

Variable-size Discovery and Representation of Atomic Propositions

Francesco Ortame¹ and Roberto Navigli^{1,2}

¹Sapienza NLP Group, Sapienza University of Rome ²Babelscape, Italy
{ortame, navigli}@diag.uniroma1.it

Abstract

Sentence embeddings represent meaning as a single fixed-size vector, regardless of the complexity of the input sentence. This design compresses all propositional structure into a single point, losing part of the internal structure of the sentence in the process. To address this, we introduce Atomic Proposition Embeddings (APE) to dynamically discover and encode atomic propositions in sentences as variable-size bags of vectors. We use a slot mechanism, inspired by object detection in computer vision, over an encoder backbone. APE is trained on a proposition-decomposed version of WikiSplit, and matches or outperforms sentence encoders up to $15\times$ its size on PAWS-WIKI and SNLI with no task-specific fine-tuning, using proposition-level scoring rules at inference time. We further introduce PROPSEARCH, a retrieval benchmark for proposition-level embeddings in which models must retrieve source sentences given atomic propositions. APE leads on PROPSEARCH over proposition-level alternatives, with its advantage growing with sentence complexity. Ablations confirm the gains are architectural rather than data-driven and that APE can generalize over its training signal. We release our code, data and models for research purposes at <https://github.com/SapienzaNLP/APE>.

1 Introduction

Consider the sentences *Humans developed complex social hierarchies early in their evolutionary history, but also some other apes have them* and *Some apes have sophisticated social structures*. These sentences are related but not equivalent: the first expresses multiple propositions, while the second expresses a clear atomic proposition. A sentence embedding model assigns both a fixed-size vector of identical dimensionality, collapsing all propositional content into a single point in space (Reimers and Gurevych, 2019). This is a structural limitation,

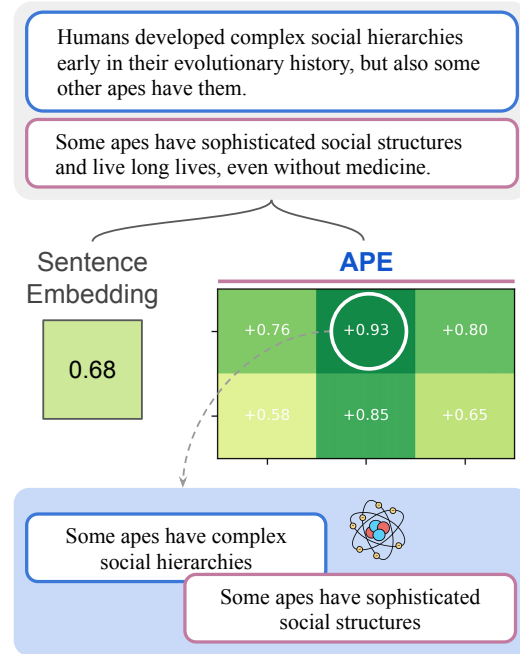


Figure 1: APE represents each sentence as a bag of proposition embeddings. The similarity matrix reveals proposition-level similarities, interpretable via proposition seeding (Section 6.3). A fixed-size sentence embedding, instead, collapses similarity into a single score.

since sentence embeddings fail to reflect the internal propositional complexity of their input (Chen et al., 2024; Xu et al., 2025).

Prior work has addressed this by learning sub-sentence representations (Chen et al., 2024), with one embedding per proposition span, but this approach relies on separate generative models and token masking to extract spans before encoding, decoupling discovery from representation. The encoder thus never learns what a proposition *is*, just how to optimize span representations.

To address this gap, we introduce **Atomic Proposition Embeddings (APE)**, a sentence representation paradigm that jointly learns to discover and represent propositions as a variable-size bag of embeddings. Given a sentence, APE produces one

dense vector per identified atomic proposition (Figure 1). Built on a pretrained encoder, APE uses a set-prediction paradigm inspired by the DETection TRansformer (DETR) (Carion et al., 2020), in which learnable slot queries attend over token representations, each converging toward an atomic proposition; an activity head then determines which slots are active. APE is trained via knowledge distillation on proposition-decomposed, coreference-resolved data, and at inference time sentence pairs are compared using proposition-level scoring rules over the resulting matrix without task-specific fine-tuning.

APE matches or outperforms models up to $15\times$ its size on PAWS-WIKI (Zhang et al., 2019) and SNLI (Bowman et al., 2015). To directly assess decomposition quality, we introduce **PROPSEARCH**, a proposition retrieval benchmark that evaluates how well a model retrieves source sentences from their atomic propositions at varying complexity levels. APE leads consistently over other proposition-level models, and is competitive with large instruction-tuned embedding models, especially as complexity increases.

Our main contributions are:

- **APE**, a new sentence representation paradigm that dynamically discovers and represents atomic propositions as a variable-size bag of vectors via a DETR-inspired slot mechanism trained on proposition-decomposed data.
- **Proposition-level scoring rules** operating directly on the proposition matrix, enabling symmetric and asymmetric zero-shot comparison.
- **PROPSEARCH**, a retrieval benchmark for evaluating proposition-level embedding models stratified by propositional complexity.

2 Related Work

APE sits at the intersection of three research areas: sentence representation learning, proposition-level semantics, and variable-size set prediction.

Sentence embeddings Neural approaches to sentence-level representation include InferSent (Conneau et al., 2017), the Universal Sentence Encoder (Cer et al., 2018), and Sentence-BERT (Reimers and Gurevych, 2019); the latter adapted siamese networks with NLI supervision to produce semantically meaningful fixed-size embeddings. Contrastive learning significantly advanced

the field, with SimCSE (Gao et al., 2021) demonstrating that dropout-based augmentation suffices for strong unsupervised representations.

Subsequent work introduced large-scale models trained on diverse retrieval and similarity objectives, including E5 (Wang et al., 2022), GTE (Li et al., 2023) and BGE (Xiao et al., 2024), as well as embedding models derived from large generative backbones, such as Qwen3-Embedding (Zhang et al., 2025) and EmbeddingGemma (Vera et al., 2025). Despite continuous improvements in quality, all of these models share a fundamental design choice: a sentence, regardless of its propositional complexity, is mapped to a single fixed-size vector.

Multi-vector retrieval models partially address the fixed-size limitation by keeping the representation at one embedding per token to enable token-level late interaction (Khattab and Zaharia, 2020; Jha et al., 2024). However, token-level representations correspond to very fine-grained units, reflecting positional, lexical or syntactic structure more than propositional content, and the number of vectors scales with token count rather than meaning.

Proposition-level representations The idea of decomposing a sentence into propositional units has deep roots in NLP, from early work on semantic role labeling (Palmer et al., 2011) and PropBank (Palmer et al., 2005) to graph-based meaning representations such as AMR (Banarescu et al., 2013). These approaches produce discrete symbolic structures rather than learned dense representations.

More recently, Rudinger et al. (2017) proposed Skip-Prop vectors with a Skip-Thought-style sequence-to-sequence objective (Kiros et al., 2015), learning one embedding per proposition over externally pre-segmented spans, evaluated only qualitatively.

Similar in nature to Skip-Prop, though more closely related to our work, is the Sub-Sentence Encoder (SSE), which produces proposition-level embeddings by span-pooling token representations over boundaries identified by an external large generative model (Chen et al., 2024). Proposition discovery and representation are thus handled by two separate models with no shared objective: the encoder learns to represent spans, but never learns what makes a span propositional. APE instead jointly learns to discover and represent propositions through a slot mechanism trained end-to-end on proposition-decomposed supervision, producing a variable number of embeddings that scales

with propositional complexity, with no dependency on external models.

More broadly, atomic propositions have emerged as a useful unit of analysis in several NLP applications, including fine-grained factuality evaluation (Min et al., 2023; Scirè et al., 2026), atomic claim verification (Zheng et al., 2025) and fact-checking (Cen et al., 2026), demonstrating that proposition-level decomposition captures meaningful semantic structure beyond what sentence-level representations expose. However, none of this work produces learned dense embeddings at the propositional level, as proposition decomposition is used as a processing step rather than as a representational paradigm.

Variable-size set prediction APE’s architecture draws on the set prediction paradigm introduced by DETR (Carion et al., 2020), which frames object detection as direct set prediction using a transformer encoder-decoder with learned object queries and a Hungarian matching loss. Slot attention (Locatello et al., 2020) similarly uses a competitive iterative attention mechanism to bind latent slot vectors to distinct entities, demonstrating that learned slots can discover compositional structure without explicit supervision. APE applies a similar order-invariant set prediction paradigm to semantic representation learning: slot queries attend over token representations to discover and encode propositions, supervised via Hungarian matching against proposition targets. To the best of our knowledge, APE is the first text representation model to apply DETR-inspired set prediction to the dynamic discovery and encoding of propositions.

3 Methodology

The goal of APE is to map a sentence to a variable-size set of proposition embeddings. Figure 2 illustrates the full architecture and training objective.

3.1 APE Architecture

APE consists of a pretrained transformer encoder and a slot-based transformer decoder. Given an input sentence, the backbone encoder produces a sequence of contextualized token representations $\mathbf{H} \in \mathbb{R}^{L \times D}$, where L is the sequence length and D is the hidden dimension.

