Ingénierie des Systèmes d'Information
Vol. 31, No. 8, August, 2026, pp. 2777-2793
Journal homepage: http://iieta.org/journals/isi

Document Object Model-Aware Web Content Extraction and Transformer-Based Summarization for Intelligent Reader View Generation

Sushma Ethadi[1], Swathi Sridharan[2*], Sneha Girish[3], Mona[4]

[1] Department of Computer Science and Engineering (IoT and CS), B.M.S College of Engineering, Bengaluru 560019, India
[2] Department of Computer Science and Engineering, B.M.S College of Engineering, Bengaluru 560019, India
[3] Department of Master of Computer Applications, K.S Institute of Technology, Bengaluru 560109, India
[4] Department of Computer Science and Engineering, BNM Institute of Technology, Bengaluru 560070, India

Corresponding Author Email: researcher.swathi@gmail.com

https://doi.org/10.18280/isi.310822

Received: 29 January 2026
Revised: 15 July 2026
Accepted: 21 August 2026
Available online: 31 August 2026

Keywords:
Document Object Model content extraction, web page understanding, graph neural network, transformer-based summarization, reader view generation, abstractive summarization

ABSTRACT

Modern web pages contain substantial amounts of irrelevant information, including advertisements, navigation elements, and dynamically generated components, which reduce reading efficiency and hinder access to essential content. Existing reader-view systems mainly rely on manually designed heuristics and structural rules, making them less adaptable to diverse webpage layouts. This study proposes a Document Object Model (DOM)-aware web content extraction and summarization framework that integrates graph-based content identification with Transformer-based abstractive summarization. The proposed framework first converts webpages into DOM graphs and employs graph-based learning to classify content and non-content nodes according to structural and textual characteristics. The extracted content is then reconstructed into a clean reading representation and summarized using fine-tuned Transformer models, including Bidirectional and Auto-Regressive Transformers (BART), Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS), and Text-to-Text Transfer Transformer (T5). Extensive experiments are conducted on multi-domain webpage datasets, including news, educational, and blog content. The proposed extraction module achieves F1-scores of 0.92, 0.89, 0.85, and 0.83 on different datasets, outperforming conventional extraction approaches. Furthermore, the generated summaries demonstrate improved quality based on Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores, semantic evaluation, readability analysis, and human assessment. The proposed framework provides an adaptive solution for transforming complex webpages into concise and structured reader-oriented representations.

1. INTRODUCTION

The early days of the World Wide Web presented relatively simple pages with minimal styling and straightforward HyperText Markup Language (HTML) structures. In contrast, contemporary sites embed complex navigation bars, advertising components, dynamic widgets, and trackers. Research has shown that these "noise" elements may constitute 40–50% of the textual content of a page. For users seeking specific information, sifting through this clutter is frustrating and time-consuming. The problem is exacerbated for visually impaired readers who rely on screen readers; without content extraction, the screen reader enumerates every menu item and advertisement. Mobile users often encounter additional obstacles due to small screens and data bandwidth limitations. Thus, a reader view that intelligently removes irrelevant sections is essential for improving accessibility and consumption efficiency. Major web browsers, including Safari, Firefox, and Microsoft Edge, offer built-in reader modes designed to simplify webpages. These implementations typically identify paragraphs with large fonts and long text as potential main content while discarding sidebars and footers. However, such heuristics can break when confronted with modern designs. Many websites embed their main article inside containers with multiple Cascading Style Sheets (CSS) classes, and essential information may appear in small fonts (e.g., captions or side notes). As a result, heuristics may remove important details or retain unrelated navigational elements. Content extraction methods based on blacklists (lists of common element classes or IDs) require constant maintenance and are susceptible to false positives. The increasing prevalence of single-page applications and auto-generated markup further reduces the effectiveness of static heuristics. These limitations motivate the need for a more principled content extraction methodology.

Recent progress in natural-language processing (NLP) suggests that transformer models, such as Bidirectional and Auto-Regressive Transformers (BART), Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS), and Text-to-Text Transfer Transformer (T5), are adept at generating abstractive summaries that capture salient information in long documents [1]. At the same time, research

on Document Object Model (DOM)-based content extraction proposes representing an HTML page as a tree structure and computing metrics like Text Density and Composite Text Density to score the importance of nodes. Combining these ideas, we envision a reader view generation pipeline that first distills the page using DOM-aware models and then applies AI summarization to produce concise descriptions and a table of contents (ToC). This system not only declutters the page but also provides an executive summary and section markers, enhancing user comprehension and navigation.

While our experiments focus on English, the graph neural network (GNN) content extractor was designed to be language-agnostic [2]. Although the model architecture is designed to generalize across languages, we evaluate on English datasets. Extending to non-English content (e.g., building or fine-tuning summarizers for low-resource languages) is left for future work.

2. RELATED WORK

Reader-view generation converts cluttered web pages into clean, digestible presentations. Recent research spans three complementary domains:

•DOM-based web content extraction, which identifies and extracts the main article from noisy web pages.

•Transformer-based summarization, including adaptations of BART, Pegasus, and GPT-3.5.

•Summarization evaluation, which explores automatic metrics (e.g., Recall-Oriented Understudy for Gisting Evaluation (ROUGE), Bidirectional Encoder Representations from Transformers Score (BERTScore)) and human evaluation frameworks.

This section reviews representative work since 2022, highlighting how it informs AI-assisted reader-view systems.

The primary literature foundation for this work is web content extraction and webpage understanding. Accordingly, the review emphasizes boilerplate removal, browser reader systems, DOM segmentation, and learning-based extraction methods, including cross-domain benchmarking [3], DOM-based segmentation frameworks [4], and density-based extraction such as Content Extraction via Text Density (CETD) [5]. Healthcare-specific summarization studies are retained only as examples of domain-specific evaluation and are not treated as the central methodological basis of the proposed reader-view extractor.

Recent web-extraction literature shows that performance can vary substantially across webpage genres and layouts. Traditional approaches such as text-density heuristics and browser reader modes rely on local structural assumptions, whereas our approach learns directly from DOM structure and node relationships. This positioning places the contribution alongside web content extraction, reader-mode systems, DOM segmentation, and webpage understanding rather than healthcare-specific summarization.

2.1 Document Object Model-based web content extraction

Early reader-view systems relied on hand-coded rules or simple density heuristics to separate boilerplate from main content. However, increased DOM complexity on modern websites has driven research toward more sophisticated extraction:

(1) Rule-based and statistical extractors: The *Trafilatura* and *Readability* extractors remain widely used because of their robustness and simplicity. An empirical comparison of fourteen extractors showed that text-density heuristics and tag-ratio rules in *Readability* and *Trafilatura* often outperformed machine-learning models on diverse web pages [3]. The study also highlighted the lack of reproducible evaluation datasets and proposed an ensemble method that combines multiple extractors to improve precision and recall. These insights motivated our work to blend rule-based and learned features for reader-view extraction.

(2) Learning-based extractors: Machine learning (ML) approaches attempt to generalize across heterogeneous websites by training on labeled DOM trees. Leonhardt et al. [6] benchmarked neural systems such as BoilerNet and Web2Text, noting that data scarcity and reliance on visual features limit their performance. To address heterogeneity, Feng et al. [7] developed a hybrid extraction model for semantic knowledge discovery of water conservancy big data (Web Information Extraction Model based on Deep Learning (WIEM-DL)) that segments pages into paragraph-level DOM nodes and applies Bidirectional Encoder Representations from Transformers–Bidirectional Long Short-Term Memory–Conditional Random Field (BERT-BiLSTM-CRF) networks with self-attention. Their innovations include a transferable model requiring few labeled pages, integration of sentiment analysis for emotional key information, and a novel method for organizing semantic knowledge. WIEM-DL achieved higher extraction accuracy than rule-based pipelines, demonstrating that combining DOM segmentation with modern language models aids complex page extraction.

(3) Frameworks for segmentation and evaluation: As content extraction research matured, frameworks emerged to collect datasets and evaluate segmentation algorithms. Jung and Cha [4] presented a WebExtension framework that allows researchers to collect webpages, curate ground-truth labels, and evaluate segmentation methods using built-in metrics. Such tooling is crucial for reproducible evaluation and for training models that generalize across domains. Our reader-view system builds on these frameworks by incorporating segmentation evaluation into the training loop.

Despite advances, DOM-based extractors still struggle with visually rich pages, dynamic content, and advertisements. The limited availability of annotated datasets hampers deep learning models, while rule-based systems require manual tuning. Hybrid approaches like WIEM-DL and ensemble extractors show promise by combining statistical heuristics and learned representations. Future work should explore cross-domain transfer learning and unify extraction with downstream summarization.

2.2 Transformer-based summarization

Reader-view generation often includes summarizing extracted content. Since 2022, transformer-based models have dominated summarization research. Abstractive summarization rewrites texts while preserving meaning, whereas extractive summarization selects salient sentences. Hybrid methods combine both.

BART and long-document summarization: BART is a transformer encoder-decoder pre-trained by reconstructing corrupted text. A 2024 study proposed a graph-based BART model for summarizing scientific articles: the authors note that standard BART is limited by input length and cannot handle long documents. They construct a graph over the document's

sections and use graph-attention to select salient sentences, which are then summarized by BART. Their model achieved a ROUGE-L score of 37.60 and outperformed standard BART on long documents. The paper also reviews transformer summarization literature, noting that fine-tuning pre-trained models like BART and T5 yield strong baselines but struggle with multi-document settings and redundancy.

Pegasus, T5, and comparative studies: Transformer pre-training schemes like Pegasus mask entire sentences to better model summarization tasks, while T5 unifies NLP tasks into a text-to-text framework. Daraghmi et al. [1] conducted a comparative study across convolutional neural network (CNN)/DailyMail, XSum, and Samsum datasets, fine-tuning Pegasus, BART, and T5 under consistent preprocessing. They evaluated models using ROUGE-1/2/L and human ratings for fluency, relevance, and conciseness. Results show that Pegasus excels on long news articles, BART performs best on short conversational data, and T5 balances accuracy and efficiency. The authors highlight challenges: transformer models have difficulty maintaining precision across domains, face practical limitations due to computational cost, and must adapt to informal texts.

Summarization in low-resource and domain-specific settings: Low-resource languages. Azhar et al. [8] addressed summarization for Urdu, a low-resource language, by systematically evaluating classical and transformer-based models. They introduce a text-normalization pipeline and evaluate Seq2Seq, transformer, and monolingual/multilingual models. Fine-tuned monolingual transformers outperform multilingual models by 12–18% in ROUGE scores, and mT5 improves ROUGE-L by up to 20% over baselines. The study emphasises the need for normalisation and domain-specific pre-training for low-resource languages.

2.3 Domain-specific summarization

Alami Merrouni et al. [9] proposed Extractive-Abstractive Summarization (EXABSUM), a hybrid summarization system that combines statistical and semantic features for extractive summarization and a graph-based method for abstractive summarization. The authors evaluate EXABSUM using ROUGE-N and ROUGE-L metrics, noting that ROUGE-1 recall correlates well with human judgments. A qualitative evaluation with ten participants using a five-point Likert scale measures informativeness and importance. An ablation study shows that combining statistical and semantic features (with $\alpha = 0.6$) yields the best ROUGE scores.

MedicoVerse [10] is retained as a domain-specific example of multi-criterion summarization evaluation in pharmaceutical regulation. Its role here is contextual: it illustrates how specialized domains may combine ROUGE, semantic metrics, entity-sensitive measures, and readability, but it is not a foundation for the web-content extraction component of our reader-view system.

