# Hybrid Retrieval Grant Search with Learned Feature Weights

**Nathan Eagle**
Granted AI
nathan@grantedai.com

## Abstract

We describe the design, implementation, and iterative optimization of a domain-specific search engine for U.S. federal and private grant opportunities. The system uses a four-stage hybrid retrieval pipeline—PostgreSQL full-text search, LLM query expansion, pgvector embedding $k$NN, and cross-encoder reranking—to retrieve candidates from $\sim$5,000 active grants ingested from 12 government and philanthropic data sources, and simultaneously fuses results from five frontier LLMs (Gemini, GPT-4.1, Claude, Grok, Perplexity) that search distinct web indices to discover grants beyond the curated catalog. All candidates—regardless of source—are scored using a unified 15-dimensional feature vector with learned weights, eliminating the score normalization problem inherent in multi-source fusion Shokouhi and Si [2011]. The dominant signal (49% of weight budget) is embedding cosine similarity from OpenAI's `text-embedding-3-small` Neelakantan et al. [2022]. We detail a knowledge distillation pipeline Hinton et al. [2015] that uses a frontier LLM (Claude Opus 4.6) to generate 1,034 relevance-labeled queries via two strategies (persona-derived and grant-centric synthetic generation Liu et al. [2024]), then applies stochastic hill climbing Kirkpatrick et al. [1983] to optimize feature weights against a composite objective of nDCG@10 Järvelin and Kekäläinen [2002], Precision@5, and poison rate. Over three optimization phases, validation P@5 progressed from 50.0% to **60.3%** while poison rate dropped from $\sim$42% to $\sim$**22%**, with the system maintaining sub-200ms scoring latency and $\sim$\$0.003/query marginal cost.

## 1 Introduction

### 1.1 Problem Statement

Grant seekers face a discovery problem: the U.S. federal government alone publishes $\sim$67,000 active opportunities through Grants.gov, with thousands more from NSF, NIH, SAM.gov, SBIR.gov, state portals, and private foundations. A nonprofit seeking environmental justice funding must sift through programs spanning defense, agriculture, healthcare, and education to find the handful that match their mission, eligibility, and geographic focus.

Existing grant search tools fall into two categories: (1) keyword-based portals (Grants.gov, SAM.gov) that return hundreds of results with no relevance ranking, and (2) commercial platforms (Instrumentl, GrantWatch) that charge subscription fees and still rely heavily on manual curation. Neither provides personalized, scored results that account for organization type, geographic eligibility, and topical relevance simultaneously. Recent advances in dense retrieval Karpukhin et al. [2020], learned sparse representations Formal et al. [2021], and LLM-based reranking Sun et al. [2023] suggest that modern IR techniques can address these shortcomings.

### 1.2 Design Goals

1. **Sub-second latency**—Scoring via precomputed features and dot-product; LLM calls (query expansion, cross-encoder) run in parallel with retrieval, adding $\sim$100–200ms.
2. **Interpretable scores**—Each grant receives a 1–99 fit score with human-readable match reasons derived from the feature vector.
3. **Low poison rate**—Results clearly irrelevant to a query should rarely appear in the top 10.
4. **Continuous improvement**—A reproducible evaluation framework with 1,034 labeled queries, automated baseline comparison, and cached-feature distillation enabling rapid weight iteration.

### 1.3 System Overview

The system operates as a four-stage hybrid retrieval pipeline:

```
Stage 1: RECALL (parallel retrieval)
  |-- PostgreSQL FTS (OR strategy) + ilike fallback
  |-- LLM Query Expansion (gpt-4.1-nano, ~100ms)
  |     -> Second FTS pass with expanded terms
  |-- Embedding kNN (pgvector, top 50)
  |     -> text-embedding-3-small cosine similarity
  --> Merged candidate pool (500-700 candidates)

Stage 2: SCORE (15-feature extraction + weighted sum)
  Per-candidate feature extraction -> dot product
  -> Scored, filtered, sorted candidates (Pass 1)

Stage 3: RERANK (cross-encoder on top 50)
  OpenAI gpt-4.1-nano (or Cohere Rerank)
  -> crossEncoderScore injected as feature #15
  -> Full re-scoring with all 15 features (Pass 2)

Stage 4: FILTER + PRESENT
  Hard filters, soft filters, deduplication
  -> Top 40 results with fit scores and reasons
```

## 2 Data Ingestion Pipeline

### 2.1 Data Sources

The system ingests grant opportunities from 12 active data providers spanning official government APIs, RSS feeds, state portals, and AI-discovered foundation grants (Table 1).

