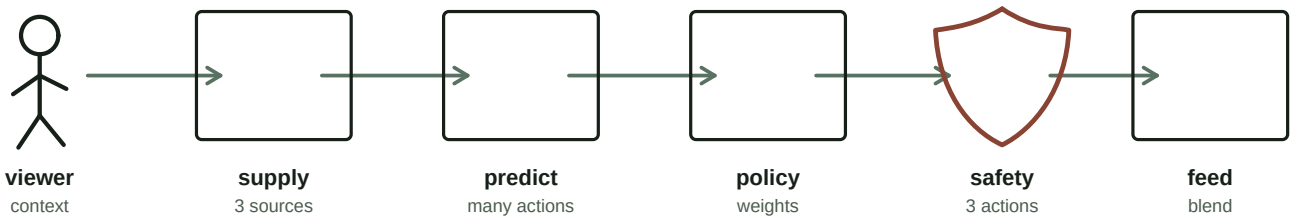
# How the X recommendation algorithm works

A source-audited map of retrieval, Phoenix ranking, slate policy, visibility enforcement, and final feed composition.




one request, several independent decisions

`commit 47c1bcdadfe4911568fd6db4f8838b194325beab`

Independent technical audit • 2,053 changed files • exact-SHA source links

**Commit:** Open-source X Recommendation Algorithm

**Parent:** `0bfc2795d308f90032544322747caacd535f75ae`

**Commit time:** 2026-08-13 17:09:58 UTC

**Audit method:** exact-SHA source checkout, complete parent-to-commit path inventory, targeted source tracing across request-time ranking, offline model systems, safety labeling, visibility enforcement, configuration, and tests.

# Executive summary

The public X recommendation algorithm in this commit is not one model and not one score. It is a layered system:

1. **Home Mixer builds viewer context.** It hydrates recent engagement sequences, follows, blocks, mutes, subscriptions, prior impressions, demographics, topics, and other request features.
2. **Multiple candidate sources retrieve posts concurrently.** Thunder supplies recent posts from followed accounts. Phoenix supplies neural out-of-network candidates. SimClusters supplies graph-and-community-based candidates. Additional sources can contribute cached posts and legacy retrieval candidates.
3. **A deterministic pipeline filters and enriches those candidates.** Duplicate, age, social-graph, prior-impression, media, topic, and inventory rules remove ineligible posts before expensive scoring.
4. **Phoenix predicts engagement probabilities.** For each candidate, it predicts probabilities for actions such as favorite, reply, repost, click, share, author follow, and negative feedback. Home Mixer converts those predictions into a scalar utility with configurable action weights.
5. **Heuristic and diversity layers alter the ordering.** Cold-start promotion, repeated-author attenuation, out-of-network attenuation, and optionally a determinantal point process, or DPP, modify which high-scoring posts remain competitive.
6. **Visibility Filtering makes the safety decision separately.** It hydrates author state, viewer relationships, tweet and media metadata, and safety labels, then evaluates ordered rules that return `Allow`, `Interstitial`, or `Drop`.
7. **The organic timeline is blended with non-organic products.** Ads, Who to Follow, prompts, feed surveys, Push to Home, and Jetfuel frames enter an outer pipeline after organic ranking.

The most important technical distinction is between **prediction**, **ranking**, and **eligibility**:

- Phoenix predicts action likelihoods.
- Home Mixer decides how much each action is worth and applies ranking policy.
- Visibility Filtering decides whether the post may be shown, shown with an interstitial, or removed.

That separation means there is no single line of code that represents "the X algorithm." The observable feed is the result of a candidate supply graph, several sequential transformations, safety policy, experiment configuration, and product blending.

This commit is a very large public reconstruction of that system. It changes **2,053 files after rename detection**, with **363,246 insertions and 11,640 deletions**. At raw path-status level it contains **2,080 path records**: 1,887 additions, 106 modifications, and 87 deletions. The 27-record difference is explained by 27 rename pairs. The release adds or substantially expands Phoenix, SimClusters, Thunder, Home Mixer, visibility filtering, bot and abuse systems, credibility scoring, label aggregation, and transparency tooling.

The release is still not a reproducible snapshot of X production. It omits production data, trained checkpoints, deployment configuration, several internal dependencies, some anti-abuse rules, Grox prompt templates, and exact production thresholds. Feature switches and experiment deciders can also override repository defaults. The correct reading is: this is detailed executable and structural evidence of the algorithm family, not proof of the exact feed any user receives at a particular moment.

# 1. What changed in this commit

## 1.1 Change census

The following counts come from a complete `git diff --numstat` and raw `git diff-tree` audit of the exact commit and its parent.

