Contents

1 Vector Algebra Lab Research Report

Last updated: 2026-05-25

1.1 Scope

The app is a math visualization for word-vector operations, not a factual language tool. The goal is to make its results as strong and legible as possible for arbitrary plus/minus word equations while avoiding obvious failures where the answer just echoes the input entity.

1.2 Research Takeaways

- Classic word-vector analogies come from static embedding work such as word2vec and GloVe. Those models were explicitly studied with equations like `king - man + woman`.
- 3CosAdd is the simple vector-offset method. 3CosMul can help classic static embeddings by reducing dominance from one large cosine term, but it performed worse on this app's tested sentence embedding models.
- Recent analogy research still treats word analogy as a specialized geometry problem. "Revisiting Word Embeddings in the LLM Era" reports that carefully handled static embeddings remain competitive on analogies, and it highlights mean-centering and relation-specific methods such as LRCos. Those methods need relation examples, so the app now treats them as relation-family inference rather than a direct raw arithmetic replacement.
- Modern embedding leaderboards such as MTEB mostly optimize retrieval, semantic similarity, classification, clustering, and reranking. High MTEB scores do not guarantee old-school analogy arithmetic behavior.
- Relation embedding work such as RelBERT supports the same conclusion from the other side: analogy quality improves when the model explicitly represents word-pair relations, not just individual word positions.
- 2026 LLM analogy papers reinforce the split between analogy generation and controllable vector arithmetic: LLMs can encode and generate relationally aligned analogies, but failures still trace to missing or misplaced relational structure, so an embedding calculator needs explicit relation evidence and calibration rather than a larger generic embedder alone.
- Concept/subspace arithmetic work in transformer LMs suggests a more serious next step: extract or learn relation subspaces instead of assuming raw sentence embeddings preserve a linear relation globally. Feucht et al. report that transforming Llama-2-7b hidden states through concept-head weights raises country-capital nearest-neighbor analogy accuracy from 47% with raw hidden states to 80% in concept space, and token-head transformations similarly improve surface-form analogies.
- Current embedding research is not simply returning to raw word2vec. Recent LLM-generated text embedding work optimizes broad retrieval/MTEB-style performance from synthetic contrastive data, while 2025 static-word-embedding work extracts cheaper static vectors from Sentence Transformers for sentence semantics. Neither line directly claims arbitrary plus/minus lexical algebra; the relevant lesson is to separate embedding quality from relation-arithmetic quality.
- Newer subspace-embedding work is also relevant because point vectors are weak for hierarchy, negation, and conjunction. Subspace embeddings model concepts as linear subspaces and support operations such as intersection, linear sum, and orthogonal complements; that is closer to the math the UI exposes, but it is not yet a drop-in lexical analogy model.
- A May 2026 literature refresh did not find a mature off-the-shelf model or benchmark for arbitrary chained word arithmetic. The strongest current evidence still points toward relation-aware methods, relation examples, and subspace learning rather than one global nearest-neighbor offset.
- A 2026-05-25 refresh supports keeping the benchmark honest: NeurIPS 2025 work on linear analogy emergence argues that classic linear analogies are a real co-occurrence geometry, while AAAI 2026 LLM work still reports

failures when relational information is missing, misplaced, or hard to transfer. The local implication is to improve relation inference and calibration, not to assume a larger embedder alone fixes arbitrary algebra.

- A follow-up 2026 refresh found the same pattern in an adjacent multimodal setting: a 2026 remote-sensing CLIP arithmetic study traces many arithmetic failures to concept entanglement and reports that text-only similarity can partially predict whether arithmetic will work. This supports the app's reliability-gating direction: predict when to trust a relation operation instead of forcing every equation through one vector offset rule.
- The newest local results reinforce that direction while also showing the danger of overfitting the input prompt. Wrapping terms as short concept phrases and applying target-side relation-class scoring now gives the best exact Google fixed-vocabulary result, 225/280 top-1 over 20,045 candidate words, and improves the BATS candidate-set slice from 807/1200 to 826/1200 for automatic relation-class scoring. The same concept wrapper regresses strict fixed-BATS automatic relation-class scoring from 655/1200 to 616/1200 and few-shot class scoring from 652/1200 to 635/1200, so it is a benchmark-specific option rather than a deployable default.
- A deployable stronger-model probe did not beat BGE-large. `Xenova/gte-large` tied BGE-large on raw 3CosAdd in the smaller fixed-Google slice but trailed on relation-aware scoring. `onnx-community/Qwen3-Embedding-0.6B-ONNX` was much worse on bare-word analogies even after switching from mean pooling to the model-card-recommended last-token pooling, but short concept wrapping recovered a large part of the gap. This is an important mixed result: state-of-the-art retrieval/MTEB embedders are not automatically state-of-the-art lexical arithmetic embedders, and input formatting can change analogy geometry by more than the model swap itself.
- A 2026-05-25 source recheck preserved the same research framing. MTEB explicitly covers broad embedding tasks rather than lexical arithmetic; Qwen3 Embedding reports state-of-the-art multilingual MTEB/retrieval performance but not classic analogy wins; recent analogy-specific papers point to co-occurrence geometry, relation examples, and concept/token subspaces as the mechanisms that recover linear analogies. The local benchmark direction therefore remains: keep BGE-large as the deployable base, add relation-aware scoring and calibrated selector evidence, and do not treat a larger retrieval embedder as a guaranteed arithmetic upgrade.
- The first validation-selected scorer gate is now the strongest learned fixed-BATS selector in this branch. It chooses between automatic relation-class scoring and five-example few-shot relation-class scoring from one relation-score threshold, beats the strongest always-class/few-shot baseline by 1.3 to 2.3 top-1 points across three held-out splits, and also beats the nearest-centroid portfolio selector on all three splits. It still trails the per-record portfolio oracle, so it is a research result rather than a solved arbitrary arithmetic model.
- The same gate idea did not transfer cleanly to fixed-Google. Three fixed-Google held-out splits underperformed the best always-class or always-few-shot baseline by 0.7 to 2.2 top-1 points, and selected less stable relation-margin/score rules. This is a useful guardrail: the current gate is a BATS improvement, not a benchmark-universal selector.
- A nested-validation variant makes that guardrail sharper. It chooses candidate gates on an inner validation split instead of the full outer-train split. On fixed-BATS it still beats the strongest constant baseline on all three outer splits, but the larger v2/v3 gains shrink. On fixed-Google it improves over the first gate overall, tying or beating the constant baseline on two splits, but still underperforms badly on one split. The selector problem is therefore real generalization, not just one noisy threshold.
- A conservative nested gate that requires one or two extra inner-validation top-1 hits before leaving the constant baseline was negative. It refused some zero-lift Google gates, but still accepted unstable target-agreement gates on BATS and did not beat the nested gate or per-split constant baselines.
- A repeated-split nested gate is also not enough. It aggregates candidate gates over seven deterministic inner validation partitions and rejects gates with unstable win/loss rates. On fixed-BATS it improves over per-split constants in aggregate, but trails the simpler nested gate. On fixed-Google it falls behind nested and guarded gates because repeated validation still misselects both relation-margin gates and constant labels. The next selector should not be another scalar acceptance threshold; it needs benchmark-conditioned reliability, explicit feature-family priors, or calibrated uncertainty around both gates and fallback constants.
- A relation-family grouped version sharpens that conclusion. It calibrates gates and constants by the inferred relation family rather than the true benchmark category. On fixed-BATS this barely edges the nested gate in aggregate, 1034/1845 versus 1033/1845, but on fixed-Google it drops to 316/407 even at the most conservative checked group-size threshold. Inferred-family calibration is therefore useful diagnostic evidence, not a deployable selector.
- A shrunk relation-family version is the best fixed-BATS selector in this branch so far among the stricter validation variants, but it still fails cross-benchmark promotion. It uses inferred-family rules only when they beat the global repeated rule by a repeated-validation margin. On fixed-BATS it reaches 1049/1845 top-1, nearly matching the non-nested gate's 1051/1845 while being more conservative than independent grouping. On fixed-Google it still reaches only 316/407 because the global repeated fallback itself chooses unstable constants and one bad class/raw

gate.

Useful sources:

- Mikolov et al., "Efficient Estimation of Word Representations in Vector Space": https://arxiv.org/abs/1301.3781
- Pennington et al., "GloVe: Global Vectors for Word Representation": https://aclanthology.org/D14-1162/
- Levy and Goldberg, "Linguistic Regularities in Sparse and Explicit Word Representations": https://aclanthology.org/W14-1618/
- Muennighoff et al., "MTEB: Massive Text Embedding Benchmark": https://aclanthology.org/2023.eacl-main.148/
- Qwen3 Embedding model card and MTEB evaluations: https://huggingface.co/Qwen/Qwen3-Embedding-0.6B
- Merullo et al., "The Geometry of Categorical and Hierarchical Concepts in Large Language Models": https://aclanthology.org/2024.naacl-long.281/
- "Revisiting Word Embeddings in the LLM Era": https://aclanthology.org/2025.ijcnlp-long.145/
- "RelBERT: Embedding Relations with Language Models": https://doi.org/10.1016/j.artint.2025.104359
- "On the Emergence of Linear Analogies in Word Embeddings": https://arxiv.org/abs/2505.18651
- "The Curious Case of Analogies: Investigating Analogical Reasoning in Large Language Models": https://doi.org/10.1609/aaai.v40i37.40414
- "The Curious Case of Analogies: Investigating Analogical Reasoning in Large Language Models" arXiv page: https://arxiv.org/abs/2511.20344
- "Parallelograms Strike Back: LLMs Generate Better Analogies than People": https://arxiv.org/abs/2603.19066
- Concept arithmetic with token and concept subspaces: https://arithmetic.baulab.info/
- Feucht et al., "Vector Arithmetic in Concept and Token Subspaces": https://arxiv.org/abs/2511.18162
- Wang et al., "Improving Text Embeddings with Large Language Models": https://arxiv.org/abs/2401.00368
- Wada et al., "Static Word Embeddings for Sentence Semantic Representation": https://arxiv.org/abs/2506.04624
- Moreira et al., "Native Logical and Hierarchical Representations with Subspace Embeddings": https://arxiv.org/abs/2508.16687
- Zhuang et al., "Vector-ICL: In-context Learning with Continuous Vector Representations": https://arxiv.org/abs/2410.05629
- Hong and Yu, "When Does Embedding Arithmetic Fail? A Systematic Analysis in Remote Sensing Vision-Language Models": https://openreview.net/forum?id=SmTDce10MP
- "Text embedding models yield detailed conceptual knowledge maps derived from short multiple-choice quizzes": https://www.nature.com/articles/s41467-026-69746-w
- Hugging Face, "Train 400x faster Static Embedding Models with Sentence Transformers": https://huggingface.co/blog/static-embeddings
- Qwen Team, "Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models": https://qwenlm.github.io/blog/qwen3-embedding/
- Zhang et al., "Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models": https://arxiv.org/abs/2506.05176
- Hugging Face ONNX Community, Qwen3-Embedding-0.6B-ONNX model card: https://huggingface.co/onnx-community/Qwen3-Embedding-0.6B-ONNX

1.3 Local Benchmark Harness

Benchmark scripts are now available for curated probes, Google analogies, fixed-vocabulary Google analogies, BATS analogies, fixed-vocabulary BATS analogies, curated chains, and BATS relation perturbations:

```
npm run benchmark:word-arithmetic
npm run benchmark:google-analogies -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --models=Xenova
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:bats-analogies -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestio
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:bats-analogies -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestio
npm run benchmark:bats-analogies -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestio
npm run benchmark:bats-analogies -- --device=webgpu --termTemplate=concept --maxCategories=40 --maxPairsPerC
npm run benchmark:bats-analogies -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestio
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
```

```
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-bats -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestionsPe
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --maxPerCategory=20 --maxCategories=14 --vocabSize=20000 -
npm run benchmark:fixed-google -- --device=webgpu --termTemplate=concept --maxPerCategory=20 --maxCategories
npm run benchmark:fixed-google -- --device=webgpu --pooling=last_token --maxPerCategory=10 --maxCategories=
npm run benchmark:fixed-google -- --device=webgpu --pooling=last_token --termTemplate=concept --maxPerCatego
npm run benchmark:chained-arithmetic -- --device=webgpu --model=Xenova/bge-large-en-v1.5 --methods=3CosAdd,
npm run benchmark:bats-chained -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=15 --maxQuestions
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
npm run benchmark:bats-composed -- --device=webgpu --maxCategories=40 --maxPairsPerCategory=50 --maxChains=5
```

The Google analogy harness downloads the public `questions-words.txt` dataset and evaluates candidate-set top-1 accuracy. This is deliberately not full-vocabulary Google analogy accuracy; it measures the app-style setting where the result is selected from a bounded candidate pool.