Table 1: Active data sources for grant ingestion.

| **Provider** | **Type** | **Cadence** | **Volume** |
|---|---|---|---|
| Grants.gov (XML) | Official API | Daily | $\sim$67,000 |
| SAM.gov | Official portal | 12h | $\sim$2,000–5,000 |
| NSF Funding | RSS feed | 12h | $\sim$500–1,000 |
| NSF Due Dates | RSS feed | 12h | $\sim$100–300 |
| California Grants | State portal | 12h | $\sim$200–500 |
| Challenge.gov | REST API | 24h | $\sim$100–500 |
| SBIR.gov | REST API | 12h | $\sim$50–200 |
| Texas eGrants | REST API | 24h | $\sim$100–300 |
| Foundation RFP Agent | AI-discovered | On-demand | $\sim$689 |

### 2.2 Normalization Pipeline

Each provider has a dedicated fetcher that normalizes records to a common schema before upserting to the `public_grants` table. The normalization pipeline applies: (1) HTML entity decoding (double-encoded entities), (2) deadline parsing to ISO 8601, (3) amount validation ($100 minimum for federal programs), (4) stable URL slug generation for deduplication, (5) quality scoring (composite 0–100 based on field completeness), (6) staleness bucketing, and (7) link health checking via HTTP HEAD with 4s timeout.

### 2.3 LLM-Based Grant Tagging

At ingestion time, each grant is asynchronously tagged using `gpt-4.1-nano` ($\sim$$0.001/grant). The tagger classifies against a fixed taxonomy of $\sim$180 topics across 12 categories. Tags are stored in a JSONB column indexed with a GIN index. As of February 2026, 4,596 of 4,597 active grants are tagged (99.98% coverage).

## 3 Feature-Based Scoring Architecture

### 3.1 Feature Vector

Each grant-query pair produces a 15-dimensional feature vector $\mathbf{f} \in \mathbb{R}^{15}$. Features are grouped into five categories.

**Text Relevance Features (4 dimensions).** `phraseMatchScore` $\in [0, 1]$: Compound phrase hits across grant fields using a dictionary of 111 known domain phrases. For each phrase found in the query, field-weighted matches are computed:

$$\texttt{phraseMatchScore} = \min\left(1, \ \frac{\sum_i \sum_j w_j \cdot \mathbf{1}[p_i \in \text{field}_j]}{|\text{phrases}|}\right) \tag{1}$$

where $w_j \in \{0.40, 0.22, 0.25, 0.13\}$ for name, summary, eligibility, and funder fields respectively.

`tokenMatchScore` $\in [0, 1]$: Individual token hits with the same field-weighted approach. Tokens shorter than 4 characters use word-boundary matching to prevent false positives.

`tokenCoverage` $\in [0, 1]$: Fraction of all query terms with at least one match in any grant field.

`topicOverlap` $\in [0, 1]$: Jaccard similarity between query topics (inferred via keyword matching against the 180-topic taxonomy) and grant topics (from pre-computed `grant_tags`):

$$\texttt{topicOverlap} = \frac{|Q_{\text{topics}} \cap G_{\text{topics}}|}{|Q_{\text{topics}} \cup G_{\text{topics}}|} \tag{2}$$

**Semantic Features (2 dimensions).** `embeddingSimilarity` $\in [0, 1]$: Cosine similarity between query and grant embeddings from OpenAI `text-embedding-3-small` Neelakantan et al. [2022] (1536 dimensions). Grant embeddings are pre-computed and stored with a pgvector IVFFlat index Johnson et al. [2019]. At query time, $k$NN retrieval returns the top 50 most similar grants:

$$\texttt{embeddingSimilarity} = \cos(\mathbf{e}_q, \mathbf{e}_g) = \mathbf{e}_q^\top \mathbf{e}_g \tag{3}$$

since OpenAI embeddings are $L_2$-normalized. This feature became the dominant ranking signal after Phase B distillation (49% of weight budget).

`crossEncoderScore` $\in [0, 1]$: Pointwise relevance from a cross-encoder Nogueira and Cho [2019] applied to the top 50 candidates after initial scoring. Two backends are supported: Cohere Rerank (`rerank-v3.5`) and an OpenAI `gpt-4.1-nano` fallback Sun et al. [2023].

**Metadata Features (5 dimensions).** `orgTypeMatch` $\in [-0.5, 1.0]$, `stateMatch` $\in [0, 1]$, `sourceQuality` $\in \{0, 0.5, 1.0\}$, `hasRfpUrl` $\in \{0, 1\}$, `hasAmount` $\in \{0, 1\}$. Organization type matching is gated by text relevance: grants matching org type but with zero focus hits are capped at 0.15, preventing metadata-only inflation.

**Freshness Features (3 dimensions).** `deadlineScore` $\in \{-1, 0, 1\}$, `freshnessScore` $\in \{0, 0.5, 1.0\}$, `qualityScore` $\in [0, 1]$.

**Penalty Features (1 dimension).** `nonUsScore` $\in \{-1, 0\}$: Binary penalty for non-US grants, triggered by detection of Canadian province names, Australian/UK/EU keywords.

