Pay-Per-Crawl Pricing for AI: The LM-Tree Agent

Richard Archer* Soheil Ghili† Nima Haghpanah‡

March 2026

Abstract

As AI systems shift from directing users to content toward consuming it directly, publishers need a new revenue model: charging AI crawlers for content access. This model, called pay-per-crawl, must solve a problem of *mechanism selection at scale*: content is too heterogeneous for a fixed pricing framework. Different sub-types warrant not only different price levels but different pricing rules based on different unstructured features, and there are too many to enumerate or design by hand. We propose the LM Tree, an adaptive pricing agent that grows a segmentation tree over the content library, using LLMs to discover what distinguishes high-value from low-value items and apply those attributes at scale, from binary purchase feedback alone. We evaluate the LM Tree on real content from a major German technology publisher, using 8,939 articles and 80,451 buyer queries with willingness-to-pay calibrated from actual AI crawler traffic. The LM Tree achieves a 65% revenue gain over a single static price and a 47% gain over two-category pricing, outperforming even the publisher's own 8-segment editorial taxonomy by 40%—recovering content distinctions the publisher's own categories miss.

*Yale School of Management.
†Yale School of Management.
‡Yale School of Management.

1 Introduction

During the search era, the business model for online content was built on direct traffic. Users found content through search engines and accessed it directly; publishers monetized via advertising, subscriptions, or commerce. This model is eroding in the AI era. AI systems now crawl content for training and for retrieval-augmented generation, consuming publisher output without directing users to the source. The traffic-based revenue model breaks down.

For content creation to remain viable, publishers need a new revenue model: charging AI systems directly for content access. This is called **pay-per-crawl** (PPC). Services like Cloudflare and Tollbit now provide the infrastructure to charge bots. What remains unsolved is the pricing problem: how should a publisher determine what to charge?

Optimal pricing for PPC is hard for two reasons that compound each other. First, pricing-relevant features are unstructured: a content item's value to an AI crawler depends on features embedded in the text itself—topic specificity, data richness, timeliness—not on structured metadata. Pricing a piece of content requires reading and understanding it, not checking a category label. Second, the space of content types is massive and hierarchical: different content sub-types warrant different pricing rules based on different features, and the problem of deciding which features matter for which content must be solved at scale. Manual rule design is infeasible. Crucially, the relevant features differ not just within a publisher's catalog but across publisher types: for a financial news outlet, recency by the second determines value; for a legal database, the applicable jurisdiction is the key dimension; for a technology review site, the tier of the product under review drives willingness-to-pay. There is no universal pricing rule—each content type or subtype warrants its own discovered mechanism.

Together, these challenges constitute a problem of *mechanism selection at scale*: a pricing methodology must discover–for any given publisher–which segments warrant distinct pricing, which textual features define those segments, and what prices they support—all from binary purchase feedback alone. The **LM Tree** is designed to solve it—an adaptive pricing agent that combines tree-based market segmentation with LLM-powered feature discovery. The agent starts with coarse content categories, explores prices, and observes only binary purchase outcomes. It deploys two specialized components: an **LLM Analyst** that reads content text and discovers what distinguishes high-value items from low-value ones—not from metadata, but from the prose itself—and an **LLM Annotator** that applies the discovered attributes to all items in the node. These attributes define split rules that partition content into finer sub-types, each priced independently. The process recurses, growing a pricing tree that matches the natural heterogeneity of the content. At inference time, no LLM is needed—split

rules reduce to simple attribute lookups.

We evaluate the LM Tree on real content from HardwareLuxx (HWL), a major German technology publisher. The dataset contains 8,939 articles spanning two observable categories—long-form reviews (*artikel*) and news items (*news*)—drawn from a corpus crawled extensively by AI systems including GPTBot, ClaudeBot, and PetalBot. Willingness-to-pay is calibrated from actual AI crawler traffic: $\text{WTP}(i) = 0.004 \times \text{observed crawler views}(i)$, reflecting the principle that crawlers' value for content scales with how actively they consume it in the current unpriced market. We generate 9 synthetic buyer queries per article (80,451 total) to simulate transaction-level demand. Articles are split into a training set (7,210 articles) on which the agent learns prices, and a test set (1,729 articles) on which learned prices are evaluated without further adjustment.

We train and compare a number of pricing policies. Each policy explores prices as our simulated queries "arrive," receives a binary purchase feedback from each query, and optimizes the pricing of different content items over time. Importantly, the binary purchase feedback is the only information that each policy receives; the underlying calibrated WTP which determines the purchase decision is hidden from the agent.

Specifically, we have four pricing policies. Single-price optimization—one price for all 8,939 articles—yields \$160 in test-set revenue. Format category pricing—one price per content format, distinguishing long-form reviews (*artikel*) from short news items (*news*)—raises this to \$179, a modest gain reflecting that this format distinction captures little of the within-format WTP heterogeneity. Moving to the publisher's own 8-segment editorial taxonomy (hardware, software, consumer electronics, and others within each format) reaches only \$189—an 18% gain over single pricing. The LM Tree, initialized with only the two format categories and discovering finer content segments from article text alone, achieves \$264—a 65% gain over single-price, a 47% gain over format category pricing, and a 40% gain over the richer 8-segment baseline.

The key finding concerns how the LM Tree structures its splits. The discovered rules cut across the publisher's formal editorial categories: an article covering high-end GPU specifications belongs to a different pricing regime than other hardware articles—a distinction the editorial taxonomy does not capture. The agent recovers this segmentation from binary purchase feedback alone, with no prior knowledge of the content hierarchy.

We expect pay-per-crawl to be the first but not the last market in which agentic pricing based on unstructured features becomes the dominant method. As AI systems intermediate more transactions—from API access to data licensing to professional services—the same core problem recurs: heterogeneous goods, unobservable willingness-to-pay, and pricing-relevant features buried in unstructured text. Pay-per-crawl provides an attractive setting in which

to study the design of pricing agents supported by language models precisely because it combines all of these challenges in a market that is forming now.

The remainder of the paper proceeds as follows. Section 2 reviews related literature. Section 3 formalizes the setting: the publisher's content library, the buyer model, and the pricing problem. Section 4 presents the LM Tree—its algorithmic structure, the roles of the LLM Analyst and Annotator, and the properties of its splitting rules. Section 5 describes the HardwareLuxx dataset and evaluation setup. Section 7 reports results, including a decomposition of the revenue gains and an analysis of the splits the LM Tree makes. Section 8 discusses broader applicability and open questions. Section 9 concludes.

2 Related Literature

Bandit pricing and demand learning. The price exploration step at each node is a multi-armed bandit problem. A substantial literature characterizes the cost of learning demand from binary purchase feedback: Kleinberg and Leighton (2003) establish regret bounds for online posted-price auctions; Besbes and Zeevi (2009) derive near-optimal algorithms for dynamic pricing without prior knowledge of the demand function; den surveys the subsequent development of these ideas across pricing contexts. In marketing, Misra et al. (2019) demonstrate the practical value of bandit experiments for dynamic online pricing in field settings, showing that adaptive price exploration yields substantial revenue gains over passive approaches. The LM Tree applies this machinery locally—one bandit problem per node—but its primary contribution is not to the price-learning step. What it adds is a principled method for deciding *which* bandit problems to run: the tree structure determines which items should be priced together, and the LLM Analyst determines when and how to split. Standard bandit pricing takes the set of items as given; the LM Tree discovers the right partition from data.