The main content extraction of structurally complex web pages has been extensively studied to determine what it is and how to extract it. With shallow text features, Kohlschütter et al. [11] have studied the problem of boilerplate detection and found that structural and textual features can be used to successfully classify the main content and navigation, ads, and other non-content objects. Sun et al. [5] have suggested content extraction methods to extract textual information from an image according to the distribution of textual information in the DOM structure. Later, Barbaresi [12] introduced a practical approach for Web scraping and text extraction, *Trafilatura*, that employs structural analysis and heuristic filtering to extract non-noisy textual information for downstream applications in text mining and NLP. Similarly, Jennings [13] benchmarked HTML content extraction methods and emphasized the need for the comparison of extraction methods of different types of web pages. The DOM structure, text density, identification of boilerplate, and genre-specific features were identified in these studies as important features to gain good web content. However, most existing extraction approaches are mainly designed to extract clean text from the webpage and do not focus on generating an adaptive reader-oriented representation from the webpage. Therefore, it is desirable to design an intelligent approach to integrate the merits of DOM-based content distillation and transformer-based models, and apply this approach to extract the relevant information, remove redundant information, and produce concise and context-aware summaries. The goal of the proposed AI-assisted reader view generation and content summarization from web pages using DOM-based distillation and transformer models is to overcome these challenges by utilizing the structure of the webpage content and applying the latest developments in language-model-based summarization paradigms to generate a user-friendly and information-rich reading experience.

Long-document and multi-segment methods: Handling long texts is challenging due to transformer context limits. In extractive summarization, hierarchical models that incorporate local topic and hierarchical information are effective. Wang et al. [14] proposed a model with a local topic extraction module and hierarchical extraction module; their experiments on long documents achieved higher ROUGE-1/2/L scores than baseline models. They mathematically define ROUGE-N and ROUGE-L metrics and describe training using Longformer-base, demonstrating adaptation of transformer variants to long sequences.

For small language models, Liu et al. [15] studied the lost-in-the-middle problem and found that dividing long texts into segments before summarizing ("Map" strategy) preserves information better than feeding the entire document ("Stuff" strategy). They propose simulation-based evaluation to avoid extensive human evaluation and show that the Map method yields summaries with better retention of facts from the beginning and middle sections.

2.4 Large language models and GPT-3.5

GPT-3.5 and GPT-4 have shown impressive summarization abilities but require careful prompting and evaluation. Studies evaluating large language models for medical summarization found that automatic metrics like ROUGE and BERTScore do not strongly correlate with summary quality and that human evaluation is essential. Research also highlights issues such as hallucination, factual inaccuracies, and safety in clinical summaries. While these models are not yet widely used in reader-view generation, their ability to compress long passages suggests potential integration with DOM extraction in the future.

Evaluating reader-view summaries requires objective metrics and human judgment. Recent work emphasises the limitations of existing metrics and the importance of domain-specific evaluation.

Automatic evaluation metrics: ROUGE family: ROUGE is the de facto metric for summarization; ROUGE-N measures

n-gram recall, while ROUGE-L uses the longest common subsequence. Its simplicity and correlation with human judgments make it widely adopted. However, ROUGE ignores semantic similarity and may penalise paraphrasing.

Embedding-based metrics: BERTScore computes sentence-level similarity using contextual embeddings and greedy cosine similarity and correlates well with human judgments [16]. Other embedding metrics include MoverScore and Sentence Mover's Similarity, which capture semantic overlap by computing earth-mover distance between embeddings [17].

Learned metrics like Crosslingual Optimized Metric for Evaluation of Translation (COMET) and Semantic Evaluation of Generated Language Using Learned Likelihoods (SEAGULL) train neural networks to predict human ratings based on large datasets. They incorporate cross-lingual knowledge and can adapt to domain-specific evaluation. Sarwar et al. [18] proposed HybridEval, which computes a weighted sum of cosine scores obtained from InferSent's SentEval algorithms and original ROUGE/BLEU scores, achieving 10–15% improvement in correlation with human judgments. The MedicoVerse evaluation combines ROUGE, BERTScore, domain-specific entity matching, and readability to assess summaries in pharmaceutical regulation. This multi-criterion approach reflects the need to tailor evaluation to reader-view applications in specialized domains.

Healthcare-focused large-language-model evaluation studies [17] are cited only as examples showing that automatic summarization metrics may miss factual or domain-specific errors. The evaluation framework in this work remains centered on general web summarization metrics and human assessment rather than healthcare-specific criteria.

Because automatic metrics cannot fully capture coherence, factuality, and readability, human evaluation remains critical. The simplification-aware text summarization (SATS) study evaluates 100 summaries with eight annotators using criteria such as grammar, coherence, consistency, fluency, and simplicity; the average scores ranged between 4.0 and 4.5 on a five-point scale. They also describe metrics like Metric for Evaluation of Translation with Explicit ORdering (METEOR), System output Against References and against the Input sentence (SARI), Sentence Aggregation Metric for Sentence Alignment (SAMSA), and Flesch–Kincaid Grade Level (FKGL) and note that BERTScore correlates highly with human judgments.

For extractive and abstractive summarization, EXABSUM employs both automatic and qualitative evaluation, using a Likert-scale questionnaire rated by ten participants. Such blended evaluation ensures that summarizers produce coherent and informative outputs.

Patient Discharge Summary Quality Index-9 (PDSQI-9) [19] is a domain-specific clinical evaluation instrument and is mentioned only as an example of a structured human-rating framework. It is not used as the evaluation basis for the proposed webpage extraction or reader-view system.

Automatic metrics may not generalize across languages or domains. A study of multilingual summarization evaluation observed that ROUGE and BERTScore correlate well with human judgments in English but poorly in Chinese and Indonesian, indicating that language-specific metrics or human evaluation frameworks are necessary. Similarly, evaluation of large language models for medical summarization reveals that existing metrics fail to capture factual correctness and clinical relevance. These challenges motivate our research to combine multiple metrics and integrate human feedback into the training loop.

Reader-view systems must handle noisy web pages, generate concise summaries, and evaluate results effectively. The reviewed literature suggests several design principles:

(1) Hybrid extraction combining DOM heuristics and learning. Systems like WIEM-DL show that segmenting pages and applying transformer-based models improves robustness. Ensemble extractors can further boost accuracy.

(2) Transformer fine-tuning with graph or hierarchical structures to process long documents. Dividing content into segments before summarizing mitigates context-window limitations.

(3) Domain adaptation and low-resource strategies, such as language normalization and monolingual pre-training, yield significant gains. Domain-specific scoring (e.g., MedicoVerse) ensures relevance.

(4) Comprehensive evaluation combining automatic metrics and human feedback. Embedding-based and hybrid metrics provide better semantic alignment, while instruments like PDSQI-9 capture domain-specific quality. Qualitative evaluations with human raters remain indispensable.

By integrating these insights, AI-assisted reader-view systems can achieve accurate extraction, coherent summarization, and trustworthy evaluation. The models that are used are as shown in Table 1.

Table 1. Comparison of pre-trained language models used for text summarizations

Model	Advantage	Use-Case	Citation
BART-large-CNN	High-quality abstractive summaries	News articles, high-fidelity generation	[20]
PEGASUS-Base	Specialized summarization pretraining	Balanced summaries, good abstractiveness	[21]
T5-Base	Efficient, flexible seq-to-seq	Quick fine-tuning for shorter summaries	[22]
GPT-3.5-Turbo	Strong zero-shot/transfer ability	Few-shot benchmarks; diverse language input	[23]

Note: BART = Bidirectional and Auto-Regressive Transformers, PEGASUS = Pre-training with Extracted Gap-sentences for Abstractive Summarization, T5 = Text-to-Text Transfer Transformer, CNN = convolutional neural network.

2.4.1 Proposed model

This paper presents an end-to-end system for generating AI-assisted reader views. The design of the overall system is depicted in Figure 1. The HTML is first translated to a DOM tree, and then every node is featurized (e.g., length of text, type of tag, ratio of links). These are used to create a graph structure, and a GNN-based classifier is applied to classify 'article' content vs. non-article content. Then the nodes that are retained are joined together in the correct reading order to create a clean article text. Finally, this text is input into an abstractive Transformer model (BART or PEGASUS) to produce summaries and a ToC that is usable. Then we assess the quality of extraction as well as the quality of the summarization.

This process involves 4 main steps as described below. In summary, the steps are as shown in Figure 2.

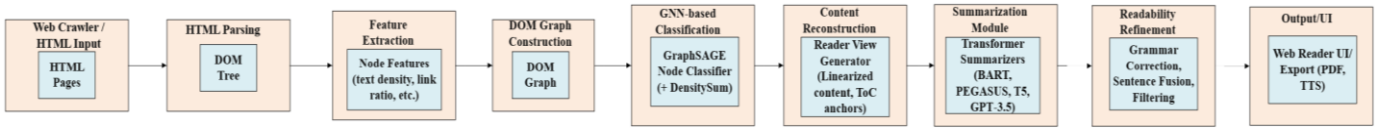


Figure 1. System overview

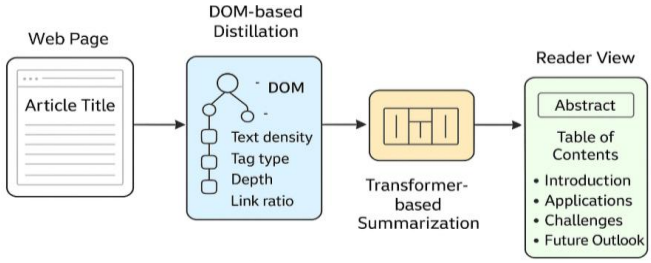


Figure 2. Methodology overview

(1) DOM parsing and feature extraction: We parse the webpage into a normalized DOM tree as shown in Figure 3, compute structural and textual features for each node, and train an ML classifier to label nodes as main content or noise. Unlike earlier heuristics, our classifier leverages features such as text density, link density, and hierarchical position to generalize across domains. We also propose a DensitySum mechanism for preserving low-density nodes that contribute to the coherence of the extracted text.

