How the X Algorithm Works: Candidate Sourcing, Ranking, and the Determinants of Reach

Ruchir Jajoo[1] and Husain

Ruchir Jajoo, Founder, **Identity Labs**, India

Husain, Technical Lead, **Identity Labs**, India; IIT Delhi

[1]Corresponding author: ruchir@identity-labs.com

Approximately 500 million posts are published on X each day, of which an individual user is shown on the order of 1,500 — a selection rate of roughly 0.0003%. The mechanism performing this selection is the platform's home timeline recommendation system, substantial portions of which were released publicly in March 2023. We reconstruct that system from its published source and organise it into three sequential stages: candidate sourcing, ranking, and filtering with mixing. We describe the feature taxonomy on which the system operates — post engagement, social graph, and user data — and the four graph-construction packages (RealGraph, GraphJet, TwHIN, SimClusters) that convert raw interaction events into weighted, directed relationships among users, posts, and clusters. We then document the ranking function, a weighted sum over ten predicted engagement probabilities, and tabulate the published weights, which span nearly five orders of magnitude and are strongly asymmetric: a predicted report carries a weight of −369 while a predicted favourite carries 0.5. A worked example traces the assembly of a single post's score end to end. We then identify four points in the pipeline at which an author's behaviour measurably alters distribution, and separate the interventions that are directly evidenced by published constants from those that are inferential. Finally we delimit what the release does not disclose — chiefly the model mapping features to probabilities — and discuss the consequences for external auditability of the system.

Recommender systems | Social media | Algorithmic ranking | Graph embeddings | Platform governance

Population-scale content ranking is now the dominant mechanism by which information reaches audiences on social platforms. On X, formerly Twitter, the asymmetry between supply and attention is extreme: on the order of 500 million posts are authored each day, while a user's home timeline surfaces approximately 1,500 candidate posts per refresh cycle before final presentation. The system that performs this reduction is therefore not a peripheral convenience but the primary determinant of what is seen, by whom, and how often.

The commercial logic is direct. X, like other social media intermediaries, derives revenue predominantly from advertising, and advertisers commit spend in proportion to expected exposure. Sustained user engagement is thus the platform's proximate economic objective, and the recommendation system is the instrument through which that objective is pursued. The system's operative goal is accordingly stated in engagement terms: identify the subset of available content on which a given user is most likely to act.

The home timeline is presented through two tabs. The *Following* tab is a reverse-chronological arrangement of posts authored by accounts the user follows, and involves no learned ranking. The *For you* tab is the algorithmic surface, and is the subject of this paper. Its task can be stated compactly: from the full corpus of recent posts, assemble approximately 1,500 candidates that the user may plausibly be interested in, order them, filter them, and interleave them with advertisements and account recommendations.

In March 2023 X published a substantial portion of the source underlying this pipeline [1, 2], accompanied by an engineering description of the system's architecture [3]. The release is partial: it includes the candidate-sourcing packages, the ranking weights, the visibility filters, and the graph-construction machinery, but excludes the trained model that maps features to engagement probabilities. This paper reconstructs the system from what was released, states the published constants explicitly, and marks the boundary beyond which the system remains opaque.

Our contribution is threefold. First, we give a consolidated account of the pipeline, organised so that each stage's inputs, transformations, and published parameters are identifiable (see Fig. 1). Second, we tabulate the ranking weights and the multiplicative boosts applied during candidate scoring, and demonstrate through a worked example how they compose into a single score. Third, we derive from these constants a set of determinants of reach — author-side behaviours whose effect on distribution follows from the published code — and we distinguish these carefully from claims that the release does not support.

System Overview

The pipeline decomposes into three sequential jobs, preceded by a feature and graph layer that all three consume. Figure 1 shows the full arrangement.

Candidate sourcing retrieves approximately 1,500 posts from a corpus several orders of magnitude larger. Retrieval is split evenly between *in-network* sources, which return posts authored by accounts the user follows, and *out-of-network* sources, which return posts from authors the user does not follow but is predicted to find relevant.

Ranking assigns each retrieved candidate a scalar score. The scoring model, referred to in the source as the Heavy Ranker, consumes on the order of a thousand features per candidate and emits ten probability estimates, each corresponding to a specific engagement outcome. These are combined by a fixed weighted sum.

Filtering, heuristics, and product features then modify the ranked list to satisfy constraints that a pure relevance ordering would violate: legal compliance, author diversity, network balance, and negative-feedback suppression. Finally, a mixing stage interleaves the surviving posts with advertisements and follow recommendations and serves the result.

Two structural observations are worth stating at the outset. First, the majority of the system's discriminative work occurs in the first two stages: candidate sourcing determines what is eligible to be seen at all, and ranking determines the order in which eligible content is seen. Filtering and mixing operate on an already heavily reduced set. Second, the two stages consume overlapping but distinct signals — candidate sourcing operates on graph structure and relationship strength, ranking operates on predicted per-user, per-post engagement — which means that an author's actions can affect distribution through two largely independent channels.