| Top-level area | Raw records | A | M | D | Insertions | Deletions | Role in the system |
|---|---|---|---|---|---|---|---|
| `botmaker` | 492 | 492 | 0 | 0 | 61,527 | 0 | Typed rule language and feature/action runtime |
| `phoenix` | 314 | 301 | 2 | 11 | 117,520 | 4,361 | Neural retrieval and ranking training/serving stack |
| `simclusters` | 232 | 232 | 0 | 0 | 35,797 | 0 | Community embeddings and approximate-nearest-neighbor retrieval |
| `grox` | 224 | 165 | 0 | 59 | 12,788 | 4,313 | Model workflow framework and safety/embedding flows |
| `home-mixer` | 215 | 113 | 86 | 16 | 28,683 | 2,249 | Timeline orchestration, ranking policy, filtering, and blending |
| `media-model-proxy` | 98 | 98 | 0 | 0 | 4,167 | 0 | Shared media hydration and model fan-out |
| `agatha` | 90 | 90 | 0 | 0 | 5,629 | 0 | PMI-based account credibility and safety scoring |
| `botmaker-rules` | 73 | 73 | 0 | 0 | 1,578 | 0 | Public anti-abuse and safety rule set |
| `visibility-filtering` | 63 | 63 | 0 | 0 | 15,861 | 0 | Ordered safety and eligibility policy engine |
| `bdsm` | 46 | 46 | 0 | 0 | 9,897 | 0 | Behavioral sequence model for coordinated or abusive activity |
| `phoenix-rankall` | 35 | 35 | 0 | 0 | 7,652 | 0 | Kafka-to-snapshot candidate corpus construction |
| `scarecrow` | 32 | 32 | 0 | 0 | 3,231 | 0 | Production wiring and execution for BotMaker rules |
| `abuse-enforcement-service` | 31 | 31 | 0 | 0 | 12,622 | 0 | Rule-driven enforcement from model and label signals |
| `under-the-hood` | 26 | 26 | 0 | 0 | 4,138 | 0 | User-visible safety-label explanations and aggregation |
| `thunder` | 21 | 13 | 8 | 0 | 29,940 | 293 | Real-time in-network post store and retrieval |
| `clip` | 20 | 20 | 0 | 0 | 2,856 | 0 | Image/text embedding model support |
| `phoenix-rankall-strato` | 17 | 17 | 0 | 0 | 1,699 | 0 | Event hydration and corpus-index inputs |
| `candidate-pipeline` | 10 | 1 | 9 | 0 | 395 | 192 | Generic staged candidate-processing framework |
| `vm-ranker` | 10 | 10 | 0 | 0 | 1,302 | 0 | Optional DPP diversity selection service |
| `user-cred-v2` | 9 | 9 | 0 | 0 | 805 | 0 | Follow-graph PageRank-style credibility score |
| `adult-content` | 8 | 8 | 0 | 0 | 482 | 0 | Adult-content classification components |
| `visibility-filtering-client` | 7 | 7 | 0 | 0 | 2,999 | 0 | Client contract for visibility decisions |
| `safety-label-user-agg` | 3 | 3 | 0 | 0 | 617 | 0 | Post-label-to-account-label aggregation |
| `README.md` | 1 | 0 | 1 | 0 | 354 | 231 | Release description and updated system explanation |
| `docs` | 1 | 1 | 0 | 0 | 495 | 0 | Added public documentation |
| `pnsfwmedia` | 1 | 1 | 0 | 0 | 212 | 0 | Experimental media NSFW model |
| `.gitignore` | 1 | 0 | 0 | 1 | 0 | 1 | Removed repository ignore file |

This is the complete top-level path inventory, not a sample. The A/M/D columns use raw path records, where a rename appears as one deletion and one addition. The line counts use Git's rename-aware diff, which is why pure moves do not add line churn. The release announcement and its own scope qualifications are in the root README.

Every changed path was included in the inventory and classified by subsystem. Algorithm claims were then traced through executable implementations, configuration, documentation, and tests. Generated interfaces, low-level support code, and synthetic fixtures remain in the change totals, but this report does not misrepresent each generated or vendored line as an independent ranking decision.

## 1.2 The architectural shift

The parent revision already contained a Home Mixer implementation and an older Phoenix demonstration. This commit does four material things:

- It turns Phoenix from a small illustrative release into a much larger trainer, serving runtime, model configuration, and synthetic end-to-end verification system.
- It publishes the SimClusters graph and ANN stack that supplies community-based candidates.
- It exposes much more of the safety path: model and rule labeling, visibility hydration, policy evaluation, enforcement, and user-facing label explanations.
- It updates Home Mixer defaults and plumbing so those systems are visible as one request-time pipeline rather than isolated components.

This is why reading only the new Phoenix model would miss much of the algorithm. Candidate generation and safety policy determine the feasible set before and after neural scoring, while Home Mixer parameters determine how predictions become ranking utility.

# 2. End-to-end architecture

### Architecture diagram source

```
flowchart TD
    A["For You request"] --> B["Query hydration"]
    B --> C1["Thunder: followed accounts"]
    B --> C2["Phoenix: neural retrieval"]
    B --> C3["SimClusters: community ANN"]
    B --> C4["Other and cached sources"]
    C1 --> D["Merge candidates"]
    C2 --> D
    C3 --> D
    C4 --> D
    D --> E["Candidate hydration"]
    E --> F["Sequential prefilters"]
    F --> G["Phoenix action probabilities"]
    G --> H["Weighted utility"]
    H --> I["Cold-start, author, OON adjustments"]
    I --> J["Optional VM/DPP diversity"]
    J --> K["Top organic candidates"]
    K --> L["Visibility hydration and rules"]
    L --> M["Allow, interstitial, or drop"]
    M --> N["Conversation deduplication"]
    N --> O["Blend ads and product modules"]
    O --> P["Final response"]
    P --> Q["Async logging and served-history side effects"]

    R["Grox, BotMaker, BDSM, Agatha, media models"] --> S["Safety labels and enforcement facts"]
    S --> L
```

There are two separate data planes:

- The **online request plane** runs Home Mixer, candidate sources, rankers, and Visibility Filtering.
- The **offline and streaming preparation plane** trains Phoenix, creates Phoenix retrieval indices, computes SimClusters embeddings, emits safety labels, and builds account credibility signals.

The two planes meet through model endpoints, candidate indexes, caches, Kafka topics, Manhattan-style stores, and feature services.

# 3. Exact request-time pipeline semantics

The generic candidate pipeline is the control-flow contract behind Home Mixer. Its execution order matters because some stages are concurrent and others deliberately observe prior mutations.

## 3.1 Stage order

1. **Query hydrators run concurrently.** Each successful result updates the query. A failed hydrator does not erase the query or terminate the entire request.
2. **Dependent query hydrators run concurrently after the first group.** They can rely on fields populated by the first hydration barrier.
3. **Candidate sources run concurrently.** Their successful candidate vectors are concatenated. Failed sources are flattened away, so a source outage normally reduces candidate supply rather than failing the full feed.
4. **Candidate hydrators run concurrently against the same candidate snapshot.** Their returned updates are merged afterward. A hydrator that returns the wrong vector length is rejected. Per-candidate hydration errors preserve the candidate's prior state.
5. **Filters run sequentially.** Later filters see the removals made by earlier filters.
6. **Scorers run sequentially.** Later scorers see earlier scores and candidate metadata. This is essential because the ranking scorer consumes Phoenix predictions, and VM Ranker consumes the scalar score produced by the ranking scorer.
7. **The selector chooses the current result set.** The organic Home Mixer selector is a top-K stage.
8. **Post-selection hydrators run concurrently.** Visibility and ancillary response data are added here, after the expensive ranking reduction.
9. **Post-selection filters run sequentially.** Visibility drops and conversation-level constraints are enforced.
10. **The result is truncated to the configured pipeline size and finalized.**
11. **Side effects run asynchronously.** Logging, cache writes, event publication, and served-history recording happen after final candidates exist and do not hold up the response.