A set of N learnable slot queries $\mathbf{Q} \in \mathbb{R}^{N \times D}$ attend over $\mathbf{H}$ in the transformer decoder via cross-attention. The decoder is a stack of M pre-norm Transformer decoder layers (Vaswani et al., 2017).

Slot representations are initialized as $\mathbf{Z}^{(0)} = \mathbf{Q}$, and updated at each layer $l = 1, \dots, M$ by

$$\begin{aligned}\hat{\mathbf{Z}}^{(l)} &= \mathbf{Z}^{(l-1)} + \text{MHSA} \left(\text{LN}(\mathbf{Z}^{(l-1)}) \right), \\ \tilde{\mathbf{Z}}^{(l)} &= \hat{\mathbf{Z}}^{(l)} + \text{MHCA} \left(\text{LN}(\hat{\mathbf{Z}}^{(l)}), \mathbf{H}, \mathbf{H} \right), \\ \mathbf{Z}^{(l)} &= \tilde{\mathbf{Z}}^{(l)} + \text{FFN} \left(\text{LN}(\tilde{\mathbf{Z}}^{(l)}) \right),\end{aligned}$$

where MHSA is multi-head self-attention among the N slots and MHCA is multi-head cross-attention with the slot states as queries and $\mathbf{H}$ as keys/values. The final slot representations are $\mathbf{Z} = \text{LN}(\mathbf{Z}^{(M)})$.

Then, each decoded slot representation is projected through two heads: an *embedding* head $f_{\text{emb}} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ that produces a candidate proposition embedding $\mathbf{e}_i \in \mathbb{R}^D$, and an *activity* head $f_{\text{act}} : \mathbb{R}^D \rightarrow \mathbb{R}$ that produces a logit $a_i \in \mathbb{R}$ to measure how active slot i is. The activity logits $\mathbf{a} = (a_1, \dots, a_N)$ thus determine the activity level of candidate embeddings $\mathbf{E} = (\mathbf{e}_1, \dots, \mathbf{e}_N)$.

At inference time, slot i is considered active and embedding $\mathbf{e}_i$ is returned if $\sigma(a_i) > \tau$, where τ is the activity threshold. The $\hat{N} \leq N$ active embeddings $\hat{\mathbf{E}}$, where $\hat{\mathbf{E}} = \{\mathbf{e}_i : \sigma(a_i) > \tau\}$, constitute the bag of proposition embeddings for the input sentence. This is the mechanism that gives APE its variable-size output.

3.2 Training objective

APE is trained to decompose a sentence into atomic proposition embeddings by supervision from a dataset of (sentence, $\{\text{prop}_1, \dots, \text{prop}_K\}$) pairs (Section 3.3). The training objective encourages each active slot to converge toward a distinct proposition, while inactive slots are suppressed.

Training proceeds via knowledge distillation from a frozen copy of the backbone encoder, which serves as the *teacher*. Each ground-truth proposition is encoded independently using mean pooling over its token representations, producing a set of K target embeddings $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_K\}$, where $K \leq N$ varies per sentence. The objective combines three losses.

Embedding loss $\mathcal{L}_{\text{emb}}$ Slot embeddings are matched to target embeddings via the Hungarian algorithm (Kuhn, 1955) on the cosine distance matrix, yielding optimal assignment π^* of slots to targets. The embedding loss is the mean cosine

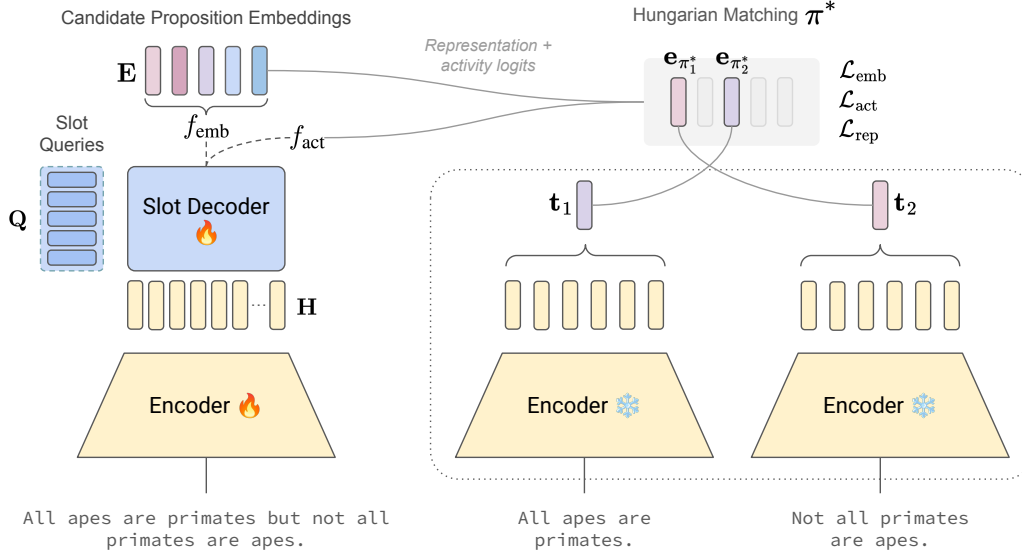


Figure 2: APE architecture and training objective.

distance over matched pairs:

$$\mathcal{L}_{\text{emb}} = \frac{1}{K} \sum_{k=1}^K \left(1 - \frac{\mathbf{e}_{\pi_k^*} \cdot \mathbf{t}_k}{\|\mathbf{e}_{\pi_k^*}\| \|\mathbf{t}_k\|} \right),$$

where $\mathbf{e}_{\pi_k^*}$ is the embedding assigned to target $\mathbf{t}_k$.

Activity loss $\mathcal{L}_{\text{act}}$ A binary cross-entropy loss supervises the activity head, with matched slots assigned a target of 1 and unmatched slots a target of 0:

$$\mathcal{L}_{\text{act}} = -\frac{1}{N} \sum_{i=1}^N (y_i \log \sigma(a_i) + (1 - y_i) \log(1 - \sigma(a_i)))$$

where $y_i = 1$ if slot i is matched by π^* , and $y_i = 0$ otherwise.

Slot repulsion loss $\mathcal{L}_{\text{rep}}$ To discourage active slots from collapsing to similar representations and to provide a clearer separation between active and inactive slots, we penalize cosine similarity between pairs where at least one slot is active:

$$\mathcal{L}_{\text{rep}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \max \left(0, \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|} \right),$$

with $\mathcal{P} = \{(i, j) : i < j, \sigma(a_i) > \tau \vee \sigma(a_j) > \tau\}$.

Then, the total APE loss is:

$$\mathcal{L}_{\text{APE}} = \lambda_{\text{emb}} \mathcal{L}_{\text{emb}} + \lambda_{\text{act}} \mathcal{L}_{\text{act}} + \lambda_{\text{rep}} \mathcal{L}_{\text{rep}}$$

3.3 Training data

APE is trained on a proposition-decomposed version of WikiSplit (Botha et al., 2018), where each sentence has been decomposed into atomic, coreference-resolved propositions using gpt-oss-120b (OpenAI, 2025), discarding connectives and discourse markers (Appendix A.1). While more fine-grained variants of WikiSplit exist (Tsukagoshi et al., 2024), their splits are not self-contained; we instead prompt the LLM to produce independent, coreference-resolved propositions. We filter the resulting dataset with an NLI cross-encoder (nli-deberta-v3-large; Reimers and Gurevych, 2019; He et al., 2021), retaining only examples where all propositions are entailed by the original sentence; this removes 10.34% of examples, leaving 890,584 multi-proposition pairs. We further augment the dataset with singleton examples to ensure robustness to single-proposition sentences, yielding 910,584 training examples in total. We keep a held-out set of 5,000 pairs for post-training activities. More details about training set creation and validation can be found in Appendix A.1.

3.4 Inference and Scoring

Given two sentences with m and n active proposition embeddings denoted, respectively, as $\hat{\mathbf{E}}^{(1)}$ and $\hat{\mathbf{E}}^{(2)}$, APE computes a scalar similarity score by aggregating over the $m \times n$ pairwise cosine similarity matrix. We define three scoring rules based on the cosine similarity between two proposition embeddings $s(\mathbf{e}, \mathbf{e}') = \frac{\mathbf{e} \cdot \mathbf{e}'}{\|\mathbf{e}\| \|\mathbf{e}'\|}$.

Hungarian Average (HA) Assume without loss of generality that $m \leq n$. HA first applies the Hungarian algorithm to find the optimal one-to-one assignment $\pi^* = \arg \max_{\pi} \sum_i^m s(\mathbf{e}_i^{(1)}, \mathbf{e}_{\pi(i)}^{(2)})$ from $\hat{\mathbf{E}}^{(1)}$ into $\hat{\mathbf{E}}^{(2)}$, matching each of the m elements of $\hat{\mathbf{E}}^{(1)}$ to a distinct element of $\hat{\mathbf{E}}^{(2)}$. HA then averages the similarities of the m matched pairs, leaving the $m - n$ unmatched elements of $\hat{\mathbf{E}}^{(2)}$ unscored:

$$\text{HA} = \frac{\sum_{i=1}^m s(\mathbf{e}_i^{(1)}, \mathbf{e}_{\pi^*(i)}^{(2)})}{m}$$

HA is symmetric and suited for tasks where both sentences are peers.