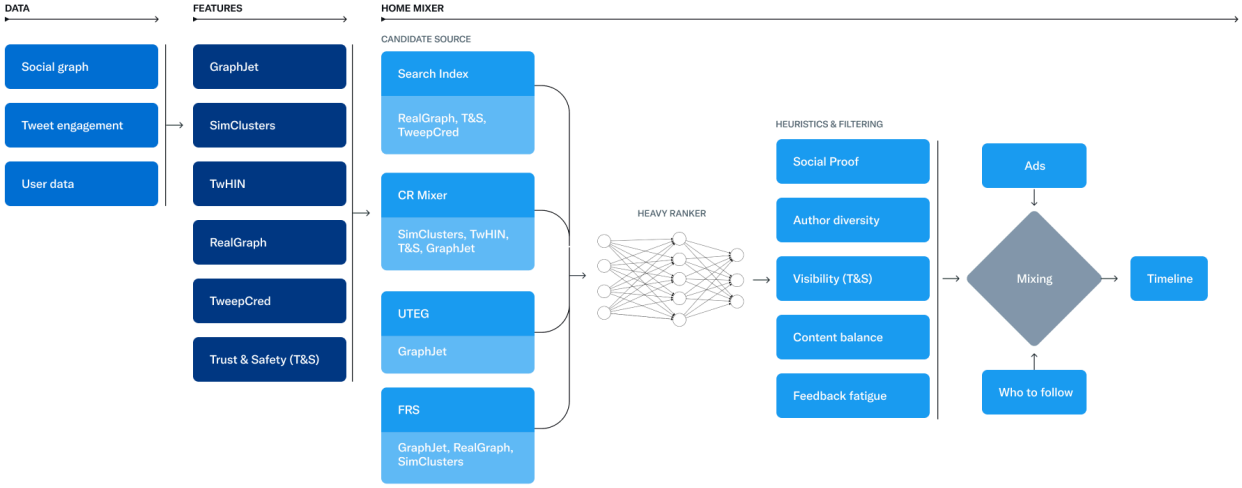

Fig. 1. Architecture of the home timeline recommendation pipeline. Raw signals (left) are grouped into three families — social graph, post engagement, and user data. These are consumed by the feature and graph-construction packages (GraphJet, SimClusters, TwHIN, RealGraph, TweepCred, and the Trust & Safety models), which produce weighted relational representations of users, posts, and clusters. The Home Mixer then invokes four candidate sources — the Search Index for in-network retrieval, and CR Mixer, UTEG, and FRS for out-of-network retrieval — each drawing on a different subset of the upstream packages. The pooled candidate set, approximately 1,500 posts, is scored by the Heavy Ranker, passed through the heuristic and filtering layer (social proof, author diversity, visibility, content balance, feedback fatigue), and finally mixed with advertisements and follow recommendations before being served to the timeline. TweepCred, shown in the feature layer, has since been deprecated.

Feature Representation and Graph Construction

Before the pipeline can rank anything it must answer a prior question: what is the relationship between any two entities on the platform? Consider two users, Alex and Bella, who may be acquaintances, strangers, or anything between. The system must represent not only the strength and direction of their mutual relationship, but their relationships to individual posts, and the relationship of both to the broader interest communities in which they participate.

Features

Features are the atomic inputs to every model in the system. The published feature set is large — on the order of thousands of named features are enumerated in the machine-learning repository [2] — but they fall into three families.

Post engagement features record actions taken on individual posts: favourites (likes), reposts, replies, clicks, profile clicks arising from a post, and dwell or playback signals for media. *Social graph* features record the follow graph and derived structures such as trusted circles. *User data* features record account-level and negative signals: blocks, mutes, unfollows, spam reports, abuse reports, and contextual attributes including geolocation and locally trending topics.

Two properties of this representation matter for what follows. Negative signals are first-class features, not merely the absence of positive ones; and the entities linked by these features are not restricted to users. Nodes in the resulting structures include users, posts, media items, and clusters, numbering in the billions.

Packages

The system's functionality is distributed across a set of services, referred to in the release as packages. Table 1 lists those that are load-bearing for the timeline. Several of these are also used elsewhere on the platform — for search result ranking and for the follow-recommendation service — so their behaviour is not specific to the home timeline.

Table 1. Core packages of the timeline recommendation system.

Package	Function
`RealGraph`	Models the relationship between pairs of users and predicts future interaction [8]
`GraphJet`	Maintains the real-time bipartite user–post interaction graph [7]
`TwHIN`	Embeds the heterogeneous network of users, posts, and other entities [6]
`SimClusters`	Assigns users and posts to overlapping interest communities and represents relationships between them [5]
`HeavyRanker`	Produces the final relevance score for each candidate post [4]
`Trust-and-safety-models`	Classifies abusive, toxic, and NSFW content [10]
`Visibility-filters`	Enforces legal compliance, blocks, mutes, and account state
`Home-mixer`	Orchestrates the pipeline and interleaves organic posts with advertisements and recommendations

Graph construction

Four packages — RealGraph, GraphJet, TwHIN, and SimClusters — jointly construct the relational representation on which candidate sourcing depends. They operate at three levels of granularity.

User–user. Users are nodes; interactions are edges. Each edge is directed, so that the relationship from Alex to Bella is represented independently of the relationship from Bella to Alex, and each edge carries a weight expressing relationship strength. A single repost interaction, for example, may contribute an edge of weight 0.56 (Fig. 2). Multiple interaction types coexist between the same pair of nodes, each contributing its own weighted edge.

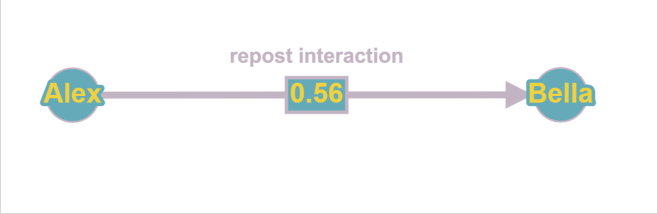

Fig. 2. A weighted, directed user–user edge. The relationship strength arising from Alex reposting Bella's content is 0.56. Many such interaction types coexist between the same pair of users, each carrying its own weight, and the edge from Alex to Bella is distinct from the edge from Bella to Alex.

