Driving Engagement in Enterprise Social Networks:

How ML-Based Feeds, Recommendations, Q&A Ranking, and Spam Detection Transformed DAU/MAU at Salesforce Community Cloud

Anjan Goswami

Abstract

Salesforce Community Cloud—an enterprise social networking platform serving both internal employee communities and customer-facing support forums—faced a persistent engagement problem: daily and monthly active user metrics (DAU/MAU) were below targets despite growing adoption. This case study describes the diagnosis, design, and deployment of four interconnected ML systems built to address this problem: (1) a feed relevance ranking system (later branded *Salesforce Highlights*) that surfaced important organizational content with carefully calibrated recency–relevance tradeoffs; (2) a multi-entity recommender system using matrix factorization across users, groups, files, and articles with adaptive multi-objective targets; (3) a Q&A answer ranking system for customer-facing forums that handled noisy responses, semantic question matching, and question routing; and (4) a spam and fraud detection system using feature hashing via Vowpal Wabbit. The work required cross-functional coordination with product management, design, legal, and operations—including establishing pilot customer agreements, building a dedicated human evaluation team, and designing simulation environments for continuous parameter tuning. The combined systems produced significant improvements in DAU and MAU across both internal and customer-facing communities.

1 The Starting Point: An Engagement Problem

Salesforce Community Cloud provided organizations with branded online communities in two primary configurations. **Internal communities** served as employee social networks—spaces for collaboration, knowledge sharing, and organizational communication. **Customer-facing communities** functioned as support forums and discussion spaces where customers asked questions, shared solutions, and interacted with company representatives.

Both configurations suffered from the same core problem: **low engagement**. Users created accounts but did not return. DAU/MAU ratios—the standard measure of engagement stickiness—were below targets. Communities felt static. Important content was buried. New users saw generic feeds with no personalization. Customer questions went unanswered or received low-quality responses buried among noise.

I was given a direct mandate: **improve DAU and MAU**. The question was where to invest.

1.1 Diagnosing Where Engagement Breaks

Before proposing solutions, I mapped the user journey across both community types to identify where engagement dropped off. The diagnosis revealed four distinct failure modes:

The key insight I brought to the PM and design teams was that **feeds touch everyone**. Every user who opens the community sees their feed first. If the feed shows irrelevant content, the user's first interaction is negative, and they leave. Feeds were the highest-leverage surface for DAU improvement—like news for an organization. Recommendations addressed the discovery problem ("what else should I engage with?"), while Q&A quality and spam detection addressed the customer-facing retention problem.

Failure Mode	Symptom	Affects	Proposed Fix
Feed irrelevance	Users see chronological noise, miss important content	Internal communities	Feed relevance ranking
Discovery failure	Users don't find relevant people, groups, files	Both	Recommender system
Q&A quality	Questions get noisy, unhelpful answers; good answers buried	Customer-facing	Answer ranking
Spam and noise	Forums polluted with spam, low-quality posts	Customer-facing	Spam detection

Table 1: Engagement failure modes and corresponding interventions.

I proposed all four workstreams, worked with PM and designers on the product framing, but drove the technical strategy and execution. The PM and design teams focused on surfaces and UX; I focused on the science and infrastructure.

2 Workstream 1: Feed Relevance Ranking (Salesforce Highlights)

2.1 The Problem with Chronological Feeds

The existing feed was strictly chronological: posts appeared in reverse time order. In a 500-person internal community, this meant a CEO's strategic announcement was buried within minutes by routine status updates. An important technical article shared by a domain expert disappeared below casual conversation. HR announcements about benefits enrollment were invisible by the afternoon.

The feed treated all content equally. But content is not equal: a comment from a company leader carries different organizational weight than a casual reply. A technical article relevant to a user's role is more valuable than a social post from a distant department. **The chronological feed optimized for recency, not relevance.**

2.2 Design Inspiration: Studying Twitter's Approach

Before building, I studied how Twitter handled the transition from chronological to ranked feeds. Twitter's approach offered several lessons:

- **Recency cannot be discarded**: Users expect feeds to feel "fresh." A purely relevance-ranked feed that surfaces week-old content feels broken, even if that content is objectively important. Twitter's solution was a relevance-weighted feed with a strong recency prior.
- **Decay functions matter**: Twitter used time-decay functions that reduced content scores as they aged, but the decay rate varied by content type and engagement level. High-engagement content decayed more slowly.
- **User control is essential**: Twitter offered users a toggle between ranked and chronological feeds. Enterprise users—particularly executives—would demand similar control.