Maximum Similarity (MaxSim) Adapted from the MaxSim operator in ColBERT (Khattab and Zaharia, 2020) to proposition-level matching, MaxSim averages both directions of best-match similarities:

$$\text{MaxSim} = \frac{1}{2} \left(\frac{1}{m} \sum_{i=1}^m \max_j s_{ij} + \frac{1}{n} \sum_{j=1}^n \max_i s_{ij} \right),$$

where $s_{ij} = s(\mathbf{e}_i^{(1)}, \mathbf{e}_j^{(2)})$. MaxSim is symmetric but does not enforce one-to-one assignments, making it less strict than HA.

Hypothesis-to-Premise (H2P) An asymmetric scoring rule that returns the worst best-match of hypothesis propositions against the premise:

$$\text{H2P}(\hat{\mathbf{E}}^{(1)}, \hat{\mathbf{E}}^{(2)}) = \min_{j \in [n]} \max_{i \in [m]} s(\mathbf{e}_i^{(1)}, \mathbf{e}_j^{(2)})$$

where $\hat{\mathbf{E}}^{(1)}$ is the premise and $\hat{\mathbf{E}}^{(2)}$ the hypothesis (Tang et al., 2024; Huang, 2025). The score is low if any hypothesis proposition lacks a counterpart in the premise, making H2P sensitive to propositional coverage and suited for entailment detection and proposition-level retrieval.

4 PropSearch

To assess the quality of proposition embeddings produced by decomposition models, we introduce PROPSEARCH: given a set of atomic propositions sampled from source sentences, the task is to retrieve the originating sentence for each proposition. Unlike standard sentence retrieval benchmarks, this directly evaluates whether a model captures propositional content at the atomic level.

PROPSEARCH comprises 13,020 source sentences with 36,337 atomic query propositions drawn from scientific abstracts (Clement et al., 2019) to test out-of-distribution generalization. Sentences vary in propositional complexity to enable fine-grained analysis as sentence complexity increases. Propositions are extracted using Claude Sonnet 4.6 (Anthropic, 2026), deliberately distinct from the training annotation pipeline to avoid annotation bias. Moreover, 300,000 sentences are added as confounds to the corpus. Such sentences do not have any corresponding query propositions. Details about the benchmark, including the full validation pipeline, can be found in Appendix A.2.

5 Experimental Design

5.1 Setup

For the main implementation of APE, we use $N = 8$ slots, which exceeds the maximum number of propositions per sentence seen during training, providing slack capacity for optimization. As the backbone, we choose BGE-small-en-v1.5 (Xiao et al., 2024), which achieves competitive sentence embedding quality on MTEB (Muennighoff et al., 2023) at only 33M parameters. Although BGE models are optimized for CLS pooling, we find that mean pooling yields better results, consistent with slot attention’s use of weighted averages over token embeddings. We experiment with alternative encoders in Appendix A.3. APE comprises 3 decoder layers with 8 attention heads each, adding around 7M parameters on top of the backbone for a total of 40M. We limit training to sentences with at most 5 propositions (83% of the training data), and experiment with scaling this number in Appendix A.4. Full hyperparameters can be found in Appendix A.5. For the loss, we set $\lambda_{\text{emb}} = \lambda_{\text{act}} = 1$ and $\lambda_{\text{rep}} = 0.1$.

5.2 Evaluation

We evaluate APE zero-shot against several baselines and across tasks to assess proposition embedding quality.

Baselines We compare APE against sentence embedding models and Sub-Sentence Encoder (SSE) variants¹ (Table 1): BGE models of all sizes (Xiao et al., 2024), GTE-large (Li et al., 2023), EmbeddingGemma (Vera et al., 2025), and Qwen3-

¹As anticipated in Section 2, SSE relies on an external T5-large generative model (770M parameters) to extract propositions to map to token spans.

Type	Model	Size	Citation
Prop	SSE-T5	879M	Chen et al. (2024)
	SSE-GTR	879M	
Sent	BGE-small	33M	Xiao et al. (2024)
	BGE-base	109M	
	BGE-large	335M	
	GTE-large	335M	Li et al. (2023)
Inst	Embed-Gemma	308M	Vera et al. (2025)
	Qwen3-Embed	596M	Zhang et al. (2025)

Table 1: Variable-size (Prop), fixed-size (Sent) and instruction-tuned fixed-size (Inst) baselines implemented along with their parameter counts.

Embedding (Zhang et al., 2025). The latter two are instruction-tuned, producing task-dependent representations; we evaluate them with task-specific instructions (Appendix A.6), which makes them a distinct comparison class from APE and other models that use a single representation² across tasks. We report BGE-small-en-v1.5 results under both mean pooling (BGE-small-mean), as it serves as APE’s training teacher, and CLS pooling (BGE-small-cls), as recommended by the authors.

To isolate the architectural contribution from the training data, we fine-tune BGE-small-en-v1.5 on the APE training data by minimizing the cosine distance between the sentence embedding and the mean of its ground-truth proposition embeddings (BGE-small-prop). A strong BGE-small-prop would suggest the gains are data-driven rather than architectural. To further support the architectural claim, we also train APE solely on SSE data (proposition-decomposed, like APE training data) and report the results in Appendix A.7.

Moreover, since APE relies on a discrete pipeline for training data generation (namely, LLM-based proposition extraction followed by independent encoding of each proposition using a pretrained sentence transformer) we also measure this pipeline’s own performance on the benchmarks. Improved APE performance relative to the discrete pipeline would provide strong evidence that, despite being orders of magnitude smaller, APE can correct biases present in its training signal. This pipeline is marked as APE-discrete.

Downstream benchmarks We evaluate APE zero-shot on two benchmarks targeting different aspects of propositional understanding.

PAWS-WIKI (Zhang et al., 2019) tests paraphrase detection on adversarially constructed pairs with high lexical overlap but distinct propositional content. This is a notoriously difficult setting for fixed-size encoders, whose collapsed representations often fail to distinguish beyond surface form. Therefore, it is particularly suited to measure whether a model’s embeddings capture subtle differences at the proposition level. Being Wikipedia-derived, this benchmark is in-distribution for APE.

SNLI (Bowman et al., 2015) tests entailment detection as binary classification, which directly probes the asymmetric H2P scoring rule: a hypothesis is entailed by a premise if and only if all its propositions are covered, which H2P is designed to capture. SNLI is out-of-distribution for APE.

Fixed-size encoders are evaluated by ranking pairs by cosine similarity, from which AUC and Average Precision (AP) are computed against binary ground-truth labels. APE and SSE are evaluated using the proposition-level scoring rules from Section 3.4; we report all three rules for APE and the best-performing rule per benchmark for SSE. Additionally, we report multi-atom results, i.e., scores restricted to pairs where APE identified multiple propositions in at least one sentence.

Diagnostic results on standard semantic textual similarity (STS) benchmarks from MTEB (Muenighoff et al., 2023) are reported in Appendix A.8 as evidence of no catastrophic forgetting compared to the teacher encoder.

PropSearch We evaluate APE’s and SSE’s decomposition quality on PROPSEARCH (Section 4), reporting Recall@1 and MRR stratified by source sentence proposition count ($n_{\text{atoms}} \in \{2, 3, 4, 5, 6+\}$). The $n_{\text{atoms}} \geq 6$ stratum tests out-of-distribution generalization beyond APE’s training bound of 5 propositions. The benchmark is also out-of-distribution in register, covering scientific abstracts versus APE’s Wikipedia-derived training data. We use H2P scoring for retrieval, and include fixed-size models as a control group under standard retrieval. While not specifically instruction-tuned, BGE models do prepend a "query:" prefix in retrieval tasks. Therefore, we apply that prefix in their PROPSEARCH evaluation.

Cardinality We evaluate whether APE and SSE recover the correct proposition count per sentence, as defined by PROPSEARCH ground-truth. APE infers cardinality implicitly from active slots, while SSE relies on an external segmenter, making this a

²Or at most two, as some models (including BGE) prepend a "query:" prefix for better results in retrieval-based tasks.

Model	AUC	AP
APE (HA)	81.1 (82.1)	77.1 (78.6)
APE (MaxSim)	80.1 (81.0)	76.4 (78.0)
APE (H2P)	77.3 (78.4)	73.1 (75.2)
APE-discrete (HA)	<u>77.8 (78.0)</u>	<u>74.5 (74.7)</u>
SSE-T5 (HA)	76.0 (76.4)	73.4 (73.9)
SSE-GTR (HA)	75.8 (76.2)	72.9 (73.5)
BGE-small-mean	60.8 (60.6)	56.3 (56.1)
BGE-small-prop	64.4 (64.4)	59.1 (59.1)
BGE-small-cls	60.5 (60.4)	56.0 (55.8)
BGE-base-cls	61.8 (61.7)	57.7 (57.6)
BGE-large-cls	64.2 (63.9)	60.6 (60.4)
GTE-large	64.5 (64.3)	59.2 (58.9)
Embed-Gemma	66.1 (66.0)	62.9 (62.9)
Qwen3-Embed	74.0 (73.9)	67.0 (66.7)

Table 2: Results on the PAWS-WIKI benchmark. Numbers in parentheses refer to the multi-atom subset.

direct comparison of learned versus pipeline-based decomposition. Since atomic decomposition is not uniquely defined, exact cardinality should be interpreted cautiously, as many disagreements reflect near-equivalent decompositions differing only in granularity; we therefore report both exact and ± 1 accuracy. Additionally, we report cardinality results on the held-out set (split into calibration and evaluation only for this task).