The fixed-Google harness uses the same analogy questions but searches an external common-English vocabulary. It reports top-1, top-5, and MRR and is closer to real nearest-neighbor evaluation than candidate-set scoring. It supports the few-shot relation-example methods and now uses the same exact candidate-vector index as fixed-BATS for streaming rank metrics and pair-rerank top-K phrase collection. Pair-phrase embeddings are requested lazily after each top-K shortlist is known and cached incrementally in JSONL files, so repeated or interrupted pair-rerank experiments can reuse completed phrase batches without blocking non-pair baselines. Pair-rerank methods can also opt into `--annShortlist=true`, a deterministic random-projection shortlist path that scores only nearby buckets exactly, or `--cosinePrefilter=true`, which first keeps the nearest target-cosine candidates and then scores that pool with the full base ranker. Exact full-index ranking remains the default audit baseline.

The BATS harness downloads BATS 3.0 and evaluates candidate-set top-1 over selected relation files. It supports multiple acceptable answers per pair, such as `dad -> mom/mum`, and generates held-out pair-to-pair analogy questions inside each relation category.

The fixed-BATS mode uses the same BATS questions and answer sets, but searches the external common-English vocabulary plus every selected BATS source and acceptable answer. This is much stricter than candidate-set BATS and reports top-1, top-5, and MRR. The evaluator now builds one finite candidate-vector index per model, scores each included candidate once for exact rank metrics, and uses exact streaming top-K shortlist selection for pair-rerank phrase collection. Pair phrases also use lazy incremental JSONL caching. This keeps the 1,200-question fixed-vocabulary slice practical enough for non-pair runs. The optional ANN shortlist path is available for pair-rerank probes, but it is reported separately because it changes the candidate shortlist searched by the reranker.

`InferredRelation3CosAdd` learns average relation offsets from relation examples, picks the nearest relation prototype from the observed input offset, and applies that prototype to the query word. Unlike `RelationAvg3CosAdd`, it does not receive the benchmark category as the answer key at prediction time. `BlendedInferredRelation3CosAdd` mixes the raw arithmetic target with the inferred relation target.

The chained-arithmetic harness runs five-operand equations such as `king - man + woman - woman + girl = princess` through the same backend as the app. It is intentionally small and diagnostic: it checks whether relation-aware modes preserve residual arithmetic terms instead of silently replacing the whole equation with the first observed analogy.

The BATS chained harness is more precisely a relation-perturbation benchmark. Inserted `+x -x` terms cancel exactly under raw vector addition, so this benchmark should not be read as proof of new semantic composition. It tests whether relation-aware modes remain stable when algebraic no-op terms are added to an otherwise standard analogy.

The BATS composed harness mines non-canceling relation chains from BATS. If one category contains `a -> b` and another contains `b -> c`, it builds a held-out equation from two separate example offsets, such as `a - x + y - u + v = c`. This tests whether a method can apply multiple real relation offsets in sequence instead of only surviving algebraic no-op perturbations.

The composed evaluator now supports `--calibrationSplitSalt` for repeatable split-resampling audits. Its calibration path caches relation prototypes, scans exact rank metrics instead of sorting the full candidate list for every trial, and precomputes raw/prototype candidate scores so blend sweeps do not repeat 1024-dimensional cosine loops. The metric definition is unchanged; these changes make multi-split calibration checks practical.

The harness also contains benchmark-only experimental methods:

- `SoftInferredRelation3CosAdd` and `BlendedSoftInferredRelation3CosAdd`: mix the top relation prototypes with a softmax instead of choosing one hard prototype.
- `AdaptiveBlendedInferredRelation3CosAdd`: changes the blend weight from the hard-prototype confidence.
- `BlendedRelationMulScore`: keeps the blended relation target but adds a small log-3CosMul score term.
- `PairRelationBlendRerank`: reranks a top-K shortlist by comparing relation phrase embeddings such as `man -> king` and `woman -> queen`.
- `RelationClassBlendScore` and `InferredRelationClassBlendScore`: LRCos-inspired class-side scoring. They keep relation-target ranking, then add a small score for being on the target side of a learned relation family and near the source pair's target word.
- `WeightedInferredRelationClassBlendScore`: benchmark-only calibration probe that keeps the same blended target as `InferredRelationClassBlendScore`, but uses a softmax-weighted mixture of the top inferred relation-class axes instead of one hard inferred class. The 2026-05-25 run tied fixed-Google and slightly lost fixed-BATS, so it is not promoted.

- `GatedInferredRelationClassBlendScore`: benchmark-only confidence probe that applies inferred relation-class scoring only when explicit confidence diagnostics clear thresholds, otherwise falling back to raw 3CosAdd. The first fixed-BATS sweeps were negative: score/margin thresholds collapsed to raw performance, target-agreement thresholds either reproduced the ungated baseline or hurt it, and disabling the gate reproduced the ungated baseline.
- `ValidatedRelationClassCentroidSelector`: benchmark-only learned selector. It trains a nearest-centroid classifier on a deterministic train split using relation score, relation margin, and raw/prototype target agreement, then chooses raw 3CosAdd or inferred relation-class scoring on held-out examples. The first fixed-BATS runs underperformed always using relation-class scoring.
- `ValidatedRelationPortfolioCentroidSelector`: benchmark-only learned selector over raw, inferred relation-class, and few-shot relation-class scoring. It uses the same relation diagnostics as the raw/class selector, then chooses one scorer label on held-out examples. The first three fixed-BATS splits beat the always-class or always-few-shot baselines by 0.2 to 1.3 top-1 points, but still trail the oracle portfolio by 4.3 to 5.3 points.
- `ValidatedRelationPortfolioRankCentroidSelector`: benchmark-only learned selector that adds answer-free candidate surface diagnostics: per-method top score, top-score margin, top-5 score spread, top-1 agreement, and top-5 overlap. The first three fixed-BATS splits were negative, so these rank-surface features are not enough with a nearest-centroid selector.
- `ValidatedRelationPortfolioGateSelector`: benchmark-only validation-selected one-feature gate over raw, inferred relation-class, and few-shot relation-class scoring. It directly optimizes held-in train records for top-1/MRR/top-5, then applies the selected rule to held-out records. The first three fixed-BATS splits selected a stable relation-score threshold that switches between class and few-shot scoring and outperformed the centroid portfolio selector.
- `ValidatedRelationPortfolioRankGateSelector`: same one-feature gate search, but with answer-free rank-surface features added. On the checked split it selected the same relation-score gate as the relation-only version, so the added rank-surface features did not yet improve the selector.
- `NestedValidatedRelationPortfolioGateSelector`: benchmark-only inner-validation gate over the same raw/class/few-shot portfolio. It generates candidate single-feature gates from an inner fit split, selects by an inner validation split, and only then evaluates on the outer held-out split. This is a stricter overfit check for the simpler validation-selected gate.
- `NestedValidatedRelationPortfolioRankGateSelector`: same nested selector with answer-free rank-surface features added. It is implemented for controlled probes, but has not earned a full sweep yet.
- `GuardedNestedValidatedRelationPortfolioGateSelector`: benchmark-only conservative nested gate. It compares the inner-validation winner against the best inner-validation constant label and falls back to the constant unless the candidate clears configurable top-1/MRR/top-5 lift thresholds. The first fixed-BATS and fixed-Google sweeps were negative, so this is evidence against simple scalar acceptance thresholds.
- `GuardedNestedValidatedRelationPortfolioRankGateSelector`: same guarded nested gate with answer-free rank-surface features added. It is implemented for controlled probes, but should wait until the non-rank guarded selector has a viable objective.
- `RepeatedNestedValidatedRelationPortfolioGateSelector`: benchmark-only repeated inner-validation gate. It evaluates candidate gates over several deterministic inner validation partitions and accepts a non-constant gate only when aggregate top-1/MRR/top-5 lift and per-round win/loss stability clear configurable thresholds. The first fixed-BATS and fixed-Google sweeps were mixed-to-negative: repeated validation helped reject some unstable gates, but it still misselected enough gates or constants to trail stricter baselines.
- `RepeatedNestedValidatedRelationPortfolioRankGateSelector`: same repeated selector with answer-free rank-surface features added. It is wired for controlled probes, but the relation-only repeated selector has not earned a rank-surface sweep yet.
- `GroupedRepeatedNestedValidatedRelationPortfolioGateSelector`: benchmark-only relation-family calibration probe. It uses the inferred relation family as an answer-free group key, then fits repeated gates/constants inside groups with enough support and falls back to the global repeated selector for sparse groups. The first sweep is mixed-to-negative: a one-hit aggregate gain over nested fixed-BATS, but a large fixed-Google regression.
- `GroupedRepeatedNestedValidatedRelationPortfolioRankGateSelector`: same grouped repeated selector with answer-free rank-surface features added. It is wired for controlled probes, but should wait because the relation-only grouped selector does not transfer.
- `ShrunkGroupedRepeatedNestedValidatedRelationPortfolioGateSelector`: benchmark-only pooled relation-family calibration probe. It first fits a global repeated selector, then only accepts a family-specific repeated selector when that family beats the global rule by configurable repeated-validation top-1/MRR/top-5 and win/loss margins. The first sweep is the best strict fixed-BATS selector in this branch, 1049/1845, but it still fails

fixed-Google at 316/407, so it is not promoted.

- `ShrunkGroupedRepeatedNestedValidatedRelationPortfolioRankGateSelector`: same shrunk grouped selector with answer-free rank-surface features added. It is wired for controlled probes, but should wait because the relation-only shrunk selector still fails fixed-Google.
- `InferredRelationClassPairRerank`: benchmark-only hybrid that first ranks with inferred relation-class scoring, then reranks the top-K shortlist with pair-phrase relation similarity.
- `InferredRelationClassMulScore`: benchmark-only hybrid that keeps inferred relation-class scoring and adds a log-3CosMul term. The May 2026 sweep did not beat the class-only method.
- `SymmetricPairRelationBlendRerank`: compares both forward and inverse relation-pair phrases. It improved fixed-Google at a 50/50 inverse mix, but regressed BATS, so it remains benchmark-only.
- `FewShotRelation3CosAdd`: simulates a user supplying a small number of correct relation examples. It builds a prototype from only those held-out same-category pairs and applies it to the query word.
- `FewShotBlendedRelation3CosAdd`: blends the few-shot prototype target with the raw offset target.
- `FewShotRelationClassBlendScore` and `FewShotBlendedRelationClassBlendScore`: add LRCos-inspired target-side class scoring from only the few-shot examples.
- `RelationPrototypeChain3CosAdd`: decomposes five-token chained equations into explicit `minus, plus` offset pairs, infers the closest BATS relation family for each pair, then adds the learned relation prototypes to the start word. This is the first non-canceling chain scorer in the harness.
- `ComposedPairRelationChainRerank`: benchmark-only composed-chain reranker. It starts from `RelationPrototypeChain3C` embeds the explicit relation path phrase such as `run -> running ; playing -> plays`, then reranks the top-K candidates by comparing that path against candidate phrases such as `walk -> walks`.
- `CalibratedComposedPairRelationChainRerank`: same composed pair reranker, but chooses `pairRelationWeight` on the train split from a small grid before evaluation. The first split-resampling run underperformed the fixed `0.35` setting, so this is a calibration negative result rather than the promoted variant.
- `MultiFeatureComposedPairRelationChainRerank`: composed-chain reranker that starts from `MultiFeatureBlendedRelat` instead of the prototype-only target, then applies the same composed pair-phrase rerank. The first four-split run underperformed the prototype-only pair reranker, so this is a rejected cascade rather than the promoted variant.
- `CalibratedMultiFeatureComposedPairRelationChainRerank`: same multi-feature-base pair reranker, but chooses `pairRelationWeight` on the train split. It recovers some top-5 hits but still trails fixed prototype-pair reranking on top-1 and MRR.
- `CalibratedJointComposedPairRelationChainRerank`: joint grid-search reranker over raw target score, prototype target score, composed pair-phrase score, and pair-score interactions with relation confidence features. It searches a union of raw/prototype top-K candidates, making it stricter and slower than the pair-only reranker. The first four-split run overfit the train partitions and underperformed pair-only, so this is a useful negative result.
- `LearnedJointComposedPairRelationChainRerank`: regularized pairwise linear reranker over raw score, prototype score, pair score, and relation-confidence pair interactions. It trains on each calibration split with a hinge-style margin update. The first four-split run is faster than the hand-grid joint search but still underperforms pair-only because one held-out split collapses.
- `FamilyLearnedJointComposedPairRelationChainRerank`: relation-chain-specific variant of the learned joint reranker. It trains a global fallback plus separate pairwise linear weights for relation chains with at least three training examples. The first four-split run is the strongest automatic composed-chain top-1/MRR result so far, but it still trades away top-5 recall.
- `ShrunkFamilyLearnedJointComposedPairRelationChainRerank`: count-shrunk variant of the family-specific learner. It linearly blends each family model back toward the global learned model by `familyRecords / (familyRecords + familyShrinkageRecords)`. The first sweeps show it can add one top-1 hit at light shrinkage, but it worsens top-5 and does not solve the recall tradeoff.
- `ValidatedFamilyLearnedJointComposedPairRelationChainRerank`: held-out shrinkage-gated family learner. It splits each relation-chain family inside the train partition, trains a temporary family model on the fit slice, chooses a blend alpha against the global model on the validation slice, then retrains the family model on all train records for that family. The first four-split run recovered one top-5 hit versus unshrunk family learning but gave up too much top-1, so it is a narrow recall tradeoff rather than a promotion.
- `ListwiseFamilyLearnedJointComposedPairRelationChainRerank`: relation-chain-specific learner with a listwise softmax cross-entropy update over the whole raw/prototype top-K shortlist. It is the first true shortlist-normalized objective in the composed-chain harness. The first four-split run improved top-5 versus unshrunk family learning but lost too much top-1 and MRR, so it is a recall-biased negative result rather than the promoted

ranker.

