

A Comprehensive Review on Designing a New Model and Architecture for Effective Web Crawling:

Somika Kumari

Research Scholar, Computer Applications, Department of Mathematics,
B R A Bihar University, Muzaffarpur

Dr. Radhe Shyam Soni

Assistant Professor, Department of IT
L N Mishra College of Business Management, Muzaffarpur Bihar

Received: 08.07.2024 Revised : 17.08.2024 Accepted: 12.10.2024

Abstract

The World Wide Web has grown into one of the largest and most complex information systems in human history, containing billions of pages across millions of domains and expanding at an unprecedented rate. Web crawling — the automated systematic traversal and retrieval of web content — forms the foundational infrastructure upon which search engines, artificial intelligence training pipelines, and large-scale data analytics systems are built. However, traditional web crawling architectures suffer from persistent and fundamental inefficiencies: redundant content retrieval, static resource allocation, poor page prioritisation, slow indexing speeds, and an inability to effectively handle the dynamically rendered content that increasingly dominates the contemporary web. This paper presents a comprehensive review of the state of the art in web crawling research, examining the evolution of crawler architectures, the taxonomy of crawler types, dominant algorithmic strategies, and the progressive integration of machine learning techniques into crawling systems. Drawing on an analysis of over forty peer-reviewed studies spanning three decades of crawling research, this review identifies critical limitations in existing models and articulates a well-defined set of research gaps centred on the absence of a unified, adaptive, AI-first crawling architecture. In response to these gaps, this paper outlines the design principles, architectural components, and algorithmic innovations of the Adaptive Intelligent Crawler (AIC), a novel web crawling model that integrates supervised learning, reinforcement learning, and natural language processing into a coherent, scalable, microservices-based crawling architecture. The proposed model is evaluated against established baselines including Breadth-First Search crawlers, PageRank-based crawlers, and the Scrapy production framework, using benchmark datasets from Common Crawl and Web Data Commons. Results demonstrate significant improvements in crawl efficiency, harvest rate, and redundancy reduction, establishing the AIC as a substantial advance over the current state of the art.

Keywords: *web crawling, intelligent crawlers, machine learning, URL prioritisation, distributed architecture, reinforcement learning, focused crawling, dynamic content, scalable systems, PageRank, information retrieval.*

1. Introduction

The World Wide Web constitutes one of the most significant technological and informational achievements of the modern era. From its origins as a small network of hyperlinked academic documents in the early 1990s, it has expanded into a global information ecosystem containing an estimated 5.6 billion indexed pages and an exponentially larger body of deep web content that remains inaccessible to standard retrieval systems (He et al., 2007). This extraordinary growth has been accompanied by a corresponding increase in the complexity, dynamism, and heterogeneity of web content, creating profound challenges for the systems tasked with navigating, retrieving, and organising this information.

Web crawling — the automated, systematic traversal of the web through hyperlink structures — is the foundational technology upon which virtually all large-scale web information retrieval is built. Search engines depend on crawlers to discover and index web pages; artificial intelligence systems depend on crawlers to collect the training corpora on which their capabilities are built; data analytics platforms depend on crawlers to gather the raw data that feeds their insights (Olston & Najork, 2010). The quality, efficiency, and intelligence of web crawling systems therefore has ramifications that extend far beyond the narrow technical domain of crawler engineering, influencing the quality of search results experienced by billions of users, the capabilities of AI systems deployed across industry and government, and the richness of the data available for scientific and commercial analysis.

Despite the critical importance of web crawling and despite three decades of sustained research effort, fundamental inefficiencies persist in the design of crawling systems. Traditional crawlers that employ simple Breadth-First Search (BFS) or Depth-First Search (DFS) traversal strategies treat all web pages as equally worthy of retrieval, without any mechanism for distinguishing between high-value and low-value content (Najork & Wiener, 2001). This egalitarian approach is computationally convenient but practically disastrous at web scale, where the distribution of page quality is highly skewed and where crawling resources must be carefully allocated to maximise the value of retrieved content within finite bandwidth, storage, and compute budgets.

The introduction of PageRank-based crawling by Brin and Page (1998) represented the first major advance beyond uninformed traversal strategies, demonstrating that link-based quality signals could significantly improve the proportion of high-importance pages discovered early in the crawl. Subsequent work on focused crawling (Chakrabarti et al., 1999), online page importance computation (Abiteboul et al., 2003), and machine learning-based URL prioritisation (Gattani et al., 2013) has progressively refined and extended this basic insight. However, as this review will demonstrate, existing approaches remain fragmented — typically incorporating a single intelligence technique into an otherwise conventional crawling architecture — and fail to address the full range of challenges faced by modern web crawling systems.

This paper makes three primary contributions to the literature. First, it provides a systematic and comprehensive review of web crawling research, covering the evolution of crawler architectures, the taxonomy of crawler types, algorithmic strategies, and the state of machine learning integration in crawling systems. Second, it conducts a rigorous analysis of the limitations of existing approaches and articulates a clear set of research gaps that motivate further investigation. Third, it outlines the design and preliminary evaluation of the Adaptive

Intelligent Crawler (AIC), a novel web crawling model that addresses the identified gaps through a principled integration of multiple AI capabilities within a scalable, microservices-based architecture.

The remainder of this paper is organised as follows. Section 2 reviews the evolution of web crawlers and the taxonomy of crawler types. Section 3 examines web crawling algorithms and architectural approaches. Section 4 reviews the integration of machine learning in crawling systems. Section 5 analyses the limitations of existing models and identifies research gaps. Section 6 presents the proposed AIC model and architecture. Section 7 describes the evaluation methodology and preliminary results. Section 8 discusses the findings and their implications, and Section 9 concludes the paper.

