and Fitness (.625). This variation suggests topic-specific linguistic patterns that different approaches capture differently. The relatively weaker performance of all systems on Sports and Fitness (best F1=.625) compared to Home and Garden (best F1=.708) indicates domain-specific challenges in procedural action vocabulary.

4.4. Error Analysis

We examine the primary failure modes of high-performing systems:

4.4.1. Over-Prediction Pattern

EVITA and VNP exhibit systematic over-prediction, identifying multiple event-like expressions per sentence where only the primary procedural action is annotated. Consider this example:

> *"Blend thoroughly until you achieve a smooth texture."*
> REF: "Blend"
> EVITA: "Blend," "achieve"
> VNP: "Blend," "achieve," "smooth"

While the silver-standard annotates only the primary procedural action ("Blend"), systems reasonably identify "achieve" as an event trigger. EVITA's news-domain rules identify all TimeML event classes—including STATE and ASPECTUAL triggers—rather than focusing solely on primary procedural actions. Similarly, VNP marks "smooth"

as a state predicate within its frame-semantic representation, reflecting a broader event conceptualization than our annotation scheme captures.

4.4.2. Semantic Drift in VNP

VNP's performance (F1=.621) despite rich semantic knowledge reveals a task alignment challenge. Despite sharing similar transformer architecture with RoBERTa—both built on fine-tuned BERT-like models—VNP substantially underperforms (F1=.621 vs. .659). This gap appears to stem from training objective differences rather than model capacity: VerbNet frames capture states, properties, and auxiliary predicates in addition to primary actions, which may lead to systematic false positives in our task. VNP identifies states as events ("ready," "hot," "smooth"), marks auxiliary actions ("have," "be," "get"), and detects perception predicates ("see," "notice"). This pattern suggests that VerbNet's broad semantic scope may not align well with the narrow procedural-action focus required by our annotation scheme.

4.4.3. Architectural Mismatch: FLAN-T5

To directly test whether architecture or model size drives performance, we fine-tuned FLAN-T5-large (780M parameters) on the same MAVEN data as RoBERTa (125M parameters). However, the seq2seq model with 6$\times$ more parameters achieves only 14.1% F1, with 82.8% of predictions completely empty.

The root cause is architectural: T5's autoregressive decoder can reproduce the input sentence unchanged, which constitutes a valid (if unhelpful) output. Under domain shift from news to procedural text, the model defaults to this conservative behavior rather than inserting `<trigger>` markup tags. A second training attempt with a custom tag-penalty loss produced identical results, confirming this is an architectural limitation rather than an optimization problem. In contrast, RoBERTa's token classification head *forces* a label decision at every position, making it unable to "opt out" of prediction; it over-predicts under uncertainty (R=.752) rather than refusing to predict. This finding suggests that encoder-only architectures with forced per-token classification are preferable to encoder-decoder generation for span detection tasks.

4.4.4. LLM Output Format Failures

LLMs occasionally fail to produce well-formed JSON output, generating malformed responses or narrative explanations instead of structured predictions. Smaller models, particularly gemma-2b, struggle with instruction following, contributing to their poor performance. Even when achieving format compliance, LLMs sometimes hallucinate trigger spans or provide incorrect character offsets.

5. Future Work

Large-Scale Annotation and Domain Generalization The RoBERTa model's 65.9% F1 represents the best single-system baseline; we propose using it to annotate the full OpenPI dataset (24,000 procedures), creating a large-scale procedural event corpus. The inverse annotation pipeline itself is domain-agnostic, and we plan to apply it to other task-oriented text datasets beyond WikiHow.

Resource Release We release the silver-standard dataset, inverse annotation pipeline code, and fine-tuned RoBERTa model in our public repository[1] to support reproducibility and enable adaptation to new domains.

6. Conclusion

Understanding procedural text requires modeling both actions and their effects, yet existing large-scale datasets typically capture only one dimension. We have presented inverse annotation, a framework for using LLMs to recover missing linguistic annotations by reversing the traditional annotation direction. Applying this to OpenPI's state-change annotations, we created OpenPI-A, a silver-standard dataset of 515 examples with event trigger annotations added to existing state-change data across five procedural domains.

Comprehensive evaluation of 14 systems validates the dataset quality and establishes baseline performance. The inverse annotation framework is domain-agnostic and applicable to other annotation recovery tasks, enabling scalable completion of partially-annotated resources without expensive dual-layer manual effort.

7. Impact Statement

This work advances procedural text understanding and provides a scalable framework for annotation recovery. Applications include instruction-following systems, task planning, and assistive technologies for interpreting complex procedural knowledge.

8. Ethical Considerations

This work uses publicly available data from WikiHow via the OpenPI dataset. The human validation

[1] https://github.com/brandeis-llc/poem-OPIA

study involved two graduate student annotators verifying existing silver-standard annotations; no new text data was collected. The procedural domains studied (cooking, home repair, etc.) present no foreseeable ethical concerns.