- `PairBlendedFamilyLearnedJointComposedPairRelationChainRerank`: pooled family/pair blend. It learns the same relation-chain family model, then chooses one train-split alpha that interpolates the learned family score with a pair-only-style baseline score over the raw/prototype shortlist. The first four-split run recovers some top-5 versus family learning, but not enough to beat pair-only top-5 or unblended family top-1/MRR.
- `GatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank`: one-feature train-selected reliability gate over the pooled family/pair alpha. It recovers family-level MRR while preserving more top-5 recall than unblended family learning, but still misses pair-only top-5.
- `ValidatedGatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank`: same bounded family/pair gate, but the gate is selected on an inner validation slice while the final family scorer is retrained on the full train partition. The first four-split run is the strongest automatic top-5 composed-chain result so far, with 559/788 top-5 hits, but it gives up top-1 versus family learning.
- `ValidatedLinearPairBlendedFamilyLearnedJointComposedPairRelationChainRerank`: constrained multi-feature alpha model. It normalizes reliability diagnostics on the inner validation slice, searches small linear alpha formulas, then interpolates between the pair baseline and full-train family scorer. It matches the unblended family learner's top-1 while recovering some top-5, but does not beat the validated one-feature gate's top-5.
- `top5-with-top1-loss` pair-family objective: validation selector that maximizes top-5 after allowing at most an absolute number of lost top-1 validation hits versus the top-1-best candidate. With `--pairFamilyBlendMaxTop1Loss=1`, it was stable but did not beat the prior `top5-with-top1-floor` gate result.
- `top5-with-bootstrap-top1-loss` pair-family objective: validation selector that bootstraps the inner validation records, filters candidates by a high-quantile top-1 loss budget, then ranks the survivors by top-5. It matches the prior best validated-gate top-5 result while making the stability constraint explicit.
- `OraclePairFamilyPortfolioRerank`: diagnostic-only oracle that picks the best per-record outcome from pair-only, family-learned, validated-gated, and validated-linear rankers. It is not a deployable model because it uses the answer rank to choose the winner, but it measures how much headroom remains if a future selector could choose among the existing rankers perfectly.
- `NearestPairFamilyPortfolioRerank`: deployable benchmark prototype that trains a nearest-neighbor selector over pair/family reliability diagnostics and chooses among pair-only, family-learned, validated-gated, and validated-linear rankers without seeing the held-out answer. The first default-split sweep was negative: it overfit the train oracle labels and did not beat the validated gate on held-out top-5 or MRR.
- `CentroidPairFamilyPortfolioRerank`: deployable benchmark prototype that smooths the oracle scorer-label choice by computing one diagnostic centroid per winning label, then selecting the nearest label center at evaluation time. The first default-split probe was a controlled negative result: the train oracle labels were dominated by the family scorer, and centroid smoothing underperformed the validated label gate on held-out top-1, top-5, and MRR.
- `ValidatedGatedPairFamilyPortfolioRerank`: deployable one-feature validation gate over scorer labels instead of alpha values. It chooses among pair-only, family-learned, validated-gated, and validated-linear rankers from an inner validation slice, then applies the selected label rule to held-out examples. The first four-split run improves aggregate top-5 over pair-only and scalar validated gates, but not top-1 or MRR.
- `MetaBlendedFamilyLearnedJointComposedPairRelationChainRerank`: unconstrained pooled meta-ranker over the family-minus-pair score delta and reliability interactions. It tied the pooled blend on top-1 but lost too much top-5, so it remains a negative result and a warning against free linear delta scoring.
- `Top5FamilyLearnedJointComposedPairRelationChainRerank`: rank-aware family learner that stops updating an example once the expected answer is already inside the top five. This was tested as a direct top-5-recall fix, but it severely underfit/overfit across relation chains and is a negative result.
- `ReciprocalRankFamilyLearnedJointComposedPairRelationChainRerank`: smoother rank-aware family learner that weights pairwise updates by the reciprocal-rank gain from moving the expected answer above higher-ranked negatives. This avoids the hard top-five stop rule, but the first tuned four-split run still underperformed pair-only and family learning, so it is also a negative result.
- `BlendedRelationPrototypeChain3CosAdd`: blends the raw composed vector target with the relation-prototype chain target.
- `ThresholdRelationPrototypeChain3CosAdd`, `CalibratedRelationPrototypeChain3CosAdd`, `ConfidenceBlendedRelatio` `MarginBlendedRelationPrototypeChain3CosAdd`, `CandidateMarginBlendedRelationPrototypeChain3CosAdd`, and `MultiFeatureBlendedRelationPrototypeChain3CosAdd`: benchmark-only confidence probes for composed chains. The first two fall back to raw arithmetic when the weakest inferred relation score is below either a fixed threshold or a threshold tuned on a held-out training split. The single-feature blended versions tune one scalar confidence source. The default multi-feature version tunes a linear score-to-prototype-weight curve

over relation score, relation margin, prototype target candidate margin, raw/prototype target agreement, and raw/prototype residual similarity. An optional `--chainIncludeCandidateOverlap=true` probe adds raw/prototype top-candidate overlap to that feature grid, but it is not enabled by default because the first held-out probe added runtime without improving metrics.
- `OracleRelationPrototypeChain3CosAdd`: same chain target as `RelationPrototypeChain3CosAdd`, but receives the true BATS category names. This is a reference ceiling for relation-category inference inside the prototype-only chain scorer, not an arbitrary solver or a ceiling for blended raw/prototype targets.

Experimental methods are deliberately not exposed in the app unless they beat the current blended relation mode across more than one benchmark slice. `PairRelationBlendRerank` cleared that bar in the May 2026 run and is now exposed as the app's pair-relation mode. `InferredRelationClassBlendScore` has also cleared that bar and is exposed as the relation-class mode.

Benchmark scripts accept `--device=webgpu` and include the device in their embedding cache keys. The local Mac runtime successfully initialized a Transformers.js WebGPU feature-extraction pipeline. The production domain remains CPU-backed for cost control. The Cloud Run GPU service is separate, private, internal-ingress only, manually scaled to 0 by default, capped at one L4 instance, protected by a Cloud Scheduler job that forces it back to manual zero every minute, and monitored by a Workflows cap guard that forces zero after 3,600 billable GPU-service seconds in a rolling 24-hour window. `benchmark:fixed-google` and `benchmark:bats-analogies` also accept `--pooling=mean|cls|last_token`; mean pooling remains the default to preserve existing BGE and MPNet runs, while last-token pooling is needed for fairer Qwen3 embedding probes. The same scripts accept `--termTemplate=raw|word|concept|lexical-concept|qwen-analogy`; raw remains the default, and non-raw templates get separate vector-cache keys.

1.4 App Prototype

Vector Algebra Lab now exposes seven backend scoring modes:

- `3CosAdd`: raw vector offset, used as the default because it is the most predictable visual baseline.
- `InferredRelation3CosAdd`: infers a relation family from the observed offset, then applies that learned relation prototype.
- `BlendedInferredRelation3CosAdd`: interpolates between the raw target and the inferred relation target with `ANALOGY_RELATION_BLEND`, default `0.55`.
- `InferredRelationClassBlendScore`: uses the inferred relation family, then adds LRCos-inspired target-side and target-anchor scores with `ANALOGY_CLASS_AXIS_WEIGHT`, default `0.15`, and `ANALOGY_CLASS_ANCHOR_WEIGHT`, default `0.1`.
- `ComposedRelationChain3CosAdd`: decomposes equations shaped as repeated `minus, plus` pairs, infers one relation prototype per explicit offset, then adds those prototypes to the start word. It falls back to raw arithmetic if any offset inference is below `ANALOGY_CHAIN_RELATION_MIN_SCORE`, default `0.25`.
- `PairRelationBlendRerank`: starts from blended relation ranking, then reranks the top `ANALOGY_PAIR_RERANK_K` candidates by relation-phrase similarity; defaults are 25 candidates, weight `0.2`, and `ANALOGY_PAIR_TEMPLATE=arrow`.
- `3CosMul`: useful in benchmark slices, but not the default because it is unstable across embedding families and loses some curated UI examples.

The backend relation prototype currently uses curated families for royal-title, country-capital, gender-feminine, and profession-place relations. This is a product prototype of the best non-oracle benchmark direction, not a complete relation model.

1.5 Results

Curated app probes:

Model	Method	Top-1
Xenova/all-mpnet-base-v2	3CosAdd	4/4
Xenova/bge-large-en-v1.5	3CosAdd	4/4
Xenova/bge-large-en-v1.5	Centered3CosAdd	4/4
Xenova/bge-base-en-v1.5	3CosAdd	2/4

Google analogy candidate-set sample, 14 categories x 20 examples:

Model	Method	Accuracy
Xenova/bge-large-en-v1.5	3CosAdd	85.0% (238/280)
Xenova/bge-large-en-v1.5	Centered3CosAdd	84.6% (237/280)
Xenova/bge-small-en-v1.5	Centered3CosAdd	78.2% (219/280)
Xenova/bge-small-en-v1.5	3CosAdd	77.5% (217/280)
Xenova/all-mpnet-base-v2	Centered3CosAdd	77.5% (217/280)
Xenova/all-mpnet-base-v2	3CosAdd	77.1% (216/280)
mixedbread-ai/mxbai-embed-xsmall-v1	3CosAdd	72.5% (203/280)
Xenova/all-MiniLM-L6-v2	Centered3CosAdd	72.1% (202/280)
Xenova/all-MiniLM-L6-v2	3CosAdd	71.4% (200/280)

Google analogy fixed-vocabulary sample, 14 categories x 20 examples:

Model	Method	Vocabulary	Devi
Xenova/bge-large-en-v1.5, `termTemplate=concept`	InferredRelationClassBlendScore	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelationClassPairRerank + ANN shortlist	20,045	Web
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore, 5 examples	20,045	Web
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore, 8 examples	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelationClassBlendScore	20,045	Web
Xenova/bge-large-en-v1.5	WeightedInferredRelationClassBlendScore	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelationClassMulScore	20,045	Web
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore, 3 examples	20,045	Web
Xenova/bge-large-en-v1.5	FewShotBlendedRelationClassBlendScore, 5 examples	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelationClassPairRerank	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelationClassPairRerank + cosine prefilter	20,045	Web
Xenova/bge-large-en-v1.5	3CosMul	20,045	Web
Xenova/bge-large-en-v1.5	BlendedRelationMulScore	20,045	Web
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	20,045	Web
Xenova/bge-large-en-v1.5	RelationClassBlendScore	20,045	Web
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	20,045	Web
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5	BlendedSoftInferredRelation3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelation3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5	FewShotRelation3CosAdd, 5 examples	20,045	Web
Xenova/bge-large-en-v1.5	3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5, `termTemplate=concept`	FewShotRelationClassBlendScore, 5 examples	20,045	Web
Xenova/bge-large-en-v1.5, `termTemplate=concept`	3CosAdd	20,045	Web
Xenova/bge-large-en-v1.5	InferredRelation3CosAdd	5,243	Web
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	5,243	Web
Xenova/bge-large-en-v1.5	3CosMul	5,243	Web
Xenova/bge-large-en-v1.5	3CosAdd	5,243	Web
Xenova/bge-large-en-v1.5	3CosMul	5,243	auto
Xenova/bge-large-en-v1.5	3CosAdd	5,243	auto
Xenova/all-mpnet-base-v2	3CosAdd	5,243	auto
Xenova/all-mpnet-base-v2	3CosMul	5,243	auto

Deployable stronger-model probe, Google fixed-vocabulary sample, 14 categories x 10 examples:

Model	Pooling	Method	Vocabulary	
Xenova/bge-large-en-v1.5	mean	FewShotRelationClassBlendScore, 5 examples	5,133	
Xenova/bge-large-en-v1.5	mean	InferredRelationClassBlendScore	5,133	
Xenova/bge-large-en-v1.5	mean	3CosAdd	5,133	
Xenova/gte-large	mean	3CosAdd	5,133	

Model	Pooling	Method	Vocabulary	
Xenova/gte-large	mean	InferredRelationClassBlendScore	5,133	
Xenova/gte-large	mean	FewShotRelationClassBlendScore, 5 examples	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	mean	3CosAdd	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	mean	InferredRelationClassBlendScore	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	mean	FewShotRelationClassBlendScore, 5 examples	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	last_token	3CosAdd	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	last_token	InferredRelationClassBlendScore	5,133	
onnx-community/Qwen3-Embedding-0.6B-ONNX	last_token	FewShotRelationClassBlendScore, 5 examples	5,133	

Term-template sweep on the same 5,133-word stronger-model slice:

- `onnx-community/Qwen3-Embedding-0.6B-ONNX`, last-token pooling, `termTemplate=concept`: 3CosAdd 58.6% (82/140), InferredRelationClassBlendScore 63.6% (89/140), FewShotRelationClassBlendScore 62.9% (88/140). This is a large recovery over bare words but still below BGE.
- `onnx-community/Qwen3-Embedding-0.6B-ONNX`, last-token pooling, `termTemplate=lexical-concept`: 3CosAdd 57.9% (81/140), InferredRelationClassBlendScore 60.7% (85/140), FewShotRelationClassBlendScore 58.6% (82/140). The longer instruction-style wrapper is worse than the short concept phrase.
- `onnx-community/Qwen3-Embedding-0.6B-ONNX`, last-token pooling, `termTemplate=qwen-analogy`: 3CosAdd 47.9% (67/140), InferredRelationClassBlendScore 54.3% (76/140), FewShotRelationClassBlendScore 54.3% (76/140). The explicit Qwen-style query instruction is worse than a simple phrase for this symmetric lexical task.
- `Xenova/bge-large-en-v1.5`, mean pooling, `termTemplate=concept`: 3CosAdd 77.1% (108/140), InferredRelationClassBlendScore 82.9% (116/140), FewShotRelationClassBlendScore 82.9% (116/140). This small-slice gain motivated the exact 20k run above.
- `Xenova/gte-large`, mean pooling, `termTemplate=concept`: 3CosAdd 74.3% (104/140), InferredRelationClassBlendScore 81.4% (114/140), FewShotRelationClassBlendScore 80.0% (112/140). Concept wrapping improves GTE relation-aware scoring but hurts raw arithmetic.