User–post. Posts are admitted as nodes alongside users, producing a heterogeneous graph in which an edge may connect two users, a user and a post, or a post and a post via shared engagement (Fig. 3). This is the structure GraphJet maintains in real time, and it is what allows retrieval of the form "posts engaged with by users who engage with the same posts as you."

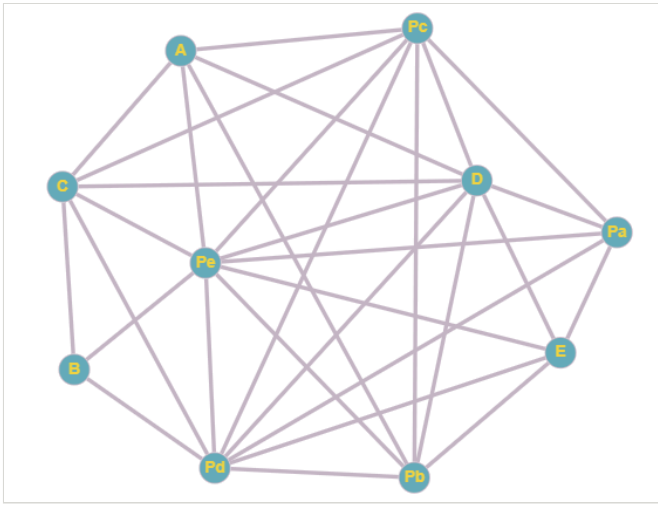

Fig. 3. A heterogeneous user–post graph. Nodes A–E are users; nodes P_a–P_e are the posts they respectively authored. Edges connect users to users, users to posts, and posts to posts via co-engagement. Retrieval traverses this structure to reach content that has no direct follow relationship to the viewer.

Clusters. SimClusters groups users and posts into overlapping interest communities and maintains a graph over the communities themselves (Fig. 4). The published description reports approximately 145,000 clusters, recomputed on a roughly three-week cycle, ranging in size from hundreds of thousands to several million members. Membership is not exclusive: a single account may belong simultaneously to clusters corresponding to, for instance, technology, business, spaceflight, and automotive interest communities. This overlapping membership is what permits recommendation across communities rather than merely within them.