Tree-based market segmentation. Recursive partitioning methods have a long history in statistics (Breiman et al., 1984) and have been adapted to a range of causal and economic objectives. Athey and Imbens (2016) introduce causal trees, which tailor split criteria to maximize treatment effect heterogeneity rather than prediction accuracy—establishing the principle that tree structure should be designed around the quantity of interest. Wager and Athey (2018) extend this to forests for valid nonparametric inference. aou bring the framework explicitly to pricing, developing market segmentation trees that split on features to maximize revenue differentiation across leaves. The LM Tree builds on this lineage and follows it closely in structure: the splitting criterion targets price differentiation, split validation uses economic rather than statistical criteria, and leaf prices are estimated from fresh exploration data. The

departure is upstream: in all existing tree methods, the feature space is fixed and given—the algorithm's task is feature *selection*, searching over columns of X. The LM Tree adds a prior step of feature *construction*, in which the LLM Analyst generates a node-specific feature representation from content text. This is what allows the tree to operate in settings where no feature matrix exists.

Text as data and LLM-based feature extraction. A growing literature in economics uses text to measure quantities that are difficult to observe directly—Gentzkow et al. (2019) survey applications ranging from political polarization to asset pricing. More recent work uses large language models for structured information extraction: given unstructured prose, an LLM can reliably extract attributes, labels, and relational facts that would otherwise require costly manual annotation. The LM Tree embeds this capability inside an adaptive economic loop. The LLM is not merely measuring a fixed target—it is discovering, at each node, which attributes are worth measuring, and the tree structure determines which discovery problems to pose. The interaction between economic feedback (H and L sets) and language model reasoning is the core mechanism; neither is useful without the other.

Economics of AI and digital content markets. The pay-per-crawl setting connects to a broader literature on the economics of data and AI. ber study how data generates value in markets with asymmetric information; Agarwal et al. (2018) provide a framework for thinking about AI as a prediction technology with implications for market structure. More directly, a nascent literature examines the economic relationship between AI developers and content producers—who owns training data, how scraping affects incentives to create, and what compensation mechanisms are feasible. The LM Tree takes the infrastructure of pay-per-crawl as given and asks how a publisher should price within it—a question this literature has not yet addressed formally.

Price discrimination and product design. At the economic level, the LM Tree is performing second-degree price discrimination: it partitions a heterogeneous product space into segments and prices each separately. The classical treatments—Mussa and Rosen (1978) on monopoly and product quality, and Maskin and Riley (1984) on monopoly with incomplete information—assume the seller knows which product dimensions matter and designs quality schedules accordingly; Ghili et al. (2025) combine theory, experimental design, and empirical estimation to study second-degree price discrimination in a unified framework. Bergemann et al. (2015) characterize the full set of consumer surplus and profit outcomes achievable under any market segmentation, establishing the theoretical boundaries of price discrimination as an information design problem. Haghpanah and Siegel (2022) study the limits of multiproduct price discrimination; Haghpanah and Siegel (2023) identify conditions under which segmentation Pareto-improves market outcomes. On bundling—an

extension of the second-degree framework to settings where the relevant quality dimensions interact—Ghili (2023) characterizes optimal bundling with nonadditive values; Haghpanah and Hartline (2021) characterize when pure bundling is optimal; and Yang (2025) studies nested bundling, showing that hierarchical bundling structures can replicate complex nonlinear pricing schemes. Shapiro and Varian (1999) establish versioning as the canonical digital-goods strategy. Dubé and Misra (2023) study personalized pricing—third-degree discrimination based on buyer identity—implemented at scale via machine learning, demonstrating large profit gains from buyer-level price targeting. The LM Tree as presented conditions prices on content characteristics rather than buyer identity, but extends naturally to a combination of second- and third-degree discrimination once buyer identity is observable; we discuss this in Section 8. All of this literature assumes the seller knows which product dimensions matter. The LM Tree relaxes this assumption: it discovers the relevant dimensions from data, making price discrimination applicable when the product space is too complex, heterogeneous, or novel to enumerate in advance.

3 Setting

A publisher operates a content library consisting of items $i \in \mathcal{I}$. Each item belongs to an observable coarse category $c(i) \in \mathcal{C}$ (e.g., hardware reviews, software news, consumer electronics coverage) and has associated text t_i—the prose body of the content. The text is the publisher's primary asset: it is what AI crawlers seek to access.

Buyers are AI crawlers operated by AI companies. Each buyer arrival q targets a content category and is characterized by a buyer type s_q (e.g., a large foundation model trainer vs. a retrieval-augmented generation service). Upon arrival, the buyer requests a content item i from the publisher, who offers access at price p.

3.1 Willingness-to-Pay

Each buyer has a willingness-to-pay $v(i, s_q)$ that depends on the content item and the buyer type. What makes the pricing problem difficult is the structure of v: it is not a simple function of a few universal features, but a deeply heterogeneous object whose relevant dimensions change across content types.

Consider a publisher whose library spans several broad categories—say, GPU and processor benchmark reviews, software and games coverage, and consumer electronics articles. Within GPU benchmarks, the value of a piece of content to an AI training pipeline depends on rasterization throughput and thermal performance under sustained load. Within software

and games coverage, these metrics are irrelevant; value depends on benchmark scores, frame rates, and driver compatibility specifics. Within consumer electronics, value depends on hands-on testing depth and product availability at time of publication.

These are not different values of the same attributes—they are entirely different attributes, measured in different units, relevant to different buyer types. A pricing rule that works for GPU benchmarks (e.g., "charge more when silicon generation is current") is meaningless for software news, where silicon generation is not a concept. Conversely, "charge more for benchmark score coverage" is meaningless for consumer electronics reviews.

The heterogeneity compounds as one looks deeper. Within GPU benchmarks, there are sub-types: flagship performance reviews of the latest silicon, budget-tier value comparisons, and legacy architecture retrospectives. Each commands different prices from different crawler types: a model trainer values cutting-edge silicon specs; a product recommendation engine values budget comparisons; a historical analysis pipeline values legacy coverage. The relevant pricing dimensions shift again at each level of granularity.

The result is that v is not one function but a large family of functions, each defined over a different feature space, applying to a different slice of the content library. The number of distinct pricing rules one would need to write—each specifying which attributes matter, how they enter v, and what price they justify—grows combinatorially with the depth of the content taxonomy. For a publisher with thousands of content items across dozens of sub-types, manually specifying these rules is infeasible.

3.2 Transactions

When the publisher offers item i to buyer q at price p, the buyer purchases if and only if the price is below their willingness-to-pay:

$$y = \mathbf{1}[p \leq v(i, s_q)]$$

The publisher observes the binary outcome y but never the value $v(i, s_q)$, the buyer type s_q, or which features of the content drove the buyer's valuation. The publisher does observe its own content—the text t_i and category $c(i)$—and the history of prices offered and outcomes received.

3.3 The Publisher's Problem

The publisher chooses a pricing policy $\pi : \mathcal{I} \to \mathbb{R}_+$ mapping content items to prices. The objective is to maximize expected revenue:

$$\max_{\pi} \mathbb{E}\left[\sum_{q} p_\pi(i_q) \cdot \mathbf{1}[p_\pi(i_q) \leq v(i_q, s_q)]\right]$$

where i_q is the item matched to query q and $p_\pi(i_q)$ is the price assigned by policy π.