9. Bibliographical References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.

Susan Windisch Brown, Julia Bonn, James Gung, Annie Zaenen, James Pustejovsky, and Martha Palmer. 2019. VerbNet Representations: Subevent Semantics for Transfer Verbs. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 154–163, Florence, Italy. Association for Computational Linguistics.

Young Min Cho, Li Zhang, and Chris Callison-Burch. 2022. Unsupervised Entity Linking with Guided Summarization and Multiple-Choice Selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9394–9401, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Singh, Zoubin Ghahramani, Neil Houlsby, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70):1–53.

Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wentau Yih, and Peter Clark. 2018. Tracking State Changes in Procedural Text: A Challenge Dataset and Models for Process Paragraph Comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

Mark Davies. 2008. The Corpus of Contemporary American English (COCA).

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Weihong Du, Wenrui Liao, Hongru Liang, and Wenqiang Lei. 2024. PAGED: A Benchmark for Procedural Graphs Extraction from Documents.

James Gung and Martha Palmer. 2021. Predicate Representations and Polysemy in VerbNet Semantic Parsing. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 51–62, Groningen, The Netherlands (online). Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, and Henning Peters. 2023. spaCy: Industrial-strength Natural Language Processing in Python. Zenodo.

Xiaomeng Jin and Heng Ji. 2024. Schema-based Data Augmentation for Event Extraction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14382–14392, Torino, Italia. ELRA and ICCL.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction.

Ghazaleh Kazeminejad, Martha Palmer, Tao Li, and Vivek Srikumar. 2021. Automatic Entity State Annotation using the VerbNet Semantic Parser. In *Proceedings of the Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 123–132, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Raymond Mooney. 2025. CaT-BENCH: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans.

Mingyu Derek Ma, Xiaoxuan Wang, Po-Nien Kung, P. Jeffrey Brantingham, Nanyun Peng, and Wei Wang. 2024. STAR: Boosting Low-Resource Information Extraction by Structure-to-Text Data Generation with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18751–18759.

Abhilash Nandy, Yash Kulkarni, Pawan Goyal, and Niloy Ganguly. 2024. Order-Based Pre-training Strategies for Procedural Text Understanding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 821–828, Mexico City, Mexico. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. 2005. Temporal and Event Information in Natural Language Text. *Language Resources and Evaluation*, 39(2):123–164.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kyeongmin Rim, Jingxuan Tu, Bingyang Ye, Marc Verhagen, Eben Holderness, and James Pustejovsky. 2023. The Coreference under Transformation Labeling Dataset: Entity Tracking in Procedural Texts Using Event Models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12448–12460, Toronto, Canada. Association for Computational Linguistics.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A Robust Event Recognizer For QA Systems. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 700–707, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A Dataset for Tracking Entities in Open Domain Procedural Text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online. Association for Computational Linguistics.

Mashiro Toyooka, Kiyoharu Aizawa, and Yoko Yamakata. 2025. A Highly Clean Recipe Dataset with Ingredient States Annotation for State Probing Task.

Jingxuan Tu, Timothy Obiso, Bingyang Ye, Kyeongmin Rim, Keer Xu, Liulu Yue, Susan Windisch Brown, Martha Palmer, and James Pustejovsky. 2024. GLAMR: Augmenting AMR with GL-VerbNet Event Structure. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7746–7759, Torino, Italy. ELRA and ICCL.

Marc Verhagen and James Pustejovsky. 2012. The TARSQI Toolkit. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2043–2048, Istanbul, Turkey. European Language Resources Association (ELRA).

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1652–1671, Online. Association for Computational Linguistics.

Te-Lin Wu, Caiqi Zhang, Qingyuan Hu, Alexander Spangher, and Nanyun Peng. 2023. Learning Action Conditions from Instructional Manuals for Instruction Understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3023–3043, Toronto, Canada. Association for Computational Linguistics.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357.

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English Recipe Flow Graph Corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023. Causal Reasoning of Entities and Events in Procedural Texts. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia. Association for Computational Linguistics.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A Robustly Optimized BERT Pre-training Approach with Post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A. Dataset Statistics

A.1. Silver-Standard Selection Pipeline

Our final dataset was created by computing the union of annotations from two top-performing models (**gemma-3n-4b** and **qwen3-14b**):

- **gemma-3n-4b:** 382 examples at threshold 0.80
- **qwen3-14b:** 343 examples at threshold 0.80
- **Common sentences:** 191 (both models annotated)
- **Agreement:** 172 examples (90.0% of common sentences)
- **Final union dataset:** 515 examples (172 agreed + 191 gemma-only + 152 qwen-only)

A.2. Span Matching Strategy Distribution

Match Type	Count	Percentage
Exact match	458	88.9%
Lemma match	29	5.6%
Semantic match	28	5.4%
Total	**515**	**100.0%**

A.3. Trigger Span Length Distribution

Length (chars)	Count	Percentage
1–3 characters	91	17.7%
4–6 characters	304	59.0%
7–10 characters	105	20.4%
$>$10 characters	15	2.9%

A.4. Most Frequent Triggers

Top 10 most common event triggers in the dataset:

1. cut (25 occurrences)
2. add (22 occurrences)
3. remove (15 occurrences)
4. place (13 occurrences)
5. attach (12 occurrences)
6. pour (11 occurrences)
7. fill (9 occurrences)
8. apply (8 occurrences)
9. heat (8 occurrences)
10. mix (7 occurrences)

The trigger vocabulary reflects the multi-domain nature of the dataset, spanning cooking actions (*cut*, *pour*, *heat*, *mix*), construction and repair (*attach*, *apply*, *fill*), and general procedural verbs (*add*, *remove*, *place*).