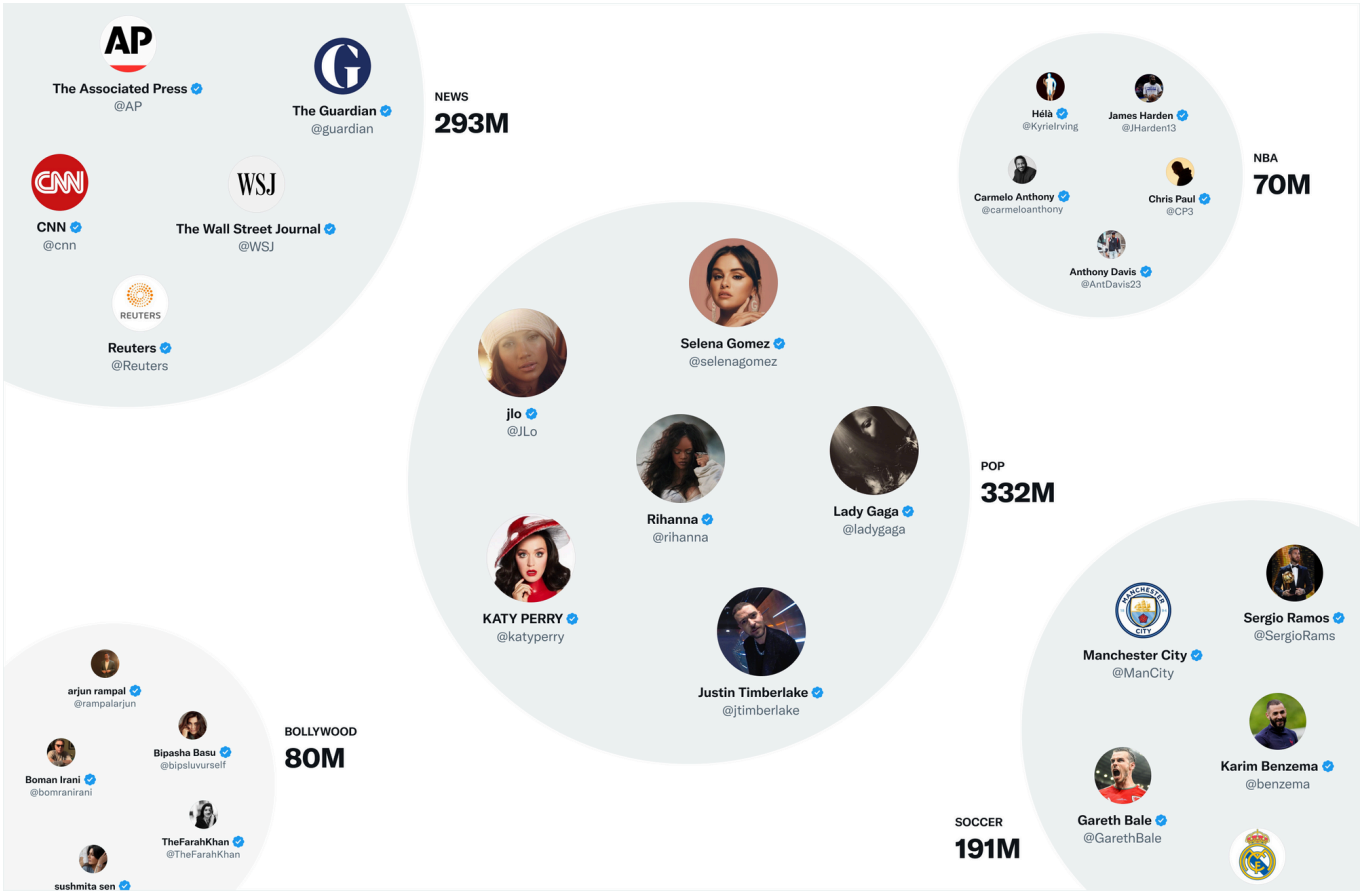

Fig. 4. Overlapping interest communities produced by SimClusters. Users and posts are assigned to communities on the basis of co-engagement rather than declared topic. Cluster sizes shown are illustrative membership counts. Approximately 145,000 such clusters are maintained and refreshed on a roughly three-week cycle; membership is overlapping, so an account may participate in many communities simultaneously, and it is this overlap that supports cross-community recommendation.

The Representation Scorer illustrates how raw features become graph weights. It partitions signals by valence — favourites, reposts, follows, shares, replies, and authored posts are treated as positive; blocks, mutes, reports, and "see fewer" requests as negative — and aggregates each over multiple time windows using both maximum and mean statistics. The parameter block is instructive:

```
// parameters used by Representation-Scorer
1: optional double fav1dLast10Max   // max, last 10 favs, 1 day
2: optional double fav1dLast10Avg   // avg, last 10 favs, 1 day
3: optional double fav7dLast10Max   // max, last 10 favs, 7 days
4: optional double fav7dLast10Avg   // avg, last 10 favs, 7 days
5: optional double retweet1dLast10Max
6: optional double retweet1dLast10Avg
7: optional double retweet7dLast10Max
8: optional double retweet7dLast10Avg
9: optional double follow7dLast10Max
10: optional double follow7dLast10Avg
11: optional double follow30dLast10Max
12: optional double follow30dLast10Avg
13: optional double share1dLast10Max
...
19: optional double reply7dLast10Max
20: optional double reply7dLast10Avg
```

Positive-signal parameters of the Representation Scorer, aggregated over 1-, 7-, and 30-day windows using both maximum and mean statistics over the ten most recent events of each type.

Negative signals are represented with the same structure and comparable granularity, occupying a dedicated identifier range:

```
// 2001 - 3000: Negative Signals
2001: optional double block1dLast10Avg
2002: optional double block1dLast10Max
2003: optional double block7dLast10Avg
2005: optional double block30dLast10Avg
2101: optional double mute1dLast10Avg
2103: optional double mute7dLast10Avg
2105: optional double mute30dLast10Avg
2201: optional double report1dLast10Avg
2203: optional double report7dLast10Avg
2205: optional double report30dLast10Avg
2301: optional double dontlike1dLast10Avg
2303: optional double dontlike7dLast10Avg
2401: optional double seeFewer1dLast10Avg
2403: optional double seeFewer7dLast10Avg
2405: optional double seeFewer30dLast10Avg
```

Negative-signal parameters, abridged. Blocks, mutes, reports, explicit dislikes, and "see fewer" requests are each tracked over 1-, 7-, and 30-day windows.

The symmetry of treatment is notable: negative feedback is not modelled as a filter applied after ranking, but as a feature that shapes the relational graph itself, with a memory of up to thirty days.

Candidate Sourcing

The first stage retrieves approximately 1,500 posts. The published description specifies an even split between two retrieval regimes.

In-network retrieval supplies roughly 750 posts, drawn from accounts the user follows and ranked primarily by a logistic model of engagement likelihood. This path is

comparatively simple: eligibility is determined by the follow graph, and the principal question is ordering.

Out-of-network retrieval supplies the remaining 750 from authors the user does not follow, and is where the graph machinery does its work. Two mechanisms dominate.

The first is *social graph traversal*, which accounts for approximately 15% of the timeline and answers questions of the form "what have the accounts I follow engaged with?" and "what have accounts with engagement patterns similar to mine engaged with?" If Alex follows Bella and Bella favourites a post by Charlie, that post becomes eligible for Alex's timeline. If Alex and Bella both favourited a post by Charlie without following one another, and Bella subsequently favourites a post by David, David's post becomes eligible for Alex. This is served principally by GraphJet through the UTEG candidate source.

The second is *embedding-space retrieval*, the most technically involved component of the system. Rather than traversing explicit engagement edges, it operates in learned representation spaces — SimClusters for community membership and TwHIN for dense entity embeddings — and retrieves content on the basis of representational proximity. Users who share no common engagements at all may nonetheless occupy adjacent regions of the space by virtue of similar interest profiles, and will be shown content prominent within those regions. This is the mechanism by which content reaches audiences with no traceable connection to the author.

The Trust and Safety models operate in parallel with all retrieval paths rather than as a subsequent stage, so that content classified as violating is not merely down-ranked but excluded from candidacy.