2. Evolution of Web Crawlers

2.1 Early Crawlers: 1993–1998

The history of web crawling begins in 1993 with the World Wide Web Wanderer, developed by Matthew Gray at the Massachusetts Institute of Technology. The Wanderer was not conceived as an indexing system but as a measurement tool, designed to track the growth rate of the web by periodically traversing its link structure and counting accessible servers (Gray, 1993). Despite its modest ambitions, the Wanderer demonstrated the fundamental feasibility of automated web traversal and, by generating traffic volumes significant enough to be misidentified as malicious activity, provided an early demonstration of the importance of crawler politeness constraints — a design consideration that remains critical to this day.

The development of the first true indexing crawlers followed within months. JumpStation, WebCrawler, and the World Wide Web Worm were among the earliest systems to systematically collect web content, build full-text indices, and make those indices searchable (Pinkerton, 1994). These systems were severely limited by the hardware and network infrastructure of the era — typically running on single workstations with modest memory and limited network connectivity — but they established the architectural pattern of crawl-index-search that would be refined and scaled over subsequent decades.

The period from 1993 to 1998 saw rapid growth in both the scale of the web and the sophistication of crawling systems. Early crawlers evolved from single-threaded, sequential fetchers to multi-threaded systems capable of maintaining dozens or hundreds of simultaneous connections. Basic politeness mechanisms — rate limiting, robots.txt compliance — were introduced in response to complaints from web server operators overwhelmed by crawler traffic. URL deduplication, essential for preventing the crawler from fetching the same page multiple times, emerged as a core architectural requirement as the web's link structure became increasingly complex.

2.2 The PageRank Era: 1998–2005

The publication of Brin and Page's (1998) description of the Google search engine marked a transformative moment in the history of web crawling. The paper introduced not only the PageRank algorithm — which ranks pages by the number and quality of their incoming links, providing a measure of authority that correlates strongly with user interest — but also described

the first web-scale crawling system that explicitly used link-based importance metrics to guide crawling decisions. By prioritising pages with high estimated PageRank scores for early retrieval, Google's crawler could achieve higher coverage of the 'important' web within a given crawl budget.

The insight that the hyperlink structure of the web encodes information about the relative importance of pages proved to be one of the most generative ideas in web information retrieval. Cho and Garcia-Molina's (1998) empirical comparison of BFS, DFS, and importance-ordered crawling provided rigorous evidence that intelligent ordering of the URL frontier could substantially improve the quality of early-crawled content, establishing the theoretical foundation for all subsequent work on crawl prioritisation. Their finding that BFS already provides reasonably good coverage of high-PageRank pages — since well-connected pages tend to be reachable in few hops from popular seeds — clarified both the strengths and the limitations of simple traversal strategies.

The Mercator crawler, described by Heydon and Najork (1999), represented the first comprehensive documentation of a production-quality distributed crawling system and introduced several architectural innovations that have since become standard practice. Mercator's separation of URL frontier management from content fetching and parsing, its use of priority queues for URL scheduling, its explicit handling of per-domain politeness constraints, and its modular, extensible design provided a blueprint for distributed crawler architecture that influenced virtually every subsequent large-scale crawling system. The documentation of Mercator in sufficient detail to enable replication was itself a significant contribution to the field, establishing the precedent of open, reproducible crawler research.

Chakrabarti et al.'s (1999) introduction of focused crawling extended the intelligent crawling paradigm to topical retrieval, demonstrating that topic-specific crawlers using relevance classifiers could achieve significantly higher precision — the proportion of fetched pages genuinely relevant to the target topic — than general-purpose crawlers within the same resource budget. Focused crawling opened a new research direction centred on the application of text classification and natural language processing to crawl prioritisation, laying the groundwork for the broader integration of machine learning into crawling systems that would characterise subsequent decades of research.

2.3 Distributed and Scalable Crawling: 2002–2015

The growth of the web to billions of pages during the early 2000s drove the development of increasingly sophisticated distributed crawling architectures. Shkapenyuk and Suel (2002) provided the first systematic treatment of the engineering challenges of high-performance distributed crawling, addressing URL assignment across crawler nodes, coordination between fetchers and parsers, and the management of distributed URL frontiers. Their work established the key trade-offs in distributed crawler design — between centralised coordination (which simplifies consistency management but creates bottlenecks) and decentralised coordination (which improves scalability but complicates consistency) — that continue to shape architectural decisions.

The UbiCrawler system (Boldi et al., 2004) advanced the state of the art in distributed crawler architecture by introducing consistent hashing for URL-to-node assignment, enabling

10.48047/jocaaa.2024.33.08.434

fully decentralised coordination without a central URL server. This architectural choice improved both fault tolerance — since the failure of individual nodes does not require reconfiguration of the entire system — and scalability, since nodes can be added or removed without disrupting ongoing crawl operations. The use of distributed hash tables (DHTs) for URL frontier management, explored by Exposto et al. (2007), extended this approach further, enabling fully peer-to-peer distributed crawling with no central point of failure.

The introduction of the Apache Nutch project (Khare et al., 2004) and its integration with the Hadoop distributed computing framework democratised access to large-scale crawling capabilities, making high-quality distributed crawler implementations available to the research community as open-source software. Nutch's adoption enabled reproducible comparative research across different crawling strategies and provided a common platform for the integration of new capabilities, accelerating the pace of innovation in the field. The subsequent development of production-ready crawling frameworks including Scrapy (Zyte, 2023), Heritrix (Internet Archive, 2023), and StormCrawler (DigitalPebble, 2023) further expanded the toolkit available to both researchers and practitioners.

2.4 AI-Driven Crawling: 2015–Present

The current era of web crawling research is characterised by the progressive integration of machine learning and artificial intelligence techniques into crawling systems, driven by advances in deep learning, reinforcement learning, and natural language processing. Gattani et al.'s (2013) deployment of gradient boosted tree models for URL prioritisation in a production web crawler at Yahoo! provided the first demonstration at industrial scale that machine learning could deliver significant and measurable improvements in crawling quality, validating the practical applicability of earlier academic research.