In principle, this is a standard dynamic pricing problem: explore prices, estimate demand, and converge to revenue-maximizing prices. But the structure of v described above transforms it into something harder. Because the value-relevant features differ across content types—and because the publisher does not know *which* features matter for *which* content—the publisher cannot simply estimate a demand curve. It must first discover the right segmentation: which items should be priced together, and on what basis they should be differentiated.

This is a problem of **mechanism selection at scale**. The publisher is not choosing one pricing rule; it is choosing a *collection* of pricing rules, one for each content segment, where the segments themselves must be discovered. For each segment, the publisher must identify the relevant features, learn how those features relate to willingness-to-pay, and set prices accordingly—all from binary purchase feedback alone. The combinatorial structure of v means that the space of possible segmentations is vast, and the cost of exploring the wrong segmentation is real revenue forgone.

The pricing agent's task, then, is to navigate this space efficiently: discover which content segments warrant distinct pricing, identify the textual features that define those segments, and learn per-segment prices—using only the content text t_i and the history of purchase outcomes as inputs.

Instantiation on HardwareLuxx. We implement and evaluate this problem on data from HardwareLuxx (HWL), a major German technology publisher. The content library consists of 8,939 articles in two observable categories—long-form reviews (*artikel*) and news items (*news*). Willingness-to-pay is calibrated from real AI crawler traffic as $v(i) = 0.004 \times \text{observed views}(i)$, reflecting the assumption that crawlers' current consumption reveals their relative valuations. The publisher observes each article's text and its coarse category, but never the crawler view counts. Articles are split into a training set (7,210) on which the agent learns, and a test set (1,729) on which learned prices are evaluated without further adjustment.

4 The LM Tree

This section develops the LM Tree, a pricing agent that solves the mechanism selection at scale problem defined in Section 3. The agent must simultaneously discover content segments, identify the textual features that define them, and learn per-segment prices—all from binary purchase feedback.

4.1 Tree-Based Pricing Policies

We begin with the standard framework for tree-based partitioning (Breiman et al., 1984). A pricing tree $\mathcal{T}$ is a binary tree that induces a partition $\Pi = \{\mathcal{I}_1, \ldots, \mathcal{I}_K\}$ of the content library $\mathcal{I}$ into K segments, with each segment k assigned a price p_k. The pricing policy maps each item to the price of its segment:

$$\pi_{\mathcal{T}}(i) = p_{\ell(i)}$$

where $\ell(i)$ is the leaf segment containing item i.

In a standard decision tree, each internal node n holds a split rule σ_n defined over a pre-specified feature vector $x_i \in \mathbb{R}^d$: the algorithm searches over all features $j \in \{1, \ldots, d\}$ and all thresholds τ to find the split (j, τ) that optimizes a criterion—Gini impurity for classification (Breiman et al., 1984), treatment effect heterogeneity for causal inference (Athey and Imbens, 2016), or response model fit for market segmentation (aou).

The publisher's problem differs from all of these in a fundamental way: Content is unstructured, hencethe feature vector x_i does not exist. The publisher has content text t_i and a coarse category label $c(i)$, but no structured feature matrix. The attributes that matter for pricing—benchmark depth, jurisdictional coverage, signal freshness—are embedded in the prose, vary by content type, and are not known in advance. There is no column to split on until someone reads the text and decides what to extract.

The LM Tree addresses this by adding a **feature construction** step before the standard split-and-estimate logic. At each node, a large language model reads the content text, discovers pricing-relevant attributes, and constructs a local feature representation. The tree then splits on these constructed features using standard logic. The key departure from the tree literature is that the feature space is not given—it is generated, and it is different at every node.

4.2 Algorithm

The LM Tree grows by alternating between three operations at each node: **price exploration**, **feature discovery**, and **split validation**. We describe each in turn.

Initialization. The tree begins with $|\mathcal{C}|$ root nodes, one per coarse category. Each root contains all items in its category: $\mathcal{I}_n = \{i : c(i) = c\}$ for $c \in \mathcal{C}$. In the HardwareLuxx evaluation, $|\mathcal{C}| = 2$, corresponding to long-form reviews (*artikel*) and news items (*news*).

Price exploration. At node n, the agent explores K price arms $\{p_1, \ldots, p_K\}$ log-spaced around a baseline. For each arm k, it runs M trials: a query q arrives, an item $i \in \mathcal{I}_n$ is offered at p_k, and the agent observes $y = \mathbf{1}[p_k \leq v(i, s_q)]$. The agent estimates per-arm conversion rates and revenue:

$$\hat{r}_k = \frac{1}{M} \sum_{m=1}^{M} y_{k,m}, \qquad \hat{R}_k = p_k \cdot \hat{r}_k$$

where $\hat{r}_k$ is the empirical conversion rate (fraction of trials resulting in a purchase) and $\hat{R}_k$ is the estimated revenue per query. The node's current optimal price is $p_n^* = \arg\max_k \hat{R}_k$.

This is the standard multi-armed bandit approach to price discovery (Kleinberg and Leighton, 2003). Used alone, it yields a single price per category—the Category Pricing benchmark in our results. The LM Tree's contribution begins with what happens next.

Feature discovery. The goal is to identify which textual features of content drive high versus low willingness-to-pay. Two challenges make this hard. First, content is unstructured: pricing-relevant features are not in a column of a table but embedded in prose, and are not known in advance. Second, WTP is never directly observed—the agent has only binary purchase outcomes under different prices, and must infer latent value from this indirect signal.

The law of demand structures what can be inferred. High-value items sell at both high and low prices; low-value items sell only at low prices. As a result, the set of items that sell at high prices is a *subset* of the set that sells at low prices—not a disjoint group. The agent's task is to find the hidden textual features that determine whether an item belongs to this high-value subset, using only the content text and the pattern of purchase outcomes under price variation.

To construct contrast sets that expose this structure, the agent partitions the K price arms into a top half and a bottom half by rank:

- H_n: items that purchased at a *top-half* price arm—revealed willingness to pay at high prices

- L_n: items that purchased at a *bottom-half* price arm[1] —revealed willingness to pay only at low prices

In a standard tree, the algorithm would search over columns of a feature matrix X to find the split that best separates H_n from L_n. The LM Tree replaces this search with a call to the **LLM Analyst**—an LLM whose role is to reason over content text and surface pricing-relevant structure.

The LLM Analyst is presented with a sample of items from H_n and L_n—specifically, their content text t_i—and asked: *what textual attributes distinguish items that sell at high prices from those that sell only at low prices?* It reads the prose and returns a set of candidate attributes $\{a_1, \ldots, a_J\}$.

This is the step that has no analogue in the existing tree literature. The LLM Analyst is performing feature construction: it examines unstructured text and proposes a structured representation that the tree can split on. The attributes it discovers are specific to the node—a different node, covering different content, will engage the LLM Analyst anew and may yield entirely different attributes.

Split rules. The discovered attributes define a split rule $\sigma_n : \mathcal{I}_n \to \{L, R\}$. The LM Tree supports two types, tried in order of preference.

Existence rules. The LLM identifies an attribute a that is *mentioned* in items from H_n but *absent* from items in L_n (or vice versa). The split rule is:

$$\sigma_n(i) = \begin{cases} R & \text{if attribute } a \text{ is mentioned in } t_i \\ L & \text{otherwise} \end{cases}$$

For example, the LLM might discover that high-value technology reviews mention "engagement lift" while low-value ones describe "data points per decision." The split is not on the *value* of engagement lift but on whether the concept appears at all.