All APE results are obtained with an activity threshold of 0.6, the optimal value on the held-out set.

6 Results

We present results on the downstream benchmarks and PROPSEARCH, followed by an analysis of slot activity prediction and slot interpretability. A qualitative analysis of APE’s behavior in presence of diverse linguistic phenomena and common failure cases can be found in Appendix A.9.

PAWS-Wiki Results are shown in Table 2. APE (HA) outperforms all baselines by a large margin: +20.3 AUC and +20.8 AP over its teacher backbone (BGE-small-mean), +7.1 AUC over the best fixed-size baseline (Qwen3-Embed), and +5 AUC over SSE models. The large gains suggest that APE is correctly extracting and representing propositions, and detecting differences between the propositional content of sentence pairs. Most fixed-size encoders assign uniformly high similarity scores, often reducing the task to near-random ranking (Zhang et al., 2019). The strong performance of APE-discrete and SSE confirms that proposition

Model	AUC	AP
APE (HA)	82.9 (83.6)	68.9 (68.7)
APE (MaxSim)	81.1 (81.2)	65.1 (63.3)
APE (H2P)	<u>84.1 (85.2)</u>	<u>72.3 (73.6)</u>
APE-discrete (H2P)	83.3 (83.2)	74.4 (74.2)
SSE-T5 (H2P)	80.7 (81.1)	69.1 (69.2)
SSE-GTR (H2P)	76.9 (76.9)	65.4 (64.8)
BGE-small-mean	81.0 (81.1)	65.2 (63.8)
BGE-small-cls	81.1 (81.4)	65.3 (64.1)
BGE-small-prop	81.4 (81.4)	65.8 (64.3)
BGE-base-cls	81.4 (81.8)	65.6 (64.4)
BGE-large-cls	82.0 (82.3)	66.6 (65.4)
GTE-large	80.6 (80.7)	64.8 (63.5)
Embed-Gemma	84.2 (84.3)	68.5 (67.0)
Qwen3-Embed	81.2 (81.3)	65.3 (64.0)

Table 3: Results on the SNLI benchmark. Numbers in parentheses refer to the multi-atom subset.

decomposition directly addresses this failure mode. On the multi-atom subset (sentences deemed propositionally complex), APE improves while all fixed-size baselines degrade, widening the gap further.

BGE-small-prop, the fixed-size encoder fine-tuned on APE’s training data, improves by only 3.6 AUC over its pretrained counterpart (BGE-small-mean), confirming that the gains are primarily architectural rather than a consequence of training distribution (see also Appendix A.7). The improved performance of APE compared to its discrete counterpart (despite relying on only 0.03% of the parameters) suggests that the model can generalize and learn beyond the biases of its teacher pipeline.

Among APE’s scoring rules, HA and MaxSim perform comparably and both outperform H2P, consistent with PAWS-WIKI being a symmetric task where H2P’s asymmetric penalty is inappropriate. The two rules produce identical scores when APE predicts equal slot counts for both sentences, which is common given the benchmark’s uniform syntactic structure.

SNLI Results are shown in Table 3. Fixed-size sentence encoders perform substantially better on SNLI than on PAWS-WIKI, as entailment detection is a more natural test of sentence-level similarity and does not rely on adversarial surface-form manipulation. Despite this, APE (H2P) still achieves +3.1 AUC and +7.1 AP over its teacher backbone. Compared to the best instruction-tuned fixed-size baseline (EmbeddingGemma), APE is almost on par (-0.1 AUC) with its overall performance while scoring +3.8 AP, and outperforms it

Model	Tot	2	3	4	5	6+
APE (H2P)	96.0	98.5	97.7	96.9	95.3	91.4
SSE-T5 (H2P)	67.7	80.4	72.6	68.4	61.9	55.4
SSE-GTR (H2P)	73.3	85.0	78.8	74.9	67.8	62.2
BGE-small-mean	88.4	96.9	92.2	89.1	84.8	78.7
BGE-small-cls	90.5	97.2	93.4	91.1	87.5	83.4
BGE-small-prop	91.4	98.3	94.7	92.1	88.7	83.5
BGE-base-cls	92.6	98.3	95.3	93.3	90.3	85.9
BGE-large-cls	92.6	98.2	95.1	93.0	90.2	86.2
GTE-large	89.7	97.2	93.4	90.6	86.2	81.0
Embed-Gemma	<u>95.8</u>	99.4	97.7	<u>96.4</u>	<u>94.7</u>	<u>90.9</u>
Qwen3-Embed	92.0	97.9	94.7	92.6	89.7	85.1

Table 4: Results on the PROPSEARCH benchmark (MRR overall and for different complexity levels). The *Tot* metric represents the macro average across classes.

on the multi-atom subset, scoring +6.6 AP. Compared to the best SSE model, APE scores +3.4 AUC and +3.2 AP overall. H2P outperforms HA by a significant margin, which confirms its fit for entailment detection.

While APE leads on AUC over its discrete counterpart, it achieves slightly lower AP. Compared to PAWS-WIKI, SNLI texts tend to be shorter and overall simpler, meaning that the discrete pipeline has a lower chance of incorrectly splitting the text and induce a chain of errors. Indeed, as complexity increases (i.e. on the multi-atom subset), APE performance improves across both AUC and AP, while APE-discrete shows the opposite behavior.

PropSearch Table 4 reports MRR on PROPSEARCH stratified by propositional complexity. APE achieves the highest MRR overall, with EmbeddingGemma a close second. At the lowest complexity level (2 propositions), EmbeddingGemma slightly outperforms APE by 0.9 MRR points. However, as complexity increases (≥ 3 propositions), APE maintains a consistent lead. Notably, APE outperforms its teacher encoder by up to 12.7 MRR points.

SSE models show particularly steep degradation with complexity, which might be attributed to their decoupled discovery-representation pipeline. Indeed, span boundaries become less reliable as sentences grow more complex, and errors propagate into the representation with no mechanism for recovery. This is confirmed by inspecting the 1-atom stratum, which is trivial for fixed-size encoders, since decomposition is the only possible source of error. Here, SSE-GTR only achieves 95.9% Precision@1, well below APE’s 99.8%.

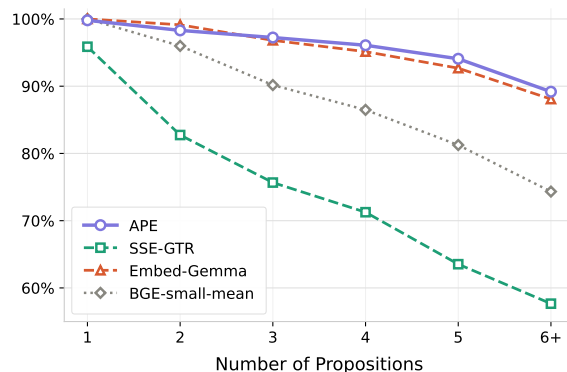


Figure 3: Performance on PropSearch by sentence complexity measured by Precision@1.

Figure 3 shows the full Precision@1 trends for four representative models: APE, SSE-GTR (best-performing SSE model), EmbeddingGemma (best-performing fixed-size encoder), and BGE-small-mean (APE’s teacher encoder). Most notable is the $n_{\text{atoms}} \geq 6$ stratum (5,369 query propositions), which lies outside APE’s training distribution: APE achieves the highest Precision@1 at 89.2%, ahead of EmbeddingGemma (88.1%). Despite never training on sentences with more than 5 propositions (and never producing more than 5 active atomic embeddings), APE’s slot mechanism generalizes gracefully beyond its training.

6.1 Efficiency

APE achieves $120\times$ higher throughput and $19\times$ lower memory footprint than the SSE pipeline (segmenter + encoder), while producing comparable proposition counts (1.73 vs 1.94 per sentence on 1,000 SNLI sentences). Over 93% of SSE’s latency comes from the external segmentation stage alone. This confirms that decoupling proposition discovery from representation is the primary computational bottleneck.

6.2 Slot Activity and Cardinality Prediction

We now focus on APE’s ability to scale its representation with input complexity. Table 5 reports the performance on proposition cardinality prediction. APE generally outperforms the SSE segmenter by a significant margin, with only a few exceptions. Most importantly, while the SSE segmenter seems to be biased toward predicting 2-3 propositions per sentence, APE remains more balanced. We empirically find that, when APE overpredicts cardinality, the additional active slots fall close to other active embeddings in the representation space. Merging

Held-out Set (Eval)			PROPSEARCH	
K	APE	SSE	APE	SSE
1	88.6 (97.3)	34.8 (92.6)	36.4 (76.5)	10.5 (62.5)
2	74.3 (94.3)	68.6 (95.0)	53.9 (89.8)	46.0 (86.5)
3	62.5 (92.4)	65.1 (96.7)	55.3 (92.3)	54.0 (94.5)
4	55.2 (97.5)	51.1 (91.4)	44.9 (91.7)	36.0 (83.2)
5	67.9 (94.1)	26.8 (73.8)	50.4 (82.4)	19.2 (55.9)

Table 5: Cardinality prediction on the held-out set and PROPSEARCH. Numbers in parentheses indicate $\pm 1\%$.