The relevant generic contracts are split across the pipeline, candidate hydrator, and query hydrator implementations.

## 3.2 Failure behavior is part of the algorithm

The pipeline is intentionally resilient, but resilience changes semantics:

| Failure | Request-time consequence |
|---|---|
| One retrieval source fails | Other sources continue; the candidate pool becomes smaller |
| One candidate hydrator fails for a candidate | That candidate keeps its previous state |
| One scorer fails for a candidate | The candidate keeps its previous score |
| Post-selection side effect fails | The already-built response is not retracted |

| Failure | Request-time consequence |
| --- | --- |
| Visibility label lookup fails internally | The visibility service can evaluate an empty label set |
| Home Mixer visibility call fails for a candidate | The generic hydrator path can leave the candidate without a drop reason, so the later filter retains it |

These are not equivalent. Some failures are explicitly fail-open inside a service; others become effectively fail-open because the generic update contract preserves a candidate on error. Any production evaluation should therefore measure both ranking quality and dependency health.

## 3.3 Changes to the generic pipeline in this commit

The generic framework changes are mostly observability improvements: pipeline summaries, scoped metric labels, selector statistics, source fetched counts, and quieter optional tracing. A behavioral extension lets cached hydrators specify both a per-query/candidate cache key and an `already_hydrated` predicate. That allows an expensive hydration to be skipped when an earlier stage already supplied the data.

# 4. Home Mixer: the organic For You pipeline

The organic pipeline is defined in Home Mixer's Phoenix candidate pipeline. Despite the filename, it accepts more than Phoenix candidates.

## 4.1 Viewer and request context

Before retrieval, Home Mixer can hydrate:

- Phoenix scoring and retrieval sequences
- followed, blocked, muted, and subscribed account IDs
- explicit and implicit engagement signals
- recent impressions and a bloom filter of seen posts
- cached posts and previously served history
- mutual-follow relationships
- demographics, inferred gender, and IP-derived context
- followed Grok topics and starter packs
- installed-app context

This context has three functions. It gives Phoenix its behavioral sequence, gives candidate sources seeds and social context, and gives filters evidence for removals.

## 4.2 Candidate sources

The organic pipeline declares these sources:

| Source | Main function | Default/public behavior visible in this commit |
| --- | --- | --- |
| Thunder | In-network retrieval | Recent eligible posts from followed accounts |
| Phoenix | Neural out-of-network retrieval | Enabled, up to a quality-adjusted 1,000 candidates |

| Source | Main function | Default/public behavior visible in this commit |
|---|---|---|
| Phoenix Topics | Topic-conditioned neural retrieval | Separate retrieval configuration |
| Phoenix mixture-of-experts | Experimental neural retrieval | Disabled by default |
| SimClusters | Graph/community retrieval | Post-seeded ANN, up to 800 final candidates |
| TweetMixer | Legacy or auxiliary retrieval | Disabled by default in the public config |
| CachedPosts | Request shortcut | Can bypass live source work when populated |

The candidate sources are not forced to return equal quantities. A high-volume source can dominate the merged pool until filtering, while the final scoring model provides a common comparison surface.

## 4.3 Candidate hydration

The pipeline adds features that retrieval responses do not necessarily contain:

- in-network and bidirectional-follow status
- core post data and quoted-post data
- author/account state
- media and subscription information
- whether the author blocked the viewer
- topics and language
- engagement counts
- semantic IDs used by retrieval and safety systems

Because these hydrators run concurrently, they must not rely on updates from another hydrator in the same group. Dependencies must be placed in query hydration or a later stage.

## 4.4 Sequential prefilters

The public pipeline applies the following order:

1. duplicate removal
2. missing core-data removal
3. maximum age, publicly configured at 48 hours
4. viewer's own post removal
5. out-of-network repost/reply policy
6. out-of-network NSFW SimClusters filter
7. repost deduplication
8. ineligible subscription removal
9. previously seen removal
10. backup seen-state filtering
11. previously served removal
12. muted-keyword filtering
13. author social-graph filtering
14. video constraints

15. topic-ID constraints
16. minimum engagement for new users
17. inventory holdout

The order avoids spending scoring capacity on obviously ineligible candidates. It also means removal reasons are order-dependent: a post failing several checks is attributed to the first filter that removes it.

# 5. Candidate generation

## 5.1 Thunder: in-network recency

Thunder consumes post-create and post-delete events and stores recent posts in in-memory concurrent maps keyed by author. It keeps separate bounded deques for original, secondary, and video inventory.

At request time, Home Mixer's Thunder source fetches inventory for followed accounts, removes seen IDs and blacklisted internal accounts, applies reply/repost constraints, sorts eligible posts by creation time descending, and takes the configured maximum.

Thunder exposes an algorithm parameter, but the public service implementation routes the declared modes to recent-first behavior, with unsupported values also falling back to recency. In the public Home Mixer defaults, the maximum is 1,200 before quality factoring. Thunder is therefore a high-recall, low-latency recency source. Phoenix scoring later decides how much that recency inventory is worth relative to out-of-network candidates.

## 5.2 Phoenix retrieval: learned candidate search

Phoenix retrieval sends the viewer retrieval sequence and request context to the Phoenix service. Its public production-cluster setting is `Experiment1Fou`, with no source-level fallback and zero configured retries. A source error is still contained by the candidate-pipeline concurrency semantics.

The flagship retrieval model is a two-tower system:

- A **user tower** encodes up to 1,023 history tokens and coarse viewer context.
- A **candidate tower** encodes post content and metadata, including semantic IDs.
- Both vectors are L2-normalized.
- Retrieval uses dot-product similarity, which is cosine similarity after normalization.
- Separate heads support Home and immersive use cases.

The main public configuration does not depend on a learned user-ID embedding. The viewer representation comes from behavior history and coarse features. The flagship candidate representation also avoids a direct post-ID embedding, improving its ability to represent newer posts from content and author/context features.

Training uses positive examples centered on favorites for Home retrieval and a wider action set for immersive retrieval. Negatives combine in-batch examples with 64 global negatives. A learned temperature controls logit scale, masks prevent invalid comparisons, and log-Q correction compensates for sampling frequency. During checkpoint/index construction, the candidate tower is run over a corpus with a configured maximum of 28,672,000 items so online retrieval can perform approximate nearest-neighbor search.