Method probe on 8 categories x 20 examples:

Model	Method	Accuracy
Xenova/all-mpnet-base-v2	3CosAdd	71.9% (115/160)
Xenova/all-mpnet-base-v2	Centered3CosAdd	72.5% (116/160)
Xenova/all-mpnet-base-v2	CSLS3CosAdd	65.6% (105/160)
Xenova/all-mpnet-base-v2	3CosMul	42.5% (68/160)
Xenova/all-MiniLM-L6-v2	3CosAdd	70.6% (113/160)
Xenova/all-MiniLM-L6-v2	Centered3CosAdd	71.9% (115/160)
Xenova/all-MiniLM-L6-v2	CSLS3CosAdd	68.1% (109/160)
Xenova/all-MiniLM-L6-v2	3CosMul	60.6% (97/160)

BATS candidate-set sample, 40 categories x 30 examples:

Model	Method	Accuracy	Interp
Xenova/bge-large-en-v1.5, `termTemplate=concept`	RelationClassBlendScore	72.0% (864/1200)	New st
Xenova/bge-large-en-v1.5, `termTemplate=concept`	InferredRelationClassBlendScore	68.8% (826/1200)	New b
Xenova/bge-large-en-v1.5, `termTemplate=concept`	FewShotRelationClassBlendScore, 5 examples	68.1% (817/1200)	Improv
Xenova/bge-large-en-v1.5, `termTemplate=concept`	3CosAdd	60.6% (727/1200)	Unlike
Xenova/bge-large-en-v1.5	RelationClassBlendScore	70.2% (842/1200)	Catego
Xenova/bge-large-en-v1.5	InferredRelationClassBlendScore	67.3% (807/1200)	Best n
Xenova/bge-large-en-v1.5	RelationAvg3CosAdd	66.1% (793/1200)	Relatio
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	65.8% (790/1200)	Arrow
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	65.2% (782/1200)	Earlier
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	64.5% (774/1200)	Tuned
Xenova/bge-large-en-v1.5	BlendedRelationMulScore	64.1% (769/1200)	Weight

Model	Method	Accuracy	Interpr
Xenova/bge-large-en-v1.5	InferredRelation3CosAdd	63.7% (765/1200)	Best n
Xenova/bge-large-en-v1.5	BlendedSoftInferredRelation3CosAdd	63.0% (756/1200)	Soft re
Xenova/bge-large-en-v1.5	AdaptiveBlendedInferredRelation3CosAdd	63.7% (765/1200)	Best a
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	62.7% (752/1200)	Improv
Xenova/bge-large-en-v1.5	3CosMul	60.1% (721/1200)	Helps
Xenova/bge-large-en-v1.5	3CosAdd	57.3% (688/1200)	Raw a

BATS fixed-vocabulary sample, 40 categories x 30 examples, 20,971 candidate words:

Model	Method
Xenova/bge-large-en-v1.5	RelationClassBlendScore
Xenova/bge-large-en-v1.5, `termTemplate=concept`	RelationClassBlendScore
Xenova/bge-large-en-v1.5	InferredRelationClassBlendScore
Xenova/bge-large-en-v1.5	WeightedInferredRelationClassBlendScore
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore, 5 examples
Xenova/bge-large-en-v1.5	InferredRelationClassMulScore
Xenova/bge-large-en-v1.5, `termTemplate=concept`	FewShotRelationClassBlendScore, 5 examples
Xenova/bge-large-en-v1.5, `termTemplate=concept`	InferredRelationClassBlendScore
Xenova/bge-large-en-v1.5	PairRelationBlendRerank
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd
Xenova/bge-large-en-v1.5	GatedInferredRelationClassBlendScore, score >= 0.5 and margin >= 0.05
Xenova/bge-large-en-v1.5, `termTemplate=concept`	3CosAdd
Xenova/bge-large-en-v1.5	3CosAdd

The exact candidate-vector index reproduced these headline fixed-BATS numbers in a fresh WebGPU-cache run after the scoring-path change: `3CosAdd` 42.9%, `InferredRelationClassBlendScore` 54.6%, `BlendedInferredRelation3CosAdd` 47.2%, and `PairRelationBlendRerank` 48.9%. The first hard-gated relation-class prototype is also a negative result. A permissive disabled gate, `relationClassGateMinScore=-1` and `relationClassGateMinMargin=-1`, exactly reproduced ungated `InferredRelationClassBlendScore` at 54.6%. Thresholds at score `0.5` and margin `0.05` fell to 45.2%, and score thresholds from `-0.9` through `0.4` or small margin thresholds collapsed to raw fixed-BATS performance, 42.9%. Target-agreement gating was also too coarse: thresholds from `-0.5` through `0.6` reproduced the ungated 54.6%, while `0.8` dropped to 51.8%. Fixed-Google target-agreement thresholds `0.6`, `0.8`, and `0.9` all reproduced the ungated automatic relation-class result, 78.9%. Raw prototype cosine, simple margin, and raw/prototype target agreement are therefore insufficient as standalone confidence features for BGE relation-class arithmetic; future gates need learned reliability features such as base-rank margin, candidate overlap, class-score lift, or held-out relation-family calibration.

The first learned raw-vs-class selector is also negative. `ValidatedRelationClassCentroidSelector` was evaluated on three deterministic 50/50 fixed-BATS splits. With `centroid-v1`, selected scoring reached 50.3% (303/602), while always-raw was 43.5%, always-class was 55.6%, and the per-record raw/class oracle was 56.1%. With `centroid-v2`, selected scoring reached 51.9% (307/592), always-class 57.1%, and oracle 57.8%. With `centroid-v3`, selected scoring reached 49.1% (298/607), always-class 55.0%, and oracle 55.7%. The useful interpretation is not "centroid failed" alone; it is that these relation-level diagnostics do not distinguish the small set of records where raw scoring beats class scoring. The raw/class oracle headroom is below one top-1 point on these splits, so the next learned selector needs a broader portfolio or candidate-rank features, not just relation diagnostics.

The first broader scorer-portfolio selector is a small positive result, but not yet enough to promote. `ValidatedRelationPortfolioC` chooses among `InferredRelationClassBlendScore`, `FewShotRelationClassBlendScore`, and raw `3CosAdd` from relation score, relation margin, and raw/prototype target agreement. On `portfolio-v1`, selected scoring reached 54.8% (325/593), beating always-class at 53.0%, always-few-shot at 54.0%, and always-raw at 43.7%; oracle was 59.4%. On `portfolio-v2`, selected scoring reached 55.8% (346/620), narrowly beating always-class at 55.6% and few-shot at 55.2%; oracle was 61.1%. On `portfolio-v3`, selected scoring reached 56.8% (359/632), beating tied class/few-shot baselines at 55.5%; oracle was 61.1%. This validates the broader portfolio direction, but the remaining oracle gap is still large enough that the selector is a research prototype rather than a UI default.

The rank-surface extension is a controlled negative result. `ValidatedRelationPortfolioRankCentroidSelector` adds answer-free candidate diagnostics to the same three-way portfolio: each method's top score, top-score margin, top-5

score spread, top-1 agreement, and top-5 overlap. On the same splits it reached only 51.8% (307/593), 54.0% (335/620), and 54.1% (342/632), each below the strongest always-class or always-few-shot baseline. The likely failure is not that candidate-surface features are useless; it is that nearest-centroid labels are too blunt for reliability calibration once the features become mixed-scale and method-specific. The next selector should use a validation-optimized objective over top-1/MRR or a constrained logistic/gated model, not another unconstrained centroid over more features.

The validation-selected portfolio gate is the strongest learned selector in this branch so far. `ValidatedRelationPortfolioGateSele` searches single-feature gates on the train partition, ranks candidates by top-1, then MRR, then top-5, and applies the selected gate to held-out records. Across `portfolio-v1`, `portfolio-v2`, and `portfolio-v3`, it selected almost the same rule: use `InferredRelationClassBlendScore` when `relationScore` is above roughly `0.366`, otherwise use `FewShotRelationClassBlendScore`; raw scoring was never selected. Held-out results were 55.3% (328/593), 57.7% (358/620), and 57.8% (365/632) top-1, beating the strongest always-class/few-shot baseline by 1.3, 2.1, and 2.3 points respectively, and beating the centroid portfolio selector by 0.5, 1.9, and 1.0 points. It still trails the oracle portfolio by 3.3 to 4.1 points, so the objective is not complete, but the evidence now supports a calibrated class-vs-few-shot gate as the next deployable-candidate direction.

`ValidatedRelationPortfolioRankGateSelector` tested whether answer-free rank-surface features improve that gate. On `portfolio-v1`, it selected the exact same `relationScore` rule and reproduced the same 55.3% top-1, 76.1% top-5, and 0.646 MRR. This is better than the rank-surface centroid result, but it does not justify the extra rank-surface cost yet. A full three-split rank-gate sweep should wait until the feature set includes more targeted reliability signals.

`NestedValidatedRelationPortfolioGateSelector` is the stricter overfit check on that same portfolio. It selects candidate gates with an inner validation split inside the outer train partition before applying the result to the outer held-out records. On fixed-BATS it still beats the strongest always-class/few-shot baseline on all three outer splits, but gives back the larger non-nested gains: `portfolio-v1` stays at 55.3% (328/593) top-1 with 76.1% top-5 and 0.646 MRR, `portfolio-v2` drops from 57.7% to 56.3% (349/620) with 78.5% top-5 and 0.658 MRR, and `portfolio-v3` drops from 57.8% to 56.3% (356/632) with 76.1% top-5 and 0.654 MRR. The selected rules remain class-vs-few-shot relation-score thresholds, but the thresholds vary more, from `0.1436` to `0.3849`. This keeps the portfolio-gate direction alive while downgrading the earlier larger gains as likely train-split calibration overfit.

`GuardedNestedValidatedRelationPortfolioGateSelector` tested the obvious conservative fix: select the inner-validation winner only if it beats the best inner-validation constant label by at least one or two top-1 hits. This was negative on fixed-BATS. With both `selectorMinCorrectGain=1` and `2`, the same target-agreement gates passed the guard and reached only 53.3% (316/593), 54.4% (337/620), and 57.0% (360/632) top-1 across `portfolio-v1`, `portfolio-v2`, and `portfolio-v3`. That is below the nested relation-score gate and below the strongest class/few-shot constant on two of the three splits. The reason is visible in the diagnostics: target-agreement gates had 2-4 extra inner-validation top-1 hits, but those gains did not transfer. A scalar validation-gain threshold is therefore not a reliable guard.

`RepeatedNestedValidatedRelationPortfolioGateSelector` tested whether several deterministic inner validation splits can replace a single fragile guard. With seven inner splits, `selectorMinWinRate=0.6`, `selectorMaxLossRate=0`, and `selectorMinCorrectGain=1`, fixed-BATS reached 53.8% (319/593), 57.7% (358/620), and 55.5% (351/632) top-1 across `portfolio-v1`, `portfolio-v2`, and `portfolio-v3`. This totals 1028/1845, above the per-split constant total of 1016/1845, but below the simpler nested gate's 1033/1845 and far below the non-nested gate's 1051/1845. The useful finding is that repeated validation can reject the v3 gate as unstable, but it still accepts a v1 relation-score gate that looks perfectly stable across inner splits and then loses one held-out hit to always-few-shot. Stability across resampled train records is not the same as transfer across held-out relation families.