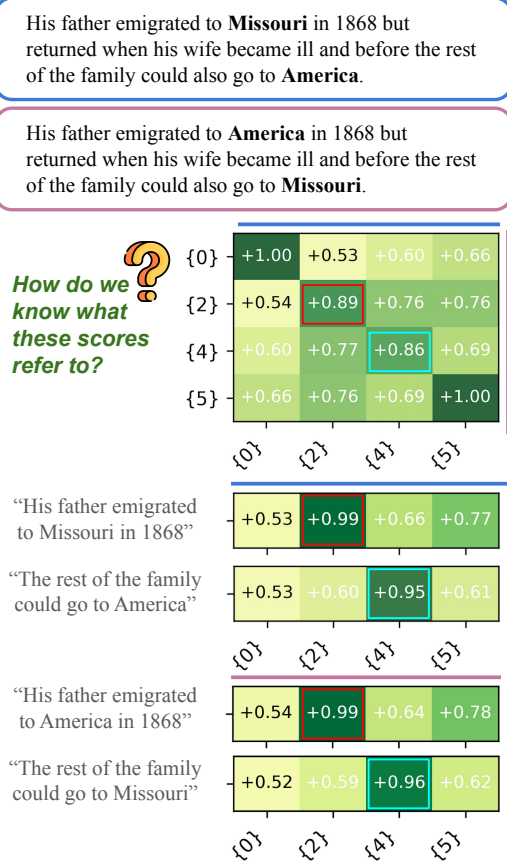


Figure 4: Slot interpretation via proposition seeding.

slots over a threshold is something we will investigate in future work.

6.3 Slot Interpretability

Since APE’s proposition discovery and representation are fully latent, we interpret slots via *proposition seeding*. Given a seed proposition p and a sentence x , APE encodes both: the seed produces a single active embedding e^p , and the sentence produces a bag $\hat{E} = \{e_1, \dots, e_N\}$. The slot most aligned with the seed is $\arg \max_i s(e_i, e^p)$. Figure 4 illustrates this on an adversarial PAWS-WIKI pair. Both sentences are encoded into four active slots each, yielding a 4×4 similarity matrix. Without seeding, slot content is opaque; with atomic propo-

sition seeds, assignments can be interpreted. Slots $\{0\}$ and $\{5\}$ show perfect self-similarity across the swapped pair, while slots $\{2\}$ and $\{4\}$ differ. Seeding reveals that the latter correspond exactly to the propositions with swapped locations (*Missouri* vs *America*). The BGE-small backbone assigns 0.996 cosine similarity to the same pair, collapsing this propositional distinction into a single score.

7 Conclusion

We introduced Atomic Proposition Embeddings (APE), a novel embedding paradigm that represents sentences as variable-size bags of proposition embeddings by applying a slot attention mechanism over a pretrained transformer backbone inspired by object detection in computer vision. Unlike fixed-size sentence embeddings, APE produces a number of vectors that scales with the propositional complexity of the input, with each vector corresponding to an atomic proposition discovered and encoded end-to-end. Such propositions are represented as self-contained units of meaning, independent from other propositions in the sentence. APE matches or outperforms much larger sentence encoders on PAWS-WIKI and SNLI with no task-specific fine-tuning. Notably, APE beats the very discrete pipeline (LLM-based proposition extraction and encoding via a pretrained sentence transformer) it learns from on several metrics, which suggests it is able to generalize beyond its training. APE also leads on PROPSEARCH, a new retrieval benchmark we introduce for evaluating proposition decomposition and representation models. Compared to existing proposition-level embedding methods, APE achieves consistently higher performance at a fraction of the parameter count and inference time.

On sentences with more than 5 propositions, i.e., beyond APE’s training bound, the slot mechanism degrades gracefully, forming a coarser representation that still outperforms other proposition-level embedding models and fixed-size baselines on retrieval. This suggests that slots capture structural properties of propositions that generalize beyond the training cardinality.

Finally, the proposition seeding technique introduced in Section 6.3 provides an interface for inspecting what individual slots encode and opens a path toward interpretable sentence representations.

Limitations

APE represents sentences as bags of proposition embeddings, deliberately discarding the relational structure that connects propositions within a sentence. That is, causal, temporal and discourse relations in general are not encoded. This makes APE a partial rather than a general-purpose sentence representation model, and tasks that require reasoning over inter-propositional relations are outside of its current scope. Even if there is no evidence of severe catastrophic forgetting, APE still slightly under-performs its teacher in such tasks.

Training APE requires proposition-decomposed supervision, which must be synthetically generated to obtain in sufficient quantity. While this pipeline is reproducible and relatively inexpensive, it introduces a dependency on the quality and consistency of LLM-generated annotations, and the resulting training data may not extensively cover all linguistic phenomena. Furthermore, APE is trained via knowledge distillation from a frozen sentence encoder, meaning the quality of its proposition embeddings is bounded by the representational capacity of the teacher. While we show that APE can achieve higher performance compared to the discrete pipeline it learns from, which mitigates the limitation, we have not investigated this further at this time.

Finally, APE is trained and evaluated exclusively on individual sentences. Whether the slot mechanism extends as-is to longer texts such as paragraphs or documents, where proposition counts are considerably larger, remains an open question that we will investigate in the future.

Acknowledgments

We acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy).

References

- Anthropic. 2026. Claude sonnet 4.6. <https://www.anthropic.com/news/claude-sonnet-4-6>. Accessed: 2026-05-24.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguis-*

tic annotation workshop and interoperability with discourse, pages 178–186.

- Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. [Learning to split and rephrase from Wikipedia edit history](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 732–737, Brussels, Belgium. Association for Computational Linguistics.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. [End-to-end object detection with transformers](#). In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg. Springer-Verlag.
- Hongyi Cen, Mingxin Wang, Yule Liu, Jingyi Zheng, Hanze Jia, Tan Tang, and Yingcai Wu. 2026. Atom-eval: Atomic evaluation of adversarial claims in fact verification. *arXiv preprint arXiv:2604.07967*.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. [Universal sentence encoder for English](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.
- Sihao Chen, Hongming Zhang, Tong Chen, Ben Zhou, Wenhao Yu, Dian Yu, Baolin Peng, Hongwei Wang, Dan Roth, and Dong Yu. 2024. [Sub-sentence encoder: Contrastive learning of propositional semantic representations](#). 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 1596–1609, Mexico City, Mexico. Association for Computational Linguistics.
- Colin B. Clement, Matthew Bierbaum, Kevin P. O’Keeffe, and Alexander A. Alemi. 2019. [On the use of arxiv as a dataset](#). *Preprint*, arXiv:1905.00075.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. [Supervised learning of universal sentence representations from natural language inference data](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

- Markus Frohmann, Igor Sterner, Ivan Vulić, Benjamin Minixhofer, and Markus Schedl. 2024. [Segment any text: A universal approach for robust, efficient and adaptable sentence segmentation](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11908–11941, Miami, Florida, USA. Association for Computational Linguistics.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [SimCSE: Simple contrastive learning of sentence embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.
- Minghui Huang. 2025. Decmetrics: Structured claim decomposition scoring for factually consistent llm outputs. *arXiv preprint arXiv:2509.04483*.
- Rohan Jha, Bo Wang, Michael Günther, Georgios Mastrotras, Saba Sturua, Isabelle Mohr, Andreas Koukounas, Mohammad Kalim Wang, Nan Wang, and Han Xiao. 2024. Jina-colbert-v2: A general-purpose multilingual late interaction retriever. In *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, pages 159–166.
- Omar Khattab and Matei Zaharia. 2020. [Colbert: Efficient and effective passage search via contextualized late interaction over bert](#). In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20*, page 39–48, New York, NY, USA. Association for Computing Machinery.
- Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. [Skip-thought vectors](#). In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.
- Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.
- Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. 2020. Object-centric learning with slot attention. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA. Curran Associates Inc.
- Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. [FActScore: Fine-grained atomic evaluation of factual precision in long form text generation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). Preprint, arXiv:2508.10925.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. [The Proposition Bank: An annotated corpus of semantic roles](#). *Computational Linguistics*, 31(1):71–106.
- Martha Palmer, Daniel Gildea, and Nianwen Xue. 2011. *Semantic role labeling*. Morgan & Claypool San Rafael.
- 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.
- Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. [Skip-prop: Representing sentences with one vector per proposition](#). In *Proceedings of the 12th International Conference on Computational Semantics (IWCS) — Short papers*.
- Alessandro Scirè, Andrei Stefan Bejgu, Simone Tedeschi, Karim Ghonim, Federico Martelli, and Roberto Navigli. 2026. [Truth or mirage? Towards end-to-end factuality evaluation with LLM-Oasis](#). *Computational Linguistics*, 52(1):1–41.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA. Curran Associates Inc.
- Liyan Tang, Philippe Laban, and Greg Durrett. 2024. [MiniCheck: Efficient fact-checking of LLMs on grounding documents](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8818–8847, Miami, Florida, USA. Association for Computational Linguistics.
- Hayato Tsukagoshi, Tsutomu Hirao, Makoto Morishita, Katsuki Chousa, Ryohei Sasano, and Koichi Takeda. 2024. [WikiSplit++: Easy data refinement for split](#)

and rephrase. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17625–17636, Torino, Italia.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panayam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, and 1 others. 2025. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxiang Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. [C-pack: Packed resources for general chinese embeddings](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’24*, page 641–649, New York, NY, USA. Association for Computing Machinery.

