Innovative Informatics and Artificial Intelligence Research (IIAIR)
Vol. 2, issue 1, pp. 10-16, 2026
Received 17 Jul 2025; accepted 25 Jan 2026; published 30 Apr 2025
https://doi.org/10.35718/iiair.v2i1.8481487

Main Content Extraction from Scientific Journal Websites Using BoilerNet

Felix Dean Janitra [1], Gusti Ahmad Fanshuri Alfarisy [1], and Aninditya Anggari Nuryono[1]

[1]Department of Informatics, Institut Teknologi Kalimantan, Balikpapan 76127, Indonesia

Corresponding author: *Gusti Ahmad Fanshuri Alfarisy (gusti.alfarisy@lecturer.itk.ac.id)*

To cite this article: F. D. Janitra, G. A. F. Alfarisy, A. A. Nuryono, "Main Content Extraction from Scientific Journal Websites Using BoilerNet" *Innovative Informatics and Artificial Intelligence Research*, vol. 2, issue 1, 2026. [Online]. Available: https://doi.org/10.35718/iiair.v2i1.8481487

Gusti Ahmad Fanshuri Alfarisy serves as an Editor of IIAIR but was not involved in the peer-review process of this article

Abstract

In the digital era, the number of scientific journal publications available online is increasing rapidly. However, the process of information extraction from these journals is often disrupted by boilerplate elements such as headers, footers, navigation bars, and metadata that are not directly related to the main content. The presence of such elements hinders text analysis and further processing in various academic applications. This study aims to develop an automated boilerplate removal method using BoilerNet, a deep learning model based on the Bi-LSTM approach. The model is designed to classify each text segment in journal pages and accurately separate the main content from boilerplate elements. The methodology includes several stages: collecting scientific journal data through web scraping, preprocessing for normalization and labeling, and training the model using Binary Cross Entropy optimization. The model is evaluated using metrics such as precision, recall, and F1-score to assess its effectiveness in removing boilerplate elements. The experiment used a total of 180 data instances, divided into 50 for training, 30 for validation, and 100 for testing. Experimental results show that the optimal BoilerNet configuration with 256 hidden units, a dropout rate of 0.4, two LSTM layers, and a dense layer size of 256 achieves an F1-score of 0.91 for the boilerplate class and 0.81 for the main content class. The overall average F1-score reached 0.863. These results indicate that BoilerNet significantly improves main content extraction performance compared to conventional methods such as Readability.JS.

Keywords: Boilerplate Removal; BoilerNet; Deep Learning; Bi-LSTM; Scientific Journals; Main Content Extraction.

1. Introduction

In the digital era, the volume of scientific journal publications available online has grown significantly. Easy access to scholarly literature is essential for researchers and academics, enabling the rapid dissemination and consumption of knowledge. However, one of the primary challenges in processing these documents is the presence of boilerplate elements on web pages, which are segments that do not directly relate to the main content. These may include headers, footers, navigation bars, advertisements, and metadata, all of which can hinder accurate information extraction [1] .

To address this issue, various approaches have been proposed for extracting the main content from web pages. These range from rule-based techniques to machine learning-based methods. For instance, Jung et al. [2] introduced an extraction technique based on the user's first impression area to locate the most relevant content. Alarte and Silva [3] proposed a method that leverages structural analysis of web pages to enhance content extraction performance. A comprehensive empirical comparison by Bevendorff et al. [4] , revealed that no single extraction algorithm consistently outperforms others across different scenarios. This finding has encouraged the use of hybrid or ensemble approaches.

In recent years, deep learning-based methods have emerged as promising solutions for improving boilerplate removal. One such model is BoilerNet, which applies neural sequence labeling to automatically identify and eliminate boilerplate elements [1] This approach leverages the sequential nature of HTML documents and learns to distinguish between relevant and irrelevant content through training on labeled data.

This study investigates the effectiveness of BoilerNet in removing boilerplate from scientific journal web pages to facilitate more accurate main content extraction. Specifically, this work aims to:

1. analyze how variations in BoilerNet's architectural parameters, including the number of LSTM layers, hidden unit dimensions, dropout rates, and dense layer sizes, affect its performance.
2. compare its effectiveness with other established methods, such as Readability.JS, using standard evaluation metrics.