`GroupedRepeatedNestedValidatedRelationPortfolioGateSelector` tested whether the answer-free inferred relation family can calibrate constants and gates better than one global rule. With `selectorMinGroupRecords=12`, fixed-BATS reached 55.0% (326/593), 55.5% (344/620), and 57.6% (364/632). The aggregate, 1034/1845, is one top-1 hit above the nested gate and six above the repeated gate, but the shape is not robust: v1 and v2 trail the nested gate, while v3 recovers most of the non-nested gate's gain. This makes inferred-family grouping a useful diagnostic and maybe a feature for a future calibrated model, but not a selector worth promoting by itself.

`ShrunkGroupedRepeatedNestedValidatedRelationPortfolioGateSelector` tested whether group-specific rules should be used only when they beat the global repeated rule on that inferred family. With `selectorMinGroupRecords=12` and `selectorMinGroupCorrectGain=3`, fixed-BATS reached 55.1% (327/593), 57.6% (357/620), and 57.8% (365/632). The aggregate, 1049/1845, is two hits below the non-nested gate's 1051/1845 and 16 above the nested gate's 1033/1845. It preserves most of the BATS gain while avoiding independent grouping's v2 loss, but it remains benchmark-only because the same selector still fails fixed-Google.

BATS few-shot relation-example sweep, 40 categories x 30 examples, 15 pairs per category, candidate set includes all selected category pairs:

Model	Method	Example pairs	Accuracy	Interpr
Xenova/bge-large-en-v1.5	RelationClassBlendScore	all held-out category pairs	70.9% (851/1200)	Categor
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore	8	68.9% (827/1200)	Best fe
Xenova/bge-large-en-v1.5	FewShotBlendedRelationClassBlendScore	8	68.8% (825/1200)	Blended
Xenova/bge-large-en-v1.5	FewShotBlendedRelationClassBlendScore	5	67.5% (810/1200)	Best 5-
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore	5	67.2% (806/1200)	Class-si
Xenova/bge-large-en-v1.5	RelationAvg3CosAdd	all held-out category pairs	65.9% (791/1200)	Categor
Xenova/bge-large-en-v1.5	FewShotBlendedRelation3CosAdd	8	65.6% (787/1200)	Nearly
Xenova/bge-large-en-v1.5	FewShotRelation3CosAdd	5	65.5% (786/1200)	Best fe
Xenova/bge-large-en-v1.5	FewShotBlendedRelationClassBlendScore	3	65.4% (785/1200)	Class-si
Xenova/bge-large-en-v1.5	FewShotRelationClassBlendScore	3	63.2% (758/1200)	Withou
Xenova/bge-large-en-v1.5	FewShotBlendedRelation3CosAdd	5	64.4% (773/1200)	Blendir
Xenova/bge-large-en-v1.5	InferredRelation3CosAdd	0 user examples	64.1% (769/1200)	Best au
Xenova/bge-large-en-v1.5	FewShotBlendedRelation3CosAdd	3	63.6% (763/1200)	Three u
Xenova/bge-large-en-v1.5	FewShotRelation3CosAdd	3	63.0% (756/1200)	Three e
Xenova/bge-large-en-v1.5	FewShotBlendedRelation3CosAdd	2	62.1% (745/1200)	Blendir
Xenova/bge-large-en-v1.5	FewShotRelation3CosAdd	2	59.9% (719/1200)	Two ex
Xenova/bge-large-en-v1.5	FewShotBlendedRelation3CosAdd	1	59.4% (713/1200)	One ex
Xenova/bge-large-en-v1.5	3CosAdd	0	58.0% (696/1200)	Raw of
Xenova/bge-large-en-v1.5	FewShotRelation3CosAdd	1	49.9% (599/1200)	A singl

This sweep supports the UI/API examples feature, but it also gives guardrails: the app should encourage multiple examples, blend or fall back when users provide only one sparse pair, and prefer class-side scoring once users provide roughly five or more examples. The result also supports a future app mode that labels examples as target-side evidence rather than using them only as an offset average.

The stricter fixed-Google few-shot run supports the same conclusion with a larger distractor set: `FewShotRelationClassBlendScor` reaches 79.6% top-1 at both five and eight examples, improving over the 78.9% automatic relation-class result. Offset-only few-shot scoring reaches only 75.0% at five examples, so the gain is not simply better offset averaging; it comes from using examples to identify the target side of the relation. The exact candidate-vector index reproduced the headline fixed-Google WebGPU-cache numbers after the scoring-path change: `3CosAdd` 74.3%, `InferredRelationClassBlendScore` 78.9%, `FewShotRelationClassBlendScore` 79.6%, `BlendedInferredRelation3CosAdd` 76.4%, and `PairRelationBlendRerank` 77.1%. The first fixed-Google portfolio-gate sweeps show why cross-benchmark validation is required. `ValidatedRelationPortfolioGateSelector` underperformed the best always-class or always-few-shot baseline on all three checked splits: 81.6% (115/141) versus a tied 82.3% baseline on `google-portfolio-v1`, 75.2% (100/133) versus 77.4% class on `google-portfolio-v2`, and 74.4% (99/133) versus 77.4% few-shot on `google-portfolio-v3`. The nested version is better but still not safe to promote. `NestedValidatedRelationPortfolioGateSelector` reaches 82.3% (116/141), 75.2% (100/133), and 78.2% (104/133) across the same splits. That ties the best constant baseline on v1 and beats it by 0.8 points on v3, but still loses v2 by 2.2 points. Averaged across these split sizes, the nested gate improves the original gate from 314/407 to 320/407 top-1, but the per-split best constant baseline is 322/407. This is a useful robustness improvement, not a deployable universal selector. The guarded nested fixed-Google result is also not enough. With `selectorMinCorrectGain=1`, it rejected the v1 and v2 gates and matched the inner-selected constant label at 82.3% (116/141) and 77.4% (103/133), but accepted a one-hit inner-validation v3 gate that fell to 75.2% (100/133). With `selectorMinCorrectGain=2`, it rejected that v3 gate too, but the inner-selected constant label was still class, also 75.2%, while held-out few-shot reached 77.4%. Both guarded settings therefore total 319/407 top-1 across the three fixed-Google splits, below the unguarded nested result at 320/407 and below the per-split best constant result at 322/407. This shows that both gate selection and constant-label selection are split-sensitive. The repeated nested fixed-Google result is worse. With the default repeated guard (`selectorMinWinRate=0.6`, `selectorMaxLossRate=0`), it reaches 82.3% (116/141), 75.2% (100/133), and 75.2% (100/133), or 316/407 total. A stricter `selectorMinWinRate=0.8` rejects the v2 gate and improves that split to 76.7% (102/133), but the total is still only 318/407. The remaining errors are exactly the dangerous ones for a deployable selector: the repeated validation baseline chooses few-shot on v2 even though class wins held-out by one hit, and chooses class on v3 even though few-shot wins held-out by three hits. A robust selector therefore needs uncertainty over constant labels,

not only uncertainty over non-constant gates. The grouped repeated fixed-Google result is a stronger negative. Sweeping `selectorMinGroupRecords` over `5`, `8`, and `12` did not recover the lost constant-label decisions. At the most conservative checked setting, `12`, grouped repeated reaches 81.6% (115/141), 73.7% (98/133), and 77.4% (103/133), or 316/407 total. Smaller group thresholds also total 316/407, with worse v1 behavior because sparse family-specific rules choose raw scoring too often. Inferred-family calibration is too data-hungry for this small Google slice and does not solve cross-benchmark reliability. The shrunk grouped fixed-Google result did not recover. Sweeping `selectorMinGroupCorrectGain` over `1`, `2`, and `3` yielded the same 80.9% (114/141), 76.7% (102/133), and 75.2% (100/133), or 316/407 total. A stricter global and group `selectorMinWinRate=0.8` also totaled 316/407. Shrinkage suppresses some sparse group damage but cannot fix global fallback mistakes, so fixed-Google still blocks promotion. The relation-class plus log-3CosMul probe is a negative result. On fixed-Google, weights `0.01`, `0.03`, and `0.05` exactly tied `InferredRelationClassBlendScore`; weight `0.2` improved top-5 from 88.2% to 89.3% but reduced top-1 from 78.9% to 78.6% and MRR from 0.838 to 0.836. On fixed-BATS, weight `0.05` reduced top-1 from 54.6% to 54.2% and top-5 from 75.3% to 74.6%. This suggests the multiplicative signal is mostly redundant or destabilizing once class-side relation evidence is included. The weighted relation-class probe is also a negative result at the default top-5 softmax setting. It tested whether uncertain relation-family inference should blend class axes rather than commit to one hard class. Fixed-Google tied the hard class scorer exactly at 78.9% top-1 / 88.2% top-5 / 0.838 MRR, while fixed-BATS slipped from 54.6% to 54.5% top-1, 75.3% to 74.9% top-5, and 0.635 to 0.634 MRR. The current evidence says class-side calibration needs learned confidence features, not just a softmax over relation prototype cosine. The first exact class-plus-pair hybrid result is not strong enough to promote: `InferredRelationClassPairRerank` ties `InferredRelationClassBlendScore` on fixed-Google top-1/MRR and only nudges top-5 from 88.2% to 88.6%. A full fixed-BATS hybrid run emitted the baseline `InferredRelationClassBlendScore` result, 54.6%, but the hybrid stage was stopped because pair-phrase embedding/scoring did not complete in a practical window. After adding the JSONL pair-phrase cache, a 50-question fixed-BATS K=5 slice finished and tied the class baseline at 94.0% top-1, 96.0% top-5, and 0.956 MRR; a cache-hit repeat finished in 12.5 seconds. The full fixed-Google class-plus-pair run populated 6,680 WebGPU pair phrases, reproduced 78.9% top-1 / 88.6% top-5 / 0.838 MRR, and a cache-hit repeat finished in 1:05.6 instead of 1:55.1. After switching pair-phrase reads and embeddings to lazy per-item requests, the same cached full fixed-Google class-plus-pair run finished in 45.57 seconds, and the same 50-question fixed-BATS slice finished in 9.48 seconds with unchanged metrics.

The first ANN shortlist probe is mixed but useful. With direct Hamming-neighbor projection lookup, fixed-Google `InferredRelationClassPairRerank` improved from 78.9% top-1 / 0.838 MRR in 25.99s exact-cache-hit time to 80.4% top-1 / 0.847 MRR in 22.00s with `annMinCandidates=128`, `annMaxHammingRadius=2`, and `annCandidateLimit=512`. On the 50-question fixed-BATS slice, ANN preserved 94.0% top-1 / 0.956 MRR but was slower, 8.72s versus 6.09s exact, so this is not a general speed win yet. The diagnostic run explains why it should not be called a faithful approximation: the fixed-Google ANN pool averaged 287 candidates but contained only 8.0% of exact top-25 base candidates, with missing exact top-K candidates on all 280 questions; the BATS slice averaged 320.7 candidates but contained only 18.8% of exact top-5 base candidates, also missing exact top-K candidates on every question. A broader pool (`annMinCandidates=512`, radius `3`, limit `2048`) raises recall but still leaves the projection path far from exact: fixed-Google recall is 18.9% of exact top-25 with 988.7 average candidates and missing exact top-K candidates on all 280 questions, while the BATS slice reaches 47.6% exact-top-5 recall with 1267.3 average candidates and misses exact top-K candidates on 46/50 questions. The ANN result is best treated as a different candidate-generation experiment, not as a replacement for exact audit metrics.

The cosine prefilter is a better recall-aware shortlist baseline. It scans by target-vector cosine, keeps a small candidate pool, then applies the full base scorer only inside that pool. With a 64-candidate pool, fixed-Google keeps the exact class-plus-pair metrics at 78.9% top-1 / 0.838 MRR and recovers 98.3% of exact top-25 base candidates; the BATS 50-question slice keeps 94.0% top-1 / 0.956 MRR with 100.0% exact-top-5 recall. Runtime is roughly tied rather than faster: fixed-Google is 26.45s versus 25.99s exact, and BATS is 6.04s versus 6.09s exact. Larger cosine pools of 128 and 512 recover 99.9-100.0% fixed-Google exact top-25 and 100.0% BATS exact top-5, but diagnostic runs show the current JavaScript scan cost dominates. This is useful as a faithful candidate-generation baseline, not yet as a performance optimization.

BATS candidate-set sample, 40 categories x 60 examples:

Model	Method	Accuracy	Interpretation
Xenova/bge-large-en-v1.5	3CosAdd	58.9% (1413/2400)	Stronger than MPNet on broad BATS.
Xenova/bge-large-en-v1.5	Shift3CosAdd	58.5% (1404/2400)	Generic shift-margin prototype did not beat raw offs
Xenova/all-mpnet-base-v2	3CosAdd	55.8% (1338/2400)	Confirms BGE-large is better beyond Google.

Model	Method	Accuracy	Interpretation
Xenova/all-mpnet-base-v2	Shift3CosAdd	55.0% (1320/2400)	Generic shift-margin also hurts MPNet.
Xenova/all-mpnet-base-v2	CenteredShift3CosAdd	54.5% (1307/2400)	Centering plus shift is not a general fix.
Xenova/all-mpnet-base-v2	3CosMul	9.8% (236/2400)	3CosMul is unstable across embedding families.