Liyan Xu, Zhenlin Su, Mo Yu, Jiangnan Li, Fandong Meng, and Jie Zhou. 2025. [Dense retrievers can fail on simple queries: Revealing the granularity dilemma of embeddings](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19295–19305, Suzhou, China. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. [PAWS: Paraphrase adversaries from word scrambling](#). 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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Liwen Zheng, Chaozhuo Li, Zheng Liu, Feiran Huang, Haoran Jia, Zaisheng Ye, and Xi Zhang. 2025. Fact in fragments: Deconstructing complex claims via llm-based atomic fact extraction and verification. *Expert Systems with Applications*, 302:130572.

A Appendix

A.1 Training Data

In this section, we go into further details of the training data generation process, namely data sourcing, proposition extraction and validation.

Generation prompt The prompt we give gpt-oss-120b (OpenAI, 2025) to generate the training set is:

Extract atomic propositions from a given sentence. Such propositions need to be self-contained and independent. That is, they must not contain implicit or explicit references to other propositions.

Return the output as a Python list.
Sentence: {sentence}

Singletons Since the vast majority of WikiSplit sentences (Botha et al., 2018) are split into at least two propositions (WikiSplit, by design, mostly contains sentences with complex propositional structure), we augment the dataset by generating (atom, paraphrased_atom) pairs using the same LLM.³ We generate over 100,000 of such pairs. However, training ablations show that 20,000 singletons was enough training signal for the model to detect single-proposition sentences.

Validation As an automatic data validation step, we run an NLI model, nli-deberta-v3-large (Reimers and Gurevych, 2019; He et al., 2021), to estimate entailment between each atom and its source sentence. Out of 993,285 non-singleton examples and 4,164,437 (atom, sentence) pairs, we estimate 134,280 non-entailed pairs (3.2% of total pairs) spread across 102,701 sentences (10.34% of total sentences). We filter out sentences that contain at least one non-entailed proposition from the training set, which leaves us with 890,584 clean, non-singleton examples. Figure 5 shows the full training data distribution.

Additionally, we manually validate a random sample of 200 training instances as a general assessment of the training set quality. We define two validation criteria:

- **Atomicity**, to check whether all extracted propositions can be considered atomic.

³Technically, we could have simply used (atom, atom) identity pairs, but we chose to include paraphrases to create more challenging training examples.

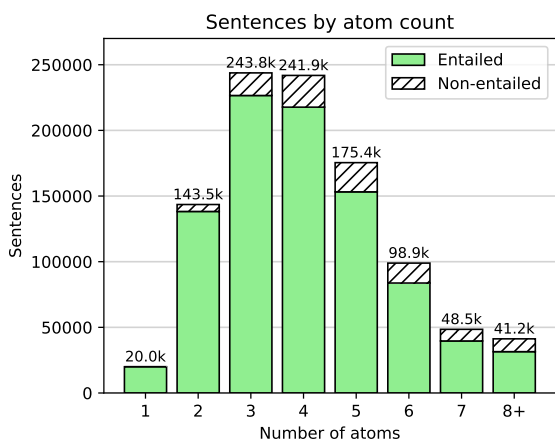


Figure 5: Sentences by atom count in APE training data. The patterned portions of the bars indicate training pairs in which `nli-deberta-v3-large` predicts that at least one atom is not entailed by the source sentence.

- **Coverage**, to check whether the whole propositional content of the source sentence is covered by the extracted propositions.

We find that, in the extracted sample, 93% of the examples satisfy the atomicity criterion, and 94% of the examples satisfy the coverage criterion.

A.2 PROPSEARCH

Here, we detail the process of the PROPSEARCH benchmark creation, including data sourcing, pre-processing, generation and validation. Figure 6 shows the retrieval task from PROPSEARCH. Given an atomic proposition, a model must retrieve the original sentence from which that proposition has been extracted.

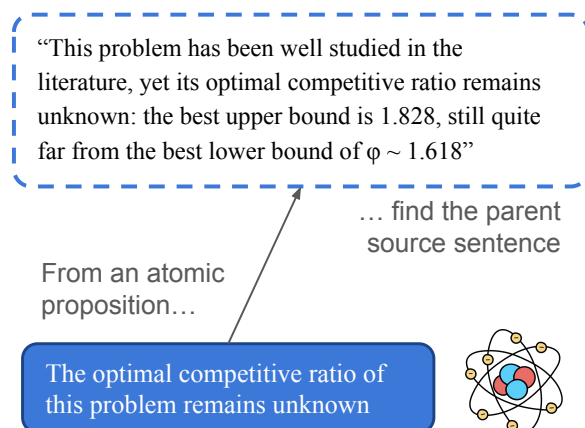


Figure 6: Illustrative example from PROPSEARCH.

Below, we describe the data collection and generation process, followed by the automatic and manual validation procedure.

Dataset creation First, we source texts from scientific abstracts (Clement et al., 2019). In particular, we randomly sample 59,985 abstracts (3% of the original 2M dataset). Then, we use segment-any-text (SaT) models (Frohmann et al., 2024), specifically `sat-3l-sm`, to segment every abstract into a list of sentences. At this point, we exclude sentences shorter than 15 characters or longer than 200 characters and randomly sample 15,000 sentences. Finally, we use Claude Sonnet 4.6 (Anthropic, 2026) to extract propositions from the given sentences. The system prompt we use is:

Extract atomic propositions from the given sentence. The propositions need to be independent (for instance, no pronouns that refer to entities outside the proposition itself) and connectives between propositions need to be discarded.

If the sentence only has one atomic proposition, return a singleton.

Return the output as a Python list.

This leaves us with 14,867 sentences and 42,971 atoms, which is the starting point of PROPSEARCH prior to validation.

To make the task more challenging, we augment the corpus with 300,000 confounding sentences from the *papers with abstracts* dataset,⁴ extracted using the same SaT model.

Benchmark validation Beyond the quality of LLM-based decomposition, we have no guarantee that one query proposition can be found in *exactly* one sentence in the augmented corpus. Therefore, we first extract, for each query proposition, the top 50 similar sentences using EmbeddingGemma (Vera et al., 2025). Then, we run an NLI cross-encoder, `nli-deberta-v3-large`, on each possible (query_proposition, candidate_sentence) pair for two distinct purposes:

- Validation of LLM-based decomposition. We remove every proposition that is not entailed by the parent sentence. However, we retain the parent sentence along with the entailed propositions keeping track of the original atom count. In total, 6.19% of propositions are

⁴<https://huggingface.co/datasets/pwc-archive/papers-with-abstracts>, derived from the Papers with Code dataset (<https://paperswithcode.com>)

deemed not entailed, and 12.42% of sentences contained at least one non-entailed proposition; 73 sentences are completely removed, as all the corresponding propositions were classified as not entailed.

- Extraction of query propositions entailed by multiple sentences in the corpus. In total, 19,178 propositions are uniquely entailed by one sentence, while 21,133 propositions are entailed by at least two sentences.

Concerning the latter point, *entailment* does not imply *containment*, which is what PROPSEARCH should test. Therefore, with the aim of disambiguating between entailment and the stricter notion of containment, we use gpt-oss-120b with the following prompt:

Given a proposition and a sentence, you need to distinguish between the proposition being:

- ****Entailed**** by the sentence (looser condition)
- ****Contained**** explicitly in the sentence (stricter condition). It does not need to be the **exact** phrasing.

Only reply with a json array: "type": "entailed" or "type": "contained".

A manual validation of a sample of 200 instances from this LLM-based classification shows that, while gpt-oss-120b shows low precision (0.53), that is, many *entailed* wrongly classified as *contained*, it achieves very high recall (0.96), that is, very few *contained* classified as *entailed*. This is actually desirable, as the second mistake is much more severe in this context than the first.

Through this process, we recover 17,159 propositions, which are classified as entailed but not contained. Propositions classified as contained by more sentences are removed. This leaves us with a total of 36,337 query propositions, 13,020 source sentences, and 300,000 confounding sentences.

Finally, we run a manual quality check on a sample of 200 (query, source_sentence) instances, and find that over 96% of instances are high quality. We also find that Claude Sonnet 4.6 tends to extract a lower number of atomic propositions compared to gpt-oss-120b.