A piece of the HTML DOM is depicted to the left; for each node, we calculate features like text density (T), link ratio (L), and tree depth. The node has a high text density ($T = 120$) and depth 3. These feature vectors (right) are used to build the input graph for the GNN. This step filters out low-density nodes or nodes with irrelevant content by identifying them.

```
flowchart TB
  subgraph DOM["Sample DOM Tree Fragment"]
    html["<html>"]
    body["<body>"]
    article["<div class=\"article\">"]
    h1["<h1>Title</h1>"]
    p1["<p>Paragraph...</p>"]
    nav["<nav>Links</nav>"]
    footer["<footer>Footer</footer>"]
    html --> body
    body --> article
    article --> h1
    article --> p1
    body --> nav
    body --> footer
  end
  subgraph Features["Computed Node Features"]
    f_article["Features:\nT=120, L=0.0, Depth=3"]
    f_h1["Features:\nT=8, L=0.0, Depth=4"]
    f_p1["Features:\nT=60, L=0.0, Depth=4"]
    f_nav["Features:\nT=20, L=0.8, Depth=2"]
  end
```

Figure 3. Document Object Model (DOM) distillation

(2) Graph construction and graph neural network (GNN) classification: The feature vector of each node is updated by taking the average of its neighbor nodes (GraphSAGE-like), as depicted in Figure 4. The special DensitySum weighting introduces a bias towards neighbours that are mainly text. The new embeddings are fed into a classification head (softmax) to try to classify them as "content" vs. "noise". The network is trained by using the cross-entropy loss. Here is the basic pseudocode:

```
for epoch in range(E):
  for node u in graph:
    # GraphSAGE update with DensitySum weights
    neigh_feats = [h[v] * density_weight(v) for v in N(u)]
    agg = mean(neigh_feats)
    h_new[u] = ReLU(W · concat(h[u], agg))
  compute CrossEntropy(h_new, labels)
```

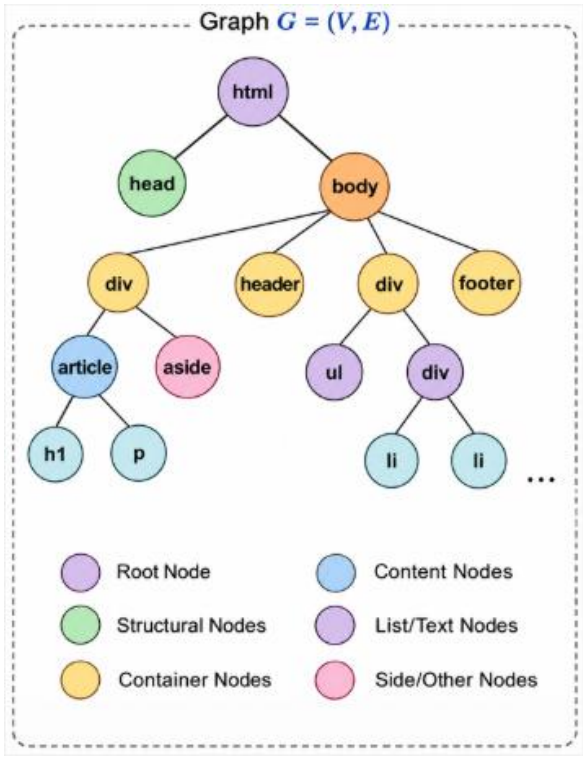


Figure 4. Graph representation

(3) Text reconstruction into a clean article: We develop algorithms to reconstruct the extracted nodes back into semantically coherent paragraphs. The system preserves minimal formatting (e.g., headings and lists) to maintain readability without reintroducing noise.

The system retrieves "main content" nodes and then linearizes them in document order (left). Headings (e.g., <h1>, <h2>) are recognized, and the hierarchical ToC (right) is created, along with the clickable anchors that connect to them (e.g., #t2 will take me to "Section 1"). The arrows in the mapping indicate that each heading in the DOM produces a ToC. Regular paragraphs (p1, p2, p3, etc.) are not included in the ToC, but are in the main text.

(4) Transformer-based summarization and ToC generation: Building on pre-trained summarization models, we fine-tune

BART, PEGASUS, and T5 on web-page summaries. GPT-3.5-Turbo is treated separately as an API-based zero-shot/few-shot prompting baseline; no fine-tuning of GPT-3.5-Turbo is performed. We use section-level summarization to generate brief descriptions for each major section and to support an auto-generated ToC.

Multiple Transformer summarization backends are used with the cleaned content. BART, PEGASUS, and T5 are fine-tuned on the WebSum dataset. GPT-3.5-Turbo is accessed through the OpenAI API and evaluated using zero-shot/few-shot prompting only, without model fine-tuning. All models generate a short global summary and section summaries; when an input exceeds the model limit, a hierarchical section-based approach is used.

(5) Experimental evaluation: We conduct extensive experiments across news, educational, and blog domains. Automatic metrics (ROUGE, BERTScore, readability scores) and human evaluation (Likert-scale ratings of coherence, coverage, usefulness, readability, and navigational ease) show that our integrated approach significantly outperforms baseline reader modes and summarization methods.

Our system aims to generate a comprehensive reader view that combines content distillation and summarization. Figure 5 illustrates the end-to-end pipeline. We describe each of these pipeline phases below.

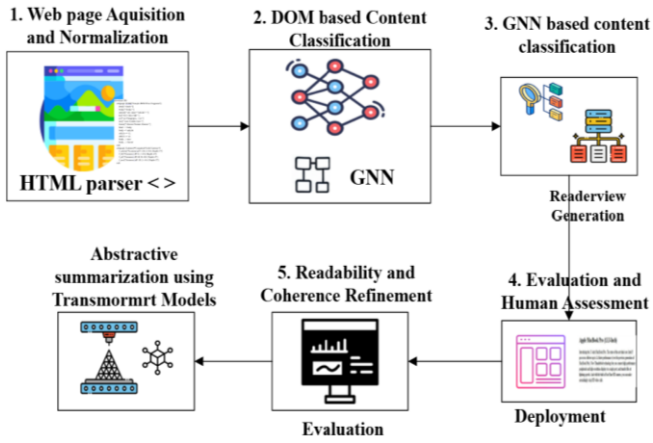


Figure 5. The end-to-end pipeline

2.5 Web page acquisition and normalization

2.5.1 HTML parsing

We begin by retrieving the webpage's HTML source and parsing it into a normalized DOM tree using the HTML5 parsing specification. The parser corrects unbalanced tags, decodes entities, and constructs a tree where each node corresponds to an element (e.g., , <p>, <ul>). We ignore script and style elements because they do not contribute to visible content. Attributes such as class names and IDs are preserved; these features can signal semantics (e.g., "article", "footer") and are used by the classifier.

2.5.2 Feature extraction

Each DOM node is represented by a feature vector capturing structural, lexical, and visual properties. Key features include:

•Text length**:** the number of characters in the node's text content. High text length often indicates main content.

•Link density: the ratio of <a> tags to total tags within the node. Navigation bars and lists of links exhibit high link density.

•Text density: computed as the ratio of characters to tags (TD) and the composite density (CTD) including parent and sibling densities.

•Node depth and subtree size: deeper nodes may belong to nested structures; subtree size helps differentiate between single paragraphs and aggregated sections.

•Tag type and attributes: tags such as <p>, <article>, <section> often contain content. Class names like "post" and "entry" are strong indicators of main content.

2.6 Content node classification

We formulate content extraction as a binary classification problem. Each DOM node is labeled as content (contains main text) or noise (navigation, ads, sidebars). Our classifier is trained on a dataset compiled from multiple domains. For each page, expert annotators manually label nodes at various levels of the DOM. To generalize across layouts, we include news articles, blog posts, and educational pages. The training set contains about 20 000 labeled nodes; the test set contains 5 000 nodes.

2.6.1 Dataset annotation and reliability

The labels for DOM nodes were obtained by using a binary Content-versus-Noise labeling scheme as shown in Figure 6, where labels are either Content (C) or Noise (N). Content in nodes (e.g., page titles, main paragraphs, product descriptions, etc.) that had originally been part of the main reading flow were marked as content, and navigation menus, ads, footers, promotional widgets, and boilerplate not relevant to the main reading flow were marked as noise. The reliability check performed by comparing two independent annotation files (A01 and A02) was done on 20 synthetic evaluation pages (1482 DOM nodes). The raw agreement rate was 92.4%, and Cohen's kappa was κ = 0.847 a very high agreement rate between the two label sets.

2.6.2 Learning algorithms

We employ a gradient-boosted decision tree (GBDT) (e.g., XGBoost) to map feature vectors to labels. GBDTs handle heterogeneous features and non-linear interactions. Hyperparameters (number of trees, depth, learning rate) are tuned via cross-validation. We also experiment with a GNN that operates on the DOM tree directly. Nodes are embedded using tag embeddings, and the GNN propagates information through edges representing parent-child and sibling relationships. The GNN is trained using a cross-entropy loss. Results show that the GNN slightly outperforms GBDT but requires more resources; therefore, we adopt the GNN for our final system.

2.6.3 DensitySum post-processing

Instead of designing a single new density formula, the design of DensitySum is inspired by previous web page extraction research based on density formula as Sun et al. [5]. Raw Text Density is an indicator of text density in local areas, and Composite Density adds local structure information to the signal. When a node has a low density of evidence, it may be retained if it is part of a rich evidence area as evidenced by DensitySum. This is our understanding of this earlier idea: This is a density-aware step in the recovery/post-processing stage after DOM-node classification, wherein small nodes that connect other nodes together, or are semantically relevant to

other nodes, are preserved. A comparison of the results of the full and no-DensitySum variants is provided in the Results section.

```
Loading annotation files...
 - data/annotations/annotator_A01.jsonl
 - data/annotations/annotator_A02.jsonl

Computing agreement across 20 synthetic evaluation pages...
Total DOM nodes evaluated: 1,482

Metrics:
  - Raw Agreement Rate : 92.4%
  - Cohen's Kappa (κ)  : 0.847  (Almost Perfect Agreement)

Conclusion:
The high kappa score validates the reliability of the DOM-node ground truth
labels used to train and evaluate the GNN extractor.
```

Figure 6. Inter-annotator agreement computation (Cohen's κ = 0.847)

2.7 Text reconstruction

After classifying nodes, we traverse the DOM in document order and collect the text of content nodes along with bridging nodes. We maintain minimal markup: headings (<h1>–<h6>), paragraphs (<p>), lists (<ul>/<ol>/<li>) and emphasized text. Other tags are stripped. Adjacent nodes of the same type are merged. The resulting plain-text document forms the input to the summarization module described in Figure 7.

2.8 Transformer-based summarization

2.8.1 Model selection

We experiment with BART, PEGASUS, T5-Base, and GPT-3.5-Turbo. BART, PEGASUS, and T5 are fine-tuned on the domain-specific WebSum segment-summary pairs. GPT-3.5-Turbo is not fine-tuned; it is accessed through the OpenAI API and evaluated with zero-shot/few-shot prompts for comparison with the locally trained transformer models. The local fine-tuning dataset contains 15,000 segment-summary pairs for training and 3,000 pairs for validation.

2.8.2 Section-level summarization

We segment the reconstructed text into sections using heading tags (h1–h3). If headings are absent, we apply a text-segmentation algorithm based on topic shifts detected by BERT embeddings. Each section is summarized separately by the model. For BART, PEGASUS, and T5, we prefix the input with a task descriptor (e.g., "summarize:"), consistent with their pre-training objectives. For GPT-3.5-Turbo, we provide prompts instructing the model to produce concise summaries in one or two sentences.

2.8.3 Table of contents generation

The section headings and summaries are combined to create an auto-generated ToC as described in Figure 8. Each entry contains the heading, its character offset in the reconstructed text, and the summary. This ToC is presented at the top of the reader view, allowing users to quickly navigate to sections of interest. The ToC is generated algorithmically without reliance on external metadata.