BATS relation-perturbation sample, 40 categories x 30 examples:

Model	Method	No inserted cancellations	Two inserted `+x -x` cancellations	I
Xenova/bge-large-en-v1.5	InferredRelationClassBlendScore	67.8% (813/1200)	66.3% (795/1200)	E
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	65.9% (791/1200)	63.1% (757/1200)	P
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	64.6% (775/1200)	62.3% (747/1200)	P
Xenova/bge-large-en-v1.5	3CosAdd	58.0% (696/1200)	58.0% (696/1200)	P

This benchmark directly addresses the cancellation concern: in pure arithmetic the inserted terms do cancel, and raw 3CosAdd proves that by producing identical scores. The useful signal is that relation-aware rankers can still be affected by canceled terms through query-term selection, relation confidence gates, and pair-rerank phrase construction. The backend now chooses the surviving positive query term for pair reranking, so a chain such as `king - man + woman - woman + girl` compares `man -> king` against `girl -> candidate`, not `woman -> candidate`.

BATS composed-relation sample, 40 categories, up to 50 pairs per category, 394 mined non-canceling chains:

Model	Method	Top-1	Top-5	MRR	Interpretation
Xenova/bge-large-en-v1.5	OracleRelationPrototypeChain3CosAdd	48.0% (189/394)	72.6%	0.593	Relation-category ce
Xenova/bge-large-en-v1.5	ComposedPairRelationChainRerank	47.2% (186/394)	70.3%	0.577	Best non-oracle com
Xenova/bge-large-en-v1.5	RelationPrototypeChain3CosAdd	46.4% (183/394)	68.0%	0.568	Strong simple chain
Xenova/bge-large-en-v1.5	BlendedRelationPrototypeChain3CosAdd	39.8% (157/394)	66.5%	0.528	Blending helps keep
Xenova/bge-large-en-v1.5	3CosAdd	27.7% (109/394)	60.2%	0.433	Raw vector addition

Held-out threshold calibration on the same composed setup, with 204 chains for threshold tuning and 190 chains for evaluation:

Model	Method	Top-1	Top-5	MRR	Inte
Xenova/bge-large-en-v1.5	MultiFeatureBlendedRelationPrototypeChain3CosAdd	46.3% (88/190)	68.4%	0.566	Add
Xenova/bge-large-en-v1.5	OracleRelationPrototypeChain3CosAdd	46.3% (88/190)	71.6%	0.584	Rela
Xenova/bge-large-en-v1.5	ComposedPairRelationChainRerank	46.3% (88/190)	67.4%	0.567	Pair
Xenova/bge-large-en-v1.5	RelationPrototypeChain3CosAdd	45.8% (87/190)	67.4%	0.562	Best
Xenova/bge-large-en-v1.5	ConfidenceBlendedRelationPrototypeChain3CosAdd	45.8% (87/190)	67.4%	0.562	Tun
Xenova/bge-large-en-v1.5	MarginBlendedRelationPrototypeChain3CosAdd	45.8% (87/190)	67.4%	0.562	Tun
Xenova/bge-large-en-v1.5	CandidateMarginBlendedRelationPrototypeChain3CosAdd	45.8% (87/190)	67.4%	0.562	Tun
Xenova/bge-large-en-v1.5	CalibratedRelationPrototypeChain3CosAdd	43.2% (82/190)	64.2%	0.534	Tun
Xenova/bge-large-en-v1.5	ThresholdRelationPrototypeChain3CosAdd	38.9% (74/190)	63.2%	0.507	Fixe
Xenova/bge-large-en-v1.5	3CosAdd	26.8% (51/190)	57.4%	0.424	Raw

The threshold-calibration, confidence-blend, margin-blend, and candidate-margin results are useful negative results. A binary fallback decision recovers some mistakes from the fixed threshold, but it throws away too many useful low-score relation prototypes. Continuous blends over a single confidence signal avoid that regression but simply tie always applying prototypes. The multi-feature blend is the first positive calibration result, but the residual-size feature is mixed: agreement-only features reached 89/190 top-1 on this default split, while residual similarity reduces that to 88/190 and improves top-5 from 128/190 to 130/190. This remains a narrow result because top-5 and MRR still trail the oracle and the feature set was tuned on the same benchmark family.

A 2026-05-25 candidate-neighborhood overlap probe tested whether raw and prototype targets agreeing on their top candidate neighborhood should increase prototype confidence. It is implemented behind `--chainIncludeCandidateOverlap=true`,

but the default split did not select the overlap feature: calibrated weights stayed {"minScore":1,"minMargin":1,"rawPrototypeR and held-out metrics stayed 46.3% top-1 / 68.4% top-5 / 0.566 MRR. Runtime for that split rose from 63.80s to 101.58s, so this is currently a rejected optional feature rather than part of the default confidence search.

The first actual pair-phrase agreement probe is stronger. `ComposedPairRelationChainRerank` uses the prototype-chain target for its base shortlist, embeds the explicit chain phrase, and reranks candidate phrases. A cache-miss full run at weight `0.2` improved prototype-only from 183/394 to 185/394 top-1 and from 268/394 to 274/394 top-5. A default-split weight sweep found `0.35` stronger for top-1/MRR: 88/190 top-1, 128/190 top-5, 0.567 MRR. At `0.35`, the full 394-chain run reached 186/394 top-1, 277/394 top-5, and 0.577 MRR. A follow-up train-split calibration run chose weights `0.2`, `0.2`, `0.05`, and `0.5` on the four partitions; this underperformed fixed `0.35` on aggregate, so the fixed setting remains the stronger benchmark variant for now.

The first cascade that combined multi-feature confidence with pair-phrase agreement was also a negative result. `MultiFeatureComposedPairRelationChainRerank` uses the calibrated multi-feature chain target for the base shortlist before pair reranking. It reached 367/788 top-1, 554/788 top-5, and 0.576 MRR across four held-out partitions, losing three top-1 hits and two top-5 hits versus prototype-base pair reranking. `CalibratedMultiFeatureComposedPairRelationChainRerank` chose pair weights `0.2`, `0.1`, `0.05`, and `0.5`; it recovered top-5 parity at 556/788 but still trailed top-1 at 368/788 and MRR at 0.574. The practical lesson is that multi-feature confidence should become a joint reranker feature, not a separate target-selection stage before pair-phrase scoring.

The first joint grid-search reranker is not good enough either. `CalibratedJointComposedPairRelationChainRerank` reranks the union of raw-target and prototype-target top-K candidates using raw score, prototype score, pair-phrase score, and one pair-score interaction feature selected on the train split. It chose four different configs across the four partitions: `minScore`, no interaction, `rawPrototypeResidualSimilarity`, and `rawPrototypeAgreement`. The beta split gained one top-1 hit over pair-only, but default, alpha, and gamma lost enough that the aggregate fell to 365/788 top-1, 551/788 top-5, and 0.573 MRR. Runtime was also much worse: roughly 5.9-6.5 minutes per split versus about 2 minutes for the staged pair runs. This says the next joint model needs stronger regularization or a real learned objective, not a hand-grid over weak interactions.

The regularized pairwise learned reranker is a partial improvement over hand-grid search but still not strong enough. `LearnedJointComposedPairRelationChainRerank` learned dense weights from each train split with prototype score still dominant, plus pair-score and agreement/residual interactions. It improved top-1 on default and gamma and tied beta, but alpha collapsed from 87/189 pair-only to 81/189. Aggregate metrics were 366/788 top-1, 544/788 top-5, and 0.572 MRR. Runtime dropped to about 52 seconds per cache-hit split, but the learned feature set is not stable across relation mixtures. The next learned model should regularize by relation family or train on a larger hand-balanced composed benchmark rather than only mined BATS chains.

The relation-chain-specific learned reranker is the first learned composed-chain improvement over the fixed pair-only benchmark on aggregate top-1 and MRR. `FamilyLearnedJointComposedPairRelationChainRerank` trains a global fallback model and per-chain models for relation-chain keys that have at least three train examples. Across the same four split salts it reached 377/788 top-1, 547/788 top-5, and 0.582 MRR. That beats pair-only by seven top-1 hits and 0.003 MRR, and it avoids the prior learned model's alpha collapse, but it loses nine top-5 hits to pair-only and 24 top-5 hits to the oracle prototype reference. Per split, it beat pair-only on default (+4), beta (+4), and gamma (+1), but still trailed pair-only on alpha (-2). This is a stronger automatic composed-chain top-1/MRR result, not a solved arbitrary arithmetic model.

Naive count-based shrinkage does not repair the top-5 recall loss. `ShrunkFamilyLearnedJointComposedPairRelationChainReran` blends family weights back toward the global learner. With `familyShrinkageRecords=6`, it fell to 374/788 top-1, 543/788 top-5, and 0.579 MRR. A lighter `familyShrinkageRecords=2` setting reached the best top-1 so far, 378/788, but kept the same weak 543/788 top-5 and only tied unshrunk family MRR at 0.582. The useful finding is that some family overfitting exists, but count-only shrinkage is too blunt; the next objective needs to optimize top-5/MRR directly or learn when to shrink from validation features.

Held-out shrinkage gates are more conservative but still not a replacement. `ValidatedFamilyLearnedJointComposedPairRelatio` chooses each family blend alpha from a small validation slice inside the train partition, then retrains that family model on all available family training records. With validation fraction `0.35` and alpha grid `0,0.25,0.5,0.75,1`, mean selected alpha stayed low, between `0.161` and `0.267` across the four split salts. Aggregate metrics were 372/788 top-1, 548/788 top-5, and about 0.579 MRR. That recovers one top-5 hit versus unshrunk family learning and five versus count-shrinkage, but it loses five top-1 hits to unshrunk family and six to light count-shrinkage. The validation signal is too sparse per relation chain to select a robust alpha by itself; any future shrinkage gate needs pooled features

across families or cross-family validation, not isolated tiny within-family validation folds.

The first listwise softmax learner changes the failure mode but does not solve it. `ListwiseFamilyLearnedJointComposedPairRelat` trains each relation-chain family with a full-shortlist softmax cross-entropy update instead of pairwise hinge updates. A small default-split sweep tested `learningRate/temperature` pairs `0.01/0.35`, `0.03/0.35`, `0.03/0.70`, `0.01/0.70`, and `0.08/0.35`; `0.03/0.35` was the best recall setting on that split, reaching 86/190 top-1, 130/190 top-5, and 0.562 MRR. Across the four split salts it reached 352/788 top-1, 552/788 top-5, and about 0.566 MRR. That is five more top-5 hits than unshrunk family learning and four fewer than pair-only, but it loses 25 top-1 hits to family learning and 18 to pair-only. The useful finding is that shortlist-normalized learning does push probability mass toward recall, but without validation-selected blending or a different objective it demotes too many first-place answers.

The pooled family/pair blend is closer to the desired tradeoff but still not enough. `PairBlendedFamilyLearnedJointComposedPair` keeps the pairwise family learner, then calibrates one global alpha over `0,0.25,0.5,0.75,1` to interpolate family scores with a pair-only-style baseline score on the raw/prototype shortlist. The default top-1-first selector chose alpha `0.5`, `1`, `0.5`, and `1` on the default, alpha, beta, and gamma train splits. Held-out aggregate metrics were 374/788 top-1, 550/788 top-5, and about 0.580 MRR. That recovers three top-5 hits versus family learning while losing three top-1 hits, but it still trails pair-only by six top-5 hits and family/shrunk-family by top-1 and MRR. A follow-up `top5-with-top1-floor` selector with a `0.98` train top-1 floor produced the same four-split result, and alpha-split probes at looser `0.95` and `0.90` floors still selected full family. The useful finding is that pooled blending can reduce the recall loss without the severe listwise top-1 collapse, but train-selected scalar alpha still does not reliably recover pair-only recall; the alpha split needs additional selection features, not just a looser scalar objective.

Feature-gated family/pair blending is a small improvement over one global alpha but still not a new best benchmark. `GatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank` precomputes family-vs-pair diagnostics such as relation-chain confidence, score margins, family record count, and family/pair top-candidate agreement, then selects a one-feature threshold gate on the train split. The optimized implementation precomputes each record's candidate ranking at each candidate alpha once, so gate search is practical for split sweeps. With the same `top5-with-top1-floor` objective and `0.98` train top-1 floor, the four split salts selected different gates: residual similarity on default, family top margin on alpha, pair-baseline top margin on beta, and family count on gamma. Held-out aggregate metrics were 376/788 top-1, 549/788 top-5, and about 0.582 MRR. This beats the pooled global blend by two top-1 hits and roughly recovers family-level MRR while preserving two more top-5 hits than unblended family learning, but it still misses family/shrunk-family top-1 and pair-only top-5. The useful finding is that reliability features contain signal, but the next version needs a pooled multi-feature gate or learned meta-ranker instead of one thresholded feature.