The public model configuration is in Phoenix XRecSys and two-tower configuration.

## 5.3 SimClusters: community-based retrieval

SimClusters provides a distinct inductive bias from Phoenix. Rather than learning only a sequence-to-item neural function, it builds sparse communities from the social and engagement graph.

### Offline graph construction

1. A user-to-user graph is built from follows and favorites.
2. Favorite edges decay with a 100-day half-life.
3. Each user retains at most 2,000 neighbors.
4. **KnownFor** assigns producers to clusters. Iterative scoring rewards agreement between a user's neighborhood and cluster membership while penalizing false positive and false negative assignments.
5. **InterestedIn** projects a viewer's edges through KnownFor producer memberships, producing up to 50 clusters per user with social-proof information.
6. Streaming jobs project post engagements into post-to-cluster embeddings and maintain top posts per cluster.

The repository includes the KnownFor update jobs, InterestedIn construction, and tweet-embedding generation.

### ANN scoring

For a source embedding `s` and candidate embedding `c`, the basic score accumulated across shared clusters is:

```
score(s, c) = sum over clusters k of s[k] * c[k]
```

The ANN implementation scans at most 50 source clusters, can pull up to 800 top posts from each cluster, applies age and source constraints, and accumulates scores by candidate. It supports dot-product-like, normalized, cosine, and heavier ranking variants before returning a final top set, commonly 200 at the ANN service boundary. See ApproximateCosineSimilarity.

### How Home Mixer actually invokes it

The request-time SimClusters source does not simply submit one InterestedIn vector. It extracts post seeds from the viewer's explicit and implicit engagement history, deduplicates and prioritizes newer seeds, and runs one post-to-post ANN query per seed.

Results are cached with a public maximum of two million entries and a 600-second TTL. Candidate lists from different seeds are interleaved in fair round-robin order and deduplicated so one prolific seed cannot monopolize the source result. The source can consider 10,000 intermediate candidates, then returns up to 800 that are younger than 48 hours and exceed a public score threshold of 0.5. The selected public model is `20M/145K/2020`, with `LOG_FAV_LONGEST_L2_EMBEDDING_TWEET` as the source embedding and `LOG_FAV_BASED_TWEET` as the candidate embedding.

Phoenix and SimClusters are therefore complementary. Phoenix learns a dense behavioral matching function; SimClusters supplies an interpretable sparse-community path and multiple engagement-seeded neighborhoods.

# 6. Phoenix engagement ranking

## 6.1 Model input and attention structure

The primary ranking configuration, `home_direct_packed`, uses:

- up to 1,022 history items
- up to 64 candidate items

- two user-prefix tokens
- 8 transformer layers
- model width 2,560
- embedding table width 1,024
- grouped-query attention with 20 query heads and 4 key/value heads
- head/key dimension 128
- feed-forward expansion factor 2
- RMS normalization, pre-normalization, and rotary position encoding

Its defining structural feature is **candidate isolation**. User and history tokens attend bidirectionally to the user/history block. They cannot attend to candidate tokens. Each candidate can attend to the user/history block and itself, but not to other candidates. The attention-mask implementation is in ranker attention utilities.

Candidate isolation has two benefits:

- A candidate's raw prediction does not change merely because another candidate happened to share the request batch.
- The service can score many candidates in one packed model call without allowing cross-candidate information leakage.

Slate-level diversity and competition are then handled explicitly by downstream ranking policy, rather than being hidden inside candidate attention.

## 6.2 Prediction heads and losses

The ranker predicts 64 discrete action targets with sigmoid outputs and eight continuous targets, primarily dwell-related quantities. Discrete targets use multilabel binary cross-entropy. Continuous targets use Tweedie losses for gallery-style dwell and mean absolute error for other continuous values. Log-Q correction adjusts for the frequency with which training examples were sampled.

The model is trained with AdamW in the public implementation, using beta values 0.95 and 0.98 and weight decay `1e-3`. Sparse embedding rows use row-wise AdaGrad with a public learning rate of 0.2. The public documentation states that X's internal dense optimizer is a tuned RMS-normalized Adam derivative, so public optimizer fidelity is intentionally incomplete. See Phoenix training documentation.

## 6.3 From probabilities to one utility score

PhoenixScorer requests action probabilities and attaches them to candidates. RankingScorer then computes the default weighted score:

```
raw_score(candidate) = sum over actions a of weight[a] * P(a | viewer, candidate)
```

Missing action predictions contribute zero. The public parameter snapshot is marked as last synchronized on 2026-08-12 in Home Mixer parameters.

| Predicted action | Public default weight |
|---|---|
| Favorite | 0.5 |
| Reply | 5.0 |
| Reply to mutual-follow author's original post | additional 15.0 |
| Repost | 1.0 |
| Photo expand | 0.05 |

| Predicted action | Public default weight |
|---|---|
| Video open | 0.05 |
| Click | 0.4 |
| Open link | 0.2 |
| Profile click | 0.0 |
| Video quality view | 0.05 |
| Share | 2.0 |
| Share by direct message | 5.0 |
| Copy link | 20.0 |
| Binary dwell | 0.0 |
| Quote | 5.0 |
| Quoted-post click | 0.05 |
| Quoted-post video quality view | 0.0 |
| Follow author | 4.0 |
| Post unexplored | 0.02 |
| Continuous dwell time | 0.004 |
| Click dwell | 0.0 |
| Active-seconds residual | 0.0 |
| Not interested | -43.2 |
| Block | -31.2 |
| Mute | -58.8 |
| Report | -234.0 |
| Not dwelled | -0.02 |

These weights show that raw engagement volume is not the objective by itself. A small predicted probability of report, mute, block, or not-interested can outweigh much larger probabilities of lightweight positive actions. Conversely, explicit sharing and copying are treated as much stronger positive evidence than a click or media expand.

Some predictions have additional gates. Video quality view depends on duration criteria. The mutual-reply boost applies only to original posts from mutual follows. `PostUnexplored` is in-network-specific, and its multiplicative variant is disabled by its public flag, with alpha also set to zero.