Threshold rules (fallback). If both groups mention the same attribute but at different magnitudes, the LLM proposes a numeric threshold: $\sigma_n(i) = \mathbf{1}[a(i) > \tau]$.

Existence rules are preferred because they are robust to the heterogeneity described in Section 3: different content sub-types discuss fundamentally different *kinds* of metrics, making presence/absence a stronger signal than numeric comparison across incommensurable scales.

Annotation. Once the LLM Analyst has identified the relevant attributes, a second component—the **LLM Annotator**—applies them to *all* items in $\mathcal{I}_n$, not just the H_n and

[1] If K is odd, the middle arm is assigned to the top half, erring toward a lower H threshold and thus more H observations. If H_n remains too small after this partitioning—because high-price arms rarely convert—the highest-ranked bottom-half arm is iteratively reassigned to the top half until H_n reaches a usable size.

L_n samples. For each item i, the LLM Annotator extracts the discovered attributes from t_i, producing a local feature vector $\hat{a}_n(i)$. The split rule σ_n then operates on $\hat{a}_n(i)$ via a simple lookup. No LLM call is needed at pricing time.

Note the parallel to honest estimation in causal trees (Athey and Imbens, 2016): the exploration data used to discover the split (H_n and L_n) is distinct from the data used to estimate prices in the child nodes (fresh exploration in children). This separation prevents overfitting the segmentation to noise in a particular set of purchase outcomes.

Split validation. After splitting node n into children n_L and n_R, the agent runs price exploration in each child independently. The split is retained if and only if the children's optimal prices differ: $p^*_{n_L} \neq p^*_{n_R}$. If both children converge to the same price, the split is discarded—it may separate items along a textually salient dimension, but not one that is economically relevant.

Recursion. At the conclusion of the procedure above, node n is in one of two states. If no valid split was found—either the LLM Analyst did not surface a useful attribute or split validation failed because the two children converged to the same price—the node remains intact as a leaf, assigned price p^*_n, and the LM Tree is finished with it. If a valid split was found, node n is partitioned into two child nodes n' and n'', each containing its respective subset of content items and each carrying its own discovered price. For each child, the entire procedure—price exploration, feature discovery, annotation, and split validation—may be repeated. The tree grows recursively in this way, but will not recurse beyond a maximum depth D. Nodes at depth D are always treated as leaves regardless of whether a further split might be beneficial.

Figure 1 illustrates one complete cycle at a single node.

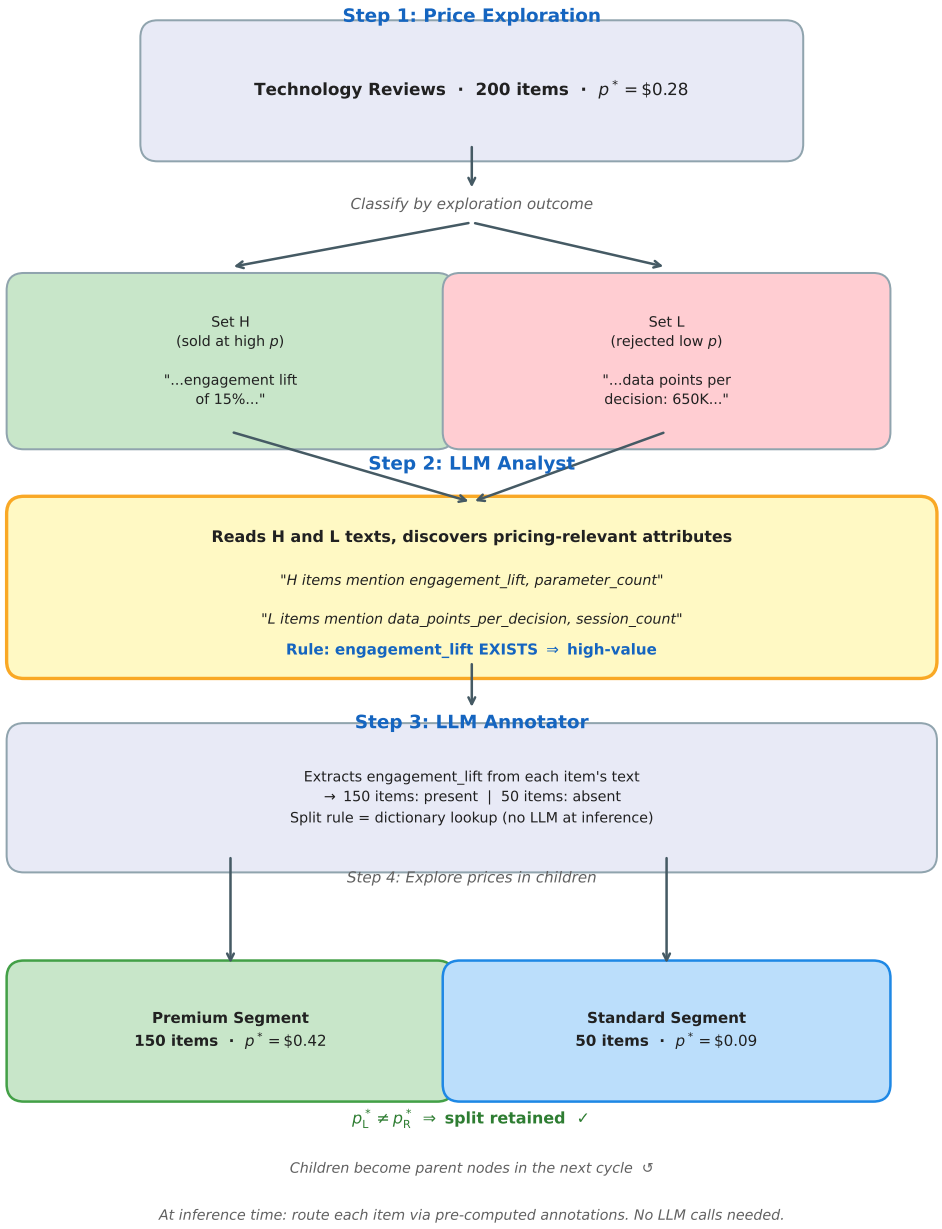


Figure 1: One cycle of the LM Tree at a single node. The LLM Analyst reads content from sets H and L to discover a pricing-relevant split rule; the LLM Annotator applies that rule to all items in the node. At inference time, routing requires only the pre-computed annotations—no LLM calls.

4.3 Algorithmic Properties

Two design choices deserve attention beyond the core algorithmic structure.

Log-scale exploration anchored to the parent's optimal price. These two choices operate as a unit. At each node n, the price arms are spaced on a log scale—multiplicatively symmetric around a baseline, so that a 10× upward exploration mirrors a 10× downward exploration. This is the natural geometry for prices, where ratios matter more than absolute differences. The baseline is not fixed globally but inherited from the parent node's optimal price p^*_{parent}, which has already been validated one level up the tree.

Together, these choices allow the agent to approximate heterogeneous optimal prices even when they differ by orders of magnitude across content types. A publisher's catalog may contain commodity news items worth fractions of a cent and flagship benchmark reviews worth several dollars—a range spanning two or more orders of magnitude. The agent does not need to know this range in advance. At the root, it explores around a rough initial baseline. At each split, child nodes inherit a refined anchor from their parent and re-explore on a log scale centered there. Over successive rounds of recursion, each branch converges toward the true optimal price for its content sub-type, regardless of where that price sits in absolute terms.