2. Related Works

This Prior studies in the field of boilerplate removal have explored a wide range of approaches, from rule-based heuristics to modern deep learning techniques. One of the most prominent contributions in this area is BoilerNet, a model proposed by Leonhardt et al. [1] which utilizes a neural sequence labeling framework to distinguish main content from non-informative sections in web pages. Unlike traditional approaches that depend on hand-crafted features, BoilerNet only uses HTML tags and the visible text as input. The model achieved promising results, with a precision of 0.95, recall of 0.86, and F1-score of 0.90 on the negative class (boilerplate), and a precision of 0.70, recall of 0.88, and F1-score of 0.78 on the positive class (main content). The authors demonstrated that BoilerNet outperforms rule-based systems such as BoilerPipe and Web2Text, highlighting the effectiveness of deep learning in this domain.

A different perspective was presented by Jung et al. [2] who introduced a content extraction method based on the First Impression Area (FIA), which refers to the part of a web page that initially attracts the user's attention. By modeling visual saliency and user attention patterns, their method aims to identify the most relevant content blocks. Their approach showed significant improvements over traditional methods, increasing F0.5-matched text block scores by up to 46% on English web pages and 42% on non-English pages. This result emphasizes the potential of visual context modeling for improving web content extraction.

In another study, Alakukku [5] proposed a domain-specific boilerplate removal technique that relies on entropy-based measurements and clustering to identify boilerplate segments in web pages. This unsupervised approach was tested on the L3S dataset and achieved a recall of up to 0.90, outperforming established methods such as jusText and BoilerPipe. However, the model suffered from low precision (0.18), indicating a tendency to include irrelevant information in the output.

Akbiyik et al. [6] introduced SORE (Semantic Outlier Removal), a novel approach for boilerplate removal that utilizes multilingual sentence embeddings and approximate nearest-neighbor search to identify and eliminate non-informative content blocks in web pages. Unlike traditional rule-based or structural methods, SORE does not rely on manually engineered features or DOM tree parsing. Instead, it computes sentence-level semantic similarity to detect outliers that deviate from the main content. Experiments conducted on multilingual datasets showed that SORE achieved competitive performance, even compared to large language model (LLM)-based methods, while maintaining low computational cost. This highlights the potential of lightweight embedding-based approaches for scalable and language-agnostic content extraction tasks.

Fernández-Pichel et al. [7] took an unsupervised approach by introducing a perplexity-based method using language models to remove boilerplate content. Their technique was evaluated on the ClueWeb12 dataset and showed a notable increase in retrieval performance, with the top-5 precision rising from 0.564 to 0.624. Additionally, the method significantly reduced the dataset size from 382GB to 96GB, making it highly efficient for large-scale processing. This study illustrates the value of leveraging language model perplexity as a metric for content quality and relevance.

While each of these approaches has shown success in particular scenarios, the absence of a universally dominant method indicates the complexity of the boilerplate removal task. Recent advances in deep learning, particularly models like BoilerNet, offer a promising direction by learning robust representations directly from raw HTML data, eliminating the need for manual feature engineering and improving generalization across diverse web page structures.

3. Proposed Dataset

The dataset used in this study was constructed from a collection of scientific journal articles sourced from multiple online repositories, including arXiv, ScienceDirect, Emerald, Nature, PMC, EuropePMC, Frontiersin, DOAJ, PsycNet, Neliti, Springer, and IIAIR. A total of 180 web page links were manually collected to ensure a diverse representation of journal websites and layouts.

Each web page was scraped using Python libraries, specifically BeautifulSoup and Selenium, to extract the raw HTML content of the article pages. These pages typically contain both the main article content and various boilerplate elements, such as headers, footers, navigation menus, metadata blocks, and advertisements. The resulting dataset serves as the foundation for training and evaluating boilerplate removal models, particularly in the context of scientific publishing platforms that often follow semi-structured yet diverse templates. The distribution of the dataset across various journal websites can be seen in Table 1.

Table 1: List of Journal Websites and Number of Documents Collected

Website	Number of Documents Collected
arXiv	20
ScienceDirect	20
Emerald	7
Journal ITK (IIAIR)	5
Nature	13
Frontiersin	12
NCBI	20
Ieeexplore	20
PsycNet	21
EuropePMC	12
Neliti	15
DOAJ	15

The dataset used throughout this research, including the raw HTML files, is publicly available at: https://drive.google.com/drive/folders/1KVlhrc7WvE31kS731E5m918rj8J8vfzK?usp=sharing

4. Methods

4.1 Data Preprocessing

To prepare the raw HTML data for training, each web page was segmented based on its DOM structure using elements such as , <p>, and <section>. This segmentation divides the document into blocks, which are then treated as classification units.