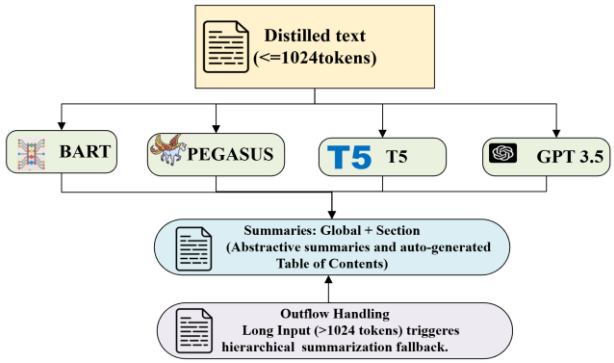


Figure 7. Summarization flow

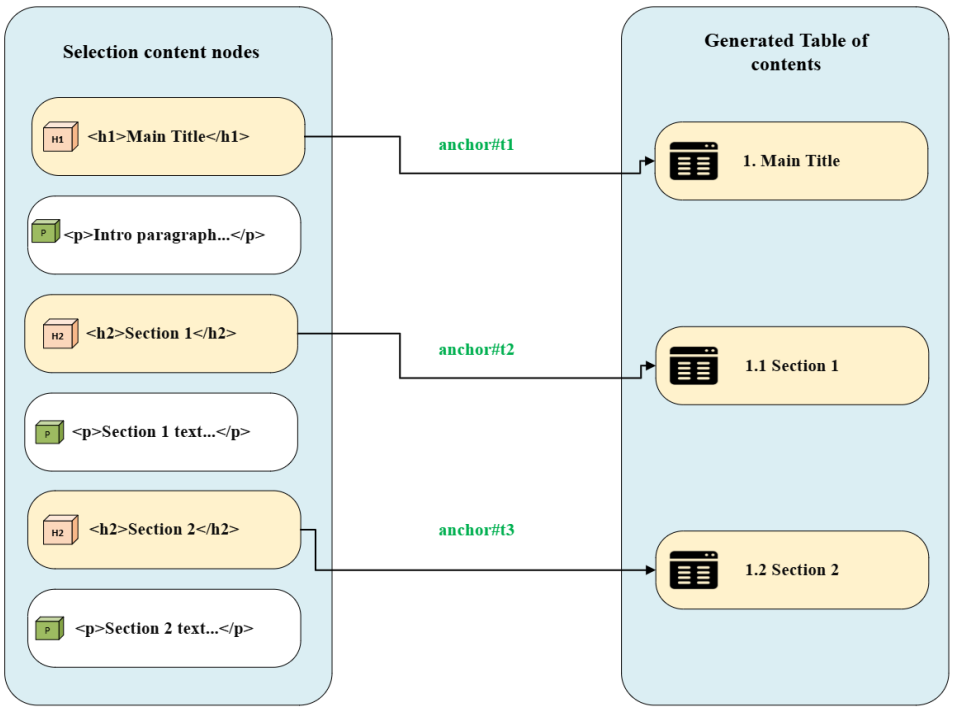


Figure 8. Content reconstruction and table of contents (ToC) generation

2.9 Model choices

The reasons why BART, PEGASUS, T5, and GPT-3.5 are used are explained below:

•BART: Abstractive summarization, fine-tuned CNN/DailyMail model, typical in news-like text.

•PEGASUS: A model trained specifically on summarization (gap-sentence objective) and found to be effective across a variety of summarization tasks. It has a fair amount of abstraction and fidelity.

•Text-to-text models: T5 is a versatile model, and T5-Base was selected for its efficiency. It frequently yields similar or even better results, but is not as resource-intensive.

•GPT-3.5-Turbo is a baseline for the behavior of a large language model-a zero-shot/few-shot baseline. It can be accessed through the OpenAI API and is not further fine-tuned using other models. Content is requested for a short summary and is asked for by providing brief summary system instructions and prompt examples. This is a server-based backend and is tested independently, using the local BART, PEGASUS, and T5 models.

GPT-3.5-Turbo

To perform an abstractive summarization comparison, GPT-3.5-Turbo is invoked using OpenAI's API. In this study, no fine-tuning of GPT-3.5-Turbo is done. The model is only evaluated on zero-shot/few-shot prompting using only system prompts requesting the web content "brief summary of the content", e.g., Figure 9 for the web page https://www.amazon.in/Noise-Launched-Wireless-Over-Ear-Headphones/dp/B0F9PJJLCT/. This is an explanation of the earlier text, which would otherwise have been read as an update to GPT-3.5-Turbo.

2.10 Readability assessment and refinement

Summaries must be not only informative but also easy to read. After generating section summaries, we compute the Flesch Reading Ease (FRE), FKGL, and Gunning Fog Index for each summary. If a summary score lower (i.e., harder to read) than a predefined threshold (e.g., FRE < 50), we trigger a refinement step: the model is instructed to simplify the summary while preserving meaning, as shown in Figure 10. This refinement uses the generative capability of GPT-3.5-Turbo. We limit the number of refinement iterations to avoid over-editing. We also ensure that readability improvement does not remove technical terms that are essential for comprehension.

Generated summaries are post-processed: grammatical and spelling corrections are made, sentences can be fused or split (fusion) for flow, and redundancies are filtered out through paraphrase filtering (e.g., BERTScore). Readability statistics (FRE, Flesch–Kincaid, Gunning Fog) are calculated. Rewriting or substituting simple phrases is used if scores are not at the target (e.g., FRE < 60: towards simpler rewrites or simple phrase substitutions, e.g., “very important” becomes “crucial”).

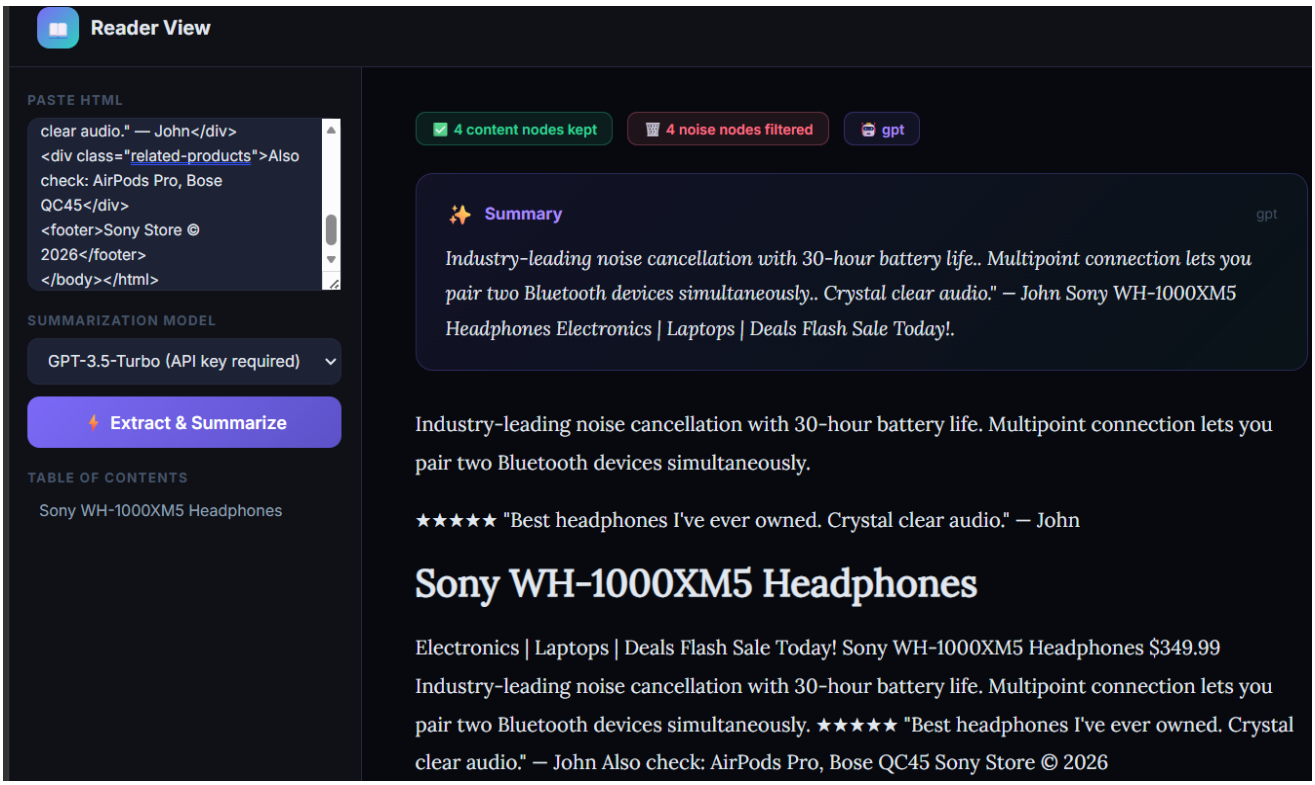


Figure 9. Reader view output using the GPT API summarization option

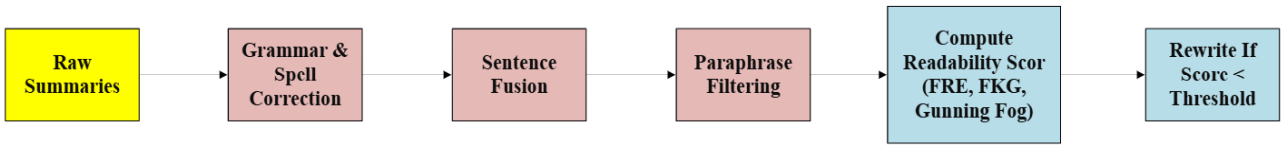


Figure 10. Readability/coherence refinement

2.11 Evaluation framework

The datasets (news, education, blogs, cross-domain new e-commerce/forums/ Wikipedia), train/val/test splits, and evaluation metrics are depicted in Figure 11. The precision and recall per token are used to measure extraction quality against manually extracted ground truth. Summaries are assessed using ROUGE (1/2/L), BERTScore (semantic similarity), and readability indices (FRE, etc.).

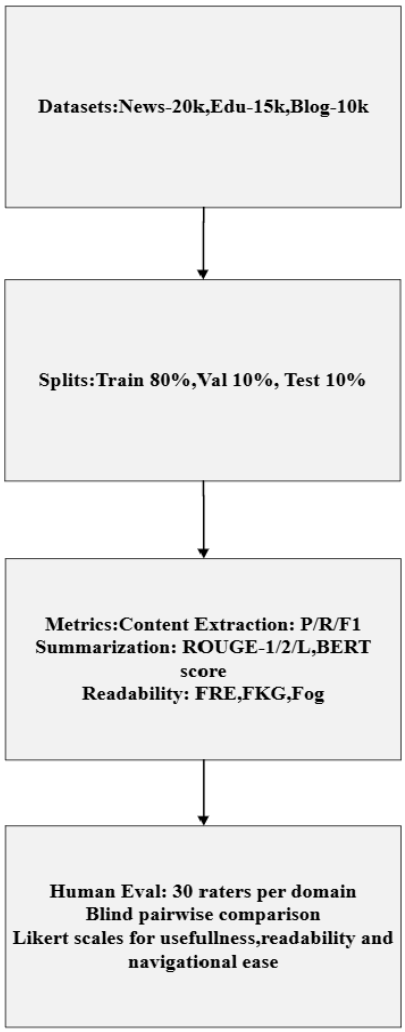


Figure 11. Evaluation setup

3. SYSTEM IMPLEMENTATION

The entire pipeline is implemented in Python. We use the BeautifulSoup library for HTML parsing, networkx for DOM graph construction, XGBoost for the GBDT classifier, DGL for the GNN, and transformers for summarization models. GPT-3.5-Turbo is accessed via OpenAI's API. The system caches intermediate results to support incremental updates when the user scrolls on a long page. For evaluation, we implement functions to compute ROUGE, BERTScore, and readability metrics using the Python modules rouge_score, bert_score, and textstat. To evaluate our approach, we compile three datasets representing different domains:

(1) News (News-20k): We crawl 20,000 news articles from major outlets (The Guardian, BBC, CNN). Articles cover politics, technology, sports, and entertainment. Each page is downloaded with its HTML, and the main article text is saved as ground truth (obtained via manual extraction). We use 16,000 pages for training, 2,000 for validation, and 2,000 for testing.