### 3.2 Weight Vector and Score Combination

The final fit score is computed as a weighted linear combination with a two-tier mapping:

$$s_{\text{raw}} = \sum_{i=1}^{15} w_i \cdot f_i \tag{4}$$

$$s_{\text{text}} = f_{\text{phrase}} + f_{\text{token}} + f_{\text{coverage}} + f_{\text{topic}} + f_{\text{embed}} + f_{\text{crossenc}} \tag{5}$$

If $s_{\text{text}} < 0.05$ (no text/semantic signal):

$$\text{score} = \text{clamp}(1 + \max(0, s_{\text{raw}} \times 80),\ 1,\ 99) \tag{6}$$

Otherwise:

$$\text{score} = \text{clamp}(10 + s_{\text{raw}} \times 120,\ 1,\ 99) \tag{7}$$

The current production weight vector (Table 2) was learned via Phase B distillation.

Table 2: Production weight vector after Phase B distillation (1,034 queries).

| **Feature** | **Weight** | **Category** |
|---|---|---|
| `phraseMatchScore` | 0.034 | Text: 13.5% |
| `tokenMatchScore` | 0.021 | |
| `tokenCoverage` | 0.080 | |
| `topicOverlap` | 0.079 | |
| `embeddingSimilarity` | 0.489 | Semantic: 61.4% |
| `crossEncoderScore` | 0.046 | |
| `orgTypeMatch` | 0.010 | |
| `stateMatch` | 0.034 | |
| `sourceQuality` | 0.001 | Metadata: 12.8% |
| `hasRfpUrl` | 0.069 | |
| `hasAmount` | 0.014 | |
| `deadlineScore` | 0.001 | |
| `freshnessScore` | 0.014 | Freshness: 1.8% |
| `qualityScore` | 0.003 | |
| `nonUsScore` | 0.105 | Penalty: 10.5% |

### 3.3 Stop Word Architecture

A critical engineering detail is the separation of stop words into two tiers. **Retrieval stop words** (24 words): true function words removed before building the FTS query (“the”, “and”, “for”, “grant”, “funding”, etc.). **Scoring stop words** (31 words): domain-common words retained for retrieval recall but filtered from scoring feature extraction (“research”, “community”, “health”, etc.). This separation ensures that long queries retain domain terms for database retrieval while excluding them from the scoring denominator.

## 4 Retrieval and Filtering

### 4.1 Multi-Stage Candidate Retrieval

Retrieval runs three strategies in parallel, merging results into a unified candidate pool.

**Full-Text Search (PostgreSQL).** Query tokens are extracted, stop-word-filtered, and joined with `OR` for maximum recall. If FTS returns empty, an `ilike` fallback queries each token against name, funder, summary, and eligibility fields.

**LLM Query Expansion.** A lightweight `gpt-4.1-nano` call (~\$0.0001/query, ~100ms, 3s timeout) expands the user’s query with synonyms and related program names. Example: “food insecurity rural communities” $\rightarrow$ “hunger, nutrition assistance, SNAP, food bank, USDA, food desert…” This approach is related to HyDE Gao et al. [2023] and Query2Doc Wang et al. [2023], but uses direct term expansion rather than hypothetical document generation. Expanded tokens are used for a second FTS pass. On failure, the system degrades gracefully to original FTS results.

**Embedding $k$NN (pgvector).** The query is embedded via `text-embedding-3-small` Neelakantan et al. [2022] and matched against pre-computed grant embeddings using pgvector's IVFFlat index (top 50). This contributes 10–30 additional candidates not found by FTS, capturing semantic matches that lexical search misses Karpukhin et al. [2020].

**Merge Strategy.** All three stages produce candidate pools merged by slug. The final pool is typically 500–700 candidates.

### 4.2 Cross-Encoder Reranking (Pass 2)

After the initial scoring pass (Pass 1, using 14 features), the top 50 candidates are reranked: (1) Build a document string per candidate (name, funder, summary, eligibility; up to 4096 chars), (2) score each document against the query using the best available backend, (3) inject `crossEncoderScore`, (4) re-score all candidates with the complete 15-feature vector (Pass 2).

### 4.3 Hard and Soft Filters

**Hard filters** (pre-scoring): removal of entries with no name/funder, terminated programs, grants expired $> 30$ days, 28 entitlement program patterns (Social Security, SNAP, Medicaid, etc.), and SBIR/STTR statutory ineligibility by org type.

**Soft filters** (post-scoring): state filter (wrong-state grants below score 55 removed), score floor at $20\%$.

## 5 Multi-Provider Fusion: Unified Scoring Across Heterogeneous Sources

The retrieval pipeline described in Section 4 operates over a curated database of $\sim$5,000 indexed grants. However, the grant landscape extends far beyond any single catalog: new programs are announced on agency websites, state portals, and foundation pages daily, with no centralized registry. To maximize recall, the system simultaneously queries five frontier LLMs—each backed by a distinct web search index—and fuses their results with the database retrieval pipeline under a single, calibrated scoring function. This section describes the federated search architecture and the score normalization problem it solves.

### 5.1 The Score Normalization Problem

In a multi-source retrieval system, each source produces relevance estimates on its own internal scale Shokouhi and Si [2011]. Classical federated search Callan [2000] addresses this via score normalization techniques such as CombMNZ Fox and Shaw [1994] or Reciprocal Rank Fusion Cormack et al. [2009]. Our setting introduces a more acute variant: the "sources" are generative LLMs with web search capabilities Nakano et al. [2021], each returning self-reported relevance scores (termed `fit_score`) alongside structured grant metadata.