Each block was manually labeled using the attribute __boilernet_label, where "1" indicates main content and "0" represents boilerplate. The labeling was carefully performed

by analyzing the layout and structure of each page. After labeling, the dataset was split into training and testing sets. Examples of the labeled blocks are provided in Table 2, which illustrates blocks marked as main content, and Table 3, which shows blocks labeled as boilerplate.

Table 2: Example of HTML Blocks Labeled as Main Content

<span __boilernet_label="1" class="title-text"> Why we post selfies: Understanding motivations for posting pictures of oneself

Table 3: Example of HTML Blocks Labeled as Boilerplate

<span __boilernet_label="0"> Get behind my selfies: The Big Five traits and social networking behaviors through selfies

4.2 BoilerNet Architecture

The architecture of BoilerNet is based on the Bidirectional Long Short-Term Memory (Bi-LSTM) model, which is capable of handling complex non-local dependencies in sequential data [1] . Bi-LSTM assigns labels to elements in a sequence by processing the data in two directions simultaneously-one from left to right and the other from right to left. This parallel processing allows the model to capture both past and future context, making it particularly effective for content extraction tasks such as boilerplate removal [8] .

In the context of BoilerNet, the use of Bi-LSTM enables the model to better distinguish between main content and boilerplate by analyzing broader contextual relationships within the HTML document. By capturing dependencies in both directions, the model is able to understand structural patterns and layout semantics that are often difficult to detect using standard LSTM models [1] .

Studies have shown that Bi-LSTM-based architectures outperform traditional LSTM models in classifying HTML elements, as they are more effective at learning the structure of semi-structured documents such as scientific journal pages [1] . Additionally, this approach has demonstrated improved performance when applied to multilingual text, which is a common challenge in main content extraction [8] .

Figure 1: Illustration of BoilerNet Architecture

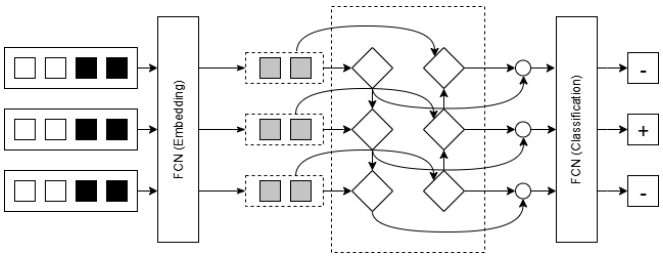

Figure 1 illustrates the representation and classification process of text blocks in a web page. The page is treated as a sequence of text blocks, where each block is represented as a sparse vector encoding information from both HTML tags and textual content. In the visualization, HTML tags are shown in white and text words in black [1] . These sparse vectors are passed through an embedding layer that transforms them into dense, low-dimensional vectors that are better suited to capturing semantic relationships between blocks [1] . The embedded representations are then fed into a classification model-such as LSTM or Transformer-based architectures—that predicts the label for each block based on its context. This process enables the model to identify key content, analyze document structure, and support advanced NLP applications.

4.3 Evaluation Metrics

In this study, the performance of the model is evaluated using three key metrics: Precision, Recall, and F1-Score. These metrics are widely used in binary classification tasks to assess how well a model distinguishes between relevant (main content) and irrelevant (boilerplate) elements [1] . Precision measures the quality of positive predictions by calculating the ratio of correctly predicted positive blocks (True Positives, TP) to all blocks predicted as positive (TP + False Positives, FP), as shown in Equation 1.

$$PREC = \frac{TP}{TP + FP} \tag{1}$$

Recall evaluates the model's ability to identify all relevant positive instances [9] . It is defined as the proportion of correctly identified positive blocks (TP) out of all actual positive blocks (TP + False Negatives, FN), as presented in Equation 2.

$$SN = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{2}$$

High recall indicates that the model is able to detect most of the main content in the web page, minimizing the risk of omitting important textual information [1] . This is particularly important in scientific articles, where losing key paragraphs or sections can lead to incomplete or misleading summaries. The F1-Score is the harmonic mean of precision and recall. It provides a single performance metric that balances both false positives and false negatives, as defined in Equation 3.

$$F\ Score = \frac{2 \times precision \times recall}{precision \times recall} \tag{3}$$

The F1-Score is especially useful in scenarios involving imbalanced datasets, such as this one, where non-content elements (boilerplate) often dominate the structure of HTML documents [9] . A high F1-Score reflects a strong overall performance in accurately extracting relevant content while excluding noise.