These observations shaped our design: the feed ranker had to balance relevance and recency through explicit, tunable parameters—not as a side effect of the model, but as a first-class design constraint.

2.3 The ML Ranking Model

The feed ranker scored each candidate post for each user. The score combined three components:

$$\text{Score}(u, p) = f_{\text{relevance}}(u, p) \cdot g_{\text{recency}}(t_p) \cdot h_{\text{authority}}(p)$$

Relevance $f_{\text{relevance}}(u, p)$: An ML model predicting the probability that user u would engage with post p. Features included:

- Topical similarity between the user's interest profile and the post content
- Organizational relationship features: same department, reporting chain proximity, shared group memberships
- Author features: the author's role level, historical engagement rate on their posts, whether the user had previously interacted with this author
- Content type: article, announcement, comment, file share, status update
- Historical engagement on this specific post: number of likes, comments, shares accumulated so far

Feature design was the critical investment. The model itself was a gradient boosted tree—the architecture was not the differentiator. **The features were the differentiator.** Features that captured organizational hierarchy ("this post is from a VP two levels above you in the reporting chain") and content category ("this is tagged as an HR announcement") encoded domain knowledge specific to enterprise social networks that no amount of behavioral data could discover on its own.

Recency decay $g_{\text{recency}}(t_p)$: A parametric decay function that reduced scores as posts aged:

$$g_{\text{recency}}(t_p) = \exp\left(-\frac{(t_{\text{now}} - t_p)}{\tau}\right)$$

where τ is the half-life parameter controlling how quickly content fades. This was the most delicate parameter in the system. Too short (e.g., $\tau = 4$ hours), and important announcements disappeared before West Coast employees saw them. Too long (e.g., $\tau = 7$ days), and the feed felt stale, filled with content users had already seen. We did not want month-old comments surfacing, but a CEO statement from two days ago should still be visible.

The solution was **content-type-specific decay rates**:

Content Type	**Half-life τ**	**Rationale**
HR/policy announcements	72–96 hours	Must reach entire org
Leadership posts	48–72 hours	High authority, wide relevance
Technical articles	48 hours	Topically relevant but less urgent
General posts	24 hours	Standard social content
Comments and replies	12–18 hours	Conversational, fast-decaying

Table 2: Content-type-specific recency decay parameters.

These parameters were not set once—they were **continuously tuned through human evaluation** (described in Section 6).

Authority $h_{\text{authority}}(p)$: A multiplier reflecting the organizational importance of the post's author and content type. Leadership posts, official announcements, and content from designated

subject matter experts received authority boosts. This component ensured that the feed reflected organizational structure, not just behavioral popularity.

2.4 The Recency–Relevance Tradeoff: What We Got Wrong First

Our first model over-indexed on relevance. It surfaced highly relevant content regardless of age, which meant users saw three-day-old articles they had already encountered through other channels. The feed felt repetitive and stale despite being "smarter." Early pilot feedback was negative: users preferred the chronological feed because at least it showed new things.

The fix was introducing the multiplicative recency decay as a hard constraint rather than a soft feature. Relevance could boost content within its freshness window, but could not resurrect content past a content-type-specific age threshold. This produced the right behavior: important content stayed visible *longer* than routine content, but nothing persisted indefinitely.

3 Workstream 2: Multi-Entity Recommender System

3.1 From Heuristics to Collaborative Filtering

The existing recommendation system was rule-based: keyword matching between user profiles and content metadata, augmented by recency and popularity heuristics. It had no semantic understanding, no behavioral learning, and no cross-entity reasoning. Person recommendations, group recommendations, file recommendations, and article recommendations operated as four independent systems.

Our first attempt—standard collaborative filtering on interaction data—failed for sparse organizations. A 200-person community generates insufficient behavioral signal for matrix factorization to converge meaningfully. Recommendations were either random or popularity-biased (recommending the same items to everyone).