Validation-selected feature-gated blending is the first automatic top-5 improvement over pair-only on the composed-chain split-resampling audit. `ValidatedGatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank` keeps the same bounded family/pair interpolation as the train-selected gate, but chooses the one-feature gate on an inner validation slice of the train partition. The final family scorer is then retrained on the full train partition and evaluated on the outer holdout. The four split salts selected family count, family count, minimum relation score, and minimum relation margin gates. Held-out aggregate metrics were 374/788 top-1, 559/788 top-5, and about 0.581 MRR. This beats pair-only top-5 by three hits and train-selected gating by ten hits while preserving MRR near the stronger family/gated methods, but it still gives up three top-1 hits versus unblended family learning and four versus light count shrinkage. The useful finding is that bounded validation selection can recover recall without the free meta-ranker's collapse.

A constrained multi-feature alpha model improves the top-1/top-5 balance but does not replace the validated one-feature gate. `ValidatedLinearPairBlendedFamilyLearnedJointComposedPairRelationChainRerank` searches small linear alpha formulas on the inner validation slice, using min-max-normalized reliability diagnostics and clamping alpha between pair-only and family-only scores. Across the four split salts it selected two-feature formulas such as `1 - minScore - familyPairTopAgreement`, `0.5 - rawPrototypeAgreement + familyCount`, `0.5 - minScore + candidateMargin`, and `0.5 - minMargin - candidateMargin`. Held-out aggregate metrics were 377/788 top-1, 551/788 top-5, and about 0.582 MRR. That matches unblended family learning on top-1/MRR and recovers four top-5 hits, but it loses eight top-5 hits to validated gating. The useful finding is that constrained multi-feature alpha is safer than free linear delta scoring, but the current grid optimizes a compromise rather than the recall-best surface; the next step should use a validation objective that explicitly optimizes a Pareto frontier between top-1 and top-5.

The absolute top-one-loss objective is a controlled negative result. `--pairFamilyBlendObjective=top5-with-top1-loss --pairFamilyBlendMaxTop1Loss=1` was added to avoid the awkwardness of ratio floors on tiny inner validation slices: candidates are allowed only one fewer top-1 validation hit than the best top-1 candidate, then sorted by top-5,

MRR, and top-1. On the default split it did exactly what was wanted: both validated gate and validated linear matched family top-1 at 92/190 and improved top-5 from 125/190 to 129/190. Across the four split salts, however, the validated gate reached 374/788 top-1, 558/788 top-5, and about 0.582 MRR, one top-5 hit below the earlier `top5-with-top1-floor` gate. The validated linear objective reproduced the prior 377/788 top-1 and 551/788 top-5 aggregate. A follow-up `--pairFamilyBlendMaxTop1Loss=2` sweep selected the same configs and reproduced the same held-out metrics on all four salts, so loosening the scalar budget is not enough. This means absolute loss is a cleaner selector knob, but not a better model. The next useful step is not another scalar acceptance rule; it should estimate uncertainty of the inner validation slice, or select gates with a cross-split/bootstrapped objective that penalizes split-specific top-1 losses.

Bootstrap-stable top-1 loss recovers the prior top-5 frontier but does not beat it. `--pairFamilyBlendObjective=top5-with-boots --pairFamilyBlendMaxTop1Loss=1 --pairFamilyBootstrapRounds=128 --pairFamilyBootstrapQuantile=0.9` keeps per-record calibration outcomes for each candidate, resamples the inner validation slice deterministically, and filters out candidates whose 90th-percentile top-1 loss exceeds one hit versus the resampled top-1 winner. On the alpha split this selected a different validated gate, `familyCount >= 1.5 ? alpha 1 : alpha 0`, improving held-out top-5 from 130/189 to 131/189 while keeping top-1 at 87/189. The four-split aggregate for the validated gate was 374/788 top-1, 559/788 top-5, and about 0.581 MRR, matching the earlier ratio-floor top-5 best and beating the absolute-loss gate by one top-5 hit. The validated linear aggregate stayed at 377/788 top-1, 551/788 top-5, and about 0.582 MRR. The useful finding is that bootstrap stability is a better selector diagnostic than a raw absolute loss count, but the ranking model itself is still bounded by the same family-vs-pair tradeoff.

The oracle portfolio gives a concrete upper bound for selector-only work. `OraclePairFamilyPortfolioRerank` evaluates pair-only, unblended family, bootstrap-validated gate, and bootstrap-validated linear scoring for each held-out record, then credits the best answer rank among those four outcomes. This is deliberately invalid as a learned model because it sees which scorer ranked the gold answer best, but it is useful for deciding whether more selector engineering is worthwhile. Across the four split salts it reached 388/788 top-1, 568/788 top-5, and about 0.597 MRR. That is eleven top-1 hits above family/validated-linear, nine top-5 hits above the validated gate, and roughly 0.015 MRR above the best real selector. The useful finding is that a better selector could still recover a small amount, especially top-5 recall, but the remaining gap is not large enough to justify many more scalar gates; the next meaningful improvement likely needs richer candidate generation, stronger relation representations, or a learned selector with real held-out labels across relation families.

The first deployable portfolio selector is a negative result. `NearestPairFamilyPortfolioRerank` uses the same ranker set as the oracle portfolio, but trains a nearest-neighbor selector from outer-train records only. Each training profile stores normalized reliability diagnostics plus the observed pair/family/gate/linear outcomes; each held-out record chooses the label whose nearest training neighbors have the strongest top-5, top-1, or MRR summary. On the default split, the answer-peeking oracle reached 93/190 top-1, 130/190 top-5, and 0.583 MRR, while the validated gate reached 92/190 top-1, 129/190 top-5, and 0.576 MRR. A sweep over objectives `top5`, `top1`, and `mrr` with `k` in `1,3,7,15` failed to recover that headroom: single-neighbor selectors exactly matched the train oracle at 102/204 top-1 and 75.0% train top-5 but fell to 91/190 top-1 and 126/190 top-5; the best held-out settings (`top1` or `mrr`, `k=3` or `k=7`) reached 92/190 top-1, 128/190 top-5, and at most 0.575 MRR. The useful finding is that local diagnostic neighborhoods are too noisy to identify which scorer will win on held-out examples. Future selector work should be cross-family validated and probably learn calibrated probabilities from richer features rather than copying nearest oracle labels.

Centroid smoothing is also a negative result. `CentroidPairFamilyPortfolioRerank` trains one normalized diagnostic centroid for each oracle-winning scorer label and picks the nearest label centroid at evaluation time. This is less variance-prone than nearest-neighbor copying, but the default split's oracle labels were highly imbalanced: `family:178`, `gate:23`, `pair:2`, `linear:1` over 204 train records. With the `top5` portfolio objective it reached 89/190 top-1, 129/190 top-5, and 0.568 MRR on the held-out split, below the validated label gate's 91/190 top-1, 130/190 top-5, and 0.574 MRR. With the `top1` objective it produced the same held-out result, while the corresponding label gate traded recall for top-1 at 91/190 top-1, 125/190 top-5, and 0.570 MRR. The useful finding is that smoothed oracle labels still do not provide enough minority-label signal; selector work needs either richer labels, balanced validation data, or a calibrated probabilistic objective rather than centroids over sparse winner classes.

A validation-selected scorer-label gate recovers a small amount of portfolio headroom without answer leakage. `ValidatedGatedPairFamilyPortfolioRerank` uses the same pair/family/gate/linear candidate outcomes as the oracle portfolio, but the train partition selects one reliability feature, threshold, direction, and pair of scorer labels on an inner validation slice. On the default split, the `top5` objective matched the oracle's 130/190 held-out top-5 while losing one top-1 hit to the scalar validated gate; `top1` and `mrr` objectives overfit the inner validation slice and fell to 125/190 top-5. Across the four split salts with the `top5` objective, the label gate reached 375/788 top-1, 561/788

top-5, and about 0.582 MRR. That is the best deployable top-5 result in this composed-chain harness so far, five hits above pair-only and two above the scalar validated gate, but it still leaves seven top-5 hits and thirteen top-1 hits to the oracle selector ceiling.

A first pooled multi-feature meta-ranker was negative. `MetaBlendedFamilyLearnedJointComposedPairRelationChainRerank` starts from the pair-baseline score, then learns pairwise hinge weights for the family-minus-pair score delta and interactions between that delta and every reliability diagnostic used by the gated method. It learned plausible positive interactions on raw/prototype agreement, residual similarity, family/pair top-candidate agreement, and family count, but the train objective pushed too much probability mass out of the top-five window. Across the four split salts, held-out metrics were 374/788 top-1, 530/788 top-5, and about 0.570 MRR. That ties the pooled global blend on top-1 but loses 20 top-5 hits and roughly 0.010 MRR. The useful finding is that unconstrained pairwise meta-learning over the family delta is too aggressive; a future meta-ranker needs explicit top-5/MRR validation, bounded alpha output, or a constrained listwise objective rather than a free linear delta scorer.

The direct top-5-aware training objective is also a negative result. `Top5FamilyLearnedJointComposedPairRelationChainRerank` only updates records whose expected answer is outside the target top-five window. That preserves pair-only top-5 on the default and alpha splits, but beta and gamma collapse sharply. Aggregate metrics fell to 341/788 top-1, 502/788 top-5, and 0.532 MRR. The failure mode is useful: simply stopping updates once the answer is top-5 removes too much rank-order pressure and lets family models learn brittle weights. A future top-5 objective needs a smoother listwise or reciprocal-rank loss, not a hard skip rule.

The first smoother reciprocal-rank objective did not solve that problem. `ReciprocalRankFamilyLearnedJointComposedPairRelat` updates answers even when they are already in the top five, weighting each pairwise correction by the reciprocal-rank gain from moving the answer above a higher-ranked negative. The naive setting, using the same step size as the hinge learner, reached only 350/788 top-1, 515/788 top-5, and 0.544 MRR because gamma again collapsed. A tuned conservative setting, learning rate `0.002` and only the top higher-ranked negative, recovered to 358/788 top-1, 533/788 top-5, and 0.565 MRR. That is much better than the hard top-five stop rule but still trails pair-only by 12 top-1 hits, 23 top-5 hits, and 0.014 MRR. The useful lesson is that reciprocal-rank weighting alone behaves like an overly weak family learner; the next attempt needs validation-selected update strength, listwise normalization over the whole shortlist, or a held-out shrinkage gate rather than another fixed update recipe.

Split-resampling check on the same 394-chain composed setup, using the default split plus salts `alpha`, `beta`, and `gamma`:

Model	Method
Xenova/bge-large-en-v1.5	OraclePairFamilyPortfolioRerank
Xenova/bge-large-en-v1.5	OracleRelationPrototypeChain3CosAdd
Xenova/bge-large-en-v1.5	ValidatedGatedPairFamilyPortfolioRerank
Xenova/bge-large-en-v1.5	ShrunkFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	FamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ValidatedLinearPairBlendedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ValidatedLinearPairBlendedFamilyLearnedJointComposedPairRelationChainRerank (`top5-with-top`
Xenova/bge-large-en-v1.5	GatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ValidatedGatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ValidatedGatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank (`top5-with-boo`
Xenova/bge-large-en-v1.5	ValidatedGatedPairBlendedFamilyLearnedJointComposedPairRelationChainRerank (`top5-with-top`
Xenova/bge-large-en-v1.5	PairBlendedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ValidatedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	MetaBlendedFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	CalibratedMultiFeatureComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	MultiFeatureComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	CalibratedComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	CalibratedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	LearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ListwiseFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	ReciprocalRankFamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	Top5FamilyLearnedJointComposedPairRelationChainRerank
Xenova/bge-large-en-v1.5	MultiFeatureBlendedRelationPrototypeChain3CosAdd

Model	Method
Xenova/bge-large-en-v1.5	RelationPrototypeChain3CosAdd
Xenova/bge-large-en-v1.5	3CosAdd

Per-split details:

Split	Train/Eval	Calibrated Parameters
default	204/190	multi-feature `{"minScore":1,"minMargin":1,"rawPrototypeResidualSimilarity":1}`; pair weight `0.`
alpha	205/189	multi-feature `{"minScore":1}`; pair weight `0.200`
beta	184/210	multi-feature `{"minScore":1}`; pair weight `0.050`
gamma	195/199	multi-feature `{"minMargin":1,"candidateMargin":1,"rawPrototypeResidualSimilarity":1}`; pair w

Smaller smoke slice, 40 categories, up to 20 pairs per category, 101 mined chains:

Model	Method	Top-1	Top-5	MRR	Interpretation
Xenova/bge-large-en-v1.5	ComposedPairRelationChainRerank	69.3% (70/101)	85.1%	0.759	Pair-chain reranking
Xenova/bge-large-en-v1.5	RelationPrototypeChain3CosAdd	68.3% (69/101)	86.1%	0.752	Strong on the mostly
Xenova/bge-large-en-v1.5	OracleRelationPrototypeChain3CosAdd	68.3% (69/101)	87.1%	0.756	Inference matched th
Xenova/bge-large-en-v1.5	BlendedRelationPrototypeChain3CosAdd	57.4% (58/101)	81.2%	0.689	Better than raw but
Xenova/bge-large-en-v1.5	3CosAdd	34.7% (35/101)	76.2%	0.545	Raw arithmetic is co

Chained five-operand diagnostic, 5 curated equations:

Model	Method	Top-1	Interpretation
Xenova/bge-large-en-v1.5	3CosAdd	5/5	Raw offset handles these cancellation-style chains cleanl
Xenova/bge-large-en-v1.5	PairRelationBlendRerank	5/5	Pair reranking preserved the raw-chain wins in this slice
Xenova/bge-large-en-v1.5	BlendedInferredRelation3CosAdd	5/5	Residual-preserving relation target plus chain confidence
Xenova/bge-large-en-v1.5	InferredRelationClassBlendScore	5/5	Stronger on standard analogy benchmarks; now falls ba

1.6 Backend Decision

The backend default is now `Xenova/bge-large-en-v1.5` with raw 3CosAdd. It was the best locally tested model that also preserved the app's visible examples.