5. Experimental Settings

In this stage, experiments were designed and configured to evaluate the performance of the BoilerNet model in removing boilerplate elements from HTML documents of scientific journals. The training and evaluation processes were conducted on a machine equipped with a 13th-generation Intel Core i7 processor, 16 GB of RAM, and an NVIDIA RTX 4060 GPU with 8 GB of memory, supporting CUDA acceleration. The software environment was managed using Anaconda to ensure library compatibility and environment isolation.

Several key libraries were utilized, including TensorFlow version 2.19.0 for the deep learning architecture, NumPy version 1.26.0 for numerical computations, and scikit-learn version 1.3.2 for performance evaluation. In addition, BeautifulSoup, html5lib, and Selenium were employed during the HTML data extraction phase. GPU-based computations were accelerated using CUDA Toolkit version 12.0 and cuDNN version 9.

The dataset used in this study consists of 180 HTML documents sourced from various scientific journal websites with diverse page structures. Each document was manually annotated to create a ground truth that distinguishes main content elements (positive) from boilerplate elements (negative). The dataset was divided into 50 documents for training, 30 for validation, and 100 for testing. This proportion was chosen to rigorously evaluate the model on unseen data, with a larger testing set to enhance the external validity of the experimental results. Although the training set contained fewer documents, each document comprised a large number of HTML tokens or elements, providing a sufficient number of instances for model learning.

The validation set was used to monitor model performance during training and to prevent overfitting. Model evaluation was conducted by calculating precision, recall, and F1-score for each class, along with the average F1-score as an overall performance indicator.

Experiments followed a controlled variable approach, where only one parameter was varied at a time while others were held constant. This allowed for precise identification of each parameter's contribution to the model's performance. Four key parameters were tested: number of hidden units (64, 128, 256, 512), dropout rate (ranging from 0.0 to 0.9), number of LSTM layers (1 to 3 layers), and size of the dense layer (64, 128, 256, 512). The results of each configuration were analyzed to determine which parameter settings yielded the best performance in extracting the main content from complex HTML structures.

As a baseline, the Readability.JS library was used. Readability.JS is a widely used JavaScript-based library for simplifying web page views by extracting the core content of a document. Its algorithm is heuristic-based, relying on features such as <article> or <main> tags, text length, and link density. Although lightweight and fast, its ability to handle unconventional page structures is limited. Therefore, evaluation of Readability.JS was conducted using the same dataset and metrics as BoilerNet, ensuring a fair and objective comparison.

6. Results and Discussions

6.1 Hidden Unit Tuning

Testing the effect of hidden unit size was conducted using the BoilerNet model, focusing on four configurations: 64, 128, 256, and 512 units. The goal of this experiment was to observe how variations in the internal memory capacity of the Bi-LSTM layers affect the model's ability to extract the main content from scientific web pages.

Based on Table 4, the performance of the model with different hidden unit configurations can be observed. The best results were obtained with 256 hidden units, achieving an average F1-Score of 0.855 and the highest F1-Score for the positive class (main content) at 0.80, with a recall of 0.81.

Table 4: Model performance with different hidden unit configurations

Hidden Unit	Negative Class			Positive Class			Avg F1
	P	R	F1	P	R	F1	
64	0.90	0.91	0.91	0.79	0.78	0.79	0.851
128	0.90	0.91	0.91	0.80	0.79	0.78	0.854
256	0.91	0.90	0.91	0.79	0.81	0.80	**0.855**
512	0.91	0.89	0.90	0.77	0.80	0.78	0.846

All configurations maintained a consistent F1-Score of around 0.90 for the negative class (boilerplate), but the ability to correctly identify the main content (positive class) varied. The smallest hidden unit sizes (64 and 128) led to reduced F1-Scores for the positive class, at 0.79 and 0.78 respectively, indicating that limited memory capacity can restrict the model's ability to differentiate content effectively.

On the other hand, increasing the hidden size to 512 did not result in performance gains. While the F1-Score for the negative class remained at 0.90, the F1-Score for the positive class dropped to 0.78, and the average F1-Score decreased to 0.846. This suggests that overly complex models may overfit to irrelevant patterns in the data, reducing generalization ability. The optimal configuration was therefore found to be 256 hidden units, offering the best balance between memory capacity and classification performance across both classes.