The solution required **combining collaborative filtering with topical features** to handle cold-start and sparse organizations.

3.2 Unified Latent Space

We constructed a shared k-dimensional latent space embedding all entity types—users ($\mathbf{U} \in \mathbb{R}^{|\mathcal{U}|\times k}$), groups ($\mathbf{G} \in \mathbb{R}^{|\mathcal{G}|\times k}$), files ($\mathbf{F} \in \mathbb{R}^{|\mathcal{F}|\times k}$), and articles ($\mathbf{A} \in \mathbb{R}^{|\mathcal{A}|\times k}$). The base affinity between user u and entity e is the inner product $\mathbf{u}_u^\top \mathbf{v}_e$, supplemented by organization-specific topical features extracted via TF-IDF against cross-organization background frequencies.

The unified space was the key architectural decision: a user's file interactions informed their group recommendations, and group memberships informed people recommendations. Four independent engines could not achieve this cross-entity discovery.

3.3 Multi-Objective Target and Adaptive Weights

Each entity type had distinct interaction signals—likes, shares, comments, follows, reads—with different engagement semantics. The composite engagement target:

$$y_{ue} = \alpha \cdot \ell_{ue} + \beta \cdot h_{ue} + \gamma \cdot c_{ue} + \delta \cdot f_{ue} + \rho \cdot r_{ue}$$

where $\boldsymbol{\theta} = (\alpha, \beta, \gamma, \delta, \rho)$ controlled what the model optimized for.

The target weights were the most consequential design decision. We initially assumed likes would dominate (highest frequency signal). Human evaluation revealed the opposite: **comments carried the highest quality signal** (γ dominant), followed by shares (β), reads (ρ for files/articles), follows (δ), and likes (α) last. Likes were dense but generic; comments required cognitive effort and indicated genuine engagement.

The data access problem: We could not access customer organization data for training. We trained default parameters on Salesforce's own internal Community Cloud deployment, then created a Bayesian adaptation mechanism for customer organizations:

$$\boldsymbol{\theta}_{\text{org}}^{(t+1)} = (1 - \eta) \cdot \boldsymbol{\theta}_{\text{org}}^{(t)} + \eta \cdot \hat{\boldsymbol{\theta}}_{\text{observed}}^{(t)}$$

where $\hat{\theta}_{\text{observed}}$ was estimated from each organization's engagement distribution, with a conservative learning rate ($\eta = 0.1$–0.2) to prevent overfitting to behavioral noise in sparse organizations.

This two-stage design—platform defaults as prior, behavioral feedback as likelihood—solved the cold-start problem for the objective function itself, not just for items.

3.4 Operational Design

The recommendation pipeline ran as a **nightly batch process**—a deliberate tradeoff of freshness for cost. Enterprise content velocity is orders of magnitude below consumer social networks; nightly updates provided sufficient freshness. We built administrative controls including a monitoring dashboard, promotional slots for pinning important content, and override rules for excluding deprecated entities.

4 Workstream 3: Q&A Answer Ranking for Customer Forums

4.1 The Noise Problem in Forum Answers

Customer-facing communities functioned as support forums where customers asked technical questions and received answers from other customers, company representatives, and community moderators. The fundamental problem was **answer quality**: threads accumulated dozens of responses, and the best answer was often buried under noise.

Forum responses were not uniformly helpful. Some comments critiqued the question rather than answering it. Some provided outdated information. Some were tangential discussions between respondents. Some were outright wrong. A naive approach—ranking by upvotes or chronological order—failed because:

- Early answers accumulated votes regardless of quality (position bias, identical to the ranking problem in search)
- Comments that critiqued or discussed the question received engagement but were not answers
- Multiple valid answers existed for some questions, and the "best" answer depended on the asker's specific context

4.2 Answer Ranking via Search Retrieval

We framed answer ranking as a **retrieval and ranking problem**: given a question, retrieve and rank candidate answers by their likelihood of resolving the question.

The retrieval stage used the question text to search across all responses in the thread and, critically, across answers from *similar questions in other threads*. This cross-thread retrieval was essential: many questions were semantically equivalent but phrased differently. A question about "error 403 when uploading" and "permission denied on file upload" should surface the same set of answers.