Distribution Figure 7 shows the distributions of atomic proposition counts in the PROPSEARCH

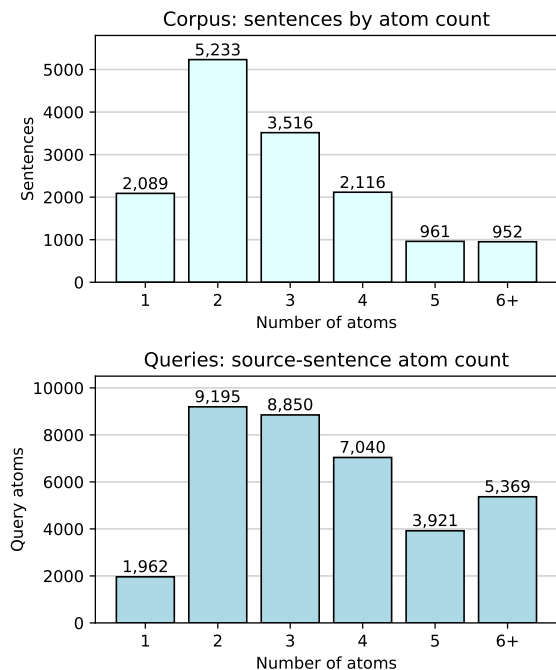


Figure 7: Atom-count distributions for the PROPSEARCH benchmark. On top, number of source sentences (excluding confounders) in the retrieval corpus (14,867 total, the 13,020 containing uniquely contained atoms plus the remaining ones containing either non-entailed atoms or not uniquely-contained atoms) by atom count. On the bottom, Distribution of source-sentence atom count over 36,337 query propositions. The 6+ bucket aggregates all sentences with six or more atoms.

dataset. While most of the source sentences distribute around the 2-3 propositions mark, the source-sentence atom count by query atom is more balanced, as the fewer sentences with higher atom counts are associated to more query propositions.

A.3 Experiments with Alternative Backbones

Table 6 shows the results of training APE using different encoder backbones. We test BGE-small-en-v1.5 and BGE-base-en-v1.5 (Xiao et al., 2024), referred to as BGE-small and BGE-base in the table for brevity, all-mpnet-base-v2 (mpnet base model introduced in Song et al. (2020), fine-tuned sentence transformer version available on Hugging Face⁵), referred to as all-mpnet in the table, and all-MiniLM-L6-v2 (Reimers and Gurevych, 2019), referred to as MiniLM in the table.

Most of the gains are on PAWS-WIKI (Zhang et al., 2019), where models go from scoring quasi-

⁵<https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

random to outperforming most fixed-size zero-shot baselines, with gains up to 20 AUC points. A sig-

Encoder	PAWS	SNLI	PropSearch
BGE-small	81.1 (+20.3)	84.1 (+3.1)	96.0 (+7.6)
all-mpnet	82.3 (+17.1)	84.7 (+1.4)	94.6 (+13.2)
BGE-base	81.8 (+19.8)	83.7 (+1.6)	96.5 (+4.7)
MiniLM	79.7 (+16.6)	81.8 (+1.1)	94.9 (+9.6)

Table 6: APE performance (AUC, MRR) across different encoder backbones. The blue number in parentheses indicates the performance gap between APE and its backbone encoder. Training uses mean-pooled representations from the teacher encoder for all presented models.

nificant performance gain can also be observed on PROPSEARCH, where models all reach the 94.5 MRR (macro average) mark after APE training, starting from very different baseline performances. With the exception of BGE-small, gains are less noticeable on SNLI (Bowman et al., 2015), where baseline models already achieve high scores.

A.4 Scaling APE

We experimented with scaling the number of atoms (1) under the same configuration, and (2) using a larger backbone with a larger decoder (6 layers, 12 heads). We test two max_atoms values: 5 (83% of the training dataset) and 8 (98.5% of the training dataset). Figure 8 shows how training APE on

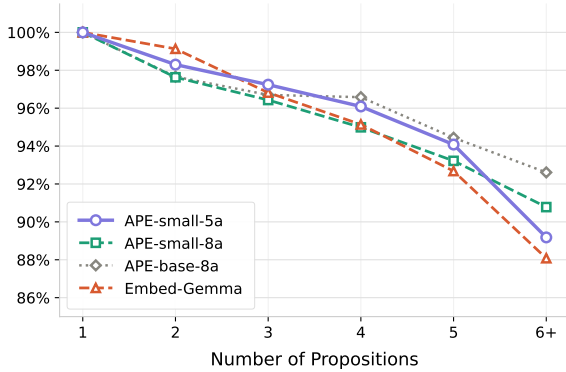


Figure 8: Performance of APE models trained with max_atoms = 5 and max_atoms = 8.

a higher number of atoms significantly improves its retrieval performance at high complexity levels, while making it slightly less precise at lower complexity levels.

A.5 Training Hyperparameters

The encoder backbone is frozen for the first 3,000 steps, during which only the decoder, embedding

head, and activity head are trained with a learning rate of 10^{-4} . The backbone is subsequently unfrozen and trained with a lower learning rate of 2×10^{-5} . Both parameter groups use AdamW with weight decay 10^{-2} . A cosine schedule with 500 warmup steps is applied throughout. Training runs for 3 epochs with a batch size of 64 and mixed precision (FP16). We experiment with different values of λ_{rep} specifically, finding that higher values led to improved cardinality prediction but worse embedding quality. The value in the tested ones $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ that achieved the best trade-off was 0.1.

With these hyperparameters, training APE with a BGE-small-en-v1.5 backbone takes approximately 1.5 hours on a single A100 GPU.

We do not carry out multiple trainings with different seeds. The reported results are therefore from a single run, for each trained model.

A.6 Gemma and Qwen Instructions

EmbeddingGemma For PAWS-WIKI, we use the “task: sentence similarity | query:” instruction for EmbeddingGemma on both sentences, as recommended in the model card. Initially, we tried both “task: fact checking | query:” and “task: entailment detection | query:” instructions for SNLI, falling back to “task: sentence similarity | query:” as it outperformed other instructions by a large margin. For PROPSEARCH, we use the standard “task: search result | query:” for propositions and “title: “none” | text: ” for sentences.

Qwen3-Embedding For PAWS-WIKI, we tried both “Instruct: Retrieve semantically similar text. Query:” and “Instruct: Retrieve semantically equivalent text. Query:”, with minimal change in performance. We report the former, as it won with a slight margin. For SNLI, we experimented with entailment-related instructions with poor results, falling back to the former instruction. For PROPSEARCH, we tested both “Instruct: Retrieve passages that verify or contradict the given claim. Query: ”, and “Instruct: Retrieve passages that contain the given claim. Query: ”, with the former slightly outperforming the latter.

A.7 APE vs Sub-Sentence Encoder

To further show the architectural gain of APE over previous methods, we train it using the same train-

ing data from Chen et al. (2024) and compare the results on the benchmarks (Table 7). The training examples are roughly 1/3 of APE’s training dataset.

Model	PAWS	SNLI	PropSearch
APE	81.1 (82.1)	84.1 (85.2)	96.0
APE*	<u>78.1 (78.2)</u>	84.1 (84.1)	<u>92.7</u>
SSE-T5	76.0 (76.4)	80.7 (81.1)	67.7
SSE-GTR	75.8 (76.2)	76.9 (76.9)	73.3

Table 7: APE performance (AUC, MRR) with different training datasets. The * indicates the APE model trained on SSE data.

A.8 Additional Benchmarks

While APE is not a general-purpose sentence representation model, as it is not trained to represent sentences holistically, we still test its performance on typical semantic textual similarity (STS) benchmarks from MTEB (Muennighoff et al., 2023) – namely STS-B and SICK-R – as a diagnostic for catastrophic forgetting with respect to its teacher encoder. Results are shown in Table 8.

Model	STS-B	SICK-R
APE (HA)	83.8 (-3.0)	78.7 (-1.1)
APE (MaxSim)	84.9 (-1.9)	78.8 (-1.0)
APE (H2P)	79.1 (-7.7)	76.7 (-3.3)
APE-discrete (MaxSim)	79.9 (-6.9)	74.8 (-5.2)
SSE-T5 (MaxSim)	80.1 (-5.1)	76.1 (-3.9)
SSE-GTR (MaxSim)	79.2 (-0.4)	73.2 (+1.7)

Table 8: Results on the STS-B and SICK-R benchmarks. The reported metric is Spearman’s rank correlation coefficient. The numbers in parentheses refer to the gap between each proposition-level model and its fixed-size teacher/backbone encoder.

First, we see how the best-performing scoring rule is MaxSim (Section 3.4), which is consistent with the STS task, as it is the scoring rule that measures overall textual similarity between two sentences the most, being both asymmetric and robust to diverging number of propositions.

Moreover, the performance of almost every proposition-level model decreases on STS tasks compared to the respective fixed-size backbone encoder. This is expected, as fixed-size sentence encoders are specifically trained to excel in such tasks, which require holistic sentence representation capabilities. Despite not being a general-purpose sentence encoder, ignoring inter-proposition relationships by design, APE still manages to achieve

strong STS performance with minimal loss compare to its backbone encoder. The behavior of SSE models (Chen et al., 2024) is less clear, as SSE-T5 shows steep degradation compared to its backbone encoder, while SSE-GTR actually improves the performance of its backbone encoder on SICK-R, albeit starting from a much lower score overall.