Price exploration plays a dual role. The exploration trials at node n serve two distinct purposes across two different rounds of the recursion. In the *current* round, they identify the best price for node n: by spreading trials across a wide log-spaced range, the agent estimates per-arm revenue and selects p^*_n. In the *next* round, the same trials generate the variation necessary to discover the right split rule and create child nodes n' and n'': because prices are spread wide, some items reveal high WTP (they sell at top-half arms) and others reveal low WTP (they fail at bottom-half arms), producing the H_n and L_n contrast sets the LLM Analyst needs. The variation required for segmentation in the next round is a byproduct of the variation required for pricing in the current round, at no additional data collection cost.

With the agent described, we next turn to our data.

5 Data: HardwareLuxx

We evaluate the LM Tree on real content from HardwareLuxx (HWL), a major German technology publisher whose editorial archive is crawled extensively by AI systems. The dataset combines two components: observed AI crawler traffic (real data from HWL's server logs) and calibrated willingness-to-pay (a constructed proxy, described below). We keep these

Table 1: AI crawler views by format category

Category	Articles	Median	Mean	Std	CV
Artikel (reviews)	1,624	15.0	22.8	26.2	1.15
News	7,315	4.0	5.8	4.8	0.81
Total	8,939	5.0	8.9	13.6	1.53

distinct throughout.

5.1 Traffic Data

HardwareLuxx records AI crawler activity via Cloudflare, capturing requests from major AI systems including GPTBot (OpenAI training), OAI-SearchBot (OpenAI retrieval), ClaudeBot (Anthropic), and PetalBot (Huawei). Over a 30-day window, these crawlers generated hundreds of thousands of requests across HWL's editorial archive, with access rates varying substantially across articles and consistent with a power-law distribution.

HWL's editorial content is organized in a three-level taxonomy derived from its Joomla CMS. The two observable **format categories** constitute the coarse partition $\mathcal{C}$ available to the agent: long-form reviews and benchmark articles (*artikel*) and shorter news items (*news*). This format distinction is coarse—both categories span hardware, software, and consumer electronics coverage—and captures article style rather than content. The finer **editorial categories** (eight content domains such as hardware, software, and consumer electronics, nested within each format) are *not* observable to the agent; they must be discovered from text.

We use 8,939 articles for which full text is available (content published from mid-2019 onward). Articles are split into a training set of 7,210 articles, on which the agent learns prices, and a test set of 1,729 articles, on which learned prices are applied without further adjustment. The split is stratified by format category and is across articles: no article appears in both sets.

Reviews attract substantially more AI crawler attention per article than news items (median 15 vs. 4 views), a 3.8× gap that reflects that long-form benchmark content is more intensively consumed by AI systems than short product announcements. Traffic is also highly concentrated: the top 10% of articles account for 42% of total crawler requests, and the top 25% account for 64%. This concentration motivates the tree structure—a flat pricing policy leaves substantial revenue on the table by treating the high-traffic tail the same as the long tail of lightly-crawled articles.

Table 2 reports crawler-view statistics by editorial category—the eight content domains

Table 2: AI crawler views by editorial category

Format	Editorial category	Articles	Median	Mean	Std	CV
Artikel	Hardware	1,371	17.0	25.0	27.5	1.10
Artikel	Software	124	6.0	6.9	4.9	0.71
Artikel	Consumer electronics	111	9.0	15.0	14.9	0.99
Artikel	Miscellaneous	18	7.0	8.7	6.7	0.77
News	Hardware	3,744	4.0	6.0	4.8	0.80
News	Software	1,663	4.0	5.4	4.7	0.87
News	General	1,209	4.0	5.4	4.5	0.83
News	Consumer electronics	699	5.0	6.5	4.7	0.73

beneath the two observable format categories.

Interpretation. Three patterns in Table 2 motivate the LM Tree design. First, the format split does meaningful work: *artikel* hardware (median 17) towers over every news sub-category (median 4–5), a gap the agent can exploit with a single coarse partition. Second, within *artikel*, hardware is a clear outlier—the remaining three sub-categories (software, consumer electronics, miscellaneous) have medians of 6–9, closer to one another than to hardware, so the relevant within-*artikel* split is "hardware vs. rest," not a smooth gradient across all four domains. Third, within *news*, the four sub-categories are nearly indistinguishable by median (all 4–5) and by CV (0.73–0.87): the editorial taxonomy adds no pricing information for news items. Whatever variation exists within news is article-level, not sub-category-level—it must be recovered from text, not from the category label. This is precisely the setting where the LM Tree's text-based splits are most valuable.

Crawler-level heterogeneity. The five crawlers in Table 3 differ substantially in the content they seek. GPTBot directs 85% of its requests to short news items, consistent with broad-coverage training data collection. OAI-SearchBot—from the same parent company—does the opposite: 73% of its requests target long-form *artikel* reviews, consistent with retrieval for search grounding where authoritative depth matters more than breadth. PetalBot, ClaudeBot, and PerplexityBot occupy intermediate positions, with a roughly even split between the two formats.

This heterogeneity implies that the same article carries different value to different crawlers—the *artikel* hardware review that OAI-SearchBot heavily targets is of little interest to a GPTBot sweep of news items. Optimal pricing would therefore combine second-degree price discrimination (segmenting by content characteristics, as the LM Tree does) with third-degree price discrimination (setting different prices for different crawler identities). Crawler identity

Table 3: AI crawler requests by bot and content format (30-day window)

Bot	Total requests	% *Artikel*	% *News*
PetalBot	468,811	44.5	55.5
GPTBot	168,370	14.7	85.3
OAI-SearchBot	42,718	73.0	27.0
ClaudeBot	13,044	41.7	58.3
PerplexityBot	9,077	45.7	54.3

Table 4: Calibrated WTP distribution ($v(i) = 0.004 \times$ AI views)

	Min	Median	Mean	Max
All articles	\$0.004	\$0.020	\$0.036	\$1.112
Artikel (reviews)	\$0.004	\$0.060	\$0.091	\$1.112
News	\$0.004	\$0.016	\$0.023	\$0.160

is observable in practice: AI systems authenticate when accessing gated content, making buyer type a feasible conditioning variable. The LM Tree framework is readily extensible in this direction—the Level-0 partition $\mathcal{C}$ can be defined over (content category $\times$ crawler identity) rather than content category alone. In this paper we focus on the content dimension only, leaving the joint segmentation for future work.

5.2 WTP Calibration

No pay-per-crawl pricing is currently live at HardwareLuxx—as is typical of the market at this stage. The infrastructure to charge AI crawlers per access (HTTP 402, Tollbit, Cloudflare's bot management layer) is only now being adopted by early-moving publishers (Stack Overflow, 2024). Transaction-level willingness-to-pay data does not yet exist for HWL or for most publishers. We therefore calibrate WTP from the traffic data:

$$v(i) = 0.004 \times \text{observed AI crawler views}(i)$$

The coefficient 0.004 is chosen so that the median article WTP is \$0.02—a plausible per-access price for the nascent pay-per-crawl market, consistent with the range discussed in early industry implementations. Table 4 reports the resulting distribution.