These LLM-reported scores exhibit three pathologies that make direct comparison impossible:

1. **Scale divergence.** One provider consistently assigns scores in $[80, 95]$ while another uses the full $[40, 95]$ range. There is no shared calibration target.
2. **Intra-provider inconsistency.** The same provider assigns different score distributions depending on query specificity—broad queries yield uniformly high scores (anchoring bias), while narrow queries produce wider variance.
3. **Cross-index incomparability.** Each provider searches a different web index (Google Search, Perplexity's crawler, OpenAI's search provider, Anthropic's search provider, xAI's search provider). A score of 85 from a provider that found the grant via a government RSS feed is not commensurable with a score of 85 from one that discovered it on a foundation's blog post.

Naïvely merging results by their self-reported scores produces rankings dominated by whichever provider inflates scores most aggressively—a degenerate outcome equivalent to letting the most optimistic judge determine the final ranking.

### 5.2 Architecture: Parallel Provider Orchestration

The authenticated search endpoint fires five LLM providers in parallel, each receiving an identical structured prompt containing the user's focus area, organization type, state, and a dynamically selected set of relevant federal agencies:

```
Parallel Provider Dispatch (5 providers, ~8-70s):
  |-- Gemini 3 Flash    (Google Search grounding)
  |-- Perplexity Sonar Pro (proprietary web crawler)
  |-- OpenAI GPT-4.1    (web_search tool)
  |-- Claude Haiku 4.5  (web_search tool)
  |-- Grok 4.1 Fast     (web_search tool)
  --> Raw candidate pools (10-15 grants each)
```

Each provider returns structured JSON conforming to a shared opportunity schema. Results stream to the client via NDJSON (newline-delimited JSON) over a `ReadableStream` as each provider completes—the fastest provider (typically Perplexity at $\sim$8s) delivers results while slower providers (Grok at $\sim$70s) are still searching. A 75-second per-provider timeout and 100-second global timeout bound worst-case latency.

Critically, each provider searches a *different underlying web index*. Google Search grounding accesses Google's crawler; Perplexity maintains its own independent index; OpenAI, Anthropic, and xAI each use proprietary or licensed search backends. This index diversity means the five providers collectively search a far larger surface of the web than any single provider, frequently discovering grants that exist only on specific agency websites, state portals, or foundation pages not yet ingested into the curated database.

### 5.3 Unified Scoring via Feature Projection

Our solution discards all LLM-reported scores and projects every candidate—whether from the curated database or from any LLM provider—through the identical 15-feature extraction pipeline (Section 3):

$$s_{\text{unified}}(q, g) = \sum_{i=1}^{15} w_i \cdot f_i(q, g) \quad \forall g \in \mathcal{G}_{\text{db}} \cup \mathcal{G}_{\text{llm}_1} \cup \cdots \cup \mathcal{G}_{\text{llm}_5} \tag{8}$$

This is not a post-hoc normalization (e.g., min-max rescaling per source or $z$-score standardization); it is a *complete re-evaluation* of each candidate against the learned feature weights. The scorer treats LLM-discovered grants identically to database grants: it extracts phrase matches, token coverage, topic overlap, organization type compatibility, geographic relevance, and all other features from the raw grant metadata. Embedding similarity and cross-encoder scores are computed only for database candidates (which have pre-computed embeddings); LLM-discovered grants receive zero for these features but can still achieve high scores through strong lexical and metadata alignment.

This design has a key theoretical property: the scoring function is *source-agnostic*. A grant's score depends solely on its textual content and metadata relative to the query—not on which provider discovered it, which search index it came from, or what confidence the originating LLM assigned. This eliminates the score normalization problem entirely rather than attempting to calibrate across incommensurable scales.

### 5.4 Incremental Deduplication with URL Resolution

Multi-source fusion introduces a deduplication challenge: the same grant frequently appears from multiple providers under slightly different names, varying summary lengths, or inconsistent metadata. We employ a two-layer incremental deduplication strategy that operates as each provider returns, rather than as a batch post-processing step:

**Layer 1: Exact key matching.** Each grant is assigned a canonical key $k$ = `lower`(name) | `lower`(funder). A shared `Set<string>` tracks all keys across the database results and all prior provider batches. Collisions trigger a merge: the duplicate's provider attribution is appended to the existing grant's `_providers[]` array, preserving multi-source provenance metadata.

**Layer 2: URL-based entity resolution.** Grants with distinct names but identical RFP URLs represent the same underlying opportunity (e.g., "NOAA Marine Debris Removal" vs. "FY25 Marine Debris Program"). A shared URL index maps normalized URLs to their owning grant's canonical key. When a background URL resolution service (Serper Google Search API) discovers that two grants share an RFP URL, the lower-scoring duplicate is merged into the higher-scoring entry.

This incremental architecture avoids the quadratic cost of all-pairs similarity computation and enables streaming: the client receives deduplicated results as each provider finishes, rather than waiting for all providers before deduplication begins.