Ranking

Given approximately 1,500 candidates, the ranking stage determines presentation order. The published description states that the ranker consumes thousands of features and outputs ten labels, each representing the probability of a specific engagement, which are then combined into a single score [3]. The combination is a fixed linear form:

$$\text{score} = \sum_i w_i \cdot p_i \qquad [1]$$

where $p_i \in [0,\ 1]$ is the model's predicted probability of engagement outcome i, and w_i is a fixed weight configured per outcome. The probabilities are computed jointly from features of the post, features of the viewing user, and features of the relationship between them. The weights are not learned at serving time; they are configuration constants, published in the scored-tweets parameter file [4], and are reproduced in Table 2.

Table 2. Ranking weights by predicted engagement outcome. Values as published in the scored-tweets parameter configuration [4].

Predicted outcome	Sign	Weight
Favourites the post	+	0.5
Reposts the post	+	1
Replies to the post	+	13.5
Opens the author's profile and favourites or replies	+	12
Watches at least half of an attached video	+	0.005
Replies, and the author engages with that reply	+	75
Opens the conversation and favourites or replies	+	11
Opens the conversation and remains ≥ 2 minutes	+	10
Requests "show less often", blocks, or mutes the author	−	−74
Reports the post	−	−369

Weights multiply *predicted* probabilities and are therefore applied before any of these actions occur. The corresponding configuration keys are `scored_tweets_model_weight_fav`, `_retweet`, `_reply`, `_good_profile_click`, `_video_playback50`, `_reply_engaged_by_author`, `_good_click`, `_good_click_v2`, `_negative_feedback_v2`, and `_report`.

Three features of this weight vector merit comment. First, its dynamic range is extreme: the largest positive weight exceeds the smallest by a factor of 1.5×10^4. The predicted probability of a video half-play, at 0.005, contributes essentially nothing relative to the predicted probability of a reply that the author then engages with, at 75. Second, the ordering does not track the observable frequency of the behaviours. Favourites are by a wide margin the most common engagement on the platform, yet they carry the second-lowest positive weight; the two highest-weighted outcomes both involve sustained conversational exchange, which is comparatively rare. The system is configured to prefer content predicted to generate conversation over content predicted to generate approval. Third, the negative weights dominate. A predicted report is weighted at −369, more than four times the magnitude of the largest positive weight, so that even a small predicted probability of a report can offset substantial predicted positive engagement.

The corresponding configuration, as published:

```
scored_tweets_model_weight_fav:                    0.5
scored_tweets_model_weight_retweet:                1.0
scored_tweets_model_weight_reply:                 13.5
scored_tweets_model_weight_good_profile_click:    12.0
scored_tweets_model_weight_video_playback50:       0.005
scored_tweets_model_weight_reply_engaged_by_author: 75.0
scored_tweets_model_weight_good_click:            11.0
scored_tweets_model_weight_good_click_v2:         10.0
scored_tweets_model_weight_negative_feedback_v2:  -74.0
scored_tweets_model_weight_report:               -369.0
```

The critical limitation of the release is located precisely here. The weights w_i are fully published; the function mapping features to the probabilities p_i is not. Equation 1 is therefore known in form and in half its parameters, while the half that carries the actual predictive content remains undisclosed. We return to the consequences in the discussion.

A Worked Example

To make the composition concrete, consider a single post and a single viewer. Let an author publish a post, denoted

ABC, and let Alex be a candidate viewer for whom ABC has been retrieved. Suppose the ranking model assigns a probability of 0.5 to each of the eight positive outcomes and 0.001 to each of the two negative outcomes. These values are stipulated for illustration; they are not measured.

WORKED EXAMPLE: SCORING ONE POST

Applying Equation 1 with the weights of Table 2:

```
(  0.5 × 0.5)  +  (  1.0 × 0.5)
+ ( 13.5 × 0.5)  +  ( 12.0 × 0.5)
+ (0.005 × 0.5)  +  ( 75.0 × 0.5)
+ ( 11.0 × 0.5)  +  ( 10.0 × 0.5)
+ (-74.0 × 0.001) + (-369.0 × 0.001)

= 61.5025 - 0.443  =  61.0595
```

The eight positive terms contribute 61.50; the two negative terms subtract 0.44. Note the composition of the positive total: the single term for "replies, and the author engages" contributes 37.5, or 61% of the entire positive score, while favourite and repost together contribute 0.75, or 1.2%.

Note further that probabilities are estimated from the union of three feature sets — those describing Alex, those describing the author, and those describing the post. Consequently, negative feedback received from *other* users raises the predicted probability of negative feedback from Alex, and thereby depresses the score, even where Alex has never previously reacted negatively to this author.

The final ordering over all 1,500 candidates is obtained by computing this score for each and sorting. The sorted list then proceeds to the filtering stage.

Filtering, Heuristics, and Product Features

A pure relevance ordering satisfies neither legal obligation nor product quality requirements, and a set of post-ranking modifications is therefore applied.

Visibility filtering removes posts from blocked or muted accounts, from suspended accounts, and from accounts or content withheld under legal instruction in the viewer's jurisdiction. *Author diversity* suppresses consecutive posts from a single author, preventing a highly-scored author from monopolising a contiguous region of the timeline. *Content balance* maintains the intended proportion of in-network and out-of-network content. *Feedback-based fatigue* reduces the visibility of content resembling that on which the user has previously given negative feedback. *Social proof* imposes a connection requirement on out-of-network content: such posts are surfaced only where some account the user follows has engaged with them, which excludes content reachable only through an unvouched path. *Conversation threading* attaches replies to their parent posts to preserve context, and *edited-post handling* replaces superseded versions with current ones.