Two arguments justify this calibration approach. The first is directional: in an unpriced market, AI systems allocate their crawl budget toward content they find most useful, so access frequency is a plausible proxy for value. Crawlers visit more what they would pay more for. The second argument is structural and more fundamental. Even if traffic is an imperfect

directional proxy for WTP, the traffic data has exactly the statistical structure that a real WTP distribution would need to have in order to test the LM Tree's core claim. As Tables 1–2 show, crawler views are partially correlated with observable content categories—reviews attract more traffic than news, and hardware reviews attract more than software reviews—but category alone leaves substantial within-group variation unexplained. That residual variation is not random noise: the top articles within each category are identifiable from their content (flagship GPU benchmark reviews, canonical CPU comparisons), suggesting that the within-category signal is recoverable from text. A WTP distribution with this structure—coarse-category signal plus large text-recoverable within-category variation—is precisely the setting in which the LM Tree is designed to add value over a category-level pricer. By calibrating WTP from traffic, we construct a test environment that is structurally valid for our evaluation regardless of whether traffic tracks WTP levels precisely.

6 Agent Training

This section describes how the pricing agents are trained on the HardwareLuxx data. We first explain how buyer queries are generated to simulate transaction-level demand, then describe the market simulation environment in which agents learn prices from binary purchase feedback.

6.1 Query Generation

Each article receives 9 synthetic buyer queries, yielding 80,451 queries in total (64,890 training, 15,561 test). Query-level WTP is drawn independently as:

$$v(i,q) \sim \mathcal{N}\big(v(i),\ \sigma^2\big)\,, \quad \sigma = 0.001, \quad \text{truncated at } 0$$

The small σ captures idiosyncratic variation across requests while keeping WTP close to the article-level center. Arrival order is randomized across all articles so the agent encounters content in a shuffled sequence rather than in batches by category.

6.2 Market Simulation and Online Training of Agents

We train three pricing agents of increasing sophistication on 8,939 real HardwareLuxx articles (7,210 train, 1,729 test). All agents use the same log-scaled grid exploration with binary purchase feedback. Each agent observes each article's text t_i and its coarse category

Table 5: Revenue by pricing strategy (HardwareLuxx; 7,210 train articles, 1,729 test articles). Percentage gains are computed on the test set.

Strategy	Train	Test	vs. Single Price	vs. Format (2)
Single Price	\$690	\$160	—	—
Category Pricing – Format (2)	\$726	\$179	+12%	—
Category Pricing (8 segments)	\$772	\$189	+18%	+6%
LM Tree	\$1,075	\$264	+65%	+47%

$c(i) \in \{\textit{artikel}, \textit{news}\}$, but never the crawler view counts or the query-level WTP. It receives only the binary purchase outcome after each pricing decision.

The three agents are:

- **Single Price:** A single price for all content, discovered by log-spaced grid exploration over the full query stream.
- **Category Pricing:** One price per category, discovered independently by exploration within each category. We consider two variants: a coarse partition into the two format categories (artikel, news), and a finer partition into the eight editorial categories (artikel/hardware, artikel/software, news/hardware, etc.).
- **LM Tree:** The agent described in Section 4. It begins with the two format categories and uses the LLM Analyst to discover finer segmentation from article text.

7 Empirical Results

We evaluate the three agents on the held-out test set and examine the pricing rules they produce. Section 7.1 compares revenue across agents. Section 7.2 looks inside the learned prices and splits to understand what drives the gains.

7.1 Performance Analysis

Table 5 reports revenue for each strategy on both the training and test sets. Percentage comparisons are based on test-set revenue.

The format categories (artikel vs. news) are a *format* distinction rather than a content distinction: both categories cover the same product domains, differing mainly in article depth and style. Accordingly, format category pricing gains only 12% over a single price. Moving to the finer eight-segment editorial partition—which separates hardware from software from consumer electronics within each format—adds a further 6%, reaching 18% above the

Table 6: Learned prices for observable-based policies. Editorial category names follow the HardwareLuxx editorial taxonomy.

Policy	Segment	Price
Single Price	All content	\$0.055
Category Pricing – Format (2)	news	\$0.028
	artikel	\$0.263
Category Pricing (8 segments)	artikel/hardware	\$0.263
	artikel/consumer-electronics	\$0.045
	artikel/sonstiges	\$0.034
	artikel/software	\$0.030
	news/hardware	\$0.028
	news/software	\$0.028
	news/consumer-electronics	\$0.028
	news/allgemein	\$0.028

single-price baseline. The editorial taxonomy helps, but both category pricing variants leave the majority of WTP heterogeneity unexploited.

The LM Tree achieves \$264 on the test set, a 65% gain over single pricing and a 47% gain over format category pricing. Notably, the LM Tree is initialized with only the two format categories; it has no access to the eight-segment editorial taxonomy. Yet it outperforms editorial category pricing by 40%. The splits the agent discovers—articles covering high-end GPU and CPU components, precise performance specifications, advanced cooling solutions—cut across the formal editorial boundaries in ways that align more closely with AI crawler WTP than the publisher's own content taxonomy.

The train–test pattern reflects the cost of online learning rather than overfitting. The LM Tree accumulates \$1,075 on the training stream, which includes the exploration cost of the log-scaled price grid; on the held-out test set, at the prices learned during training, it earns \$264. The gap is proportional across strategies: format category pricing shows a similar train/test ratio, confirming that the drop is a feature of the exploration protocol, not a property of the LM Tree specifically.

7.2 Prices and Splits

7.2.1 Observable-Based Policies

Table 6 shows the prices chosen by each observable-based policy after training.

Moving from a single price to the two format categories reveals a sharp format premium: artikel is priced at \$0.263, nearly ten times the news price of \$0.028. This reflects the aggregate

Table 7: Learned prices for the LM Tree. Each leaf corresponds to one split rule applied within a format category.

Format category	Leaf	Split rule	Price
news	high-value	Market value $\geq$ \$1,000 (threshold)	\$0.023
	low-value	otherwise	\$0.019
artikel	high-value	High-end GPU mentioned (existence)	\$0.148
	low-value	otherwise	\$0.081

WTP difference between in-depth hardware reviews and news announcements. Moving to the eight-segment editorial partition breaks the artikel category apart: hardware reviews retain the high price (\$0.263), while consumer-electronics articles (\$0.045), miscellaneous articles (\$0.034), and software articles (\$0.030) are priced down closer to their true WTP centers. All four news sub-types converge to the same price (\$0.028), indicating that WTP is homogeneous within the news format regardless of topic. The finer partition thus enables more precise pricing for the heterogeneous artikel segment, which is where most of the revenue opportunity lies.

7.2.2 The LM Tree

Table 7 shows the four prices learned by the LM Tree.

The full rule descriptions produced by the LLM Analyst are:

- **news:** "Select items that have a market value of at least 1000 USD, indicating they are likely to be high value content."
- **artikel:** "Identify high-performing content by checking the presence of high-end GPU specifications like NVIDIA GeForce RTX 30 series or AMD equivalent in the content. This attribute is likely to be present in high-value content but absent in low-value content."

Two features of the learned tree stand out. First, the artikel split produces the largest price separation: GPU and CPU review articles are priced at \$0.148, while other hardware reviews are priced at \$0.081—both above the highest news price (\$0.023), consistent with the format premium. Second, the news split is a threshold rule ("market value $\geq$ \$1,000") rather than an existence rule, separating news coverage of high-end equipment from commodity product announcements with a value benchmark.

The LM Tree's splits do not coincide with editorial category boundaries. Table 8 shows the share of queries assigned to the high-value leaf within each editorial category.

Table 8: Share of queries assigned to the high-value leaf, by editorial category. No editorial category maps cleanly onto either leaf, confirming that the LM Tree's splits cut across the formal content taxonomy.