### 5.5 Terminal Reranking

After all providers complete and deduplication converges, the top 40 candidates (drawn from both database and LLM sources) undergo a final listwise reranking pass via GPT-4.1-mini Sun et al. [2023]. The reranker sees each candidate's name, funder, summary, and eligibility text and produces a permutation of the candidate indices ordered by holistic relevance to the query. This terminal reranking serves as the final authority on ordering, capturing cross-candidate comparative judgments that the pointwise feature scorer cannot express.

The reranker fires unconditionally on every search with $\geq 5$ candidates ($\sim$\$0.001–\$0.005/call), replacing an earlier conditional strategy that only fired when score variance was low. Empirically, unconditional reranking improved result quality even for well-separated score distributions, because the reranker captures semantic nuances ("is this grant *actually* about the user's topic, or merely keyword-adjacent?") that the feature-based scorer cannot.

### 5.6 Provenance Tracking and Analysis

Every grant in the final result set carries a `_providers[]` array recording which sources contributed it. This enables post-hoc analysis of provider utility: which providers consistently discover grants that score highly under the unified scoring function, and which contribute mostly low-scoring or duplicate results. Structured events logged to the `app_events` table record per-provider result counts, duplication rates (both name-based and URL-based), and latency, enabling continuous monitoring of the federated pipeline's health.

We deliberately chose *not* to apply a provenance boost (e.g., $+5$ points for multi-provider grants) despite the intuition that cross-source consensus signals relevance Fox and Shaw [1994], Aslam and Montague [2001]. The terminal reranker already observes all evidence and can implicitly factor in multi-provider consensus, and a provenance boost risks artificially inflating mediocre grants that happen to appear in multiple indices while suppressing niche grants discoverable by only one specialized provider.

## 6 Knowledge Distillation Pipeline

### 6.1 Motivation

The 15-dimensional weight vector was initially set by domain intuition, then iteratively optimized through three phases of knowledge distillation Hinton et al. [2015] that transfer the relevance judgments of frontier LLMs into the lightweight linear scorer.

### 6.2 Label Generation

**Phase 7: Two-Tier Labeling (Failed Approach).** The initial approach used GPT-5-mini for both query generation and relevance labeling (251 queries), with 24 hand-crafted gold queries evaluated separately. This created a **calibration gap**: GPT-5-mini rates more conservatively (only 23% of candidates rated 2–3 vs. the gold standard's threshold), causing the optimizer to learn weights that improved nDCG $+7.8\%$ but regressed P@5 by $-11.8\%$. *Key insight:* Using different LLMs for training labels vs. evaluation labels creates divergent optimization targets.

**Phase 8: Unified Labeling (Production Approach).** All 224 queries were re-labeled by a single judge (Claude Opus 4.6) with consistent persona-grounded methodology. Queries span 13 taxonomy categories, 6+ organization types, and 43 U.S. states. Each candidate is rated on a 4-point scale: 3 (must-have), 2 (good), 1 (acceptable), 0 (irrelevant). The judge additionally identifies "poison" grants that should never appear.

**Label statistics:** 224 files, 5,530 ratings; distribution: 0=57.9%, 1=20.2%, 2=12.9%, 3=9.1%; 1,967 poison indices.

**Phase B: Grant-Centric Synthetic Expansion.** Phase 8's 224 queries provided only $\sim 15$ datapoints per weight parameter—insufficient for the optimizer to discover the importance of embedding and cross-encoder features. Phase B expanded to 1,034 queries via grant-centric generation: (1) 406 grants sampled from the database, stratified by category, (2) GPT-5-mini generates 2 queries per grant (2–8 words, different org types), (3) deduplicated via bigram Jaccard similarity ($> 0.7$ threshold), producing 812 new queries, (4) all labeled by GPT-5-mini using the same 4-point scale. This synthetic data expansion follows best practices outlined by Liu et al. [2024].

## 6.3 Train/Validation Split

The 1,034 queries are split 85%/15% using a deterministic MD5 hash of the query ID, yielding $\sim 889$ training and $\sim 145$ validation queries ($\sim 59$ datapoints per parameter).

## 6.4 Hill-Climbing Optimizer

The optimizer uses stochastic hill climbing with simulated annealing Kirkpatrick et al. [1983]. Perturbation magnitude decreases linearly from 0.4 to 0.1:

$$\text{mag}(i) = 0.4 \times (1 - i/I_{\text{max}}) + 0.1 \tag{9}$$

**Structural constraints.** (1) Text + semantic features $\geq 50\%$ of total weight, (2) no single weight $> 0.50$, (3) `nonUsScore` capped at 0.25 before normalization.

**Composite objective.**

$$J(\mathbf{w}) = 0.30 \cdot \text{nDCG@10} + 0.60 \cdot \text{P@5} + 0.10 \cdot (1 - \text{poisonRate}) \tag{10}$$

**Acceptance criteria.** A candidate is accepted iff: (1) $J(\mathbf{w}') > J(\mathbf{w}^*)$, (2) $\text{P@5}(\mathbf{w}') \geq 0.80 \times \text{P@5}_{\text{baseline}}$, and (3) $\text{P@5}(\mathbf{w}') \geq 0.30$.