The application of deep reinforcement learning to web crawling, pioneered by Fang et al. (2022), represents the current state of the art in intelligent crawling research. By framing URL selection as a sequential decision problem and using deep neural networks to represent the complex relationships between crawler state, candidate URL features, and crawling value, reinforcement learning approaches have demonstrated substantial improvements over both BFS and PageRank-based baselines on established benchmark datasets. The integration of graph neural networks for representing the structure of the discovered web graph has further improved the capacity of RL-based crawlers to exploit the rich structural information encoded in web link patterns.

Despite these advances, a significant gap remains between the capabilities demonstrated in academic research settings and the full complexity of production web crawling at scale. Most machine learning-enhanced crawlers have been evaluated on relatively small benchmark datasets, raising questions about the scalability of their performance advantages. Furthermore, as this review will demonstrate, existing machine learning approaches have typically been applied to individual components of the crawling pipeline rather than integrated into a comprehensive, end-to-end intelligent architecture. Addressing this gap is the central motivation for the AIC model proposed in this paper.

3. Web Crawling Algorithms and Architectures

3.1 Traversal Algorithms

Web crawling algorithms differ primarily in the strategy used to order the URL frontier — the set of discovered but unvisited URLs — determining which pages are retrieved early in the crawl and which are deferred. The choice of traversal strategy has profound consequences for crawl quality, since the distribution of page quality in the web is highly heterogeneous and the crawl is almost always budget-constrained.

Breadth-First Search (BFS) ordering implements a FIFO queue, fetching pages in the order of their discovery. Despite its simplicity, BFS has attractive properties for web crawling: it tends to discover high-quality pages early, since well-connected pages with high PageRank scores are typically reachable in few hops from popular seed URLs (Najork & Wiener, 2001). BFS is computationally straightforward and memory-efficient relative to alternatives, making it the natural baseline against which more sophisticated approaches are compared. However, BFS treats all pages within the same discovery level as equally worthy, providing no mechanism for distinguishing among them.

Depth-First Search (DFS) ordering, in which the crawler follows links to their deepest extent before backtracking, is rarely used in general-purpose crawling due to its tendency to explore deep, narrow paths that quickly descend into low-quality or sparsely linked regions of the web. Empirical comparisons have consistently demonstrated that BFS outperforms DFS in terms of early discovery of high-quality pages (Najork & Wiener, 2001). DFS retains utility in site-specific crawling scenarios where comprehensive depth of exploration within a single domain is more important than breadth.

PageRank-based ordering prioritises URLs by their estimated PageRank score, approximated from available information including the PageRank of the referring page and link anchor text. Cho and Garcia-Molina (1998) demonstrated that PageRank-based ordering substantially outperforms both BFS and DFS in terms of the quality of early-crawled content, establishing it as the standard quality-oriented alternative to BFS. The OPIC algorithm (Abiteboul et al., 2003) provides an efficient online approximation of PageRank-based ordering that can be computed during the crawl without requiring complete graph information, making it practical for real-world deployment.

Topic-based and best-first crawling algorithms order the URL frontier by estimated topical relevance, using classifier models to score candidate URLs before fetching. Chakrabarti et al.'s (1999) best-first focused crawler uses a naive Bayes classifier trained on topic-relevant examples to score fetched pages and a context-based model to estimate the relevance of unfetched URLs. Diligenti et al.'s (2000) context graph approach extended this framework by representing the link neighbourhood of candidate URLs to improve pre-fetch relevance estimation. More recent deep learning approaches have demonstrated substantially higher accuracy in URL relevance prediction than earlier feature-engineering-based methods (Gouriten et al., 2014).

3.2 Crawler Architecture Patterns

The architecture of a web crawling system defines how its functional components — URL scheduling, fetching, parsing, duplicate detection, and storage — are organised and how

they interact. Architectural choices have profound consequences for the performance, scalability, and maintainability of the system.

Centralised architectures concentrate all crawling functions within a single machine or tightly coupled cluster. While simple to design and operate, centralised architectures face fundamental scalability limitations — a single machine has finite network bandwidth, CPU, memory, and storage — and represent single points of failure that are unacceptable for continuous, mission-critical crawling operations (Heydon & Najork, 1999). Centralised architectures remain appropriate for small to medium-scale crawling tasks but are unsuitable for web-scale deployment.

Distributed architectures partition the crawling workload across multiple machines, enabling throughput and capacity levels beyond any single machine's capabilities. The canonical Mercator architecture (Heydon & Najork, 1999) distributes fetching across multiple nodes coordinated by a central URL server, while UbiCrawler (Boldi et al., 2004) eliminates the central coordinator through consistent hashing, improving fault tolerance and scalability. Modern cloud-native distributed crawlers leverage containerisation and orchestration platforms such as Kubernetes to enable elastic scaling, dynamically adding or removing crawler capacity in response to workload changes.

Parallel crawling architectures exploit intra-machine concurrency — through multi-threading or asynchronous I/O — to maximise utilisation of available network bandwidth. Since web page fetching involves substantial waiting time, a crawler maintaining many simultaneous connections in progress achieves much higher throughput than a sequential fetcher (Shkapenyuk & Suel, 2002). Event-driven, asynchronous architectures such as that used by Scrapy provide high concurrency within a single process, avoiding the synchronisation complexity of multi-threaded approaches while achieving comparable throughput.

Microservices-based architectures, inspired by practices in cloud-native software engineering (Newman, 2015), decompose the crawling system into loosely coupled, independently deployable services, each responsible for a specific function. This approach enables independent scaling of individual components, facilitates incremental development and deployment, and improves resilience through service isolation. The AIC model proposed in this paper adopts a microservices architecture as the foundation for its scalable, AI-integrated design.

4. Machine Learning Integration in Web Crawling

4.1 Supervised Learning for Page Classification

The application of supervised machine learning to web crawling was pioneered by Chakrabarti et al. (1999), who used Naive Bayes text classifiers to assess the topical relevance of fetched pages in their focused crawler. Subsequent work rapidly expanded the range of classification models applied to crawling tasks. Support Vector Machines (SVMs) trained on TF-IDF text features achieved state-of-the-art topic classification performance (Joachims, 2002), while maximum entropy models and decision tree ensembles demonstrated competitive accuracy with different computational trade-offs.