## 6.4 Score offset and negative ordering

The scorer transforms the raw total so later components receive a nonnegative value while preserving useful order:

```
if total_weight_sum == 0:
    score = max(raw_score, 0)
else if raw_score < 0:
    score = ((raw_score + sum_of_negative_weight_magnitudes) / total_weight_sum) * 0.001
else:
    score = raw_score + 0.001
```

The intent is to place negative candidates into a tiny interval below positive candidates without collapsing all negative examples to the same value. Positive candidates retain their meaningful scale with a small offset.

The repository also contains a `dwell_regret_sigmoid` or gated utility mode. It computes positive-action ratios relative to the current slate and attenuates dwell value with negative-feedback risk. The public default `ValueModelMode` is `weighted`, so the simpler weighted sum is the primary path unless configuration overrides it.

# 7. Policy adjustments after the weighted score

The score emitted by the weighted action model is not final.

## 7.1 Cold-start promotion

Author cold-start scoring is enabled by default. It considers original posts from authors with at most 1,000 followers and posts with fewer than 1,000 impressions. Eligibility is limited to candidates within the top 85 percent of nonzero results. In the treatment branch it additionally requires the post to be at most one day old.

The score target is selected at an integer rank drawn from `[ColdStartSlotMin, ColdStartSlotMax)`. The public defaults `[15, 16)` therefore select zero-based rank 15 exactly, the 16th-ranked score. The highest-scoring eligible candidate is raised to at least that target. This creates one controlled discovery opportunity around the 16th position rather than globally boosting every small account.

## 7.2 Repeated-author attenuation

Author diversity is enabled by default. Candidates are first ordered by score. For the `k`th earlier occurrence of the same author, Home Mixer applies:

```
multiplier(k) = (1 - floor) * decay^k + floor
```

With public defaults `decay = 0.5` and `floor = 0.25`:

| Earlier posts from same author, k | Multiplier |
|---|---|
| 0 | 1.0000 |
| 1 | 0.6250 |
| 2 | 0.4375 |
| 3 | 0.34375 |
| large k | approaches 0.25 |

This does not hard-cap an author. It makes repeated appearances progressively less competitive while retaining a quarter of the score in the limit.

## 7.3 Out-of-network attenuation

The public default multiplier for ordinary out-of-network candidates is 0.75. Topic out-of-network candidates receive 0.5. The same 0.75 factor can apply to replies and reposts from followed accounts because the content relationship is treated differently from a direct original in-network post.

A separate new-user out-of-network factor is publicly set to `0.00001`, but its activation requires a user-age/follow threshold whose public age threshold is zero. Under that configuration the special path is effectively disabled for users who satisfy the required follow-count condition.

## 7.4 Optional MPN path

The repository contains an MPN ranking mode with feature-specific multipliers. It is disabled by default. When enabled, it scales positive net scores, applies the score offset, and then runs cold-start handling. This is another example of source code exposing an alternative policy without establishing that the policy is live.

# 8. VM Ranker and DPP diversity

Home Mixer enables the VM Ranker client by default and points to model ID `dpp`, with cluster `Experiment3`, `theta = 0.65`, maximum rank 150, score-weight sending disabled, and fallback disabled. The request integration is in Home Mixer VM Ranker.

There is a deployment-sensitive caveat: the standalone VM Ranker executable has DPP disabled by default. If the service is not launched with `--dpp-enabled`, it echoes the input scores. Repository configuration on both sides must therefore agree before DPP is actually active.

## 8.1 DPP kernel

For the top `max_rank` candidates, the service obtains a 1,024-dimensional embedding. A repost uses the original post ID as its embedding key. A missing embedding is replaced with a random unit vector.

Let the normalized base quality be:

```
q_i = score_i / max_j(score_j)
```

The public theta-to-quality scale is:

```
alpha = theta / (2 * (1 - theta))
quality_i = exp(alpha * q_i)
```

The DPP L-ensemble kernel is:

```
L_ij = quality_i * quality_j * cosine(embedding_i, embedding_j)
```

The DPP implementation greedily performs a Cholesky-style selection. It first picks the item with the largest diagonal term, then repeatedly chooses the item with the greatest remaining conditional variance. It stops at the configured top-K or when the remaining numerical rank is exhausted.

## 8.2 It is a selector, not a smooth rescorer

The DPP model wrapper returns selected IDs in their original score order. The service preserves original scores for selected candidates and assigns zero to unselected candidates. Home Mixer then performs top-K selection.

So the practical effect is:

- preserve the base score ordering within the diverse chosen subset
- turn non-selected candidates into zero-score tail items
- allow quality and embedding dissimilarity to trade off during membership selection

If the VM call fails and fallback is disabled, the candidate-pipeline scorer semantics preserve each candidate's existing RankingScorer score. The failure does not automatically zero the whole feed.

# 9. Organic selection and outer feed blending

The organic Phoenix pipeline selects 50 candidates before post-selection hydration. Visibility and conversation filtering can reduce this set. The outer For You pipeline then combines organic posts with:

- ads
- Who to Follow
- prompts
- Push to Home

- Jetfuel frames
- feed surveys

The public constants are 35 ordinary result slots, four feed-module slots, and up to eight Jetfuel frames, producing an outer maximum of 47 items. Prompt placement starts at position 0, Who to Follow at position 6, survey at position 12, and Push to Home uses a top placement path.

Ads are not simply inserted every N posts. The default partition organic blender divides organic posts by risk and looks for a safe-post, ad, safe-post triple. It checks collision, keyword, and brand-safety constraints, fills remaining positions, truncates to 35, and avoids ending the timeline on an ad. Alternative safe-gap and time-gap blenders are present.

This outer layer is why the ranker's top 35 organic posts do not map one-to-one to final screen positions.

# 10. Visibility Filtering

Visibility Filtering is a separate policy engine, not a Phoenix prediction head. Its inputs and rule order are public in the visibility registry and rule evaluator.

## 10.1 Hydration graph

The visibility hydration executes several branches concurrently:

- viewer state
- post and media safety labels
- Tweet Event Service metadata for the post and media
- exclusive-content state
- author core-data lookup, followed by author Gizmoduck state and viewer-author social graph

If the author cannot be resolved, hydration does not proceed as if the author were healthy. The final action becomes `Drop` with an `unresolved_author_id` reason.

## 10.2 Safety-label lookup and caching

Safety-label hydration uses a local cache with up to one million entries. Posts younger than five minutes receive a short TTL capped around 30 seconds so newly arriving labels propagate quickly. Older posts receive a 60-second local TTL. The label source queries a remote Twemcache layer and falls back to a Manhattan-backed source.