## 6.5 Cached-Feature Distillation

The original distillation approach ran the full search pipeline per optimizer iteration per query. With 889 queries $\times$ 200 iterations = 177,800 evaluations, this was infeasible.

**Key insight:** Retrieval is constant across iterations—only weights change. Feature extraction is deterministic given the same grant row and search context. Only the dot product depends on weights.

**Solution:** Pre-compute feature vectors once per query-candidate pair ($\sim$64 minutes for 1,034 queries, $\sim$215K candidates), then optimize weights purely in-memory (78 seconds for 500 iterations). This achieves a $>$ 50,000$\times$ speedup.

## 6.6 Distillation Results

Table 3: Phase B distillation results (1,034 queries, 15 features, cached-feature approach).

| | nDCG@10 | | P@5 | | Poison | |
|---|---|---|---|---|---|---|
| **Dataset** | Phase A | Phase B | Phase A | Phase B | Phase A | Phase B |
| Train (889) | 65.3% | **72.6%** | 51.8% | **60.1%** | 42.1% | **22.4%** |
| Validation (145) | 63.9% | **70.4%** | 52.3% | **60.3%** | 42.8% | **21.6%** |

Key findings: (1) `embeddingSimilarity` surged from 0.047 to 0.489 (49% of weight budget), (2) `nonUsScore` dropped from 0.264 to 0.105 as embeddings provided better relevance discrimination, (3) lexical features dropped from $\sim 35\%$ to $\sim 5.5\%$, (4) validation metrics tracked training closely, confirming no overfitting.

## 7 Evaluation Framework

### 7.1 Metrics

Three metrics are computed per query and averaged:

**nDCG@10 Järvelin and Kekäläinen [2002].**

$$\text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}, \quad \text{DCG@}k = \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{11}$$

where IDCG is computed from actual results' matched relevances sorted optimally.

**P@5.** $\text{P@5} = |\{r \in \text{top-5} : \text{rel}(r) \geq 2\}| / \min(5, |R|)$. Only relevance levels 2 and 3 count as relevant.

**Poison Rate.** Fraction of queries with any clearly irrelevant result in the top 10.

### 7.2 Gold Standard

25 hand-labeled test queries derived from 18 QA personas (real organizational profiles) and 7 edge cases (very broad, very narrow, acronym-heavy, no-state, tribal, SBIR eligibility).

## 8 Production Bug Fixes and Lessons

**Long Query Failure.** Queries with 10+ words returned zero results due to cascading failures: overly aggressive stop words stripped all meaningful tokens, the token limit of 6 truncated survivors, and the score gate at 28% filtered remaining results. *Fix:* Three-tier stop word system, token limits raised to 15. *Lesson:* Stop words for scoring precision can catastrophically break retrieval recall—the two concerns must be separated.

**SBIR Statutory Eligibility.** SBIR/STTR grants appeared for ineligible org types because the soft penalty (`orgTypeMatch` weight 0.011) only cost ∼3% score. *Fix:* Hard filter before scoring. *Lesson:* Statutory eligibility rules should be hard filters, not soft scoring signals.

**State Filter Coupling.** A tribal college in North Dakota went from 34 results to 1 after a weight update because the `nonUsScore` penalty (0.408) dropped grants below the state filter threshold calibrated for pre-update weights. *Lesson:* Filter thresholds are coupled to the weight vector and must be re-calibrated when weights change.

**nDCG Metric Inflation.** Reported nDCG of 93.3% masked true performance of 72.3% because IDCG was computed from the expected list rather than actual results. *Lesson:* Always validate metric implementations against known-good examples.

## 9 AI Model Inventory and Costs

Table 4: Development and operational costs.

| Phase | One-Time | Per-Query |
|---|---|---|
| Phases 0–P8 (scoring + eval) | ∼\$3 | \$0 |
| Phase A (embed + expand + rerank) | ∼\$0.01 | ∼\$0.003 |
| Phase B (synthetic data + distill) | ∼\$10 | \$0 |
| **Total** | ∼\$13 | ∼\$0.003 |

The per-query cost is dominated by cross-encoder reranking (∼\$0.001). At 1,000 queries/day, this is ∼\$90/month.

The retrieval pipeline uses four model calls per query: query expansion (`gpt-4.1-nano`, $\sim$\$0.0001), query embedding (`text-embedding-3-small`, $\sim$\$0.000002), pgvector $k$NN (database, \$0), and cross-encoder reranking (`gpt-4.1-nano`, $\sim$\$0.001).