10.48047/jocaaa.2024.33.08.434

Deep learning has transformed the landscape of page classification for crawling. Convolutional Neural Networks (CNNs) applied to character-level URL representations demonstrated that high-accuracy relevance prediction was achievable from URL strings alone, without requiring full page content retrieval (Gouriten et al., 2014). Transformer-based models pre-trained on large web corpora — BERT (Devlin et al., 2019) and its successors — have achieved substantial further improvements in page relevance classification accuracy, particularly for complex topical queries requiring semantic understanding beyond keyword matching. The deployment of such models in production crawling environments is constrained by their computational requirements, motivating ongoing research into model compression and efficient inference techniques.

Beyond topical relevance, supervised learning has been applied to predicting multiple dimensions of page quality relevant to crawling decisions, including PageRank score approximation, spam detection, freshness prediction, and near-duplicate probability estimation. Multi-task learning approaches that jointly predict multiple quality dimensions have shown promise for improving the efficiency of the scoring pipeline by amortising the cost of model inference across multiple quality signals (Gattani et al., 2013).

4.2 Reinforcement Learning for Adaptive Prioritisation

Reinforcement learning (RL) offers a principled framework for the URL prioritisation problem that naturally accommodates its sequential, long-horizon character. A crawling agent that must decide which URL to fetch next faces a decision problem whose value depends not only on the immediate quality of the fetched page but on the future discovery opportunities it opens — a property that greedy, single-step quality maximisation fails to exploit. Rennie and McCallum (1999) provided the first demonstration that tabular Q-learning could learn effective URL selection policies for focused crawling, outperforming greedy relevance-maximising baselines particularly in web graph regions where relevant pages are connected through chains of less relevant intermediate pages.

Deep reinforcement learning approaches have substantially extended the practical applicability of RL-based crawling. Deep Q-Networks (DQN) (Mnih et al., 2015) and Actor-Critic architectures have been applied to crawl prioritisation with promising results on standard benchmarks. Fang et al. (2022) achieved the current state of the art by combining graph neural network representations of the discovered web graph with deep RL policy learning, enabling the crawler to exploit complex structural patterns in the web graph to guide its prioritisation decisions. Their experiments on the ClueWeb benchmark demonstrated significant improvements over BFS, PageRank-based, and simpler machine learning baselines.

A critical practical challenge for RL-based crawling is sample efficiency — the number of web pages that must be fetched before the RL agent learns an effective policy. Since fetching web pages is expensive, RL approaches that require millions of environment interactions to converge are impractical for real-world deployment. Recent advances in transfer learning, where RL agents are initialised from policies pre-trained on related tasks, and in model-based RL, where a learned world model enables policy improvement without environment interaction, offer promising paths toward more sample-efficient intelligent crawlers.

4.3 Natural Language Processing for Content Understanding

Natural language processing (NLP) provides tools for understanding the semantic content of web pages and URLs that go beyond the bag-of-words representations used in early machine learning approaches to crawling. Named entity recognition, topic modelling, sentiment analysis, and semantic similarity computation have all been applied to various aspects of the crawling problem, including topical relevance assessment, near-duplicate detection, and content quality estimation.

The integration of pre-trained language models into crawling pipelines represents a particularly significant development. Models such as BERT (Devlin et al., 2019) and its successors encode rich semantic representations of text that capture nuanced relationships between concepts far beyond what keyword-based methods can express. When applied to the assessment of page relevance and quality, these representations enable substantially more accurate prioritisation decisions, particularly for complex topical queries that cannot be addressed by simple keyword matching. The computational cost of deploying large language models in the crawling pipeline remains a significant engineering challenge, but ongoing advances in model distillation and efficient inference are making this increasingly practical.

Table 1
Comparison of Machine Learning Techniques Applied to Web Crawling

ML Technique	Application	Key Features	Performance	Limitation	Reference
Naive Bayes	Page relevance classification	Bag-of-words content	Improved topic precision vs. BFS	Simple model; limited features	Chakrabarti et al. (1999)
SVM	URL/page classification	TF-IDF text features	State-of-the-art topic classif.	Requires labelled training data	Joachims (2002)
Q-Learning	URL selection policy	Page relevance signal	Outperforms greedy on focused crawl	Tabular; limited state space	Rennie & McCallum (1999)
Gradient Boosted Trees	URL prioritisation scoring	URL features, link text, domain signals	Significant harvest rate improvement at scale	Feature engineering intensive	Gattani et al. (2013)
Deep Neural Network	URL quality prediction	Character-level URL features	High accuracy without page content	Requires large labelled datasets	Gouriten et al. (2014)
Deep Q-Network (DQN)	Adaptive crawl policy	Page content embedding, graph state	Outperforms BFS and PageRank on benchmarks	High computational cost	Fang et al. (2022)
GNN + Deep RL	Web graph representation for RL	Hyperlink graph topology	State-of-the-art on ClueWeb	Limited scalability to full web	Fang et al. (2022)

5. Limitations of Existing Models and Research Gaps

5.1 Scalability Limitations

A pervasive limitation of existing machine learning-enhanced crawling systems is their insufficient attention to scalability. Most machine learning approaches to crawl prioritisation have been demonstrated on benchmark datasets comprising millions of pages — a scale two to three orders of magnitude smaller than the billions of pages that production search engine crawlers must handle. Whether the performance advantages of these approaches persist at web scale, where the computational demands of machine learning models may become prohibitive and where the diversity and complexity of the crawl environment greatly increase, remains an open and critical research question (Gattani et al., 2013).

The computational cost of training and applying machine learning models within the crawling pipeline must be carefully considered in the context of a system that may process millions of URL prioritisation decisions per hour. Models that achieve high accuracy but require substantial computation per inference may reduce overall system throughput, negating their quality advantages. Efficient model architectures, batched GPU inference, and model caching strategies are necessary to make machine learning integration practical at web scale, but these engineering considerations have received insufficient attention in the academic literature, which has focused primarily on model accuracy rather than computational efficiency.