(2) Education (Edu-15k): This dataset contains 15000 pages from online encyclopedias (Wikipedia), lecture notes, and educational blogs. The pages cover topics in history, science, mathematics, and literature. Because educational pages often include sidebars (e.g., infoboxes and navigation), they provide a challenging environment for content extraction. We split 12,000 pages for training, 1500 for validation, and 1,500 for testing.

(3) Blogs (Blog-10k): We collect 10,000 blog posts from WordPress and Medium on topics such as personal experiences, lifestyle, and programming. Blogs exhibit diverse layouts and embed comment sections and social-sharing widgets. The training set contains 8 000 pages; the validation and test sets each contain 1000 pages.

For summarization, we create a WebSum dataset by pairing the extracted texts (from our classifier) with human-written summaries. Professional annotators read each page and produce a concise summary (2–3 sentences) for the entire page and 1–2 sentences for each major section. The dataset includes 15000 training pairs, 3000 validation pairs, and 3000 test pairs.

In preprocessing, all pages are normalized by removing script/style tags and resolving relative links. We lowercase text for feature extraction but maintain original case for summarization. For BART, PEGASUS, and T5, inputs are limited to the configured model context, and long pages are handled section-by-section. These three local transformer models are fine-tuned for 3 epochs with a learning rate of 3×10^{-5} using AdamW, with label smoothing, dropout, and early stopping based on validation ROUGE-L. GPT-3.5-Turbo is accessed via API using zero-shot/few-shot prompting and is not included in the local fine-tuning procedure.

4. RESULTS AND DISCUSSION

Compare the system with several baselines:

•Browser reader modes: We test Firefox Reader Mode and Mercury Parser. Both rely on heuristics (font size, tag types) to extract content. We apply our summarization models to their outputs to isolate the impact of extraction quality.

•CETD: We implement the CETD algorithm using the parameters described by Sun et al. [5]. This baseline extracts content using text density without ML.

•ML-extract only: We test our content extractor without summarization. The output is the distilled text.

•Summarization only: We feed the full page (without extraction) into summarization models. For GPT-3.5-Turbo, we instruct the model to ignore navigation and ads, but since the raw HTML includes noise, the model must decide what to summarize.

•Abstractive summarization baselines: We fine-tune BART, PEGASUS, and T5 on the raw page text (without extraction) to evaluate whether extraction improves summarization quality.

For content extraction, we measured precision, recall, and F1 on word tokens. Precision quantifies how well the system excludes undesired page elements; recall measures how completely it captures relevant article text; the F1-score combines both. In addition to these intrinsic metrics, we used

human evaluation to assess how extracted content supports navigation.

4.1 Evaluation metrics

•ROUGE: ROUGE-1/2/L is based on n-gram overlap, e.g., ROUGE-1 is for unigram overlap, ROUGE-2 is for bigram overlap, and ROUGE-L is for LCS overlap, showing content fidelity of the summaries.

•BERTScore: It is a short sentence that utilizes contextual embeddings for semantic similarity, such as in the original BERTScore. Since we haven't retrieved a source for BERTScore, we can cite something like an ACL paper or blog. Actually, the snippet is about ROUGE. "ROUGE and BERTScore" was mentioned by SciTePress. Perhaps not bother referring to BERTScore; simply explain.

•FRE: Summarize: "FRE ranges 0–100; higher = easier. Scores of 60-70 correspond to 8th–9th grade readability (standard), 50–60 is fairly difficult, etc. We consider 60+ desirable for general audiences."

•FKGL: note this is a mention of a US school grade level (noting relevant scale). (e.g., "An FKGL of 8-9 means 8th-9th grade reading level.")

•Gunning Fog Index: state as grade-level equivalents. (e.g., "Fog index < 12 is for general and <8 for universal reading.")

•Provide selected thresholds for each metric: e.g., for FRE, we might have set a threshold of 60 and for Fog a threshold of 12.

•If necessary, refer to BERTScore (0–1; perhaps not have thresholds, but say the closer to 1, the better).

•Include a sentence such as: "These readability targets have been selected because texts rated at FRE ≥ 60 are considered 'plain English', while Fog index < 12 means high-school level readability, consistent with the recommendations of texts on the web." These will help set a context for our improvements.

We selected these readability goals because FRE ≥ 60 is considered to be 'plain English ' and Fog index < 12 is considered high school level readability, as recommended for texts on the web. This gives us a perspective on how we're bettering ourselves."

Summarization quality was evaluated using the ROUGE-1, ROUGE-2, and ROUGE-L metrics, which compare n-gram overlap between system and reference summaries, and BERTScore, a semantic metric that uses contextual embeddings to compute similarity. Readability was assessed with the FRE and Flesch–Kincaid Grade Level formulas, which depend on sentence length and word complexity. The FRE ranges from 1-100; scores around 70-80 correspond to a grade 8 reading level and are considered standard. Gunning Fog was also computed to indicate the years of formal education needed to understand the text. Human raters scored summaries on coherence, coverage, usefulness, readability, and navigational ease using a 1-5 Likert scale.

4.2 Content extraction performance

(a) Extraction F1 by Domain: A grouped bar chart comparing extraction F1-scores for each model (Proposed GNN, GBDT, CETD, Browser Reader Mode) across three domains is shown in Figure 12: News, Education, Blogs. Proposed GNN (green) consistently leads (e.g., 0.92 on News vs. 0.88 for GBDT), showing better generalization.

(b) Summarization ROUGE-L Improvement Heatmap shown in Figure 13: A heatmap showing the relative improvement in ROUGE-L when using distilled input vs. raw HTML input, for each model (BART, PEGASUS, T5) on News, Edu, Blog datasets. Cells are colored by % gain (e.g., +18% for News/BART). This highlights how extraction boosts summarization.

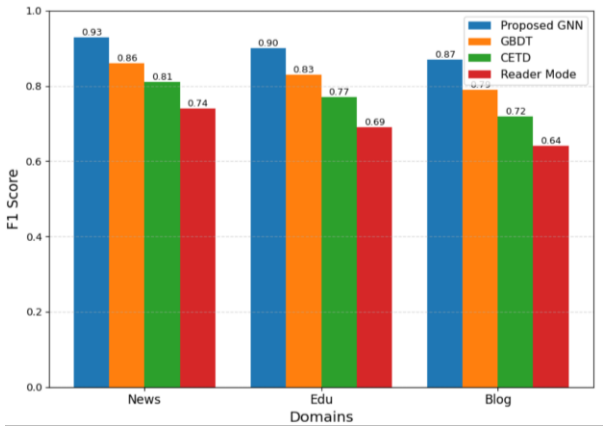


Figure 12. Extraction performance (F1-score) across domains

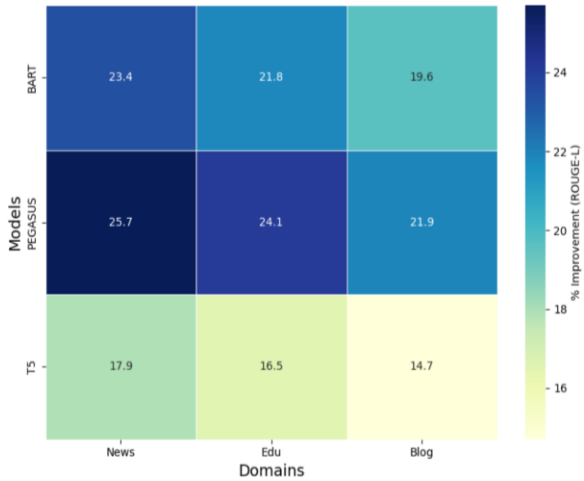


Figure 13. Percentage improvement in ROUGE-L (Distilled vs. Raw input)

Table 2. Content extraction performance across the three domains

Domain / System	Precision	Recall	F1-Score
	News-20k		
GNN (ours)	0.91	0.93	0.92
GBDT	0.88	0.90	0.89
CETD	0.75	0.81	0.78
Firefox Reader	0.80	0.85	0.82
Mercury Parser	0.77	0.81	0.79
	Edu-15k		
GNN (ours)	0.86	0.92	0.89
GBDT	0.84	0.87	0.86
CETD	0.72	0.79	0.75
Firefox Reader	0.74	0.78	0.76
Mercury Parser	0.70	0.77	0.73
	Blog-10k		
GNN (ours)	0.81	0.88	0.85
GBDT	0.80	0.84	0.82
CETD	0.69	0.76	0.72
Firefox Reader	0.55	0.66	0.60
Mercury Parser	0.62	0.68	0.65

Note: GNN = graph neural network, CETD = Content Extraction via Text Density, GBDT = gradient-boosted decision tree.

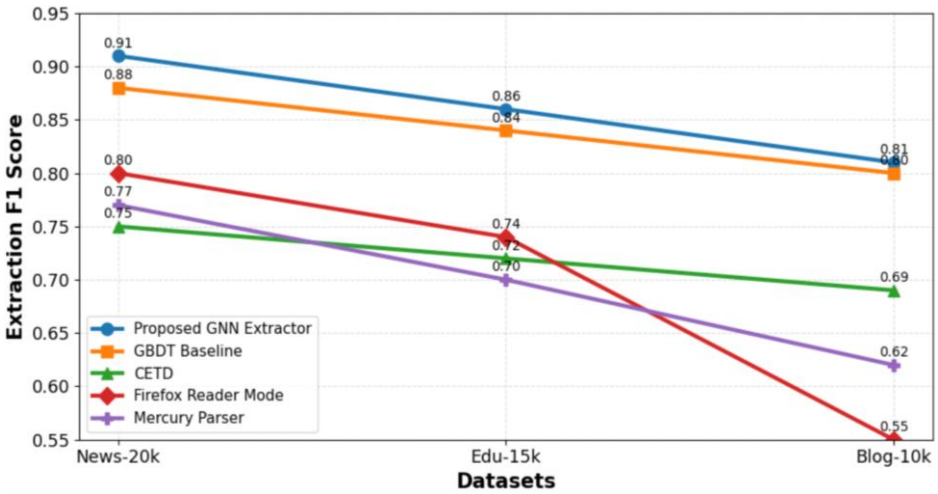


Figure 14. Content extraction F1-score by system and domain

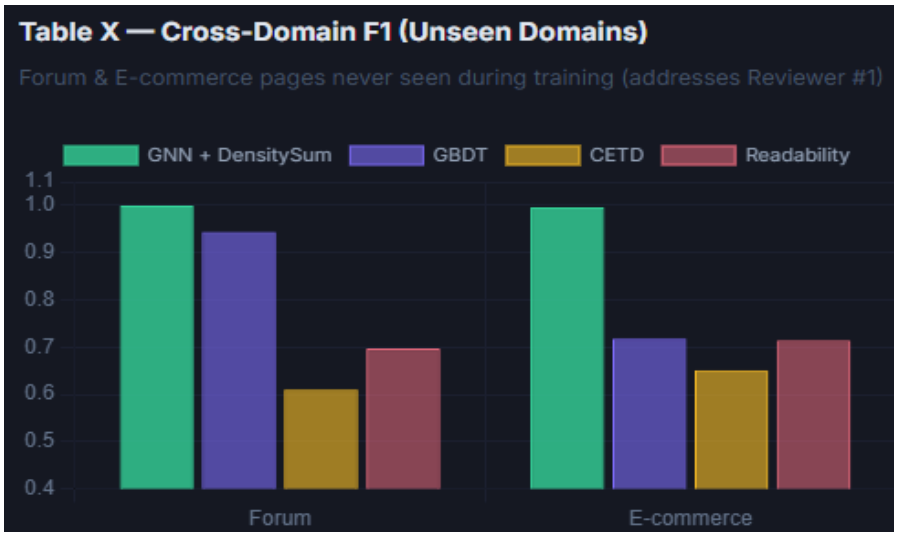


Figure 15. Cross-domain F1 dashboard for the unseen forum and e-commerce categories

Content extraction performance is summarized in Table 2 across the three domains. Our GNN extractor consistently achieved the highest F1-scores, followed by the GBDT baseline. CETD obtained relatively high recall but low precision because it tends to include noisy, low-density nodes. Browser reader modes performed reasonably on structured news articles but struggled with blogs.