6.2 Dropout Rate Tuning

To investigate the effect of dropout rate on model generalization, the BoilerNet model was tested with ten dropout configurations: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Dropout serves as a regularization method by randomly deactivating a portion of neurons during training, thereby reducing overfitting and encouraging the model to learn more robust patterns.

Table 5: Model performance across different dropout rates

Dropout Rate	Negative Class			Positive Class			Avg F1
	P	R	F1	P	R	F1	
0.0	0.89	0.91	0.90	0.79	0.76	0.78	0.844
0.1	0.91	0.90	0.90	0.78	0.80	0.79	0.849
0.2	0.91	0.87	0.89	0.75	0.81	0.78	0.840
0.3	0.92	0.89	0.90	0.78	0.83	0.80	0.858
0.4	0.91	0.91	0.91	0.80	0.81	0.81	**0.863**
0.5	0.89	0.91	0.90	0.79	0.77	0.78	0.845
0.6	0.87	0.91	0.89	0.78	0.70	0.74	0.817
0.7	0.90	0.91	0.91	0.80	0.77	0.79	0.850
0.8	0.92	0.89	0.90	0.77	0.82	0.80	0.854
0.9	0.91	0.91	0.91	0.80	0.81	0.81	0.862

Based on Table 5, the model achieved the highest average F1-Score of 0.863 with a dropout rate of 0.4, making it the most effective configuration among all tested. At this rate, the model reached an F1-Score of 0.81 for the positive class (main content), along with balanced precision (0.80) and recall (0.81). This indicates that moderate regularization

successfully prevents overfitting while preserving the model's ability to generalize across varying HTML structures.

In contrast, configurations with low dropout (e.g., 0.0 and 0.2) showed decreased performance, particularly in precision, leading to lower F1-Scores for the positive class (0.78). Excessive dropout, such as 0.9, also led to performance degradation. Although the negative class performance remained relatively stable across all settings (F1-score ranging from 0.89 to 0.91), the positive class was more sensitive to changes, reflecting its higher complexity and structural variability.

Overall, the experiment confirms that a dropout rate of 0.4 provides the optimal balance between underfitting and overfitting, resulting in the most stable and accurate performance across both classes. This value was therefore selected for use in subsequent experiments.

6.3 LSTM Layer Tuning

To investigate the effect of dropout rate on model generalization, the BoilerNet model was tested with ten dropout configurations: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Dropout serves as a regularization method by randomly deactivating a portion of neurons during training, thereby reducing overfitting and encouraging the model to learn more robust patterns. Table 6 presents the classification performance for each LSTM layer configuration.

Table 6: Model performance across different LSTM layer configurations

LSTM Layer	Negative Class			Positive Class			Avg F1
	P	R	F1	P	R	F1	
1	0.92	0.89	0.91	0.78	0.83	0.80	0.858
2	0.91	0.91	0.91	0.80	0.81	0.81	**0.863**
3	0.90	0.91	0.91	0.81	0.81	0.81	0.862

As shown in Table 6, the model achieved the best average F1-score of 0.863 using two LSTM layers. This configuration maintained stable performance on the negative class with an F1-score of 0.91, while achieving balanced precision (0.80) and recall (0.81) for the positive class (main content). This suggests that two layers provide sufficient depth to learn structural and contextual patterns from HTML blocks without sacrificing training efficiency.

In contrast, the single-layer configuration, while still competitive with an average F1-score of 0.858, showed slightly lower recall on the positive class (0.80), indicating limited depth in capturing complex variation between boilerplate and main content.

Adding a third layer did not significantly improve performance. Although it maintained an F1-score of 0.81 for the positive class, the average F1-score decreased slightly to 0.862. This marginal gain, coupled with increased model complexity and potential overfitting risks, highlights the diminishing returns of deeper LSTM stacks-especially when training on limited datasets.

Overall, the results indicate that two LSTM layers strike the best balance between representational capacity and computational efficiency. This configuration was therefore selected for subsequent experiments.

6.4 Dense Layer Size Tuning

This experiment evaluates the effect of varying the size of the dense layer in the model architecture. The dense layer plays a critical role as the final transformation stage before prediction, converting the extracted features into classification outputs. The tested configurations include dense sizes of 64, 128, 256, and 512. The goal is to determine which configuration offers the best balance between classification performance and computational efficiency. Table 7 shows the evaluation results across different dense layer sizes.