5.2 Bandwidth and Resource Inefficiency

Existing crawling systems exhibit significant bandwidth inefficiency due to imperfect duplicate detection and suboptimal revisit scheduling. Near-duplicate detection — identifying pages that are similar but not identical to already-indexed content, such as pages that differ only in advertisements, navigation elements, or minor formatting differences — remains computationally challenging at scale. While locality-sensitive hashing techniques such as SimHash (Broder et al., 1997) and MinHash provide approximate near-duplicate detection with practical computational complexity, their accuracy degrades as the size of the index grows, and their threshold parameters require careful tuning to balance false positive and false negative rates.

Revisit scheduling — determining when to re-crawl each page to check for updates — is performed using simple statistical models in most existing systems, typically based on historical change frequency data for individual pages or domain-level aggregates (Cho & Garcia-Molina, 2003). These models do not incorporate external signals that might predict imminent content changes — such as breaking news events, seasonal content patterns, or changes in linking behaviour — and therefore systematically over-crawl stable pages and under-crawl rapidly changing ones. The integration of predictive change models that exploit contextual signals represents an important opportunity for improving crawling efficiency that has received limited attention in the literature.

5.3 Handling of Dynamic Web Content

The challenge of dynamic web content — pages whose meaningful content is generated by client-side JavaScript execution rather than server-side rendering — represents one of the most pressing and insufficiently addressed limitations of existing crawling systems. The widespread adoption of JavaScript frameworks including React, Angular, and Vue.js has resulted

in a web in which a large and growing proportion of pages are, from the perspective of a traditional HTML-only crawler, effectively empty shells whose content is invisible to retrieval (Fetterly et al., 2004).

The computational cost of JavaScript rendering — which requires spinning up a headless browser, executing the page's JavaScript, and waiting for asynchronous content loading — is typically an order of magnitude higher than simple HTTP fetching. This cost makes blanket application of JavaScript rendering to all fetched pages infeasible at scale. Selective rendering, applied only to pages where rendering is likely to yield additional valuable content, is the practically necessary approach but requires the ability to predict which pages require rendering before they are fetched — a classification problem that has received limited systematic attention in the literature.

5.4 Absence of Unified AI-First Architecture

Perhaps the most significant research gap identified by this review is the absence of a comprehensive, end-to-end web crawling architecture designed from the ground up to leverage the full capabilities of modern artificial intelligence. Existing machine learning-enhanced crawlers typically incorporate a single AI capability — URL classification, reinforcement learning prioritisation, or near-duplicate detection — as an add-on to an otherwise conventional crawling architecture, without redesigning the architecture to optimally support AI integration. This piecemeal approach limits the potential benefits of AI, since individual AI components cannot fully leverage the information available in other components.

An architecture designed specifically for AI integration — in which information flows freely between components and AI models have access to the full context of the crawling operation — has the potential to achieve substantially better performance than architectures that incorporate AI incrementally. Furthermore, the absence of standardised evaluation protocols and benchmark datasets for comparing different intelligent crawling approaches makes it difficult to measure progress in the field or to identify the most promising research directions. Establishing such standards is an important community challenge that complements the technical work of designing better crawling systems.

Table 2

Research Gap Analysis: Existing Literature vs. Proposed Contribution

Dimension	State of Literature	Gap Identified	Proposed Contribution
Crawl prioritisation	PageRank-based and single-technique ML approaches	No unified multi-technique adaptive prioritisation	Integrated RL + supervised scoring framework
Duplicate detection	SimHash/MinHash well-established; limited adaptive thresholding	Accuracy degrades at scale; static configuration	Adaptive LSH with dynamic threshold adjustment
Dynamic content	Headless browser	No principled selective	ML-predicted selective rendering

Dimension	State of Literature	Gap Identified	Proposed Contribution
	explored; mostly ad hoc	rendering pipeline	with cost-quality trade-off
Architecture	AI added to conventional architectures	No AI-first distributed architecture	Microservices architecture designed for AI integration
Resource allocation	Mostly static policies	No learned dynamic resource allocation	RL-based dynamic resource allocation across nodes
Evaluation	Varied metrics; limited standardisation	Inconsistent benchmarking impedes comparison	Standardised protocol with public benchmark datasets

6. The Adaptive Intelligent Crawler: Proposed Model and Architecture

6.1 Design Principles

The Adaptive Intelligent Crawler (AIC) is designed according to five core principles that address the limitations and gaps identified in the preceding review. The Intelligence First principle holds that every crawling decision that can benefit from machine learning should be made with machine learning assistance, with computational cost managed through model efficiency and selective application rather than by limiting the scope of AI integration. The Adaptive Operation principle requires that the system continuously learn from the outcomes of its own crawling decisions, improving its prioritisation, duplicate detection, and resource allocation policies over time. The Scalable by Design principle mandates that the architecture be designed from the outset for horizontal scalability, with no component that cannot be replicated to increase capacity. The Comprehensive Coverage principle requires that the system effectively crawl all types of web content including dynamically rendered pages. The Resource Efficiency principle demands that wasteful resource consumption be minimised at every stage of the crawling pipeline.

6.2 Architectural Components

The AIC architecture comprises eight primary components organised in a microservices pattern with bidirectional feedback flows enabling adaptive learning. The URL Frontier Manager (UFM) maintains the global priority queue of URLs, implementing the multi-factor prioritisation algorithm and exposing an API for fetcher nodes to request URL batches and report fetch results. The ML Scoring Service (MSS) computes machine learning-based quality and relevance scores for candidate URLs, running multiple learned models in parallel on GPU accelerators to achieve the throughput required for web-scale operation.

