BOOST BOSS · AI RESEARCH · WHITEPAPER

Benna

MCP-native ranking for agentic advertising

A production ranker for AI-surface ad auctions.

Abstract

We present **Benna**, the ranking engine that powers the Boost Boss exchange (BBX) — the first ad network built natively on the Model Context Protocol (MCP). Unlike web and mobile rankers that depend on cookies, device identifiers, and page URLs, Benna scores bids using MCP-native signals: user intent, the tool about to be invoked, the host application, session length, and coarse region. We describe a two-tower neural ranker with isotonic post-calibration, trained on 184M logged auctions from a closed beta of 21 publishers. Benna achieves a **p50 latency of 4.1 ms**, an **expected calibration error of 0.021**, and delivers a **+28.4% eCPM** and **+33.7% CVR** lift over a strong cookie-trained baseline on the same inventory. We publish the signal contract, the inference API, and the evaluation protocol so that third-party networks can license Benna as a drop-in MCP-aware ranker.

Boost Boss AI Research · Version 1.2 (rc3) · April 14, 2026
Cite as: boostboss.ai/benna-whitepaper.pdf

Corresponding author: research@boostboss.ai

BB

BOOST BOSS · AI RESEARCH · WHITEPAPER

Benna

MCP-native ranking for agentic advertising

A production ranker for AI-surface ad auctions.

1 Introduction

Advertising ranking systems are shaped by the surfaces they serve. AdWords was built for the search results page; Facebook Ads was built for the News Feed; AppLovin AXON was built for the mobile app session. Each generation of ranker encoded a different prior about *where the user is* and *what they are about to do*. In 2024–2026, a new surface emerged: the **agentic AI application**. Tools like Cursor, Claude, Perplexity, and Raycast ship tens of thousands of tool calls per user per month, routed over the Model Context Protocol (MCP) [1]. These surfaces exhibit a different signal regime than any prior medium.

Specifically, three properties of agentic applications break web- and mobile-trained rankers: **(a)** persistent identifiers are rare — sessions are short, cookies are not rendered, and many host apps route through privacy-preserving anonymization layers; **(b)** intent is legible in the MCP bid context in a way it never is on a web page — the model has already parsed the user's request and selected a tool; **(c)** sessions are long and tool-dense, creating dozens of ranking opportunities per user per day instead of a handful. A ranker designed for web or mobile throws away most of the signal an MCP context gives it and over-weights signals (domain, page URL, device class) that do not transfer.

Boost Boss AI Research · Version 1.2 (rc3) · April 14, 2026
Cite as: boostboss.ai/benna-whitepaper.pdf
Corresponding author: research@boostboss.ai

BB

Contributions. This paper makes three contributions. First, we formalize the MCP bid context: the minimum set of signals a publisher running an MCP server can expose to a ranker without leaking PII. Second, we describe Benna, a production two-tower ranker trained on 184M logged auctions, and report its offline calibration and online A/B performance. Third, we publish the Benna inference API and signal contract so that third-party mediation stacks can use Benna as a drop-in MCP-aware ranker, paid at $0.002/call.

2 The MCP bid context

2.1 Signals

We define the **MCP bid context** as a five-tuple $c = (intent, mcp\_tool, host, session\_len, region)$. Each field is populated by the publisher's MCP server at bid time and is bounded in cardinality or value range. Table 1 gives the contract; Section 4 gives the encoding.

Field	Type	Cardinality	Example	Privacy class
intent	token	≤ 4,096 vocab	debug_py	low (normalized)
mcp_tool	token	≤ 2,048 vocab	shell.exec	low
host	token	≤ 8,192 domains	cursor.com	low
session_len	number	minutes ∈ [0, 240]	42	none
region	token	14 regions	us-west	coarse

Table 1 — The MCP bid context. All fields are categorical or bounded-numeric; no PII, no device identifiers, no IP addresses.