Format	Editorial category	Share assigned high
news	news/allgemein	16.7%
	news/hardware	11.1%
	news/consumer-electronics	9.2%
	news/software	1.4%
artikel	artikel/sonstiges	40.0%
	artikel/hardware	17.4%
	artikel/software	6.9%
	artikel/consumer-electronics	5.3%

For the artikel split (high-end GPU rule), artikel/hardware has the highest share assigned high (17.4%), as expected—but the rule also reaches into artikel/sonstiges (40%, small sample) and artikel/software (6.9%). For the news split (market-value threshold), news/allgemein tops the table at 16.7%, ahead of news/hardware (11.1%)—general news coverage of high-value products triggers the threshold rule more than dedicated hardware news. No single editorial category is cleanly mapped to either leaf.

This cross-cutting structure is precisely what allows the LM Tree to outperform editorial category pricing despite being initialized with only the two format categories. The editorial taxonomy groups articles by topic; the LM Tree groups them by the textual signals that predict AI crawler WTP. These two partitions are correlated—GPU reviews do dominate the high-value leaf—but they are not the same. The agent recovers a segmentation that the publisher's own content hierarchy does not provide.

8 Discussion

8.1 Alternative Business Models for AI Content Access

The emergence of AI systems as content consumers raises a prior question: what business model should govern this market? Three candidates are in play. Pay-per-crawl is one; the others each have a natural precedent in adjacent markets, but neither transfers cleanly to this setting.

Bulk licensing. The most visible current model is negotiated bulk licensing: a lump-sum payment grants an AI company broad access to a publisher's archive. Google's reported $60 million deal with Reddit (The New York Times, 2024) is the most prominent example.

Such deals work well when both parties are large and the content is broadly relevant—they avoid per-transaction negotiation costs and provide predictable revenue. But they do not scale to the long tail of publishers, and they price content in aggregate: high-value technical documentation and commodity news are bundled at the same rate. A platform brokering deals on behalf of thousands of small publishers would face the mechanism selection problem in its most acute form, with no market signal and no discovery mechanism.

Auctions. Online advertising established keyword auctions—most famously, sponsored search—as the dominant mechanism for pricing scarce digital attention. The case for auctions rests on genuine slot scarcity: each query generates one sponsored position, and bidding resolves the allocation efficiently. No analogous scarcity exists in the content-to-AI market. AI crawlers can access any piece of content independently and without displacing any other buyer; there is no slot to auction. The efficiency rationale that makes search auctions work does not carry over.

Pay-per-crawl. Per-access pricing avoids both limitations. It scales without upfront negotiation—access is priced automatically per request—and it can reflect heterogeneous content value at the item level, provided the pricing mechanism is sophisticated enough to discover those differences. The latter is the problem the LM Tree is designed to solve. Rothschild et al. (2025) conjecture more broadly that micro-transaction-based pricing will become a natural architecture of the agentic economy, as AI agents transact at a volume and granularity that bulk deals cannot accommodate.

Early market evidence suggests pay-per-crawl is gaining traction. Cloudflare's AI Audit product and Tollbit now provide infrastructure for per-access charging, and publishers including Stack Overflow have begun gating AI crawler access under such arrangements (Stack Overflow, 2024). The market is nascent, but the infrastructure is in place. Whether pay-per-crawl becomes a dominant model—alone or alongside bulk licensing for the largest publishers—will depend on whether pricing mechanisms can adapt to the heterogeneity of content. That is the gap this paper addresses.

8.2 Beyond Pay-Per-Crawl

The pay-per-crawl setting is the motivating application, but the LM Tree is designed for a problem class, not a single market. The relevant conditions are three: goods are heterogeneous, WTP is unobservable, and the features that drive value are embedded in the text of the goods themselves rather than in a structured metadata table.

API access pricing. AI companies and infrastructure providers increasingly sell access to capabilities—model inference, retrieval pipelines, specialized tools—at per-call or per-token

prices. The value of a particular API endpoint depends on what the endpoint does, which is described in documentation prose, not in a structured schema. An LM Tree could partition the endpoint catalog by the *kinds of tasks* buyers use each endpoint for, discovering that endpoints described in terms of "reasoning over long contexts" command different prices than those described in terms of "fast single-turn classification"—without requiring the seller to specify these categories in advance.

Data licensing. Data vendors license datasets to buyers whose intended use determines their WTP. The LM Tree's architecture applies directly: explore prices, identify high- and low-WTP buyers, ask the LLM Analyst what the descriptions of high-revenue datasets have in common, annotate the catalog, and split.

Professional and expert services. Pricing for consulting engagements, legal research, or specialized reports is often set by negotiation precisely because the value-relevant features—scope, depth, the specific expertise mobilized—are embedded in the engagement description, not in a billing code. A tree that discovers, from historical outcomes, which textual features of engagements predict high client WTP provides a principled pricing floor for future proposals.

In each of these markets, the same structural insight applies: the LM Tree does not require the seller to know in advance which features of their offerings matter to which buyers. The existence-rules finding extends naturally to these settings: products that mention "real-time" are categorically different from products that do not—the presence of the concept signals a different use case, a different infrastructure cost, and a different buyer profile.

9 Conclusion

As AI systems shift from delivering search results to consuming content directly, publishers need a new revenue model—and a new pricing technology. Pay-per-crawl creates the market; the LM Tree provides the pricing agent. It grows a segmentation tree over the content library, using an LLM Analyst to discover pricing-relevant attributes from content text at each node and an LLM Annotator to apply them at scale. The result is a system that discovers both the right segmentation and the right prices from binary purchase feedback, with no prior knowledge of which content features matter or how they interact with buyer willingness-to-pay.

Evaluated on 8,939 real HardwareLuxx articles, the LM Tree achieves a 65% revenue gain over a single static price and a 47% gain over two-category pricing—outperforming even the publisher's own 8-segment editorial taxonomy by 40%—using only article text and binary purchase feedback. The agent's discovered splits cut across the publisher's formal editorial categories, recovering a segmentation based on textual signals that aligns more closely with

AI crawler willingness-to-pay than the editorial taxonomy does. Content covering high-end GPU and CPU components belongs to a different pricing regime than commodity news about the same products—a distinction the publisher's own category labels do not capture.

The deeper contribution is to the tree literature: the LM Tree replaces feature selection with feature construction, enabling tree-based pricing in markets where the relevant dimensions are unknown, content-specific, and too numerous to enumerate—markets that share the structure of mechanism selection at scale. The value is not only in finding the right prices but in discovering the right pricing rule. A financial news outlet needs recency-based pricing; a legal database needs jurisdiction-based pricing; a product review site needs tier-of-product pricing. The LM Tree discovers which dimensions of a publisher's content drive AI crawler willingness-to-pay—without requiring the publisher to know this in advance. This is not unique to pay-per-crawl. Any market in which goods are described in text and willingness-to-pay is unobservable—API access, data licensing, professional services—shares the same structure. As AI intermediates more transactions, agents that learn from language as well as from prices will become increasingly important.

References

Ajay Agarwal, Joshua Gans, and Avi Goldfarb. *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press, 2018.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

Dirk Bergemann, Benjamin Brooks, and Stephen Morris. The limits of price discrimination. *American Economic Review*, 105(3):921–957, 2015.

Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57(6):1407–1420, 2009.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

Jean-Pierre Dubé and Sanjog Misra. Personalized pricing and consumer welfare. *Journal of Political Economy*, 131(1):131–189, 2023.

Matthew Gentzkow, Bryan Kelly, and Matt Taddy. Text as data. *Journal of Economic Literature*, 57(3):535–574, 2019.