The Fetcher Service implements asynchronous, event-driven HTTP retrieval using non-blocking I/O, maintaining thousands of simultaneous connections with configurable per-domain rate limiting enforced by the Politeness Manager. Based on the dynamic content prediction score provided by the MSS, fetched pages are routed to either the Static Parser for standard HTML processing or the Dynamic Renderer for JavaScript execution using a headless Chromium browser managed by Playwright. The Duplicate Detector implements a two-stage deduplication pipeline: exact duplicate detection via SHA-256 hashing followed by near-duplicate detection

via adaptive SimHash with dynamically tuned similarity thresholds. The Storage Service persists crawled content and metadata in a distributed object store, with structured metadata available for real-time consumption by the MSS for model retraining.

6.3 URL Prioritisation Algorithm

The URL prioritisation algorithm in the AIC computes a composite quality score $Q(u)$ for each candidate URL u as a weighted combination of four component scores: $Q(u) = w1*S_quality(u) + w2*S_novelty(u) + w3*S_recency(u) + w4*S_topology(u)$, where $S_quality$ is the machine learning-predicted page quality score; $S_novelty$ is the estimated degree to which the page's likely content adds new information to the existing index; $S_recency$ is the estimated change probability since last visit; and $S_topology$ is the approximate PageRank-based importance score derived from the current partial web graph. The weights $w1$ through $w4$ are themselves learned parameters, updated via a meta-level reinforcement learning agent that optimises for long-term crawling reward, enabling the system to adapt its prioritisation strategy to the characteristics of different crawling missions and web environments.

This multi-objective formulation addresses the limitation of single-objective prioritisation identified in Section 5. By incorporating novelty alongside quality, the algorithm avoids over-investing in re-crawling already well-indexed high-quality pages while under-investing in discovering novel content. By incorporating recency predictions, the algorithm naturally implements an intelligent revisit scheduling policy without requiring a separate scheduling system. The adaptive weighting mechanism allows the system to specialise for different use cases — prioritising quality for search engine indexing, novelty for AI dataset construction, or recency for news monitoring — without requiring manual reconfiguration.

6.4 Dynamic Content Handling

The AIC's approach to dynamic content is governed by a selective rendering policy implemented by the MSS's dynamic content predictor. This component computes a rendering probability score $P_render(u)$ for each fetched URL, based on URL-level features including domain patterns associated with JavaScript frameworks, URL path characteristics, and historical rendering outcomes for the same domain. Pages with $P_render(u)$ above a configurable threshold are routed to the Dynamic Renderer; those below the threshold are processed by the more efficient Static Parser. The threshold is tunable to adjust the trade-off between dynamic content recall and computational cost, enabling deployment configurations ranging from high-recall (aggressive rendering) to high-efficiency (minimal rendering).

The Dynamic Renderer implements adaptive timeout management, monitoring page loading progress and terminating rendering for pages that have not loaded meaningful content within a configurable timeout window. This prevents a small number of pathological pages with slow or runaway JavaScript from monopolising rendering resources. Rendering results are stored alongside the rendering probability score used to trigger rendering, enabling the dynamic content predictor to be retrained periodically on accumulated outcomes, continuously improving its accuracy and adapting to changes in web development practices.

7. Evaluation Methodology and Preliminary Results

7.1 Experimental Setup

The empirical evaluation of the AIC is conducted using three benchmark datasets. The Common Crawl Benchmark Subset comprises 50 million URLs drawn from the October 2022 Common Crawl snapshot, providing a realistic sample of the open web with diverse content types and quality levels. The Web Data Commons Hyperlink Graph (Meusel et al., 2015) provides a complete directed graph of hyperlinks for evaluating coverage and prioritisation performance, with PageRank-derived quality labels serving as ground truth for harvest rate computation. A custom Dynamic Content Test Set of 10,000 JavaScript-rendered URLs provides evaluation data for the dynamic content handling capabilities of the system.

Three baseline crawlers are used for comparison: a standard BFS crawler implementing FIFO URL scheduling; a PageRank-based crawler implementing the OPIC algorithm (Abiteboul et al., 2003); and Scrapy (Zyte, 2023) configured with default settings as a production framework baseline. All systems are evaluated on equivalent hardware configurations with equivalent total crawl budgets, ensuring fair comparison. Evaluation metrics include Harvest Rate (proportion of fetched pages above the 75th PageRank percentile), Redundancy Rate (proportion of fetched pages that are duplicates or near-duplicates of existing index content), Throughput (pages fetched per unit time), Coverage (proportion of benchmark pages fetched within budget), and Dynamic Content Recall (proportion of dynamically rendered content correctly extracted).

7.2 Results

Preliminary evaluation results demonstrate significant performance improvements for the AIC across all primary metrics. The AIC achieves a harvest rate of 0.71 after 1 million page fetches, compared to 0.52 for the OPIC-based PageRank crawler, 0.48 for Scrapy, and 0.44 for the BFS baseline — representing a 37% improvement over the best alternative. The redundancy rate is reduced from 0.34 (BFS), 0.28 (Scrapy), and 0.22 (OPIC) to 0.13 for the AIC, a 41% reduction relative to the best baseline. Throughput for the AIC is 12,400 pages per minute in the distributed 8-node configuration, compared to 9,800 for Scrapy and 8,200 for the BFS baseline, representing a 27% throughput improvement. Dynamic content recall is 0.88 for the AIC's selective rendering approach, compared to 0.61 for baseline HTML-only crawlers, demonstrating the effectiveness of the selective rendering pipeline. These results are summarised in Table 3.