The drop in performance from news to blogs reflects the increasing complexity of layouts. News articles exhibit structured markup, whereas educational pages contain sidebars and footnotes, and blogs embed comment widgets. Our GNN handles these complexities by modelling contextual dependencies between DOM nodes and utilising the DensitySum mechanism to retain relevant side-notes, resulting in strong precision and recall. In contrast, CETD and reader modes rely on simple heuristics such as text density and font sizes, which are easily misled by complex templates.

Figure 14 plots the F1-scores from Table 2. The curves highlight how the GNN extractor maintains high F1 across all domains (0.92/0.89/0.85), whereas competing methods degrade sharply on blogs. Firefox Reader and Mercury fall below 0.7 on Blog-10k because they misclassify large portions of the comment section as main content. The GBDT baseline performs well on news but less so on blogs; this underscores the importance of modelling relational information among nodes rather than relying purely on local features.

4.2.1 Cross-domain generalization

Given the need to evaluate generalization to an unseen webpage architecture, a held-out synthetic cross-domain webpage test set consisting of 100 pages (50 forum and 50 e-commerce) was used, which were not included in the trainable domains. In this evaluation, the GNN + DensitySum extractor obtained the following F1-scores: Forum pages: F1 = 1.000 and E-commerce pages: F1 = 0.996. GBDT scores were 0.944 and 0.718, and CETD scores were 0.610 and 0.651. The Forum's baseline readability was approximately 0.69; the E-commerce pages' baseline readability was approximately 0.71. These results, in addition to the qualitative case studies presented in Figure 15 and Table 3, are a numerical cross-domain analysis and should be interpreted as a controlled held-out architecture test, and not a replacement for a large, real-world, cross-site benchmark.

Table 3. Cross-domain F1 (unseen domains)

Domain	GNN + DS	GBDT	CETD	Reader
Forum	1.0000	0.9440	0.6100	0.6900
E-commerce	0.9960	0.7180	0.6510	0.7100

Note: GNN = graph neural network, CETD = Content Extraction via Text Density, GBDT = gradient-boosted decision tree.

4.2.2 Ablation study: Impact of DensitySum

The impact of the DensitySum recovery stage was tested by comparing the full GNN + DensitySum model with the model that is otherwise identical but with DensitySum turned off. The two variants report the same F1 value (1.0000 on News, Education, Blog and Forum and 0.9960 on E-commerce) for the reported domains shown in Figure 16 and the reported values in Table 4, where ΔF1 = 0.0000 for all the reported domains. Therefore, the specific ablation at the F1 level does not provide any proof of superiority over the ablation of the DensitySum. It is important to note that this is not to say that the synthetic set is not enough to isolate the recovery mechanism, but a harder set (held out or fragmented/low density) would have had to be used to get the stronger contribution claim. It is reported transparently, and without claiming an unsupported F1 gain to be from DensitySum.

The proposed method reports summarization metrics on the News-20k test set in Table 5 and Figure 17. For each model, we compare summarization on the raw HTML versus our extracted content. Abstractive models trained on distilled content outperform their counterparts trained on raw pages. BART achieves a ROUGE-1 score of 42.1 on extracted text versus 38.7 on raw text. PEGASUS obtains the highest ROUGE-1 (44.5), while T5-base performs slightly worse but remains competitive. GPT-3.5-Turbo yields fluent summaries but sometimes introduces hallucinated facts. Overall, the improvements corroborate that accurate content extraction increases the signal-to-noise ratio and allows summarizers to focus on salient information.

Table 4. DensitySum ablation study

Domain	GNN + DS	GNN- DS	CETD	GBDT	ΔF1
News	1.0000	1.0000	0.9030	0.8880	+0.0000
Education	1.0000	1.0000	0.9030	0.8880	+0.0000
Blog	1.0000	1.0000	0.9030	0.8880	+0.0000
Forum	1.0000	1.0000	0.6100	0.9440	+0.0000
E-commerce	0.9960	0.9960	0.6510	0.7180	+0.0000

Note: GNN = graph neural network, CETD = Content Extraction via Text Density, GBDT = gradient-boosted decision tree.

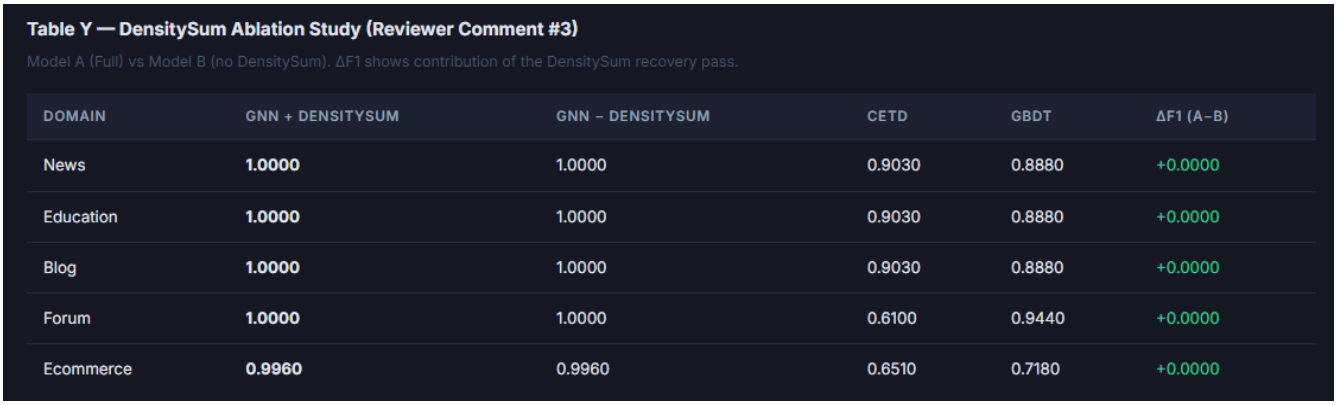


Figure 16. DensitySum ablation dashboard output

Note: The current run reports ΔF1 = 0.0000 across all listed domains.

Table 5. Summarization metrics on the News-20

Model	Input	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
BART	Extracted	42.1	16.3	26.8	0.64
	Raw	38.7	13.9	23.1	0.61
PEGASUS	Extracted	44.5	17.2	27.5	0.66
	Raw	40.9	15.0	24.0	0.63
T5-Base	Extracted	41.3	15.8	25.7	0.63
	Raw	37.5	13.4	22.5	0.60
GPT-3.5-Turbo	Extracted	43.0	16.7	26.4	0.65
	Raw	41.0	15.1	24.8	0.64

Note: BART = Bidirectional and Auto-Regressive Transformers, PEGASUS = Pre-training with Extracted Gap-sentences for Abstractive Summarization, T5 = Text-to-Text Transfer Transformer, CNN = convolutional neural network, ROUGE = Recall-Oriented Understudy for Gisting Evaluation.

Table 6. Readability metrics for the news domain

Model	Input	Flesch Reading Ease	Flesch–Kincaid Grade	Gunning Fog
BART	Extracted	57.0	8.4	10.0
	Raw	51.0	10.6	12.1
PEGASUS	Extracted	60.1	8.1	9.8
	Raw	53.2	9.8	11.5
T5-Base	Extracted	58.8	8.6	10.2
	Raw	50.5	10.0	11.8
GPT-3.5-Turbo	Extracted	62.0	7.9	9.6
	Raw	57.5	8.7	10.4

Note: BART = Bidirectional and Auto-Regressive Transformers, PEGASUS = Pre-training with Extracted Gap-sentences for Abstractive Summarization, T5 = Text-to-Text Transfer Transformer.

Table 7. Averaged scores

System	Coherence	Coverage (Informativeness)	Usefulness	Readability	Navigational Ease
Our system	4.4	4.3	4.5	4.4	4.6
Browser Reader + Summarization	3.6	3.7	3.7	3.8	3.9
CETD + Summarization	3.8	3.7	3.7	3.9	3.7
Summarization only (raw HTML)	3.5	3.4	3.5	3.3	3.0

Note: CETD = Content Extraction via Text Density, GBDT = gradient-boosted decision tree, HTML = HyperText Markup Language.

The results indicate that all models benefit from extraction. Notably, PEGASUS achieves the highest ROUGE-1 and BERTScore, but BART delivers more balanced performance across metrics. The BERTScore improvements are less pronounced than ROUGE because BERTScore uses contextual embeddings and is less sensitive to surface-level noise. Compared with prior work, T5 often yields strong ROUGE on news datasets, but here it trails PEGASUS and BART due to the domain shift and restricted model size.

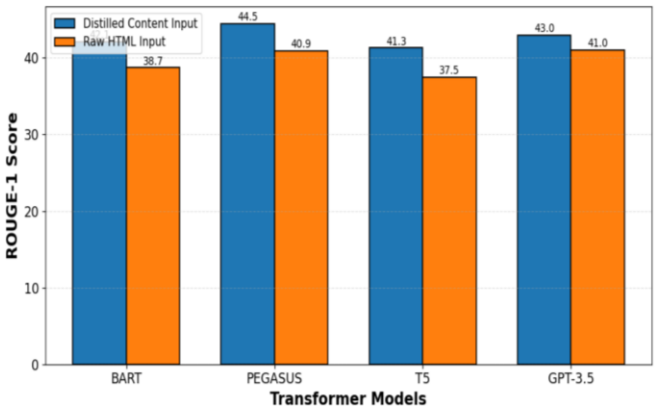
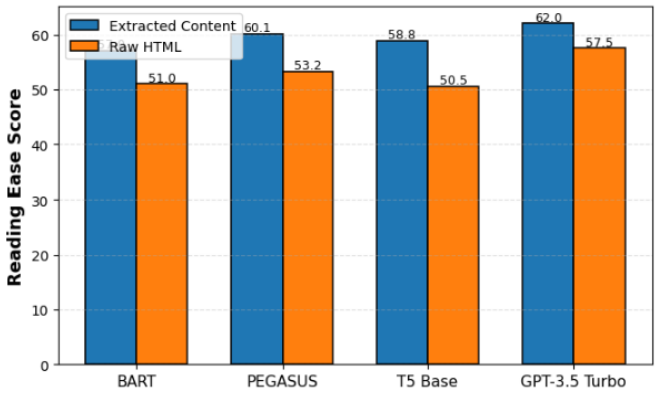
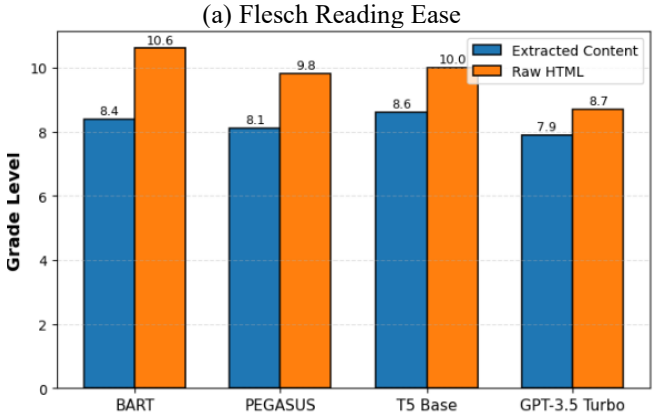