## 10 System Evolution

Table 5: Metrics evolution across development phases. Early phases use the gold standard (24 queries); search upgrade phases use the validation set (145 queries).

| Phase | nDCG | P@5 | Poison | Key Change |
|---|---|---|---|---|
| *Gold standard evaluation (24 hand-labeled queries):* | | | | |
| Pre-P1 | $\sim$45% | $\sim$35% | $\sim$20% | No eval framework |
| P1–P5 (inflated) | 93.3% | 52.5% | 0.0% | nDCG bug masked perf. |
| P6: Corrected | 72.3% | 65.8% | 4.2% | True metrics |
| P6: + Reranker | 72.2% | 65.0% | 0.0% | Poison eliminated |
| P7: Distillation | 79.6% | 54.0% | 0.0% | Calibration gap |
| P8: Unified | 85.8% | 47.9% | 0.0% | nonUsScore discovery |
| *Search upgrade phases (145 LLM-labeled queries):* | | | | |
| Phase 0 | $\sim$61% | 50.0% | $\sim$42% | Stop words, 224 queries |
| Phase A | $\sim$64% | 53.3% | $\sim$42% | +embed, +expand, +rerank |
| **Phase B** | **70.4%** | **60.3%** | **$\sim$22%** | 1,034 queries, cached distill |

## 11 Limitations and Future Work

### 11.1 Current Limitations

1. **P@5 at 60%, not 80%+.** The linear model cannot capture feature interactions. A non-linear model is needed for the next leap.
2. **Poison rate at $\sim$22%.** Some irrelevant grants score high on embedding similarity. A purpose-built cross-encoder would help.
3. **No query-type routing.** All queries use the same weight vector despite fundamentally different retrieval characteristics.
4. **LLM labels are a proxy.** Real user click signals would provide ground truth for the actual discovery task.
5. **General-purpose embeddings.** `text-embedding-3-small` Neelakantan et al. [2022] was trained on web text, not grant-specific language. Domain-specific fine-tuning Reimers and Gurevych [2019] could improve the vocabulary bridge.

### 11.2 Phase C: Learning-to-Rank

When GA4 click data provides 1,000+ queries with engagement signals, **LambdaMART** Burges [2010] replaces the linear scorer with a gradient-boosted decision tree Chen and Guestrin [2016], Ke et al. [2017] trained via the LambdaRank objective:

$$\text{Linear: } s = \mathbf{w}^\top \mathbf{f} \quad \longrightarrow \quad \text{GBDT: } s = \sum_{t=1}^{T} h_t(\mathbf{f}) \tag{12}$$

This captures feature interactions (e.g., "high embedding AND matching org type") that the linear model averages away. Expected impact: +5–10pp P@5.

### 11.3 Scaling: ColBERT and SPLADE

At 50K+ grants, the retrieval architecture evolves:

**ColBERT Santhanam et al. [2022], Xiao et al. [2024].** ColBERT stores per-token embeddings and computes a MaxSim operator:

$$s(q, d) = \sum_i \max_j \cos(\mathbf{q}_i, \mathbf{d}_j) \tag{13}$$

This enables fine-grained matching of technical terms averaged away by single-vector embeddings.

**SPLADE Formal et al. [2021].** SPLADE learns sparse, high-dimensional representations that naturally bridge vocabulary gaps, potentially replacing the current LLM query expansion step with a faster learned expansion.

```
Stage 1: RECALL
  |-- SPLADE sparse retrieval (replaces FTS + expansion)
  |-- ColBERT late interaction (replaces kNN)
  |-- BM25 fallback (exact name/acronym matches)

Stage 2: CROSS-ENCODER RERANK (top 50-100)
  Fine-tuned ModernBERT cross-encoder [Warner et al., 2024]

Stage 3: LambdaMART SCORING (15+ features)
  GBDT with feature interactions
```

Projected marginal cost at 50K grants: ~\$0/query (all self-hosted inference vs. current ~\$0.003/query from API calls).

## 12 Related Work

**Dense Retrieval and Embeddings.** Dense passage retrieval Karpukhin et al. [2020] demonstrated that learned dense representations can outperform BM25 Robertson and Zaragoza [2009] for open-domain QA. Our embedding retrieval stage uses OpenAI's contrastive pre-trained embeddings Neelakantan et al. [2022], which provide general-purpose semantic matching without domain-specific fine-tuning. Sentence-BERT Reimers and Gurevych [2019] established the bi-encoder paradigm we adopt, where queries and documents are independently embedded for efficient $k$NN retrieval. BGE M3-Embedding Chen et al. [2024] showed that multi-granularity embeddings can serve multiple retrieval functions simultaneously.

**Learned Sparse Retrieval.** SPLADE Formal et al. [2021] learns sparse, high-dimensional representations that bridge vocabulary gaps through learned term expansion—a capability we currently achieve via LLM query expansion but plan to replace with SPLADE at scale (Section 10.3).

**Late Interaction Models.** ColBERTv2 Santhanam et al. [2022] introduced efficient late interaction via token-level MaxSim, enabling fine-grained matching that single-vector embeddings miss. Jina-ColBERT-v2 Xiao et al. [2024] extended this to multilingual settings with Matryoshka dimension reduction for storage efficiency. These models represent our planned retrieval upgrade path at 50K+ grants.