The app now lets users switch between raw offset, inferred relation, blended relation, pair-relation reranking, and 3CosMul scoring. Backend responses return the chosen method and, when applicable, the inferred relation family. A WebGPU blend sweep moved the blended relation default from `0.35` to `0.55`, improving BATS from 62.7% to 64.5% while keeping the best tested blended fixed-Google result at 76.4%.

`InferredRelationClassBlendScore` is the best current automatic non-oracle prototype. It keeps the blended relation target, infers the relation family, then adds a small target-side class-axis score and a target-anchor score. It improves BATS from 65.8% to 67.3% over the previous pair-relation best, and improves fixed-Google from 77.1% to 78.9%, with MRR rising from 0.830 to 0.838. With `termTemplate=concept`, it reaches 68.8% on BATS and 80.4% on exact fixed-Google, but the concept wrapper is not promoted as a universal default because it regresses strict fixed-BATS automatic scoring from 54.6% to 51.3% and fixed-Google raw arithmetic/few-shot top-1. When user-style relation examples are available, raw-term `FewShotRelationClassBlendScore` remains stronger on fixed-Google at 79.6% top-1 and stronger on fixed-BATS at 54.3% top-1.

`GatedInferredRelationClassBlendScore` is retained only as a benchmark probe. Its first fixed-BATS and fixed-Google sweeps show that raw relation prototype score, margin, and raw/prototype target agreement are not reliable enough to decide when relation-class scoring should be trusted. The useful next selector should learn from richer diagnostics instead of applying a single hard threshold.

ValidatedRelationClassCentroidSelector is likewise retained only as a benchmark probe. Its held-out fixed-BATS splits show that nearest-centroid learning over relation score, margin, and target agreement chooses raw scoring too often and underperforms the always-class baseline. The remaining headroom is in candidate-ranking diagnostics or larger method portfolios, not in a raw-vs-class decision over these three relation features.

ValidatedRelationPortfolioGateSelector is the strongest learned selector in this branch. It replaces nearest-centroid label smoothing with a validation-selected top-1/MRR/top-5 gate, and consistently chooses a relation-score threshold that switches between automatic relation-class scoring and five-example few-shot relation-class scoring. It improves over the strongest fixed-BATS baseline by 1.3 to 2.3 points across three deterministic portfolio splits and over the centroid portfolio by 0.5 to 1.9 points. It still trails the per-record oracle by 3.3 to 4.1 points, so the current research direction is not done; the next useful step is a still-constrained multi-feature selector that preserves this simple gate's stability while recovering more oracle headroom.

NestedValidatedRelationPortfolioGateSelector is a stricter version of that gate, and the result is mixed enough that the selector should stay benchmark-only. On fixed-BATS it still beats the strongest class/few-shot baseline on all three splits, but it reduces the v2 and v3 gains from 2.1-2.3 points to 0.7-0.8 points. On fixed-Google it improves the original gate and reaches 320/407 top-1 over three held-out splits, but the per-split best constant baseline reaches 322/407. That means inner validation helps with overfit but does not yet produce a benchmark-universal scorer selector.

GuardedNestedValidatedRelationPortfolioGateSelector is a negative result. It makes the selection rule sound more conservative, but the validation slices still accepted unstable BATS target-agreement gates and unstable fixed-Google constant labels. This rules out a simple "require N extra validation hits" guard as the next selector. A stronger selector needs repeated split evidence, a prior against known-unstable feature families, or calibrated uncertainty around validation gains.

RepeatedNestedValidatedRelationPortfolioGateSelector is a controlled negative-to-mixed result. It adds the repeated split evidence that the guarded selector lacked, and it does reject one unstable fixed-BATS gate, but it still trails the nested gate on fixed-BATS and falls behind nested, guarded, and per-split constants on fixed-Google. The failure mode is now sharper: repeated validation can make an unstable decision look stable when the train split lacks the held-out relation-family mix, and the fallback constant label itself needs calibration.

GroupedRepeatedNestedValidatedRelationPortfolioGateSelector is also benchmark-only. Using inferred relation family as a group key recovers one aggregate fixed-BATS hit over the nested gate, but it regresses fixed-Google to 316/407. That means relation family is a real reliability feature, but naive per-family calibration overfits sparse families and should be replaced by a pooled or shrinkage model.

ShrunkGroupedRepeatedNestedValidatedRelationPortfolioGateSelector is the best fixed-BATS selector so far among the stricter validation variants, but it is still not deployable. It reaches 1049/1845 top-1 on fixed-BATS, nearly recovering the non-nested gate while adding a real family-level acceptance guard. It still reaches only 316/407 on fixed-Google, so the next selector has to calibrate the global fallback and constant-label choice, not only shrink family-specific gates.

ValidatedRelationPortfolioRankCentroidSelector and the one-split ValidatedRelationPortfolioRankGateSelector show that merely adding top-candidate margins and method-overlap features is not enough. The centroid version underperformed the class/few-shot baselines on all three splits. The gate version ignored those features on the checked split and selected the same relation-score rule as the cheaper relation-only gate.

PairRelationBlendRerank remains useful as a prompt-based relation check. It keeps the blended relation target, takes the top 25 candidates, embeds pair phrases such as `man -> king` and `woman -> queen`, and adds a small relation-pair score. It improves BATS from 64.5% to 65.8%, fixed-Google from 76.4% to 77.1%, and fixed-BATS from 47.2% to 48.9%. Pure pair-relation reranking performed poorly, and fixed-BATS shows pair-rerank still trails relation-class scoring, so the pair relation signal is useful as a reranker, not as a replacement for class-side relation evidence.

For chained equations, the backend now preserves residual terms when applying an inferred relation. For example, it treats `king - man + woman - woman + girl` as replacing only the `king - man + woman` relation sub-expression, then carrying `- woman + girl` forward. It also gates low-confidence relation inference on chains with `ANALOGY_CHAIN_RELATION_MIN_SCORE`, default `0.25`, falling back to raw arithmetic when the inferred family is too weak.

A pair-template sweep tested `is-to`, `arrow`, `colon`, `from-to`, `relation`, and `becomes`. `arrow` was best balanced: fixed-Google `77.1%` and BATS `65.8%`. `colon` nearly matched BATS but trailed fixed-Google, while longer natural-language

templates underperformed.

A symmetric pair-relation test compared both `a -> b` vs. `c -> d` and `b -> a` vs. `d -> c`. A 50/50 inverse mix improved fixed-Google from `77.1%` to `77.5%`, but it regressed BATS from `65.8%` to `65.5%`. A smaller inverse weight of `0.15` removed the fixed-Google gain, so this is not promoted.

The bad/good stress case is handled by a ranker gate, not by pretending the equation has a true linguistic answer. For equations that subtract and add value terms, candidates must move in the requested value direction and must not be near-base aliases. This changes the observed failure case from entity echoes like `adolf/fuhrer/mussolini` to a more plausible association such as `churchill`.

`Shift3CosAdd` was prototyped as a generic arbitrary-equation ranker: keep the normal offset score, then reward candidates that are closer to added terms than subtracted terms. It improved some individual categories but lost overall on Google and BATS, so it is available for experiments through `ANALOGY_SCORING=Shift3CosAdd` but should not replace the default.

`RelationClassBlendScore` is the strongest relation-aware ceiling in the current benchmark. It adds target-side relation-class information and improves the 1,200-question BATS slice from 57.3% to 70.2%. This is not an arbitrary equation solver yet because it assumes the relation category is known. It shows that the next useful model should infer or ask for relation families, then apply class-side scoring.

`InferredRelation3CosAdd` infers a relation family from the observed offset and then applies the learned relation prototype. It improves BATS from 57.3% to 63.7% and improves the 20k Google fixed-vocabulary slice from 74.3% to 75.4%, but inferred class-side scoring is now the stronger non-oracle path.

`ComposedRelationChain3CosAdd` is now available in the backend and UI as the first app-facing prototype for multiple explicit relation offsets. It is not the default because the relation family inventory is still small, but it addresses the specific research gap exposed by the cancellation discussion: if the equation contains real `-x + y` relation examples rather than `+x -x` no-ops, the scorer can apply each learned offset separately. A low-confidence guard prevents it from forcing unrelated offsets into a relation family.

The API and UI now also accept user-supplied relation examples. Each line in the UI examples box is parsed as a pair such as `man -> king`, `doctor:hospital`, or `france,paris`. Relation-aware modes turn those pairs into a request-local `user-examples` prototype family and compare it against built-in relation families. For composed chains, lines can be scoped to a specific offset with forms such as `[1] source -> target` or `2) source -> target`; these become per-offset request families like `user-examples-1` and are only considered for the matching `minus, plus` pair. Example terms are embedded as support evidence but are excluded from answer candidates, so the model does not simply return one of the examples. Raw offset modes do not submit examples, preserving the default `king - man + woman = queen` behavior.

1.7 Next Research Steps

1. Build a cross-benchmark scorer selector that must beat the best constant baseline on both fixed-BATS and fixed-Google held-out splits before promotion. The nested gate improved over the first fixed-Google gate but still lost one split, the guarded nested gate showed that a simple validation-gain threshold is not enough, the repeated nested gate showed that resampling stability alone can still misselect both gates and constant labels, the grouped repeated gate showed that inferred-family calibration overfits sparse groups, and the shrunk grouped gate showed that family shrinkage helps BATS but still fails Google. The next selector should calibrate the global fallback and constant-label choice, model uncertainty over both non-constant gates and fallback constants, pool relation-family reliability, and report confidence intervals or bootstrap loss budgets rather than another scalar gate.
2. Optimize the recall-aware cosine prefilter with cached target-cosine shortlists or vectorized scoring, then validate on larger fixed-BATS slices. The projection ANN path can improve fixed-Google top-1 but has low exact-top-K recall; the cosine prefilter preserves exact behavior but is not faster yet.
3. Add an app-facing few-shot class-side mode once the UI can explain that example targets are used as target-side evidence, not just offset examples.
4. Validate scoped user-supplied examples on a larger composed-chain slice. The backend and UI now support named per-offset request families for composed chains; the next question is whether user-provided relation supports improve real candidate ranks beyond the current built-in family inference.
5. Add relation-aware modes for known categories: gender, capital-country, pluralization, comparative, superlative, tense, synonym, antonym, hypernym, and meronym.

6. Test transformer concept-subspace arithmetic for relation axes learned from prompt pairs.
7. Add a learned reranker over candidate lists so arbitrary equations can combine vector offset, relation direction, base-echo suppression, candidate type, and composed-chain confidence.
8. Expand the non-canceling composition benchmark beyond mined BATS overlap. Hand-curate balanced relation products such as gender plus family role, capital plus country, morphology plus tense, and adjective degree plus negation.
9. Replace the fixed chain confidence threshold with a learned or calibrated confidence model over relation score, margin, residual size, pair-rerank agreement, and chain length. The held-out threshold-only probe improved over the hand-set `0.25` threshold but lost to always applying prototypes, while score-only, margin-only, and candidate-margin continuous blends merely tied the prototype method. Four split-resampling runs now show the multi-feature calibration search gives a small but repeatable top-1 lift over always applying prototypes. Raw/prototype agreement gave the best aggregate top-1 so far, residual similarity improved top-5 but cost one top-1, and raw/prototype candidate-neighborhood overlap was rejected after adding runtime without being selected. Actual pair-phrase agreement remains a strong baseline. Naive train-split weight calibration, cascading multi-feature confidence, hand-grid joint reranking, and the first global pairwise learner all underperform pair-only. Relation-chain-specific pairwise learning is now the best top-1/MRR result, but it sacrifices top-5. Count-based shrinkage can add one top-1 hit at light strength but worsens top-5, held-out family alpha gates recover only one top-5 hit while losing too much top-1, a hard top-5 stop rule collapses, tuned reciprocal-rank still trails the simpler family hinge learner, the first full-shortlist softmax learner is recall-biased but demotes too many first-place answers, and pooled family/pair blending only partially recovers top-5. A top-5-first alpha selector with a train top-1 floor did not improve over top-1-first alpha selection. Label-portfolio selection recovered a small top-5 gain with a validated one-feature gate, but nearest-neighbor and centroid-smoothed oracle-label selectors both failed on the default split. The next version should learn selector probabilities from balanced held-out family features or expand the benchmark, rather than adding more scalar gates or unbalanced winner-label smoothers.

Current status: improved, measured, and deployable as an app visualization. Not a frontier benchmark result.