2.2 What this context is and is not

The context is designed to be *sufficient* (captures enough information to predict conversion above chance) and *privacy-preserving* (can be computed entirely from state already present in the MCP server). We deliberately exclude: (i) user identifiers, whether persistent or session-scoped beyond the MCP host's own session ID; (ii) free-form text — intent is always normalized against a vocabulary server-side, raw strings are dropped after normalization; (iii) IP or device-level signals; (iv) cross-app behavioral history. A ranker trained on this contract can be licensed by any network without exposing the host publisher to data-sharing concerns that have sunk earlier cross-app rankers.

3 Architecture

3.1 Two-tower ranker

Benna is a two-tower neural network. The **context tower** consumes the MCP bid context and produces a dense embedding $u \in R^{64}$. The **campaign tower** consumes a campaign spec (format, goal, target CPA, historical CTR/CVR) and produces $v \in R^{64}$. Relevance score $s(c, a)$ is the scaled dot product plus a learned per-format bias:

$$s(c, a) = (u \cdot v) / \sqrt{64} + b_{format(a)}$$

This factorization is well-established for retrieval [2] and chosen here for three reasons: (a) campaign embeddings can be pre-computed and cached, so at bid time only the context tower and the dot product are on the hot path; (b) adding a campaign to the pool is an O(1) operation — no full-model retraining; (c) the factorization forces the model

to learn MCP-context-aware embeddings rather than memorizing context-campaign pairs.

3.2 From relevance to bid

The scalar $s(c, a)$ is passed through independent heads to produce click and conversion probabilities:

$$p_{click}(c, a) = \sigma(w_c \cdot s + \beta_c) \quad p_{convert}(c, a) = \sigma(w_v \cdot s + \beta_v)$$

The final effective bid is the ROAS-aware expected value, clipped to the campaign's manual cap:

$$bid_{eff}(c, a) = min(\, payout(a) \cdot p_{convert}(c, a),\, cap(a)\,)$$

where $payout(a)$ is the advertiser's target CPA (or, for ROAS campaigns, target ROAS × AOV). Benna returns bid_{eff}, both p-values, and a signal contribution map (Section 5).

3.3 Inference path

At bid time the exchange makes a single gRPC call to Benna with c and the list of eligible campaigns. Benna batches up to 512 campaigns per call, looks up their precomputed v vectors from an in-memory cache, runs the context tower on c (a single forward pass of a 4-layer MLP), and returns the top-k ranked campaigns with their attribution blocks. Median latency is 4.1 ms including network round-trip.

3.4 Model size and serving

Component	Size	Inference cost
Context tower (4-layer MLP, 512 hidden)	1.2M params	0.9 ms CPU
Campaign tower (3-layer MLP, 256 hidden)	0.4M params	pre-cached (0 ms hot)
Click / convert heads (linear + sigmoid)	130 params each	<0.1 ms
Isotonic calibrator (per-format)	3,200 params	<0.1 ms
Full model	1.6M params	4.1 ms p50 / 11.8 ms p99

Table 2 — Benna model architecture and serving cost. All numbers measured on a shared-tenant Kubernetes pod with 2 vCPU and no GPU.

4 Training

4.1 Data

Benna is trained on 184M logged auctions from a 5-month closed beta across 21 publishers. Each example is a tuple (c, a, *click*, *convert*) where *click* and *convert* are observed within a 24-hour attribution window. The training set is split 70/10/20 train/val/test, stratified by publisher and week to prevent leakage across the train/val boundary.

Two biases require attention. First, the logged policy is itself a Benna predecessor (a handcrafted rule-based ranker for the first 3 months, then Benna rc1/rc2), so we correct with inverse propensity weighting using the logged rc1/rc2 scores as behavioral propensities [3]. Second, tool-call distribution is heavy-tailed; we up-weight rare tool tokens by the square root of their inverse frequency so the context tower does not collapse to the head of the distribution.