Table 3

Comparative Performance Evaluation: AIC vs. Baseline Crawlers

Metric	BFS Baseline	OPIC (PageRank)	Scrapy Default	AIC (Proposed)	AIC Improvem ent
Harvest Rate (@ 1M pages)	0.44	0.52	0.48	0.71	+37% vs. best
Redundancy Rate	0.34	0.22	0.28	0.13	-41% vs. best

Metric	BFS Baseline	OPIC (PageRank)	Scrapy Default	AIC (Proposed)	AIC Improvement
Throughput (pages/min, 8-node)	8,200	7,900	9,800	12,400	+27% vs. best
Coverage (@ 10M budget)	0.61	0.67	0.65	0.79	+18% vs. best
Dynamic Content Recall	0.61	0.63	0.65	0.88	+35% vs. best
Scalability (nodes 1→8)	5.2x	4.8x	6.1x	7.6x	+25% vs. best

The scalability evaluation demonstrates near-linear throughput scaling for the AIC as nodes are added from 1 to 8 (achieving 7.6x throughput at 8 nodes versus theoretical maximum of 8x), compared to 6.1x for Scrapy and 5.2x for the BFS crawler. This demonstrates that the microservices architecture successfully achieves horizontal scalability without significant coordination overhead at the tested scale. Further evaluation at higher node counts is required to characterise the scaling behaviour as the system approaches the limits of the distributed coordination mechanisms.

8. Discussion

8.1 Interpretation of Findings

The evaluation results demonstrate that the AIC's multi-objective, adaptive prioritisation approach delivers substantial improvements over existing baselines across all measured dimensions of crawling quality. The 37% improvement in harvest rate represents a significant practical advance: for a search engine operating a crawl of 10 billion pages, this difference translates to hundreds of millions of additional high-quality pages indexed within the same resource budget. Similarly, the 41% reduction in redundancy rate represents a proportional reduction in wasted bandwidth and storage, enabling those resources to be redirected toward discovering novel content.

The dynamic content recall improvement from 0.65 (best baseline) to 0.88 (AIC) is particularly significant given the trajectory of web development toward increasingly JavaScript-heavy applications. As the proportion of dynamic content on the web continues to grow, the gap between crawlers capable of handling dynamic content and those that cannot will widen, making the selective rendering approach a critical differentiator. The AIC's ML-predicted selective rendering achieves near-comprehensive dynamic content recall while rendering only approximately 23% of fetched pages — demonstrating that intelligent selectivity enables high recall without prohibitive computational cost.

8.2 Limitations of the Proposed Model

Several important limitations of the AIC must be acknowledged. First, the preliminary evaluation results are based on benchmark datasets that, while large, are still several orders of

magnitude smaller than the full web. Whether the performance advantages demonstrated here persist at true web scale — where the diversity and complexity of the crawl environment greatly increase and where the machine learning models must generalise across an even wider distribution of web content — cannot be established from the current evaluation alone. Evaluation at larger scale, using production-grade infrastructure, is a critical direction for future work.

Second, the AIC's machine learning components require labelled training data for supervised learning and accumulated experience for reinforcement learning, introducing a cold start problem in new deployments. Strategies for effective initialisation of new AIC deployments — through transfer learning from existing models, through bootstrap training on public datasets, or through rapid adaptation from limited initial experience — have not been fully developed and represent an important area for further investigation.

Third, the evaluation is conducted under laboratory conditions using publicly available benchmark datasets, which may not fully capture the practical challenges of real-world web crawling including server unreliability, anti-crawling mechanisms, content personalisation, and geographically distributed content delivery. Robustness to these real-world conditions is an important property that requires evaluation in more realistic deployment settings.

8.3 Implications for Research and Practice

The findings of this study have several important implications for both research and practice. For the research community, the demonstration that an end-to-end AI-first architecture can substantially outperform systems that incorporate AI piecemeal strengthens the case for architectural innovation as a research priority, alongside the algorithmic innovation that has dominated the machine learning for web crawling literature. Future research should invest in developing standardised benchmark datasets and evaluation protocols that enable rigorous comparison of different intelligent crawling approaches, addressing the inconsistency in evaluation methodology identified in the literature review.

For practitioners operating web crawling infrastructure, the AIC provides both a set of design principles that can guide the development of more intelligent crawling systems and a concrete set of techniques — multi-objective prioritisation, adaptive near-duplicate detection, selective dynamic rendering, and microservices-based scaling — that can be applied incrementally to improve existing systems. The open-source availability of the AIC implementation (planned for publication alongside the full thesis) will enable practitioners to adopt and adapt these techniques for their specific use cases.

9. Conclusion

This paper has presented a comprehensive review of the state of the art in web crawling research and has proposed the Adaptive Intelligent Crawler (AIC), a novel crawling model and architecture that addresses the fundamental limitations identified in existing systems. The review identified four critical research gaps: the absence of a unified AI-first crawling architecture; insufficient treatment of scalability challenges; inadequate handling of dynamically rendered web content; and the lack of standardised evaluation protocols for comparing intelligent crawling systems. The AIC addresses each of these gaps through a principled integration of supervised

learning, reinforcement learning, and natural language processing within a scalable, microservices-based architecture.

Preliminary empirical evaluation on benchmark datasets from Common Crawl and Web Data Commons demonstrates that the AIC achieves substantial improvements over state-of-the-art baselines, including a 37% improvement in harvest rate, a 41% reduction in redundancy rate, a 27% improvement in throughput, and a 35% improvement in dynamic content recall. The near-linear scalability demonstrated in the distributed evaluation confirms the architectural soundness of the microservices design and establishes the AIC as a credible foundation for web-scale intelligent crawling.

The contributions of this work extend beyond the specific AIC system to include a systematic review of the web crawling literature, a rigorous analysis of the limitations of existing approaches, a multi-objective URL prioritisation framework that can be adapted to different crawling missions, and an evaluation methodology with standardised metrics and public benchmark datasets. Together, these contributions provide a foundation for continued progress in the design of intelligent web crawling systems equal to the demands of the contemporary and future web.

Future work will focus on large-scale validation of the AIC at production web scale, investigation of cold start strategies for new deployments, integration of additional AI capabilities including multilingual content understanding and structured data extraction, and empirical investigation of the AIC's behaviour in real-world deployment conditions. The long-term vision is a new generation of web crawling systems that are not merely fast and efficient, but genuinely intelligent — systems that understand the web, learn from their experience, and continuously improve their ability to serve the information needs of users, researchers, and AI systems.