Tests explicitly establish that label lookup errors fail open to an empty label set. This improves timeline availability but means dependency health is security-relevant telemetry.

## 10.3 Ordered actions

Rules return one of three actions:

- `Allow`: show normally
- `Interstitial`: retain the post but require a warning or content cover in a rendering layer
- `Drop`: remove the post

Evaluation is ordered. The first `Drop` short-circuits evaluation. The first `Interstitial` is retained unless a later rule produces `Drop`. If every applicable rule allows, the final result is `Allow`.

The repository defines multiple safety levels. Home Mixer mainly uses:

- `TimelineHome` for in-network posts and the original post behind an in-network repost
- `TimelineHomeRecommendations` for out-of-network posts, conversation ancestors, quotes, and other ancillary content

The recommendation level is stricter because amplification requires a different policy bar from showing content a viewer directly followed.

## 10.4 Main policy categories

`TimelineHome` can drop content because of:

- suspended, deactivated, erased, offboarded, or protected authors
- viewer blocks, mutes, and muted repost relationships
- public-interest or platform labels such as spam, bounce, emergency, and FOSNR categories
- nullcast, stale, legal, and local-law restrictions
- sensitive content for logged-out or underage viewers
- exclusive-content constraints

It can interstitial NSFW high-precision content, gore, card-related sensitive content, and NSFW-author content.

`TimelineHomeRecommendations` adds or tightens rules for:

- DMCA and geographic media restrictions
- NSFW account/admin states and tweet-level NSFW flags
- NSFW recall and precision labels
- gore, card, do-not-amplify, malicious URL, and high-recall spam labels
- NSFW text and FOSNR abuse/insult labels
- account labels for high-recall/high-precision NSFW, spam, compromise, read-only state, impersonation, abusive behavior, sensitive profile imagery, and near-perfect or do-not-amplify categories

Exact mappings are configuration and rule-order dependent. The important system property is that these are hard or interstitial policy decisions downstream of predicted engagement.

## 10.5 How Home Mixer applies the result

The Home Mixer visibility hydrator partitions primary and related posts by safety level and queries them concurrently. The later filter removes `Drop` decisions and other explicit filtered reasons. It does not remove `Interstitial`, because the rendering behavior belongs downstream and is not included in this repository.

Ancillary filtering also drops a candidate when a non-tombstoned ancestor, quoted post, or retweeted original receives a `Drop`. This prevents a permitted wrapper post from reintroducing disallowed embedded content.

One important availability boundary follows from generic hydrator semantics. If the Home Mixer call to Visibility Filtering returns an error for a primary candidate, the candidate update can remain absent and the later filter sees no drop reason. That path is effectively fail-open at the request layer. The same pattern applies to ancillary lookup errors. This is a source-level inference from the interaction of the two components, not an explicit policy statement.

# 11. How labels are produced

The release exposes several independent systems that produce the facts Visibility Filtering and abuse enforcement consume.

## 11.1 Grox workflow engine

Grox is reorganized into a core DAG/runtime framework and separately registered flows. Public flows cover post safety, multimodal embeddings, reply spam/ranking, and user/post assessment.

A representative post-safety flow performs filtering, rate limiting and deduplication, media hydration, category and policy inference, specific safe-sex/nudity handling, and an annotation sink. Grox can hydrate media and ASR data, invoke model samplers asynchronously, and write annotations or embeddings.

The commit deliberately excludes `.j2` prompt templates to reduce gameability. The orchestration code still references those prompts. That makes control flow auditable but prevents exact reproduction of the prompt-based labels.

## 11.2 Media Model Proxy, CLIP, and adult-content models

The Media Model Proxy loads media once from blob storage and fans the decoded input to multiple DeepBird models. It returns named scores; callers decide persistence and policy interpretation.

The CLIP module builds normalized image and text embeddings. The public default references a Twitter ViT-B/32-256 variant, but public model URLs are placeholders rather than usable checkpoints.

The adult-content stack includes an MLP over 256-dimensional embeddings. The experimental `pnsfwmedia` model combines a CLIP image score with account-level Agatha NSFW score, text NSFW score, consumer/follower context, log follower counts, and missingness flags before batch normalization, a 50-unit ReLU layer, and a sigmoid output.

## 11.3 Agatha account scoring

Agatha is a batch statistical system rather than a neural ranker. It uses 14 days of prediction history and 30-day model-building windows for labels including general reports-per-favorite, spam reports-per-favorite, suspended spam, and NSFW.

It crosses user features with labels and estimates smoothed pointwise mutual information, exponential moments, variance, and sparsity. For users present in the labeled data, it computes leave-one-out statistics so a user's own outcome does not trivially leak into that user's features. It then aggregates feature-class and label moments into account-level safety signals.

## 11.4 BDSM behavioral sequence model

BDSM models sequences of account actions with a bidirectional transformer:

- sequence length 512
- width 1,024
- 256 action types
- time-aware rotary position encoding based on timestamps
- grouped-query attention, RMSNorm, and SwiGLU
- eight behavior heads: FollowBot, LikeBot, EngagementAmplifier, ReplySpamBot, TweetSpamBot, RTBot, MultiActionBot, and LegitimateUser

The backbone is frozen while separate MLP heads train with class-balanced binary cross-entropy, reverse cross-entropy, and focal components. The runtime consumes Kafka events, accumulates feature sequences in Rust, prefetches from cache/store, performs JAX GPU inference, applies thresholds, deduplication, and cooldowns, then emits challenge/suspend actions and audit outputs.

The public thresholds are redacted to the sentinel `9.99`, so they never fire as checked in. Appeal text is also redacted, and an internal key-value sidecar is absent. The architecture is exposed, but production enforcement behavior is not reproduced.

## 11.5 BotMaker and Scarecrow

BotMaker is a typed DSL, compiler, and runtime for rules over fetched and derived features. It supports caches, counters, rate limits, and action outputs. Scarecrow wires the registries and executes compiled `.bot` rules.

The 73 public rules include Grox post-safety label handling, spam URL patterns, NSFW and gore handling, and duplicate-text behavior. The repository states that some rules are withheld, so the public rule set is informative but incomplete.