Semantic question matching used TF-IDF similarity with organization-specific vocabulary weighting (reusing the topical vocabulary pipeline from the recommender system). Questions with high semantic similarity were linked, and their answer pools were merged for ranking.

The ranking model scored each candidate answer on:

- **Answer-question relevance**: Semantic similarity between the answer text and the question, weighted by term importance
- **Answer quality signals**: Length, presence of code blocks or structured steps, specificity of language ("click Settings > Permissions" vs. "try changing your settings")
- **Author credibility**: Historical answer acceptance rate, role (company representative vs. community member), tenure
- **Noise suppression features**: Signals that a response was a critique, a clarifying question, or off-topic discussion rather than an answer. We trained a classifier to distinguish answers from non-answers using labeled examples from the human evaluation team
- **Recency-adjusted engagement**: Upvotes and "helpful" marks normalized by exposure time and thread age

4.3 Routing Questions to Relevant People

Beyond ranking existing answers, we built a **question routing system** that identified community members most likely to provide a high-quality answer to a new question. The routing model matched question topics to members' historical answer domains, current activity status, and response rate. This reduced time-to-first-answer and increased the probability that new questions received expert responses rather than languishing unanswered.

5 Workstream 4: Spam and Fraud Detection

5.1 The Forum Pollution Problem

Customer-facing forums attracted spam: promotional posts, phishing attempts, bot-generated content, and low-effort posts that degraded community quality. Spam was not just an annoyance—it directly impacted DAU. Users who encountered spam in their first few forum visits were significantly less likely to return.

5.2 Feature Hashing with Vowpal Wabbit

We deployed a spam detection system using **Vowpal Wabbit** (VW), a fast online learning library written in C. VW's key advantage for this problem was its **feature hashing trick**: rather than maintaining an explicit feature vocabulary (which would grow unboundedly as spammers adapted their language), VW hashed features into a fixed-size vector space. This provided:

- **Constant memory footprint**: The model size was fixed regardless of vocabulary growth, critical for a system processing posts across thousands of communities
- **Online learning**: VW updated the model incrementally as new labeled examples arrived, allowing the spam detector to adapt to evolving spam patterns without full retraining
- **Speed**: VW's C implementation processed posts in microseconds, enabling real-time spam scoring at post-submission time—before the spam was visible to other users

Features included n-grams from post text, URL patterns, author account age and activity patterns, posting frequency, and cross-community behavioral signals (an account posting identical content across multiple communities was a strong spam indicator). The hashing trick allowed us to include high-dimensional features (character-level n-grams, URL token combinations) without explicit feature engineering or vocabulary management.

Detected spam was flagged for moderator review rather than silently deleted, maintaining administrator control and providing labeled data for model updates.

6 Cross-Cutting Infrastructure: Human Evaluation

6.1 Building the Labeling Operation

All four workstreams depended on human evaluation for calibration, validation, and parameter tuning. I recruited a dedicated labeling team through a vendor in India, establishing a **continuous evaluation operation** rather than a one-time labeling exercise.

6.2 The Simulation Environment

For the feed ranker and recommender system, static relevance judgments were insufficient. The quality of a ranked feed depends on the *interaction* between relevance, recency, diversity, and authority—properties that can only be evaluated in the context of a complete feed, not item by item.

We built a **simulation environment** that presented evaluators with complete feed experiences: given a user profile and organizational context, evaluators saw a rendered feed under different parameter configurations and rated the overall quality. This allowed us to tune parameters—recency decay rates, authority weights, content-type-specific thresholds—through human evaluation of the holistic experience.

The simulation was designed for **continuous use**: as we adjusted parameters, new configurations were routed to evaluators for rating. This created a human-in-the-loop optimization cycle:

1. Generate candidate parameter configurations
2. Render simulated feeds under each configuration
3. Evaluators rate feed quality on relevance, freshness, diversity, and organizational appropriateness
4. Select parameters that maximize human-judged quality
5. Deploy and measure behavioral engagement
6. Feed behavioral results back into the next round of parameter candidates

This cycle was critical for the recency–relevance tradeoff in feeds, the target weight calibration in recommendations, and the noise suppression threshold in Q&A ranking.