Timing enters at this stage through an age-decay function. The published parameters specify a half-life of 360 minutes:

```
struct ThriftAgeDecayRankingParams {
  // rate at which older posts' scores decrease
  1: optional double slope    = 6.003
  // age in minutes at which the age score is
  // half that of the newest post
  2: optional double halflife = 360.0
  // floor value of the age decay score
  3: optional double base     = 0.6
}(persisted='true')
```

A post's age-derived contribution therefore halves every six hours, subject to a floor of 0.6. The practical implication is that the engagement a post accrues in its first hours is disproportionately consequential: engagement arriving after the decay has taken effect is multiplied by a smaller factor and propagates less far.

Mixing and Serving

The final stage is performed by the Home Mixer, which interleaves the filtered organic posts with advertisements and with follow recommendations from the recommendation service:

```
Ranked posts + Advertisements
             + Follow recommendations
             = "For you" timeline
```

This is the point at which the platform's revenue mechanism is applied to the ranked output, and it is also the only stage at which paid placement can substitute for algorithmic eligibility.

Determinants of Reach

We now invert the analysis. Given the structure above, at which points can an author's behaviour alter the distribution their content receives? Four such points exist, corresponding to the four stages. We treat each in turn, and we distinguish throughout between effects that follow directly from published constants and effects that are inferred.

At the candidate-sourcing level

The graph-construction packages represent an author and their posts as nodes among billions. The objective at this level is to increase both the number of edges incident on those nodes and the weight carried by each. The linear ranking parameters used in candidate scoring are published in full:

```
private def getLinearRankingParams: ThriftRankingParams = {
  ThriftRankingParams(
    `type` = Some(ThriftScoringFunctionType.Linear),
    minScore = -1.0e100,
    retweetCountParams = Some(...(weight = 20.0)),
    replyCountParams   = Some(...(weight =  1.0)),
    reputationParams   = Some(...(weight =  0.2)),
    luceneScoreParams  = Some(...(weight =  2.0)),
    textScoreParams    = Some(...(weight =  0.18)),
    urlParams          = Some(...(weight =  2.0)),
    isReplyParams      = Some(...(weight =  1.0)),
    favCountParams     = Some(...(weight = 30.0)),
    langEnglishUIBoost         = 0.5,
    langEnglishTweetBoost      = 0.2,
    langDefaultBoost           = 0.02,
    unknownLanguageBoost       = 0.05,
    offensiveBoost             = 0.1,
    inTrustedCircleBoost       = 3.0,
    multipleHashtagsOrTrendsBoost = 0.6,
    inDirectFollowBoost        = 4.0,
    tweetHasTrendBoost         = 1.1,
    selfTweetBoost             = 2.0,
    tweetHasImageUrlBoost      = 2.0,
    tweetHasVideoUrlBoost      = 2.0,
    useUserLanguageInfo        = true,
    ageDecayParams = Some(ThriftAgeDecayRankingParams(
        slope = 0.005, base = 1.0))
```

Linear ranking parameters governing candidate scoring. Boosts are multiplicative; values above 1 amplify and values below 1 attenuate.

These constants separate along the three feature families. Tables 3 and 4 reorganise them by direction of effect.

Table 3. Multiplicative boosts applied during candidate scoring.

Condition	Factor
Post receives a favourite (like or bookmark)	30×
Post is reposted	20×
Author is directly followed by the viewer	4×
Author is within the viewer's trusted circle	3×
Post contains an image	2×
Post contains a video	2×
Post aligns with a current trend	1.1×
Post receives a reply	1×

Table 4. Attenuations applied during candidate scoring. Factors below 1 reduce the score; the equivalent divisor is given for interpretability.

Condition	Factor	Equivalent
Unknown language or unrecognised words	0.05	÷ 20
Offensive terms present	0.1	÷ 10
Multiple hashtags or trend terms	0.6	÷ 1.7
Out-of-network reply (subtractive penalty)	−10	—

The out-of-network reply penalty is subtractive rather than multiplicative and is applied after boosts: `120: optional double OutOfNetworkReplyPenalty = 10.0`.

Separately, verification status enters as a multiplier on the author rather than the post, with distinct values for in-network and out-of-network distribution:

```
object BlueVerifiedAuthorInNetworkMultiplierParam
  extends FSBoundedParam[Double](
    name =
"home_mixer_blue_verified_author_in_network_multiplier",
    default = 4.0, min = 0.6, max = 100.0)

object BlueVerifiedAuthorOutOfNetworkMultiplierParam
  extends FSBoundedParam[Double](
    name =
"home_mixer_blue_verified_author_out_of_network_multiplier",
    default = 2.0, min = 0.6, max = 100.0)
```

A verified author's content therefore receives a fourfold multiplier when distributed to their own followers and a twofold multiplier when distributed beyond them. We note that these are declared as bounded parameters with configurable minima and maxima, not as constants: the defaults are 4.0 and 2.0, but the permitted range extends from 0.6 to 100.0, and the operative value at any given time is set by remote configuration rather than by the code.

Negative user-data signals are aggregated into a single edge-feature union:

```
val allEdgeFeatures: SCollection[Edge] =
  getEdgeFeature(SCollection.unionAll(
    Seq(blocks, mutes, abuseReports,
        spamReports, unfollows)))

val negativeFeatures: SCollection[KeyVal[Long, UserSession]] =
  allEdgeFeatures
    .keyBy(_.sourceId)
    .topByKey(maxDestinationIds)(Ordering.by(_.features.size))
    .map { case (srcId, pqEdges) =>
        val topKNeg = pqEdges.toSeq
          .flatMap(toRealGraphEdgeFeatures(hasNegativeFeatures))
        KeyVal(srcId, UserSession(...)) }
```

Blocks, mutes, abuse reports, spam reports, and unfollows all contribute, and their effect persists for up to ninety days from the triggering event. Unfollows are penalised less heavily than the other four. Where any of these are present between a viewer and an author, the relationship is not merely weakened but effectively suppressed, and the same holds for content classified as NSFW, abusive, or toxic.