## 11.6 Abuse Enforcement Service

The Abuse Enforcement Service consumes model scores, labels, and facts. It compiles YAML policy to CEL expressions and evaluates ordered, first-match decisions. Dynamic overrides are cached with last-good behavior.

It supports allowlists, credibility prechecks, and actions including suspension, account/post labels with TTLs, Arkose challenges, CAPTCHA, and spam-liveness workflows. Composite `act_all` actions, deduplication classes, and audit records are built in.

Public rules include a clearly disclosed mock follower threshold of 12.34 to reduce gaming. Dynamic production overrides may differ. A terminal user rule can suspend the account, while a terminal post rule skips further processing for that post.

## 11.7 UserCredV2

UserCredV2 computes a PageRank-style credibility signal over the follow graph after removing linked-account edges.

Its teleport distribution blends:

- a uniform distribution over premium users
- seven-day favorite and repost flow weighted by prior PageRank mass

The public blend beta is 0.5. PageRank uses jump probability 0.2, convergence threshold 0.001, a maximum of 50 iterations, and a warm start from the prior snapshot. Final mass is mapped to a 0-to-100 score:

```
credibility = clamp(165.2 + 7.07 * ln(pagerank_mass), 0, 100)
```

Snapshot publication includes safety gates and quarantine behavior, visible in UserCredV2App.

## 11.8 Post-label to account-label aggregation

The safety-label user aggregator consumes post safety-label events with a public 10-minute processing delay and 20-minute SLO. Config-driven rules count matching labels across recent posts. Limits include 10 rules, 10 windows, 60-day post age, and seven-day TTL.

The public aggregation can apply `POSSIBLY_NSFW_ACCOUNT` or `NSFW_HIGH_PRECISION`. It excludes accounts already carrying certain NSFW/admin states, high PageRank users, gray-listed users, and configured custom labels. It fetches recent posts, optionally includes media labels, and chooses the longest applicable TTL. See aggregation rules.

## 11.9 Under the Hood

Under the Hood aggregates historical account and post labels into daily/monthly data, publishes Manhattan-backed snapshots, and serves a Strato/GraphQL report with human-readable explanations of labels and their effects. Its public feature switch defaults to false. It is a transparency surface over safety state, not a recommendation scorer.

# 12. Phoenix training and serving reality

The new Phoenix tree is executable model engineering, not only pseudocode. It includes XREX trainers, Pallas kernels, distributed tensor/sharding utilities, checkpointing, serving, RPC contracts, synthetic-data generation, and a deterministic retrieve-then-rank harness. The Phoenix README describes the runnable path.

The public package contains smaller `nano` models, commonly width 512 with four layers, that preserve the same retrieval/ranking contracts for manageable testing. A synthetic end-to-end flow can build an index, retrieve candidates, and score them through gRPC. CUDA-capable compiled engine code is present.

What that proves:

- the public architecture can execute through training/serving mechanics
- candidate isolation, packing, losses, checkpoint flow, and retrieve-rank RPC integration are testable
- the source is substantially closer to the real model family than the deleted Grok-1 demonstration

What it does not prove:

- production recommendation quality
- parity with X's production data distribution
- parity with X's production checkpoints
- multi-host production orchestration or throughput
- the exact dense optimizer used internally
- exact serving topology or active feature-switch state

The synthetic data is deterministic and suitable for mechanics validation. It is explicitly not evidence of model quality.

# 13. Phoenix RankAll: building the retrieval corpus

Phoenix retrieval depends on a fresh candidate corpus. The Strato processor hydrates post-creation and favorite events with metadata, engagement counts, and annotations. It skips communities, replies, and reposts, then routes eligible posts into several logical indices, including post creation, one-favorite, 32-favorite, video, topic, imagine, metadata, multimodal metadata, NSFW, and evergreen variants. See the RankAll candidate processor.

The processor applies `TimelineHomeRecommendations` visibility policy without viewer context. Adult or visibility-dropped posts are excluded from the general index, with a specialized NSFW-video path treated separately.

The Rust RankAll service consumes Kafka and maintains windowed snapshots. Public windows include:

- post creation: 1 day
- one-favorite: 1 and 2 days
- 32-favorite: 1 day
- video: 2, 4, 7, 14, and 30 days

- NSFW: 2 and 7 days
- evergreen: 5 years
- evergreen Grok: 30 days
- topics: 1 day
- metadata: 1 to 3 days
- semantic-ID variants, including NSFW 14 days and imagine 4 days

Semantic-ID fetch waits for a minimum post age of 180 seconds, uses a 10-second timeout, and backfills the most recent hour every 30 seconds. Snapshot files are written as versioned Snappy Parquet and published through an atomic symlink, retaining three versions by default. With `commit_after_dump = true`, Kafka offsets are committed only after a durable snapshot is written. The behavior is defined across RankAll config, pipeline, and writer.

Two declared index variants, multimodal metadata and ads, report themselves as unimplemented in the public Rust enum. Their presence is a schema or roadmap signal, not evidence that the public service builds them.

# 14. Configuration, experiments, and what actually runs

Public defaults are useful evidence, but they are not the same as live production state.

## 14.1 Four configuration layers

1. **Checked-in parameter defaults** define values such as action weights, source caps, age filters, and diversity constants.
2. **Feature switches and deciders** can enable, disable, or override paths for a request cohort.
3. **Service deployment arguments** matter independently. VM Ranker's `--dpp-enabled` is a concrete example.
4. **Dynamic policy or model configuration** can change enforcement rules, model endpoints, thresholds, and experiment routing without changing this commit.

The main public parameter wiring is in Home Mixer config.

## 14.2 Defaults visible in the commit

| Mechanism | Public default or snapshot |
|---|---|
| Phoenix retrieval | Enabled |
| Phoenix topics | Configured as a separate retrieval path |
| Phoenix mixture-of-experts | Disabled |
| SimClusters | Declared and configured as a candidate source |
| TweetMixer | Disabled |
| Weighted Phoenix value model | Enabled/default |
| Cold-start promotion | Enabled |
| Author diversity attenuation | Enabled |
| OON multiplier | 0.75 |
| Topic OON multiplier | 0.5 |

| Mechanism | Public default or snapshot |
|---|---|
| VM Ranker client | Enabled |
| VM Ranker service-side DPP | Disabled unless deployment flag enables it |
| Organic selector size | 50 |
| Final ordinary result slots | 35 |
| Maximum post age | 48 hours |

No static audit can establish cohort allocations or live switch values that are not present in the repository. Statements in this report therefore distinguish **implemented**, **defaulted**, and **provably live**. Only the first two are established here.