B. RoBERTa Implementation Details

Training Configuration The RoBERTa-base model was fine-tuned for token classification on the MAVEN dataset using the following configuration:

- **Base model:** RoBERTa-base (125M parameters)
- **Training data:** MAVEN train split (32,431 sentences, 118,732 events)
- **Task formulation:** Token classification with 3 labels (O, B-TRIGGER, I-TRIGGER)
- **Hyperparameters:** Learning rate 2e-5, batch size 16, 3 epochs
- **Training performance:** The model achieved 95.91% token-level F1 on the MAVEN validation set. This score reflects strong performance on the in-domain task and is not comparable to the final zero-shot span-level evaluation on our procedural dataset.

Inference Process To generate trigger predictions for our dataset, we used the following process:

1. Tokenize the input sentence using the RoBERTa subword tokenizer.
2. Run a forward pass to obtain BIO label predictions for each token.
3. Reconstruct trigger spans by identifying B-TRIGGER starts and subsequent I-TRIGGER continuations.
4. Map the resulting token-level positions to character-level spans in the original, untokenized sentence.

C. Generation Prompt Template

The following prompt template is used for LLM-based trigger generation from OpenPI state-change records. The `{changes_table}` placeholder is populated with the tab-separated state-change tuples for the current procedural step:

```
You are a data processing service. Your sole function
is to analyze state changes and return a structured
JSON output.

Your task is to identify the 5 most probable events
based on the "state_changes" table provided below.

You must return your answer ONLY as a valid JSON array
of objects. Do not include any explanations, introductory
text, or markdown. Your entire response must be ONLY the
raw JSON array, starting with '[' and ending with ']'.

The JSON array must contain exactly five objects. Each
object must conform to the following structure:
1. A key named "event" with a string value. The value
   must be a brief, lower-case verb phrase that describes
   the event (e.g., "mix ingredients").
2. A key named "example" with a string value. The value
   must be a short example sentence in the imperative
   mood, not exceeding 20 words.

Here is the "state_changes" table:
'''
{changes_table}
'''

Here is the required JSON output:
```

Example Input and Output Given the state-change table for a procedural step about cutting dough:

```
dough   shape   whole   cut into oblongs
```

A typical model output:

```
[
  {"event": "cut", "example": "Cut the dough into
   4 x 3-inch oblong shapes."},
  {"event": "slice", "example": "Slice the dough
   into oblong pieces."},
  ...
]
```

The post-generation filter then aligns the top-scoring trigger ("cut") to the original sentence as a character-level span using exact match.

D. LLM Few-Shot Evaluation Prompt

For few-shot trigger detection evaluation (Section 4.2), each LLM receives the following prompt with 10 worked examples spanning all procedural domains. The examples demonstrate both single- and multi-trigger sentences, phrasal verbs, and edge cases:

```
Your task is to act as an expert linguist and identify
the event triggers in a sentence. An event trigger is
the specific word or multi-word phrase that most clearly
signifies an event or action.

**Guidelines:**
- Only mark the main predicate, not auxiliaries
  (e.g., "will cook" -> mark "cook", not "will")
- Include particles with phrasal verbs
```

```
  (e.g., "turn on", "break down")
- For light verbs, mark only the verb, not the full
  phrase (e.g., "take a walk" -> mark "take")
- Use conservative judgment for ambiguous cases
- Provide a confidence score (0.0-1.0) for each trigger

**Instructions:**
1. Identify all event triggers in the sentence.
2. For each trigger, provide its exact text, start and
   end character offsets, and confidence.
3. Respond with a single JSON object containing a key
   "triggers".
4. The value of "triggers" should be a list of JSON
   objects, where each object has the keys "text",
   "start", "end", and "confidence".
5. If no triggers are found, return an empty list.

Examples:

Sentence: "Check the oil level and inflate the tires."
{"triggers": [
  {"text":"Check","start":0,"end":5,"confidence":1.0},
  {"text":"inflate","start":24,"end":31,"confidence":1.0}
]}

Sentence: "You should turn on the headlights ..."
{"triggers": [
  {"text":"turn on","start":11,"end":18,"confidence":1.0},
  {"text":"driving","start":41,"end":48,"confidence":0.9}
]}

[... 8 additional examples across domains ...]

Sentence: "{input_sentence}"
Response:
```