Table 7: Model performance across different dense size configurations

Dense Size	Negative Class			Positive Class			Avg F1
	P	R	F1	P	R	F1	
64	0.92	0.90	0.91	0.78	0.83	0.81	0.862
128	0.91	0.90	0.91	0.79	0.81	0.80	0.857
256	0.91	0.91	0.91	0.80	0.81	0.81	**0.863**
512	0.91	0.91	0.91	0.81	0.81	0.81	0.863

As shown in Table 7, the best performance was achieved when using a dense size of 256, with an average F1-score of 0.863. The model achieved 0.91 F1-score on the negative class and 0.81 F1-score on the positive class, with balanced precision (0.80) and recall (0.81). This configuration demonstrates that a 256-unit dense layer is sufficient to capture relevant high-level features from the LSTM output without unnecessary complexity.

In contrast, a dense size of 64 also achieved a decent F1-score (0.81) on the positive class, but the lower precision (0.78) suggests a higher rate of false positives. The 128-unit configuration resulted in slightly reduced performance, with an average F1-score of 0.857, indicating inadequate capacity to represent complex patterns extracted from the HTML sequences.

Although 512 units performed on par with 256 in terms of metrics, the increased computational load and risk of overfitting-especially with limited training data-make it less favorable. Larger dense sizes can slow down training and lead to diminished generalization capability if not properly regularized.

6.5 Performance Comparison with Readability.JS

After determining the optimal configuration for the BoilerNet model, a performance comparison was conducted against a rule-based method, Readability.JS. This comparison aimed to assess how well a deep learning-based approach can improve the accuracy and consistency of main content extraction from scientific journal web pages. Table 8 presents the comparative evaluation results of both methods across positive and negative classes.

Table 8: Performance comparison between BoilerNet and Readability.JS

	Negative Class			Positive Class			Avg F1
	P	R	F1	P	R	F1	
Readabiliy .JS	0.85	0.82	0.83	0.44	0.30	0.36	0.600
BoilerNet	0.91	0.91	0.91	0.80	0.81	0.81	0.863

As shown in Table 8, BoilerNet significantly outperforms Readability.JS, particularly in extracting main content (positive class). While both models perform reasonably well on the boilerplate class (negative class), with Readability.JS achieving an F1-score of 0.83, BoilerNet achieves a higher F1-score of 0.91, indicating a stronger consistency and precision in filtering out non-essential page elements.

The difference becomes even more prominent when examining the positive class, which represents the core journal content. Readability.JS only achieves a precision of 0.44, a recall of 0.30, and a resulting F1-score of 0.36. These metrics indicate a high rate of missed content and false positives. In contrast, BoilerNet demonstrates balanced and robust performance with a precision of 0.80, recall of 0.81, and F1-score of 0.81-more than doubling the effectiveness in identifying key content.

Overall, the average F1-score of BoilerNet reaches 0.863, significantly surpassing the 0.600 achieved by Readability.JS. This substantial margin confirms that deep learning models like BoilerNet are not only more adaptive to varying web structures but also more accurate and reliable in extracting essential content from scientific journal HTML pages.

7. Conclusions and Future Study

This study presents BoilerNet, a deep learning-based approach using a Bi-LSTM architecture for boilerplate removal and main content extraction from scientific journal web pages. The proposed method was tested on a dataset consisting of 180 HTML documents collected from various academic journal sources. Through a series of experiments, the model achieved its best performance using 256 hidden units, 0.4 dropout rate, two LSTM layers, and a dense size of 256, resulting in an average F1-score of 0.863, with 0.91 F1-score for the negative class (boilerplate) and 0.81 for the positive class (main content). Furthermore, when compared with a rule-based approach such as Readability.JS, BoilerNet outperformed significantly-Readability.JS only achieved 0.36 F1-score for the positive class, while BoilerNet achieved 0.81, demonstrating superior robustness in identifying meaningful content in varied HTML structures.

For future studies, the dataset can be expanded to include more diverse and multilingual scientific sources, potentially improving the generalization of the model across domains. Further research could also explore alternative architectures, such as Transformer-based models, to evaluate performance on long and complex HTML sequences. Additionally, this work could be extended to develop downstream tasks, such as scientific metadata extraction (title, authors, abstract) or integration into real-time indexing and intelligent search engines, enhancing practical usability in digital libraries and academic repositories.