# 15. Important engineering observations

## 15.1 Source availability changes the effective objective

Because sources fail independently, a request with Phoenix unavailable is not just a lower-quality version of the same ranking problem. It has a different feasible set, likely dominated by Thunder or SimClusters. Monitoring should correlate source health with source mix and final engagement, not only total request success.

## 15.2 Prediction independence is followed by explicit slate dependence

Phoenix candidate isolation makes raw action predictions independent of other candidates in the same batch. Author attenuation, cold start, DPP, ads, modules, and conversation deduplication then introduce deliberate slate dependence. This is a clean separation between item utility and feed composition policy.

## 15.3 Negative feedback has asymmetric leverage

The public negative weights are far larger in magnitude than most positive weights. Report has weight -234 versus favorite 0.5 and reply 5. The system can therefore demote a candidate with low predicted negative-event probability even when it predicts several ordinary engagements.

## 15.4 In-network does not mean unfiltered

Thunder inventory enters a ranking pool and later passes core, age, social-graph, seen-state, Phoenix scoring, diversity, and TimelineHome visibility checks. Being followed affects supply and policy level; it is not an unconditional pass to the final timeline.

## 15.5 Safety is both upstream and downstream

Safety affects the feed before ranking through source/index eligibility and prefilters, and after ranking through Visibility Filtering. Phoenix RankAll already excludes some visibility-dropped content from general retrieval. Home Mixer repeats viewer-specific and recommendation-specific policy after selection.

## 15.6 Interstitial is not a ranking penalty

In the public code, `Interstitial` remains eligible through the Home Mixer visibility filter. The warning UI is external. Treating it as equivalent to `Drop` would misdescribe the implementation.

## 15.7 DPP activation cannot be inferred from Home Mixer alone

Home Mixer can request model `dpp` while the VM Ranker process still runs with DPP disabled and echoes scores. Any claim that the live feed uses DPP requires deployment-argument or runtime evidence in addition to source defaults.

## 15.8 Public thresholds are sometimes intentionally nonfunctional

BDSM's `9.99` thresholds and the Abuse Enforcement mock value 12.34 are explicit redactions. A reader should not tune or evaluate policy from them. Similarly, absent Grox prompts and placeholder CLIP URLs prevent exact output reproduction even when control flow is available.

# 16. What is public, partially public, and absent

| Status | Examples |
|---|---|
| Public and structurally executable | Candidate-pipeline control flow, Home Mixer orchestration, action weighting, SimClusters jobs, DPP algorithm, Phoenix synthetic trainer/serving path |
| Public but deployment-dependent | Feature switches, experiment clusters, VM DPP activation, service endpoints, dynamic policy overrides |
| Public with redacted or placeholder values | BDSM thresholds, selected abuse thresholds, CLIP model URLs |
| Structurally visible but content omitted | Grox prompt-driven flows without `.j2` templates |
| Not included | Production data, production checkpoints, exact live experiment allocation, full anti-abuse rule corpus, some internal dependencies, production orchestration and deployment topology |

The omission boundary prevents three common overclaims:

- The repository does not let an outsider recreate the exact live For You feed.
- The synthetic Phoenix harness does not demonstrate recommendation quality.
- Checked-in parameters do not prove which experiments or dynamic overrides are active for a viewer.

# 17. Compact algorithm specification

For a viewer `u`, the public request-time algorithm can be summarized as:

```
Q = hydrate_viewer_and_request(u)

C = union_concurrently(
      Thunder(Q),
      PhoenixRetrieve(Q),
      PhoenixTopicRetrieve(Q),
      SimClustersRetrieve(Q),
      optional_or_cached_sources(Q)
    )

C = hydrate_candidate_features(C, Q)
C = sequential_prefilters(C, Q)
```

```
for each candidate c in C:
    p[c] = PhoenixRanker(Q.history, c)
    s[c] = sum_a weight[a] * p[c, a]
    s[c] = offset_negative_scores(s[c])

s = cold_start_adjust(s, C)
s = repeated_author_attenuation(s, C)
s = out_of_network_attenuation(s, C)
s = optional_MPN(s, C)
s = optional_DPP_membership_selection(s, embeddings(C))

C = top_50_by_score(C, s)
V = visibility_hydrate(C, Q.viewer_context)
C = remove_drop_keep_allow_and_interstitial(V)
C = ancillary_visibility_and_conversation_dedup(C)

F = blend(
      organic=C,
      ads,
      who_to_follow,
      prompts,
      surveys,
      push_to_home,
      jetfuel_frames
    )

async_record_logging_cache_events_and_served_history(F)
return F
```

This specification is accurate at the public control-flow level. The functions remain subject to runtime switches, service health, model snapshots, and policy configuration.

# 18. Source map

The highest-value primary sources for independent verification are:

- Commit diff and metadata
- Release README and scope boundary
- Generic candidate-pipeline execution
- Organic Home Mixer pipeline
- Outer For You blending pipeline
- Public Home Mixer parameters
- Phoenix scoring
- Weighted ranking policy
- Phoenix training qualifications
- SimClusters request source
- DPP implementation
- Visibility rule registry
- Visibility hydration
- Phoenix retrieval corpus builder
- BDSM model and runtime
- UserCredV2 PageRank implementation

# Conclusion

The commit reveals a recommendation architecture built around broad candidate supply, a multi-action neural value model, explicit slate policy, and a separate safety decision system.

The feed's core ranking signal is a weighted sum of Phoenix-predicted actions, but that score only has meaning inside the rest of the pipeline. Thunder and SimClusters determine what can be considered. Filters and retrieval indices remove ineligible inventory. Cold-start, author attenuation, out-of-network multipliers, and optional DPP reshape the slate. Visibility rules decide whether content is allowed, placed behind an interstitial, or dropped. Finally, ads and product modules alter final positions.

The public source is detailed enough to reconstruct the algorithm's control flow and much of its mathematics. It is not detailed enough to reproduce the live service outcome without production data, checkpoints, active experiments, dynamic configuration, withheld safety assets, and deployment evidence.