The Trust and Safety layer applies four classifiers, whose published descriptions are:

```
pNSFWMedia: detects posts with NSFW imagery,
  including adult and pornographic content.
pNSFWText:  detects posts with NSFW text and
  adult or sexual topics.
pToxicity:  detects toxic posts. Includes marginal
  content such as insults and certain forms of
  harassment. Toxic content does not necessarily
  violate the terms of service.
pAbuse:     detects abusive content, including
  hate speech, targeted harassment, and abusive
  behaviour that violates the terms of service.
```

The distinction drawn in the third of these is significant for interpretation: `pToxicity` explicitly captures content that does *not* violate the terms of service. Classification under it therefore imposes a distribution penalty without a corresponding policy violation, and the affected author receives no notification of enforcement because none formally occurs.

The set of terms and topics attracting attenuation is not fixed. The media safety label enumeration [9] is versioned in the public repository and has changed materially over time; posts concerning the war in Ukraine were attenuated during one period and subsequently were not. Former head of Trust

and Safety Yoel Roth has stated publicly that enforcement was expanded rather than reduced following the 2022 change of ownership [11], which is consistent with the observable growth of the label set.

One package documented in earlier versions of the release, TweepCred, adjusted author credibility according to the ratio of followers to followed accounts. It has since been deprecated, and analyses that rely on it are no longer applicable.

At the ranking level

The ranking stage admits influence through two channels, of which only one is accessible from the published material. The feature-to-probability model is undisclosed, so the mapping from author behaviour to predicted probability cannot be characterised from the release. The weight vector, however, is fully published (Table 2), and it determines which predicted behaviours are worth inducing.

Read as a specification of what the system values, the vector is unambiguous. Content predicted to produce a reply that the author then engages with is weighted at 75, fifteen times the weight of a predicted repost and a hundred and fifty times that of a predicted favourite. Content predicted to hold a viewer in a conversation for two minutes or more is weighted at 10. Content predicted to draw a profile visit that converts to engagement is weighted at 12. In each case the weighted outcome is a sustained interaction rather than a momentary one.

At the filtering and heuristics level

This level is governed principally by the age-decay function and by the legal and safety filters. The six-hour half-life implies that the timing of publication relative to the target audience's activity window is consequential, and that early engagement compounds: engagement received while the decay multiplier is near its maximum both raises the score directly and extends the window during which the post remains competitive against newer content. The decay is smooth and bounded below by a floor, so the effect is a steep reduction in competitiveness rather than removal.

Because the audience activity distribution differs by author, no general optimum publication time follows from the published parameters. The determination is empirical.

At the mixing level

The mixing stage admits one form of influence that bypasses the preceding three entirely: paid placement. Advertisements are interleaved by the Home Mixer independently of the organic ranking, so purchased distribution is not subject to the candidate-sourcing and ranking constraints described above. This is the only channel in the pipeline where eligibility can be obtained rather than earned.

Practical Implications

We group the implications of the foregoing by the strength of the evidence supporting them.

Directly supported by published constants

- *Verification.* A verified author receives a 4× multiplier to followers and 2× beyond them, subject to the caveat above regarding remote configuration.
- *Hashtag restraint.* Multiple hashtags or trend terms attract a 0.6 multiplier, equivalent to division by 1.7. A single relevant term avoids this while retaining the 1.1× trend boost.
- *Media attachment.* An image or video carries a 2× multiplier.
- *Language and orthography.* Unrecognised words or languages attract a 0.05 multiplier — division by twenty, the single largest attenuation in the published set.
- *Offensive terms.* A 0.1 multiplier applies, independently of whether any policy is violated.
- *Author replies to commenters.* The outcome weighted at 75 requires the author to engage with a reply. It is the only high-weight outcome whose realisation is partly under the author's direct control.
- *Long-form content.* The outcome weighted at 10 requires a viewer to remain in a conversation for at least two minutes, which places a floor on the content length capable of producing it.
- *Profile presentation.* The outcome weighted at 12 requires a profile visit that converts to engagement, making the state of the profile a determinant of the score of every post the author publishes.
- *Topical concentration.* Consistent posting within a narrow topic strengthens membership in a small number of SimClusters, and a trusted-circle relationship carries a 3× multiplier.
- *Out-of-network replies.* A subtractive penalty of 10 applies, which must be offset by engagement on the reply itself to be worthwhile.

Supported by structure but not by explicit constants

- *Publication timing.* The six-hour half-life makes timing consequential, but the optimum is audience-specific and must be established empirically.
- *Content quality.* Every published weight is a function of predicted engagement, and predicted engagement is estimated from realised engagement on comparable content. Quality is therefore upstream of every term in the score, though it is not itself a parameter.

Risks and asymmetries

- *Negative feedback is durable and transferable.* Blocks, mutes, and reports persist in the graph for up to ninety days, and because probabilities are estimated from author-level as well as viewer-level features, negative feedback from one audience depresses distribution to all others.
- *The penalty asymmetry is severe.* At −369, a predicted report outweighs the largest positive term by more than fourfold. Content that reliably attracts both strong engagement and reports may be net-negative in score notwithstanding the engagement.
- *Attenuation is silent.* The `pToxicity` classifier by its own description covers content that does not violate the terms of service, so an author may be attenuated without any enforcement action having been taken and without notification.