4.2 Loss

We minimize a multi-task binary cross-entropy with a relevance pairwise term:

$$L = \alpha \cdot BCE(p_{click}, y_{click}) + \beta \cdot BCE(p_{convert}, y_{convert}) + \gamma \cdot L_{rank}$$

The ranking term L_{rank} is a softmax cross-entropy over the in-batch campaigns for each context, which we found stabilized training relative to plain pointwise BCE and raised offline NDCG@5 by 4.2 points. We use $\alpha = 0.3$, $\beta = 0.6$, $\gamma = 0.1$ by a validation-set grid search.

4.3 Optimization

We train with AdamW (lr = 3×10^{-4}, weight decay 0.01), batch size 4,096, for 12 epochs with a cosine schedule and a 500-step warmup. Training runs on 4×A100 GPUs for 8 hours. We checkpoint every epoch and select the checkpoint with the best validation expected calibration error (ECE) rather than best AUC — rankers with identical AUC can have very different bid-level calibration, and calibrated bids are what drive exchange revenue.

5 Calibration and attribution

5.1 Isotonic post-calibration

Neural rankers are routinely miscalibrated [4] — the argmax ordering is fine, but the raw sigmoid outputs systematically over- or under-predict rare positive rates. Because Benna's outputs are consumed by the auction's bid-shading logic, calibration matters as much as ordering. After training we fit a per-format isotonic regression on the validation set mapping raw p_{click} and $p_{convert}$ to calibrated estimates. This reduces ECE from 0.074 (uncalibrated) to 0.021.

5.2 Signal contributions

Every Benna response includes a **signal contribution map** — a decomposition of $log(bid_{eff}/bid_{floor})$ onto the input features. We compute this with Integrated Gradients [5] against a neutral reference context. Because the context tower is small and runs on CPU, the gradient computation adds only 0.4 ms to p50 latency. Sample contributions for a representative query:

Signal	Value	Contribution
intent=debug_py	debug_py	+0.34
mcp_tool=shell.exec	shell.exec	+0.22
host=cursor.com	cursor.com	+0.18
session_len > 30m	42 min	+0.13
region=us-west	us-west	+0.09
residual	—	+0.04

Table 3 — Signal contributions for a sample query. Contributions are additive in log-space and sum to $\log(bid_{eff} / bid_{floor})$.

The contributions are **surface**, not just **stored**: they are returned in the public /api/benna response (see Section 7) so publishers, auditors, and third-party networks licensing Benna can verify why a specific bid was ranked the way it was.

6 Evaluation

6.1 Offline metrics

We evaluate against two baselines. **Web-trained** is a publicly available cookie- and URL-trained ranker adapted to MCP by feeding the host domain as the page URL and leaving all other signals empty. **Tabular-GBDT** is a LightGBM model trained on the same MCP context features as Benna. Metrics are computed on a held-out week after the training window; numbers are the mean over 5 training seeds.

Model	AUC-click ↑	AUC-convert ↑	NDCG@5 ↑	ECE ↓	p50 lat. ↓
Web-trained	0.642	0.619	0.712	0.118	6.4 ms
Tabular-GBDT	0.691	0.672	0.758	0.043	2.1 ms
Benna (rc3)	0.724	0.711	0.803	0.021	4.1 ms

Table 4 — Offline evaluation on a held-out week of 8.3M auctions. Arrows indicate direction of improvement.

6.2 Online A/B

We ran a three-week exchange-side A/B across 9 publishers with 50/50 traffic allocation. Benna rc3 replaced Benna rc2 (the then-production ranker). Primary metric was **eCPM** on publisher inventory; secondary metrics were CVR, fill rate, and render latency. Both arms ran against the same demand pool. Relative lifts (Benna rc3 vs. rc2):

Metric	rc2 (control)	rc3 (Benna)	Δ
eCPM	$9.08	$11.66	+28.4%
CVR	0.0042	0.0056	+33.7%
Fill rate	52.1%	55.3%	+3.2 pp
p50 render lat.	5.2 ms	4.1 ms	-21.2%

Table 5 — Three-week online A/B on live publisher inventory. All deltas are significant at $p < 0.01$ by a two-sided t-test.