Soheil Ghili. A characterization for optimal bundling of products with nonadditive values. *American Economic Review: Insights*, 5(3):311–326, 2023.

Soheil Ghili, K. Sudhir, Nitish Jain, and Ankur Garg. Second-degree price discrimination: Theoretical analysis, experiment design, and empirical estimation. Working paper, 2025.

Nima Haghpanah and Jason D. Hartline. When is pure bundling optimal? *Review of Economic Studies*, 88(3):1127–1156, 2021.

Nima Haghpanah and Ron Siegel. The limits of multiproduct price discrimination. *American Economic Review: Insights*, 4(4):443–458, 2022.

Nima Haghpanah and Ron Siegel. Pareto-improving segmentation of multiproduct markets. *Journal of Political Economy*, 131(6):1546–1575, 2023.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 594–605, 2003.

Eric Maskin and John Riley. Monopoly with incomplete information. *RAND Journal of Economics*, 15(2):171–196, 1984.

Kanishka Misra, Eric M. Schwartz, and Jacob Abernethy. Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science*, 38(2): 226–252, 2019.

Michael Mussa and Sherwin Rosen. Monopoly and product quality. *Journal of Economic Theory*, 18(2):301–317, 1978.

David M. Rothschild, Markus Mobius, Jake M. Hofman, Eleanor Dillon, Daniel G. Goldstein, Nicole Immorlica, Sonia Jaffe, Brendan Lucier, Aleksandrs Slivkins, and Matthew Vogel. The agentic economy. arXiv:2505.15799, 2025. Microsoft Research.

Carl Shapiro and Hal R. Varian. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press, 1999.

Stack Overflow. Stack overflow and Cloudflare partner to help developers protect and monetize content in the AI era. `https://stackoverflow.blog/2024/09/19/stack-overflow-and-cloudflare-partner-to-help-developers-protect-and-monetize-content` 2024. September 2024.

The New York Times. Reddit in $60 million deal for training A.I. systems. `https://www.nytimes.com/2024/02/22/technology/reddit-google-artificial-intelligence.html`, 2024. February 22, 2024.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Frank Yang. Nested bundling. *American Economic Review*, 115(9):2970–3013, 2025.

Appendices

A Data Preparation

This appendix describes how the HardwareLuxx dataset was constructed from raw publisher data. The pipeline has four stages: (1) data extraction from the HWL content management system; (2) WTP calibration from AI-crawler traffic records; (3) train/test splitting; and (4) query generation. The pipeline is designed around the same invariant as the theoretical setting: the agent observes only the coarse category $c(i)$ and content text t_i of each article. Crawler view counts, query-level WTP, and the full editorial taxonomy are hidden. The only feedback available to the agent is the binary purchase outcome $y = \mathbf{1}[p \leq v(i, q)]$.

A.1 Data Extraction

HardwareLuxx publishes articles through a Joomla content management system. The data was extracted via a direct database export coordinated with the HWL technical team. Two tables were used: the article metadata table (containing article ID, title, publication date, and Joomla category ID) and the article content table (containing the full German-language article text). A third table—the Joomla categories table—provides the content taxonomy.

The raw export contained all articles published from summer 2019 onward (article IDs $\geq$ 50,000). Articles with missing or empty body text were discarded. The final dataset contains 8,939 unique articles.

A.2 Content Taxonomy

HardwareLuxx organizes articles in a three-level hierarchy derived from the Joomla categories table. Each category record has a `path` field that encodes the full ancestry as a slash-delimited string (e.g., `artikel/hardware/grafikkarten`). The format, editorial, and sub-category labels are read directly from the path components; no inference is required.

The two observable format categories are:

- **artikel**: long-form hardware reviews, test reports, and buying guides.
- **news**: shorter news articles covering product announcements, industry events, and market developments.

The editorial layer contains five content categories (hardware, software, consumer-electronics, allgemein, sonstiges) under *artikel* and three under *news*. The sub-category layer contains 70 leaf categories in total (e.g., grafikkarten, prozessoren, mainboards under

hardware). The format and editorial category labels are available as structured fields in the dataset; the sub-category label is derivable from the category path but is not exposed to the agent.

Table 9: HWL taxonomy: article counts by format and editorial category

Format	Editorial category	Articles
artikel	hardware	1,371
artikel	software	124
artikel	consumer-electronics	111
artikel	sonstiges	18
news	hardware	3,744
news	software	1,663
news	allgemein	1,209
news	consumer-electronics	699
Total		8,939

A.3 WTP Calibration

Willingness-to-pay is calibrated from AI-crawler traffic records. HardwareLuxx's server logs record the number of page views attributable to known AI-crawler user agents (GPTBot, ClaudeBot, PerplexityBot, and others) for each article. The article-level WTP center is:

$$v(i) = 0.004 \times \text{AI-crawler views for article } i$$

The coefficient \$0.004 per view is a calibration parameter chosen to produce a WTP distribution in a plausible per-article pay-per-crawl pricing range (cents to dollars). The query-level WTP is drawn independently for each query:

$$v(i,q) \sim \mathcal{N}\big(v(i),\ \sigma^2\big)\,, \quad \sigma = 0.001, \quad \text{truncated at } 0$$

The small σ captures idiosyncratic variation across requests while keeping per-query WTP close to the article-level center.

Table 4 summarizes the resulting WTP distribution by format category. The large gap between *artikel* and *news* medians (\$0.060 vs. \$0.016) reflects that long-form hardware reviews attract substantially more AI-crawler traffic than short news items, likely because they contain the dense specifications and benchmark tables that AI systems extract.

A.4 Train/Test Split

Articles are split into a training set (7,210 articles, 80.7%) and a test set (1,729 articles, 19.3%), stratified by format category to preserve the artikel/news ratio in both sets. Stratification ensures that the format category distribution the agent encounters during training matches the distribution it is evaluated on.

Nine queries are generated per article, yielding 64,890 training queries and 15,561 test queries (80,451 total). Within each split, query arrival order is randomized across all articles so the agent does not encounter content in format-category batches.

A.5 Query Generation

Each article generates nine synthetic queries. Queries are generated using a language model that reads the article title and a short excerpt, then produces plausible information-seeking requests that an AI crawler might issue. The nine queries per article vary in specificity and framing (e.g., broad topic requests, specific component questions, benchmark comparisons) to represent the distribution of crawler intents.

The agent does not observe the query text directly; it observes only the article text t_i and the format category label $c(i)$. Query-level WTP is drawn independently per query as described in Section A.3.

A.6 Information Asymmetry

The HWL evaluation instantiates the same information asymmetry as the theoretical setting. Table 10 summarizes what exists in the dataset and what the agent can observe.

Table 10: Information structure (HardwareLuxx evaluation)

Data element	Exists in dataset	Agent-observable
Format category $c(i)$ (artikel / news)	✓	✓
Article title and full text t_i	✓	✓
Purchase outcome $y = \mathbf{1}[p \leq v(i,q)]$	✓	✓
Editorial / sub-category labels	✓	×
AI-crawler view count	✓	×
Article-level WTP center $v(i)$	✓	×
Query-level WTP $v(i,q)$	✓	×

The agent has access to the same information a publisher would have in practice: the article text it already possesses and the coarse format label (review vs. news). All pricing-

relevant structure—crawler traffic, WTP, and the finer content taxonomy—is hidden. The only feedback is binary: did the crawler purchase at the posted price or not.