References

- Abiteboul, S., Preda, M., & Cobena, G. (2003). Adaptive on-line page importance computation. *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, 280–290. <https://doi.org/10.1145/775152.775192>
- Aggarwal, C. C., Al-Garawi, F., & Yu, P. S. (2001). Intelligent crawling on the World Wide Web with arbitrary predicates. *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, 96–105. <https://doi.org/10.1145/371920.371935>
- Boldi, P., Codenotti, B., Santini, M., & Vigna, S. (2004). UbiCrawler: A scalable fully distributed web crawler. *Software: Practice and Experience*, 34(8), 711–726. <https://doi.org/10.1002/spe.587>
- Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7), 107–117. [https://doi.org/10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X)
- Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8–13), 1157–1166. [https://doi.org/10.1016/S0169-7552\(97\)00031-7](https://doi.org/10.1016/S0169-7552(97)00031-7)

10.48047/jocaaa.2024.33.08.434

- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.
- Chakrabarti, S., van den Berg, M., & Dom, B. (1999). Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11–16), 1623–1640. [https://doi.org/10.1016/S1389-1286\(99\)00052-3](https://doi.org/10.1016/S1389-1286(99)00052-3)
- Cho, J., & Garcia-Molina, H. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7), 161–172. [https://doi.org/10.1016/S0169-7552\(98\)00108-1](https://doi.org/10.1016/S0169-7552(98)00108-1)
- Cho, J., & Garcia-Molina, H. (2003). Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3), 256–290. <https://doi.org/10.1145/857166.857170>
- Common Crawl. (2023). Common crawl corpus. <https://commoncrawl.org>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2019)*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- DigitalPebble. (2023). StormCrawler: An open source SDK for building low-latency, scalable web crawlers on Apache Storm. <https://stormcrawler.net>
- Diligenti, M., Coetzee, F. M., Lawrence, S., Giles, C. L., & Gori, M. (2000). Focused crawling using context graphs. *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, 527–534.
- Exposto, J., Macedo, J., Carvalho, J., & Pina, A. (2007). Geographically distributed web crawling with DHT-based frontier management. *Proceedings of the International Conference on Web Intelligence*, 522–525. <https://doi.org/10.1109/WI.2007.111>
- Fang, Z., Zhang, Y., & Wang, J. (2022). Adaptive web crawling via deep reinforcement learning with graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(9), 4218–4232. <https://doi.org/10.1109/TKDE.2021.3055537>
- Fetterly, D., Manasse, M., Najork, M., & Wiener, J. L. (2004). A large-scale study of the evolution of web pages. *Software: Practice and Experience*, 34(2), 213–237. <https://doi.org/10.1002/spe.577>
- Gattani, A., Lamba, D. S., Garera, N., Tiwari, M., Chai, X., Das, S., Subramaniam, S., Rajaraman, A., Harinarayan, V., & Doan, A. (2013). Entity extraction, linking, classification, and tagging for social media: A Wikipedia-based approach. *Proceedings of the VLDB Endowment*, 6(11), 1126–1137. <https://doi.org/10.14778/2536222.2536237>
- Google. (2023). How Google crawls, indexes, and serves the web. Google Search Central. <https://developers.google.com/search/docs/crawling-indexing/overview-google-crawlers>

10.48047/jocaaa.2024.33.08.434

- Gouriten, G., Maniu, S., & Manolescu, I. (2014). Scalable, generic, and adaptive systems for focused crawling. *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, 35–44. <https://doi.org/10.1145/2631775.2631804>
- Gray, M. (1993). *World Wide Web Wanderer*. MIT Laboratory for Computer Science. <http://www.mit.edu/~mkgray/net>
- He, B., Patel, M., Zhang, Z., & Chang, K. C.-C. (2007). Accessing the deep web. *Communications of the ACM*, 50(5), 94–101. <https://doi.org/10.1145/1230819.1241670>
- Heydon, A., & Najork, M. (1999). Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4), 219–229. <https://doi.org/10.1023/A:1019213109274>
- Internet Archive. (2023). Heritrix web crawler. <https://github.com/internetarchive/heritrix3>
- Joachims, T. (2002). Learning to classify text using support vector machines. *The Springer International Series in Engineering and Computer Science*, 668. Springer. <https://doi.org/10.1007/978-1-4615-0907-3>
- Khare, R., Cutting, D., Sitaker, K., & Rifkin, A. (2004). Nutch: A flexible and scalable open-source web search engine. Technical Report. University of California, Irvine.
- Meusel, R., Vigna, S., Lehmberg, O., & Bizer, C. (2015). The graph structure in the web — Analyzed on different aggregation levels. *The Journal of Web Science*, 1(1), 33–47. <https://doi.org/10.1561/106.000000003>
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533. <https://doi.org/10.1038/nature14236>
- Najork, M., & Wiener, J. L. (2001). Breadth-first crawling yields high-quality pages. *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, 114–118. <https://doi.org/10.1145/371920.371965>
- Newman, S. (2015). *Building microservices: Designing fine-grained systems*. O'Reilly Media.
- Olston, C., & Najork, M. (2010). Web crawling. *Foundations and Trends in Information Retrieval*, 4(3), 175–246. <https://doi.org/10.1561/15000000017>
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web (Technical Report No. 1999-66). Stanford InfoLab. <http://ilpubs.stanford.edu:8090/422/>
- Pinkerton, B. (1994). Finding what people want: Experiences with the WebCrawler. *Proceedings of the 1st International World Wide Web Conference*, 17. CERN.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 1–67.

10.48047/jocaaa.2024.33.08.434

- Rennie, J., & McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. Proceedings of the 16th International Conference on Machine Learning (ICML '99), 335–343.
- Shkapenyuk, V., & Suel, T. (2002). Design and implementation of a high-performance distributed web crawler. Proceedings of the 18th International Conference on Data Engineering (ICDE '02), 357–368. <https://doi.org/10.1109/ICDE.2002.994750>
- Zyte. (2023). Scrapy: A fast and powerful scraping and web crawling framework. <https://scrapy.org>