**Cross-Encoder Reranking.** BERT-based cross-encoders Nogueira and Cho [2019] remain the gold standard for pointwise relevance scoring. Recent work has explored using LLMs as rerankers Sun et al. [2023], Pradeep et al. [2023], with Rank1 Weller et al. [2025] demonstrating that test-time compute can match larger rerankers. Our system uses an LLM-based cross-encoder (gpt-4.1-nano) as a fallback when purpose-built rerankers are unavailable. ModernBERT Warner et al. [2024] provides a strong base for future fine-tuned cross-encoders with 8K context and improved efficiency.

**Query Expansion.** HyDE Gao et al. [2023] generates hypothetical documents to bridge the query-document vocabulary gap. Query2Doc Wang et al. [2023] and LLM-based expansion Jagerman et al. [2023] use language models to generate expansion terms. Our approach is closest to direct term expansion, using a lightweight LLM call to produce synonyms and related program names.

**Learning to Rank.** LambdaMART Burges [2010], implemented in XGBoost Chen and Guestrin [2016] and LightGBM Ke et al. [2017], is the standard gradient-boosted approach for learning-to-rank. Our current linear scorer is a deliberate simplification that we plan to replace with LambdaMART once click data provides sufficient training signal. GritLM Muennighoff et al. [2024] demonstrated unified embedding and generation models, suggesting a potential architecture simplification.

**Federated Search and Result Fusion.** Distributed information retrieval Callan [2000], Shokouhi and Si [2011] addresses the problem of querying multiple heterogeneous collections and merging results into a single ranked list. Classical fusion methods include CombMNZ Fox and Shaw [1994], Condorcet-based voting Aslam and Montague [2001], and Reciprocal Rank Fusion Cormack et al. [2009]. Our multi-provider architecture faces an analogous challenge, but our "sources" are generative LLMs with web search tools Nakano et al. [2021] rather than traditional document indexes. Rather than attempting cross-source score normalization, we eliminate the problem entirely by re-scoring all candidates through a single learned feature function—a strategy more akin to centralized re-evaluation than distributed fusion. BEIR Thakur et al. [2021] demonstrated that retrieval models exhibit significant performance variance across heterogeneous domains, motivating our domain-specific feature design.

**Knowledge Distillation and Synthetic Data.** Hinton et al. [2015] established the framework for transferring knowledge from large models to smaller ones. Liu et al. [2024] provided best practices for LLM-generated training data, which informed our two-strategy query generation pipeline. Our cached-feature distillation approach is related to offline feature extraction in deployed ranking systems, but applied to weight optimization rather than model training.

## 13 Conclusion

We presented a domain-specific grant search engine that combines four-stage hybrid retrieval with a 15-feature linear scorer optimized via knowledge distillation, deployed as the unified scoring backbone for a federated search system spanning five frontier LLM providers and a curated grant database. The key technical contributions are: (1) a source-agnostic feature projection that eliminates the score normalization problem in multi-provider fusion by re-evaluating all candidates—whether from a curated database or discovered by any of five LLMs searching distinct web indices—through an identical learned scoring function, (2) a cached-feature distillation approach that achieves $> 50{,}000\times$ speedup over naïve per-iteration evaluation, (3) a two-strategy query generation pipeline (persona-derived + grant-centric) that expanded training data from 224 to 1,034 labeled queries, and (4) the empirical finding that embedding cosine similarity dominates lexical features for domain-specific search with vocabulary mismatch. Over three optimization phases, validation P@5 improved from 50.0% to 60.3% while poison rate halved from $\sim$42% to $\sim$22%, at a marginal cost of $\sim$\$0.003/query.

## References

Javed A. Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–284. ACM, 2001.

Christopher J.C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. In *Microsoft Research Technical Report MSR-TR-2010-82*. Microsoft Research, 2010.

Jamie Callan. Distributed information retrieval. In *Advances in Information Retrieval*, pages 127–150. Springer, 2000.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

Gordon V. Cormack, Charles L.A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759. ACM, 2009.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292. ACM, 2021.

Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 243–252, 1994.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1762–1777. Association for Computational Linguistics, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2015.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. In *arXiv preprint arXiv:2305.03653*, 2023.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 2017.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. Best practices and lessons learned on synthetic data for language models. In *Proceedings of the Conference on Language Modeling (COLM)*, 2024.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. GritLM: Generative representational instruction tuning. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. In *arXiv preprint arXiv:2112.09332*, 2021.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nicholas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. In *arXiv preprint arXiv:1901.04085*, 2019.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 3715–3734. Association for Computational Linguistics, 2022.

Milad Shokouhi and Luo Si. Federated search. *Foundations and Trends in Information Retrieval*, 5 (1):1–102, 2011.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14918–14937, 2023.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9414–9423, 2023.

Benjamin Warner, Benjamin Clavié, James Thorne, Iker Chern, Stefan Lüdtke, and Sean O'Connell. ModernBERT: A modern approach to encoder-only transformers. *arXiv preprint arXiv:2412.13663*, 2024.

Orion Weller, Chaitanya Malaviya, Arman Cohan, and Dani Yogatama. Rank1: Test-time compute for reranking in information retrieval. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. Jina-ColBERT-v2: A general-purpose multilingual late interaction retriever. *arXiv preprint arXiv:2408.16672*, 2024.