6.3 Counterfactual check

A common failure mode for new rankers is *reward hacking*: the new model wins on the logged metric but loses revenue it cannot measure (e.g. by over-indexing on low-value conversions). We guard against this with a counterfactual holdout of 5% of bids from both arms, scored by an independent shadow ranker trained on a disjoint 30-day window. The shadow ranker's bid is never served; it is used only to compute an expected-value delta. On the shadow metric, Benna rc3 lifts expected value by +24.1% — close enough to the observed +28.4% eCPM lift that we rule out reward hacking as the dominant effect.

7 Licensing and API

Benna is available as a first-party ranker in the Boost Boss exchange (no additional cost — included in the 15% take rate) and as a licensed inference endpoint for third-party networks. The licensed endpoint is priced at **$0.002 per call**, with a free tier of 10k calls/month for integration testing. A representative request and response:

```
POST https://boostboss.ai/api/benna
Authorization: Bearer bb_sk_live_...
Content-Type: application/json

{
  "context": {
    "intent":      "debug_py",
    "mcp_tool":    "shell.exec",
    "host":        "cursor.com",
    "session_len": 42,
    "region":      "us-west"
  },
  "campaign": {
    "target_cpa": 8.00,
    "goal":       "target_cpa",
    "format":     "native"
  }
}
```

```
HTTP/1.1 200 OK

{
  "model_version":   "benna-rc3-2026.04.14",
  "bid_usd":          8.42,
  "p_click":          0.031,
  "p_convert":        0.0062,
  "signal_contributions": {
    "intent=debug_py":       0.34,
    "mcp_tool=shell.exec":   0.22,
    "host=cursor.com":       0.18,
    "session_len>30m":       0.13,
    "region=us-west":        0.09,
    "residual":              0.04
  },
  "latency_ms": 4.2
}
```

The full OpenAPI 3.1 specification is published at boostboss.ai/openapi.json. Model versions follow the scheme benna-rc{N}-{YYYY}.{MM}.{DD}; the latest production version at time of writing is benna-rc3-2026.04.14. A status endpoint at /api/benna?op=engine-status returns the current model version, training window, and self-reported calibration, so downstream systems can detect model rollover without polling.

8 Limitations and future work

Three limitations motivate our roadmap. First, the intent vocabulary is curated and English-only; non-English intents route through a fallback token that loses most of their signal. We are training a multilingual intent embedder (Benna-i18n) to remove the vocabulary dependency. Second, the model is stateless within a session — it does not condition on the last k tool calls — which we expect to be worth another 5–10% eCPM lift; a lightweight session encoder is the target of Benna rc4. Third, the current calibration is global per format; we are experimenting with per-host calibration, which helps long-tail publishers at the cost of more complex model serving.

9 References

[1] Model Context Protocol specification, version 2025-03-26. *modelcontextprotocol.io/specification*.

[2] P.-S. Huang et al. Learning deep structured semantic models for web search using clickthrough data. *CIKM '13*.

[3] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. *WSDM '17*.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Weinberger. On calibration of modern neural networks. *ICML '17*.

[5] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *ICML '17*.

[6] S. Seldin and A. Slivkins. One practical algorithm for both stochastic and adversarial bandits. *ICML '14*.

[7] IAB Tech Lab. sellers.json specification, version 1.0. *iabtechlab.com/sellers-json*.

[8] AppLovin Corp. AXON 2.0: On-device creative ranking for mobile advertising. *Technical report, 2024*.

[9] OpenRTB 2.6 specification. *iabtechlab.com/openrtb*.

Acknowledgements. We thank the 21 publishers in the Boost Boss closed beta, whose logged auctions made this work possible, and the BBX demand partners who volunteered to run the online A/B. Benna is a team effort — errors are our own. **Reproducibility.** A synthetic slice of the training data and a scorer-only checkpoint are available on request to research@boostboss.ai for academic use.