Interestingly, if we compare APE’s performance to APE-discrete (the full gpt-oss-120b decoding plus BGE-small-en-v1.5 encoding process), we can see how APE improves significantly over its teacher pipeline (+5.0 Spearman’s rank correlation points on STS-B and +4.0 on SICK-R), which provides further evidence that APE can generalize over the biases present in its training signal.

A.9 Qualitative Analysis

In this section, we qualitatively analyze (1) gpt-oss-120b’s decomposition behavior in the presence of several linguistic phenomena, and (2) APE’s behavior in the presence of the same phenomena.

A.9.1 Temporality

The intended decomposition behavior for training data generation is to discard the temporal relation (*before, after, then, ...*) completely. Failed decompositions are very rare in this case, and they materialize non-atomic propositions that contain the temporal relation. Some examples:

Sentence: *By that time, Jahangir had more or less reverted to tolerant policies of Akbar and after finding Hargobind innocent and harmless ordered his release.*

Propositions: [1] *Jahangir had more or less reverted to tolerant policies of Akbar.* [2] *Jahangir found Hargobind innocent.* [3] *Jahangir found Hargobind harmless.* [4] *Jahangir ordered the release of Hargobind.*

Note: Good decomposition, the temporal relation is discarded.

Sentence: *On 9 February, as "Tarleton" was leaving Port-au-Prince, presumably having taken refuge there after her previous encounter with the British brig, she encountered a British frigate and a brig.*

Propositions: [1] *The date was 9 February.* [2] *Tarleton was leaving Port-au-Prince.* [3] *Tarleton had previously encountered a British brig.* [4] *Tarleton had presumably taken refuge in Port-au-Prince after that encounter.* [5] *Tarleton encoun-*

tered a British frigate. [6] Tarleton encountered a brig.

Note: Bad decomposition, [4] is not atomic; many propositions are repeated.

Concerning APE's behavior, the presence of complex temporal relations in the source sentence occasionally leads to unclear interpretation of some atomic embeddings.

A.9.2 Causality/Conditionality

The intended decomposition behavior for training data generation is to extract the cause/condition and consequence as isolated atomic propositions. The most common failure mode (still rare) is to include both in the same proposition. Some examples:

Sentence: *The people of Sili rely on subsistence agriculture for their livelihood, and because it is located away from the sea, fishing has minimal impact on the village economy.*

Propositions: [1] *The people of Sili rely on subsistence agriculture for their livelihood.* [2] *Sili is located away from the sea.* [3] *Fishing has minimal impact on Sili's economy.*

Note: Good decomposition, cause and effect are isolated propositions.

Sentence: *On occasion, when the winner does not qualify (due to age) for either contest, a runner-up is sent, traditionally the winner selected in the previous year to the Miss Universe pageant.*

Propositions: [1] *The winner sometimes does not qualify for either contest due to age.* [2] *If the winner does not qualify for either contest due to age, a runner-up is sent.* [3] *The runner-up that is sent is traditionally the winner selected in the previous year to the Miss Universe pageant.* [4] *The winner selected in the previous year to the Miss Universe pageant is traditionally sent as a runner-up.*

Note: Bad decomposition, condition and consequence are merged. Propositions are not atomic and repeated.

Concerning APE's behavior, especially on longer sentences, it might merge cause/condition and consequence into the same embedding.

A.9.3 Non-Entailment

Some propositions, taken in isolation, might not be entailed from their source sentence. If such propositions are not verifiable looking at the rest of the

sentence, the intended behavior for training data generation is to retain the uncertainty in the phrasing of the proposition. The most common failure mode is to extract such propositions as certain facts. The NLI filtering we apply to remove hallucinated training instances mostly solves this issue. Some examples:

Sentence: *St. Peter's and St. Paul's, is a church situated in Kolenchery believed to have established in Seventh Century.*

Propositions: [1] *St. Peter's and St. Paul's is a church.* [2] *St. Peter's and St. Paul's is situated in Kolenchery.* [3] *St. Peter's and St. Paul's is believed to have been established in the Seventh Century.*

Note: Good decomposition, uncertainty is kept ("is believed to have been established...").

Sentence: *Anna hides in a closet, and Ben attempts to send Janine away who then makes a heartfelt plea to save their marriage, and there are indications she succeeds in seducing him.*

Propositions: [1] *Anna hides in a closet.* [2] *Ben attempts to send Janine away.* [3] *Janine makes a heartfelt plea to save their marriage.* [4] *There are indications that Janine succeeds in seducing Ben.* [5] *Janine succeeds in seducing Ben.*

Note: Bad decomposition, the uncertain proposition is reported as a certain fact [5] (in addition to being reported correctly [4]).

APE's behavior is particularly interesting in such cases. To analyze it, we can exploit examples from the held-out set.

Example A *The British were confused by the papers wrapped around the declaration, which they thought were a secret cypher.*

Here, gpt-oss-120b oversplits the sentence as:

- 1) *The British were confused by the papers wrapped around the declaration*
- 2) *The papers were wrapped around the declaration*
- 3) *The British thought the papers wrapped around the declaration were a secret cypher*
- 4) *The British thought the papers were a secret cypher*

If we encode the same sentence with APE, we successfully obtain 2 atomic embeddings. Applying proposition seeding:

- **Seed:** *The British were confused by the papers wrapped around the declaration* → 0.99 similarity with Atom 2, 0.78 with Atom 1.
- **Seed:** *The British thought the papers wrapped around the declaration were a secret cypher* → 0.96 similarity with Atom 1, 0.85 with Atom 2.

If we instead remove the uncertainty from the latter seed:

- **Seed:** *The papers wrapped around the declaration were a secret cypher.* → 0.92 (−0.042) similarity with Atom 1, 0.79 with Atom 2.

Here, APE actually corrects gpt-oss-120b’s over-splitting, in addition to being consistent with the hypothetical representation of the uncertain proposition.

Example B *On 10 January 2008, the signing was made permanent, for an undisclosed fee, believed to be £1 million.*

This is correctly split by gpt-oss-120b as:

- 1) *The signing was made permanent on 10 January 2008*
- 2) *The fee for the signing was undisclosed*
- 3) *The fee for the signing was believed to be £1 million*

If we encode this sentence with APE, we correctly obtain 3 atomic embeddings. Again, we apply proposition seeding to interpret them:

- **Seed:** *The signing was made permanent on 10 January 2008* → 0.98 similarity with Atom 1, around 0.68 similarity with other atoms (0.65, 0.71).
- **Seed:** *The fee for the signing was undisclosed* → 0.97 similarity with Atom 3, around 0.73 similarity with other atoms (0.66, 0.79).
- **Seed:** *The fee for the signing was believed to be £1 million* → 0.95 similarity with Atom 2, around 0.76 similarity with other atoms (0.65, 0.87).

If we remove the uncertainty from the last seed:

- **Seed:** *The fee for the signing was £1 million* → 0.93 (−0.022) similarity with Atom 2, around 0.76 similarity with other atoms (0.65, 0.86).

A small difference, but if we compare “*The fee for the signing was believed to be £1 million*” and “*The fee for the signing was £1 million*” with the teacher encoder (BGE-small-en-v1.5), we get 0.983 cosine similarity, thus what we find is consistent with the teacher’s latent space.

A.10 Licensing and Intended Use

Existing artifacts All artifacts used in this work are employed in a manner consistent with their licenses and intended use. Our training data is derived from WikiSplit (Botha et al., 2018), released by Google under the CC-BY-SA 4.0 license, which permits derivative works under share-alike terms. Among the baseline models we evaluate, BGE (Xiao et al., 2024), GTE (Li et al., 2023), and Qwen3-Embedding (Zhang et al., 2025) are released under permissive open-source licenses (MIT for BGE and GTE, Apache 2.0 for Qwen3-Embedding), and EmbeddingGemma (Vera et al., 2025) is released under the Gemma Terms of Use, which permits research and responsible use; our use of all four for benchmarking is within their intended scope. The Sub-Sentence Encoder (Chen et al., 2024) is a publicly released research artifact, used here for research comparison consistent with its intended use. Training-set annotations were generated with gpt-oss-120b (OpenAI, 2025), released under Apache 2.0. The PROPSEARCH benchmark annotations were generated with Claude Sonnet 4.6 (Anthropic, 2026); these outputs are used solely to construct an evaluation benchmark, consistent with Anthropic’s usage terms, and are not used as training data. In particular, APE itself is trained only on the gpt-oss-120b-annotated WikiSplit data and never on Claude-generated outputs.

Created artifacts We release two artifacts: the APE model (with its proposition-decomposed training set) and the PROPSEARCH benchmark. Both are intended for research use only. The APE training data is an LLM-decomposed derivative of WikiSplit and therefore inherits WikiSplit’s CC-BY-SA 4.0 share-alike conditions; we release it under the same license. PROPSEARCH is constructed from publicly available sources: arXiv abstracts (Clement et al., 2019), released under a CC0-1.0 public-domain dedication. We release

PROPSEARCH for research use only. Its source sentences are drawn from scientific paper abstracts, released under a CC0-1.0 public-domain dedication, and contain no personal or sensitive information beyond standard scholarly text. As derivatives of data accessed and produced for research purposes, neither PROPSEARCH nor the APE model is intended for deployment outside research contexts.