6.3 Pilot Customer Program

We could not access customer data without explicit agreements. I worked with the legal team to establish **three pilot customer partnerships**—organizations that granted data access in exchange for early access to the ML-powered features. These pilots provided:

- Real organizational data for training and validation (supplementing Salesforce's internal data)
- Diverse community types: one enterprise internal community, one customer support forum, one partner ecosystem—ensuring the system generalized beyond Salesforce's own usage patterns
- Feedback loops from actual administrators and community managers, informing the administrative dashboard and override mechanism design

Securing these partnerships required Director-level engagement with customer success teams and legal review of data sharing agreements—organizational work that was as essential as the technical implementation.

7 Results

The four workstreams were deployed progressively over multiple quarters. The feed relevance system (branded *Salesforce Highlights*) launched first as the highest-impact surface, followed by the recommender system, Q&A ranking, and spam detection.

Metric	**Result**
DAU improvement	Significant increase across deployed communities
MAU improvement	Significant increase across deployed communities
Feed engagement	>20% lift in feed interaction rates
Recommendation engagement	>20% lift over heuristic baseline
Q&A answer quality	Improved time-to-accepted-answer; reduced unanswered rate
Spam detection	Real-time flagging before community exposure

Table 3: Aggregate impact across all four workstreams.

The DAU/MAU improvement was driven primarily by the feed system and recommendations—the two surfaces that affected every user on every visit. Q&A ranking and spam detection had outsized impact on customer-facing communities specifically, where answer quality and forum cleanliness directly determined whether customers returned.

8 Generalizable Principles

1. Feeds are the highest-leverage surface for engagement. Every user sees their feed first. If the feed is irrelevant, the user's first interaction is negative, and they leave. When asked to improve DAU, start with the feed—it touches everyone, every session.

2. Recency and relevance must be balanced through explicit parameters, not implicit features. Treating recency as "just another feature" in an ML model produces unpredictable decay behavior. A multiplicative decay function with content-type-specific half-lives gives explicit control over how long content persists, making the system debuggable and tunable.

3. In enterprise platforms, organizational hierarchy is a first-class signal. A post from a CEO is not the same as a post from a peer, and the ranking system must encode this. Authority features derived from organizational structure—reporting chains, role levels, designated expertise—capture value that behavioral signals alone cannot express.

4. The target function is the product in multi-objective systems. The composite engagement score and its weights determined what the recommender system optimized for. Getting the weights wrong—overweighting likes, underweighting comments—produced popular but shallow recommendations. Human evaluation was essential for calibrating weights that behavioral data alone could not reveal.

5. Train defaults on your own data; adapt from customer feedback. When customer data is inaccessible, the vendor's own deployment provides the prior. Salesforce's internal community data trained the default parameters; per-organization behavioral adaptation provided the likelihood. This two-stage pattern generalizes to any enterprise ML product.

6. Forum Q&A is a retrieval problem with a noise suppression requirement. Not all responses are answers. Comments, critiques, and tangential discussions must be distinguished from genuine answers before ranking. Cross-thread semantic question matching multiplies the effective answer pool. Routing unanswered questions to qualified responders closes the loop.

7. Spam detection requires online adaptation. Spammers evolve. A static classifier trained on historical spam degrades within weeks. Vowpal Wabbit's online learning with feature hashing provided both the adaptability (incremental updates) and the scalability (constant memory, microsecond scoring) required for real-time detection across thousands of communities.

8. Human evaluation is continuous infrastructure, not a one-time exercise. The simulation environment and labeling team were not project expenses—they were platform infrastructure.

Every parameter change, every new feature, every recency threshold adjustment was validated through human evaluation before deployment. The cost of the labeling operation was a fraction of the cost of deploying a poorly tuned system to millions of users.

9. Improving engagement requires multiple coordinated interventions. No single system—not feeds alone, not recommendations alone—would have produced the DAU/MAU improvement. The combination of relevant feeds (driving return visits), personalized recommendations (driving discovery), quality Q&A (driving customer retention), and spam suppression (protecting community quality) created a reinforcing cycle. The Director-level contribution was identifying all four levers and executing them as a coordinated program rather than isolated projects.