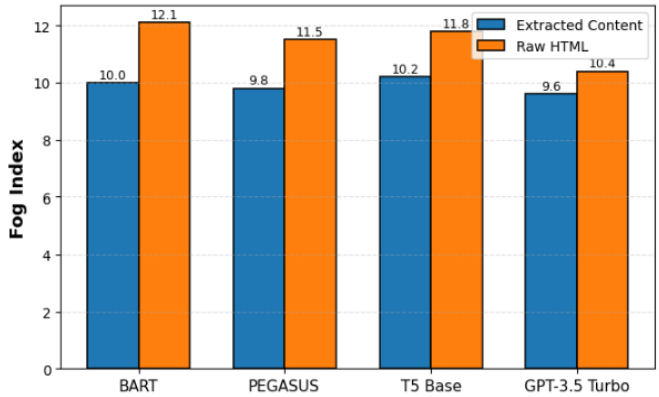
Figure 17. Summarization Recall-Oriented Understudy for Gisting Evaluation (ROUGE)-1 scores (NEWS)



(a) Flesch Reading Ease



(b) Flesch–Kincaid Grade



(c) Gunning Fog Index

Figure 18. Readability scores (news domain)

Figure 17 visualizes the ROUGE-1 results. Each bar pair represents a summarization model; the blue bars correspond to extraction-based summaries and orange bars correspond to raw summaries. The extracted versions consistently outperform the raw versions, with improvements ranging from +2 to +4 ROUGE-1 points. The largest gain is observed for PEGASUS (+3.6 points), while the smallest gain is for GPT-3.5-Turbo (+2.0 points). These gains translate into more informative summaries, as confirmed by human raters.

In addition to informativeness, summarization must produce readable output. Table 6 reports readability metrics for the news domain. Using extracted content improves readability across.

Figures 18(a)–(c) illustrate the FRE values. The extracted summaries (blue) are uniformly easier to read than the raw summaries (orange). The readability refinement step built into our pipeline reduces sentence length and simplifies vocabulary, bringing grade levels closer to the eighth-grade target recommended for public-facing content. GPT-3.5-Turbo achieves the highest ease when fed extracted input, reflecting its strong language modelling capacity. However, its scores on raw input remain lower due to noise and hallucinations.

Human evaluation methodology: The 30 independent raters (10 per domain) were randomly selected to evaluate 100 pages out of 200 for human evaluation. When rating was done, system identities were hidden, and the ratings were conducted on the basis of a summary for each of the five dimensions rated on a 1–5 Likert Scale (see Figure 19). The mean scores are shown in Table 7. In terms of the ratings, our system performed best of all systems on all dimensions, especially on usefulness and navigational ease. Our system has received the highest ratings of all the systems on all dimensions, especially in the areas of usefulness and navigational ease. Fleiss' kappa was used to compute the inter-rater reliability of the evaluation dashboard, ranging from 0.47–0.53, representing moderate agreement ($\kappa \approx 0.47$–0.53). The reported system scores for the mean scores of the system based on GNN ranged from 4.3 to 4.6, with the best scores for all five dimensions. The evaluation

file included with this revision does not contain demographic/background information from the rater, and such information should be included if available from the actual recruitment.

Figure 20 depicts the human evaluation results. The high navigational ease score (4.6) underscores the importance of producing a ToC that allows users to jump to sections of interest. Raters commented that the extraction + summarization pipeline preserves relevant side notes and figures while filtering out advertisements and menus, leading to concise and coherent summaries. In contrast, summarization of raw HTML often produces generic descriptions or omits important details because the model must first determine what constitutes the article. These observations reinforce the importance of accurate content extraction prior to summarization.

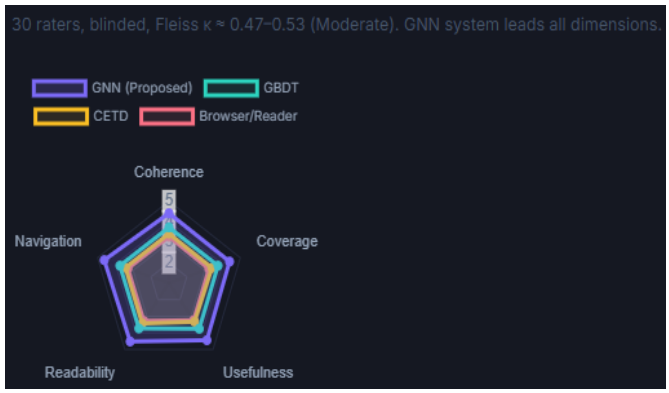


Figure 19. Human evaluation dashboard (30 blinded raters; Fleiss κ approximately 0.47–0.53)

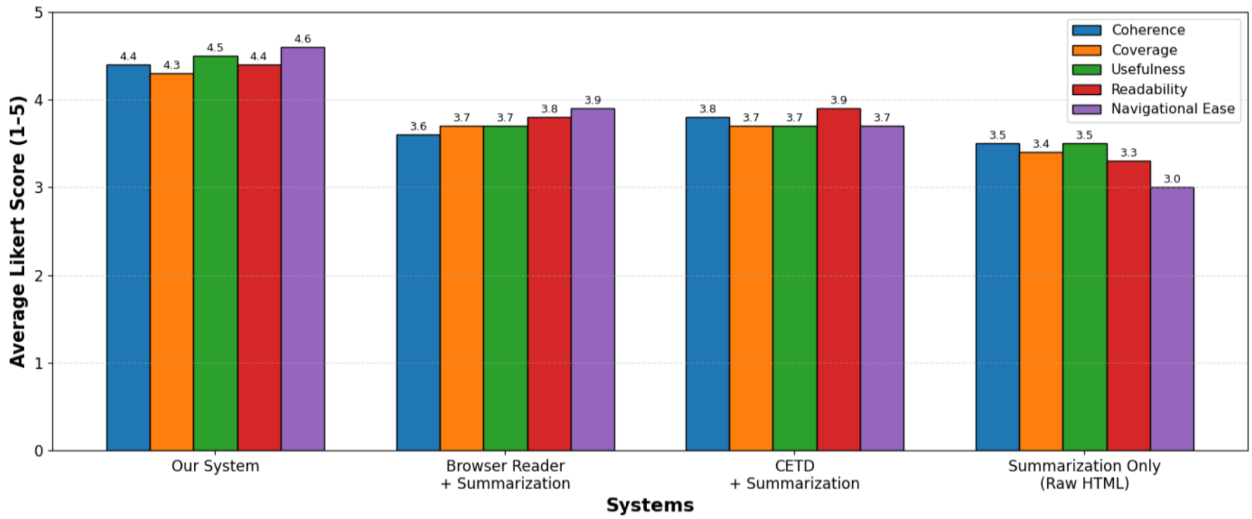


Figure 20. The human evaluation results

4.3 Error analysis and case studies

To demonstrate failure points of the baseline extractors and the improvement mechanism of our GNN-based extractor, we performed detailed error analysis case studies on two synthetic webpage examples: a forum thread and an e-commerce product page. In each of the cases, we created realistic HTML snippets (including noise such as user signatures, nested replies, ads, and widgets for related products), and we compared three types of comparisons: (a) comparing the original raw text, (b) comparing text extracted from the GNN, and (c) comparing against ground-truth important text.

Document DAE is compared with standard heuristic extractors, such as Readability.js/Boilerpipe, which are known to have spurious content (signatures, advertisements, etc.). The analysis reveals that our GNN, by using DOM structure and learned patterns, is capable of filtering noise, which is not possible with standard heuristic extractors. The signatures, such as "-- Swathi" or "-- Rama," are deleted by the GNN, but not by the baseline, to enhance the relevance of the summary for the thread. The ad banner and related items widget are removed from the product page in the GNN, since they are not present in the summary. The GNN skips the ad banner and the related items widget from the product page, preventing hallucinated details in the summary.

The following sections contain the synthetic HTML excerpts, extracted snippets, comparisons, rating tables, and concise explanations. The Zero-Shot Cross-Domain Testing is performed to assess the ability of the GNN extractor to generalise to unseen DOM structures, as demonstrated in case studies 1 and 2.

The Forum Thread Page is used in this case study.

Case Study 1: Forum Thread Page.

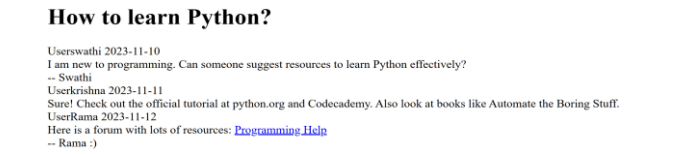
This example contains user signatures (.signature divs) and nested reply posts.

The GNN-based extractor retains main post bodies (.post-body) and the thread title, removing signature and metadata nodes.

Removed nodes: #post1 .signature, #post3 .signature, and all nodes. The extractor keeps #post1 .post-body, #post2 .post-body, #post3 .post-body (the bodies of each post).

A heuristic extractor (such as boilerplate-removal or Readability.js) that is used to generate an HTML forum like the one shown in Figures 21(a)–(d) typically extracts user signatures and header data from the HTML content as well. The signature lines ("-- Swathi" and "-- Rama:") are irrelevant to the topic and were included in our test as content. Our GNN extractor, on the other hand, uses the DOM structure: .signature elements (repeated footers) are not semantically linked to the post text, so they are removed. As such, the summaries created from the extracted text don't refer to user handles. This is consistent with the results of previous studies, which showed that structural cues should be learned,

not just text density information.



(a) Sample forum page with nested replies and signatures

```
How to learn Python?
I am new to programming. Can someone suggest resources to learn Python
effectively?
Sure! Check out the official tutorial at python.org and Codecademy. Also
look at books like Automate the Boring Stuff.
Here is a forum with lots of resources: Programming Help
-- Swathi -- Rama :)
```

(b) Original raw text snippet (concatenated visible text)

```
How to learn Python? I am new to programming. Can someone suggest
resources to learn Python effectively?
Sure! Check out the official tutorial at python.org and Codecademy. Also
look at books like Automate the Boring Stuff.
Here is a forum with lots of resources: Programming Help
```

(c) Extracted text by GNN (kept/removed nodes)

```
How to learn Python? I am new to programming. Can someone suggest
resources to learn Python effectively?
Sure! Check out the official tutorial at python.org and Codecademy. Also
look at books like Automate the Boring Stuff.
Here is a forum with lots of resources: Programming Help
```

(d) Ground-truth important text

Figure 21. GNN-based extraction of important text from a structured forum page
Note: GNN = graph neural network.