Limitations

Several limitations bound what can be concluded from the published material, and we state them explicitly.

The predictive model is not disclosed. Equation 1 is published in full form and its weights are public, but the function producing the probabilities p_i is not part of the release. Since that function contains all of the system's learned behaviour, external parties cannot determine why a particular post received a particular score, cannot audit the model for differential treatment across groups, and cannot reproduce any ranking decision. The published weights indicate what the system optimises for; they do not indicate how it decides.

Configuration supersedes code. Multiple parameters examined here are declared as remotely configurable bounded values rather than as constants. The verification multipliers, for instance, default to 4.0 and 2.0 within a permitted range of 0.6 to 100.0. The operative values at any given moment are not observable from the repository, and conclusions drawn from defaults may not describe production behaviour.

The release is a snapshot and has drifted. The published source reflects the system as of its release date. TweepCred has been deprecated since. The media safety labels are versioned and have changed materially. The proportion of production behaviour still described by the release is not determinable from outside.

The published material is internally inconsistent in places. We encountered several discrepancies and resolve them as follows. The age-decay parameters appear twice with different values: `slope = 6.003, base = 0.6` in the age-decay struct and `slope = 0.005, base = 1.0` in the linear ranking parameters. The half-life of 360 minutes is consistent across both, and we have used it. The reply weight appears as 13.5 in the scored-tweets configuration and as 27 in one feature listing; we have used the former, which is the operative configuration file. Where a discrepancy could not be resolved we have preferred the configuration file over documentation.

Scope. This analysis addresses the home timeline only. Search ranking, notifications, and the follow-recommendation service share several packages but are governed by distinct parameters and are outside our scope.

Conclusion

The X home timeline reduces a corpus of roughly 500 million daily posts to approximately 1,500 candidates and then to an ordered feed, by way of a pipeline whose structure is now substantially public. The reduction proceeds in three stages, of which the first two carry nearly all of the discriminative weight: candidate sourcing determines eligibility through weighted graph traversal and embedding-space retrieval, and ranking determines order through a fixed linear combination of ten predicted engagement probabilities.

Reading the published weights as a statement of the system's objective yields a clear picture. The platform is configured to prefer content predicted to generate sustained conversational exchange over content predicted to generate passive approval, by a margin of two orders of magnitude, and to penalise predicted negative response far more heavily than it rewards predicted positive response. These are design choices, expressed as constants in a configuration file, and they are more legible than most such choices are in commercial recommender systems.

What remains undisclosed is the part that matters most for accountability. The mapping from features to probabilities — the component that determines what the system actually predicts about any specific post and any specific viewer — was not released, and without it no ranking decision can be reproduced or audited from outside. The release therefore establishes what the system is trying to do while leaving open how it does it. Meaningful external oversight of ranking on this platform requires the second, and it is not currently available.

References

1. X Corp. (2023) The Algorithm: source code for the X recommendation algorithm. GitHub repository. github.com/twitter/the-algorithm
2. X Corp. (2023) The Algorithm (ML): model training and feature definitions. GitHub repository, `projects/home/recap/FEATURES.md`. github.com/twitter/the-algorithm-ml
3. X Engineering (2023) Twitter's Recommendation Algorithm. Engineering blog, 31 March 2023. blog.x.com/engineering/en_us/topics/open-source/2023/twitter-recommendation-algorithm
4. X Corp. (2023) `ScoredTweetsParam.scala`, home-mixer scored-tweets parameter configuration. In [1]. home-mixer/…/scored_tweets/param/ScoredTweetsParam.scala
5. Satuluri V, Parthasarathy S, Ramanathan M, *et al.* (2020) SimClusters: Community-based representations for generalizing large-scale and diverse recommendation systems. *Proc. 26th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pp 3183–3193. doi.org/10.1145/3394486.3403370
6. El-Kishky A, Markovich T, Park S, *et al.* (2022) TwHIN: Embedding the Twitter heterogeneous information network for personalized recommendation. *Proc. 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pp 2842–2850. doi.org/10.1145/3534678.3539080
7. Sharma A, Jiang J, Bommannavar P, Larson B, Lin J (2016) GraphJet: Real-time content recommendations at Twitter. *Proc. VLDB Endowment* 9(13):1281–1292. vldb.org/pvldb/vol9/p1281-sharma.pdf
8. Kamath K, Sharma A, Wang D, Yin Z (2014) RealGraph: User interaction prediction at Twitter. *Workshop on User Engagement Optimization*, ACM SIGKDD. ueo-workshop.com/wp-content/uploads/2014/04/sig-alternate.pdf
9. X Corp. (2023) `MediaSafetyLabelType.scala`, visibility-library label models. In [1]. visibilitylib/…/visibility/models/MediaSafetyLabelType.scala
10. X Corp. (2023) Trust and Safety models: `pNSFWMedia`, `pNSFWText`, `pToxicity`, `pAbuse`. In [1]. github.com/twitter/the-algorithm/tree/main/trust_and_safety_models
11. Roth Y, remarks on content moderation enforcement following the 2022 change of platform ownership, as reported in the press, November 2022.
12. X Corp. (2023) `ThriftRankingParams` and `ThriftAgeDecayRankingParams`, Earlybird search index ranking configuration. In [1]. src/thrift/…/search/common/ranking/ranking.thrift