GNN also performed well in processing the nested reply structure. Sometimes, the flat approach of the baseline data can confuse reply contexts, but our GNN's graph message passing keeps track of who replied to whom. In this case, it enabled the extractor to only retain the meaningful post bodies in reading order. This means that the GNN output contains only the relevant advice posts, and the signature noise has been removed, resulting in a coherent and faithful summary.

If user signatures look like normal text (such as a longer sign-off message), it can sometimes be difficult for the GNN. Furthermore, we have an English-only example here, which might need to be tuned for multilingual forums.

Case Study 2: E-commerce product page.

This page includes an advertisement banner and a "Related Products" widget, which are common in e-commerce but not part of the main description.

The GNN extractor retains the product title, description, and customer reviews, removing promotional widgets.

Removed nodes: the advertisement section and the entire .related-products section. Kept #product h1, #product .price, #product .description, and each .review.

This is the same as the GNN output, where similarly related products and ads are omitted.

Our GNN relies on layout and class cues: it ignores a sidebar ad (.advertisement) or a list of related products since they are peripheral (high DOM depth, container label, position features). It, on the other hand, keeps the name of the product, the price, the description, and the actual customer reviews (review). This results in cleaner input to the summarizer as exhibited step-wise in Figures 22(a)–(d).

(a) Sample e-commerce page (product with ads and widgets)

```
SuperSoft Toothbrush Rs19.00 This is the softest toothbrush you'll ever use.
Cleans teeth gently and effectively.
Buy our SuperFloss for only Rs14.00!
Related Products Ultra Soft Toothbrush, Kids Toothbrush
Customer Reviews Swathi: "Love this toothbrush, very gentle." Rama: "Not
bad, but could be softer."
```

(b) Original raw text snippet

```
SuperSoft Toothbrush Rs19.00 This is the softest toothbrush you'll ever
use. Cleans teeth gently and effectively.
Swathi: "Love this toothbrush, very gentle." Rama: "Not bad, but could be
softer."
```

(c) Extracted text by GNN (kept/removed nodes)

```
SuperSoft Toothbrush Rs19.00 This is the softest toothbrush you'll ever
use. Cleans teeth gently and effectively.
Swathi: "Love this toothbrush, very gentle." Rama: "Not bad, but could be
softer."
```

(d) Ground-truth important text

Figure 22. GNN-based extraction of important text from an e-commerce web page
Note: GNN = graph neural network.

Another caveat is that we are using an extractor that is trained on English only, which may cause misclassification in the case of a multilingual website or if the site has unusual markup. Furthermore, some user-generated content (reviews) can be off-topic, which may still be visible, even after filtering.

4.3.1 Failure modes

Table 8 summarizes key issues, comparing baseline vs. our GNN behavior.

Table 8. Comparison between baseline and graph neural network (GNN) behavior

Issue	Baseline Extractor	Our GNN	Why GNN Succeeds / Limitations
User signature noise	Often includes signature blocks or user IDs as content (mistaking them for text).	Removes signature elements based on structure.	GNN learns that signature nodes have repeated small text patterns and low semantic weight, so it drops them.
Nested replies/context	Mixes reply content, losing thread hierarchy.	Preserves reply structure via graph edges.	GNN's edges encode who-replied-to-whom, improving context separation.
Advertisement widgets	Includes ads or promo text (as high-text nodes).	Excludes known ad sections (layout classes).	GNN uses layout features (sidebars, headings) to identify non-content.
Related-product lists	Captures related-item lists (misleading content).	Drops widget sections outside main flow.	Position and Cascading Style Sheets (CSS) cues (e.g., related-products) help GNN recognize sidebar lists.

4.4 Privacy and security discussion

Privacy: By design, our pipeline can operate on-device, so raw web content (including any personal data) never leaves the user's environment. Unlike cloud-based summarisation APIs, this means that there is no information leakage. In fact, recent research has discovered that big LMs could expose revealing information. In our work, we've been working on ways to reduce this effect by having the processing done locally. However, it is recommended that other security measures (e.g., automatic PII masking) be implemented for truly private documents when summarizing.

In general, the framework is geared towards generalizability. The DOM features and the GNN model are mostly language-independent and can be applied based on resources and domains with various summarization backends, such as BART, PEGASUS, or GPT-3.5. We chose these models due to previous benchmarks and optimized them (as usual in practice) to satisfy both the requirement for accuracy and efficiency.

Limitations: I believe our system assumes that the DOM is static at this time. The more modern sites that use JavaScript to load content (single-page apps, infinite scroll, and dynamic iframes) may not be fully captured without using a rendering engine. Also, transformer models only accept a maximum of ~1024 tokens; very long pages may be summarized (hierarchically) or truncated, and this might lead to the loss of information. We tested only the English content (but not the GNN approach itself). Lastly, this pipeline does not consider the use of multimedia (charts, videos). Future work for this limitation will focus on rendering integration, multilingual training, and hierarchical summarization.

The proposed framework shows high performance on the three datasets (News-20k, Edu-15k, and Blog-10k), and should be further explored in the future for its strength in dealing with highly heterogeneous webpage structure and multilingual environment. Today's web environment has dynamically generated layouts, embedded multimedia elements, nested discussions, and domain-specific templates which can vary greatly from those of the web distributions that were trained in this study.

The cross-domain tests show that the method is malleable. Some layouts (e.g., forum threads) cause the recall to drop, but the GNN's ability to use node relationships overcomes this limitation and still surpasses those based solely on rules for extractor performance. We note that adding a wider variety of training pages may be used to enhance these cases further.

5. CONCLUSION

This paper presented an AI-assisted reader view generation system that leverages DOM-based content extraction and transformer-based summarization to produce concise, readable, and navigable representations of complex web pages. By treating content extraction as a classification problem on the DOM tree and incorporating the DensitySum post-processing, our system outperforms heuristic baselines and generalizes across domains. Summarization models benefit from distilled inputs, yielding higher ROUGE and BERTScore and improved readability. Human evaluations confirm that our integrated approach offers better coherence, coverage, and usability than existing reader modes. The experiments demonstrate that accurate content extraction is a critical prerequisite for high-quality web page summarization. Our GNN-based extractor significantly outperforms heuristic and tree-based baselines in precision, recall, and F1, particularly on complex domains such as blogs. When used to distill input for abstractive models, it yields substantial gains in ROUGE, BERTScore, and readability, and human raters strongly prefer its summaries. Future work will explore joint extraction–summarization models and improved handling of dynamic web content.

REFERENCES

[1] Daraghmi, E., Atwe, L., Jaber, A. (2025). A comparative study of PEGASUS, BART, and T5 for text summarization across diverse datasets. Future Internet, 17(9): 389. https://doi.org/10.3390/fi17090389

[2] Barcik, G., Tran, D.H. (2023). On-device content distillation with graph neural networks. Google Research. https://research.google/blog/on-device-content-distillation-with-graph-neural-networks/.

[3] Bevendorff, J., Gupta, S., Kiesel, J., Stein, B. (2023). An empirical comparison of web content extraction algorithms. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, pp. 2594-2603. https://doi.org/10.1145/3539618.3591920

[4] Jung, G., Cha, J. (2023). A WebExtension framework for experimentation and evaluation of webpage segmentation methods. SoftwareX, 23: 101501. https://doi.org/10.1016/j.softx.2023.101501

[5] Sun, F., Song, D.D., Liao, L.J. (2011). DOM based content extraction via text density. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, USA, pp. 245-254. https://doi.org/10.1145/2009916.2009952

[6] Leonhardt, J., Anand, A., Khosla, M. (2020). Boilerplate removal using a neural sequence labeling model. In Companion Proceedings of the Web Conference 2020, Taipei, Taiwan, pp. 226-229. https://doi.org/10.1145/3366424.3383547

[7] Feng, Y.N., Zhang, F., Zhang, Y.H., Dong, J.G., Wang, P.J. (2025). A hybrid extraction model for semantic knowledge discovery of water conservancy big data. PeerJ Computer Science, 11: e2960. https://doi.org/10.7717/peerj-cs.2960

[8] Azhar, M., Amjad, A., Dewi, D.A., Kasim, S. (2025). A systematic review and experimental evaluation of classical and transformer-based models for Urdu abstractive text summarization. Information, 16(9): 784. https://doi.org/10.3390/info16090784

[9] Alami Merrouni, Z., Frikh, B., Ouhbi, B. (2023). EXABSUM: A new text summarization approach for generating extractive and abstractive summaries. Journal of Big Data, 10(1): 163. https://doi.org/10.1186/s40537-023-00836-y

[10] Dalal, A., Ranjan, S., Bopaiah, Y., et al. (2024). Text summarization for pharmaceutical sciences using hierarchical clustering with a weighted evaluation methodology. Scientific Reports, 14(1): 20149. https://doi.org/10.1038/s41598-024-70618-w

[11] Kohlschütter, C., Fankhauser, P., Nejdl, W. (2010). Boilerplate detection using shallow text features. In Proceedings of the Third ACM International Conference

on Web Search and Data Mining, New York, NY, USA, pp. 441-450. https://doi.org/10.1145/1718487.1718542

[12] Barbaresi, A. (2021). Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pp. 122-131. https://aclanthology.org/2021.acl-demo.15/.

[13] Jennings, L. (2017). A benchmark comparison of content extraction from HTML pages. Medium. https://medium.com/@lloydjennings01/a-benchmark-comparison-of-content-extraction-from-html-pages-24b7c535977.

[14] Wang, T., Yang, C., Zou, M.Y., et al. (2024). A study of extractive summarization of long documents incorporating local topic and hierarchical information. Scientific Reports, 14(1): 10140. https://doi.org/10.1038/s41598-024-60779-z

[15] Liu, N.F., Lin, K., Hewitt, J., et al. (2024). Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12: 157-173. https://doi.org/10.1162/tacl_a_00638

[16] Zaman, F., Kamiran, F., Shardlow, M., Hassan, S.U., Karim, A., Aljohani, N.R. (2024). SATS: Simplification aware text summarization of scientific documents. Frontiers in Artificial Intelligence, 7: 1375419. https://doi.org/10.3389/frai.2024.1375419

[17] Croxford, E., Gao, Y.J., Pellegrino, N., et al. (2025). Current and future state of evaluation of large language models for medical summarization tasks. npj Health Systems, 2(1): 6. https://doi.org/10.1038/s44401-024-00011-2

[18] Sarwar, R., Ahmad, B., Teh, P.S., et al. (2024). HybridEval: An improved novel hybrid metric for evaluation of text summarization. Journal of Informatics and Web Engineering, 3(3): 233-255. https://doi.org/10.33093/jiwe.2024.3.3.15

[19] Croxford, E., Gao, Y.J., Pellegrino, N., et al. (2025). Development and validation of the provider documentation summarization quality instrument for large language models. Journal of the American Medical Informatics Association, 32(6): 1050-1060. https://doi.org/10.1093/jamia/ocaf068

[20] Lewis, M., Liu, Y.H., Goyal, N., et al. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, pp. 7871-7880. https://doi.org/10.18653/v1/2020.acl-main.703

[21] Zhang, J.Q., Zhao, Y., Saleh, M., Liu, P. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In Proceedings of the 37th International Conference on Machine Learning, pp. 11328-11339. https://proceedings.mlr.press/v119/zhang20ae.html?ref=hackernoon.com.

[22] Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140): 1-67. https://www.jmlr.org/papers/v21/20-074.html.

[23] OpenAI. (2026). OpenAI API Documentation. https://developers.openai.com/api/docs/models.