Authorship Contribution Statement

Felix Dean Janitra: Responsible for writing first draft, conducting experiments, preparing the dataset, analyzing the performance results, and writing the initial draft of the manuscript.

Gusti Ahmad Fanshuri Alfarisy: Contributed to conceptualizing the research, refining the problem and objectives, revising the experimental design, analyzing the results, and providing revisions to the manuscript.

Aninditya Anggari Nuryono: Contributed to analyzing the results, validated the results, providing additional insights, and reviewed the final version of the manuscript.

All authors have reviewed and approved the final version of the manuscript for submission.

Declaration of Conflicting Interests

The authors declare that there are no competing interests that could have influenced the work of our study.

References

[1] J. Leonhardt, A. Anand, and M. Khosla, "Boilerplate Removal using a Neural Sequence Labeling Model," *Companion Proc. Web Conf. 2018*, pp. 226–229, 2020. doi: 10.1145/3366424.3383547.

[2] G. Jung, S. Han, H. Kim, K. Kim, and J. Cha, "Extracting the Main Content of Web Pages Using the First Impression Area," *IEEE Access*, vol. 10, pp. 129958–129969, 2022. doi: 10.1109/ACCESS.2022.3229080.

[3] J. Alarte and J. Silva, "Page-level main content extraction from heterogeneous webpages," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 6, pp. 1–105, 2021. doi: 10.1145/3451168.

[4] J. Bevendorff, S. Gupta, J. Kiesel, and B. Stein, "An Empirical Comparison of Web Content Extraction Algorithms," in *Proc. 46th ACM SIGIR Conf.*, 2023. doi: 10.1145/3539618.3591920.

[5] L. Alakukku, "Domain-Specific Boilerplate Removal from Web Pages with Entropy and Clustering," *Tesis*, Aalto University, 2022.

[6] E. Akbiyik, J. Almeida, R. Melis, R. Sriram, V. Petrescu, and V. Vilhjálmsson, "Semantic Outlier Removal with Embedding Models and LLMs," *arXiv preprint*, arXiv:2506.16644, Jun. 2025. doi: 10.48550/arXiv.2506.16644.

[7] M. Fernández-Pichel and M. Prada-Corral, "An Unsupervised Perplexity-based Method for Boilerplate Removal," *Nat. Lang. Eng.*, vol. 30, no. 1, 2024. doi: 10.1017/S1351324923000049.

[8] M. J. Hamayel and A. Y. Owda, "A novel cryptocurrency price prediction model using GRU, LSTM and bi-LSTM machine learning algorithms," *AI*, vol. 2, no. 4, pp. 477–496, 2021. doi: 10.3390/ai2040030.

[9] Z. Vujovic, "Classification Model Evaluation Metrics," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, pp. 599–606, 2021. doi: 10.14569/IJACSA.2021.0120670.

[10] M. Desai and M. Shah, "An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN)," *Clin. eHealth*, vol. 4, pp. 1–11, 2020. doi: 10.1016/j.ceh.2020.11.002.

[11] A. A. Ifty, "Introduction to the Perceptron and Its Applications," 2023. doi: 10.13140/RG.2.2.10439.10404.

[12] A. Ismail and K. Kuppusamy, "Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 3, pp. 901–911, 2019. doi: 10.1016/j.jksuci.2019.03.011.

[13] M. A. Khder, "Web scraping or web crawling: State of art, techniques, approaches and application," *Int. J. Adv. Soft Comput. Appl.*, vol. 13, no. 11, pp. 1–15, 2021. doi: 10.15849/IJASCA.211128.11.

[14] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate Detection using Shallow Text Features," in *Proc. 3rd ACM Int. Conf. Web Search Data Min. (WSDM '10)*, pp. 441–450, 2010. doi: 10.1145/1718487.1718542.

[15] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks," *Comput. Ind.*, vol. 131, 2021. doi: 10.1016/j.compind.2021.103498.

[16] C. Muehlethaler and R. Albert, "Collecting data on textiles from the internet using web crawling and web scraping tools," *Forensic Sci. Int.*, vol. 322, 2021. doi: 10.1016/j.forsciint.2021.110753.

[17] M. Dampfhoffer, T. Mesquida, and J. Kummert, "Backpropagation-based learning techniques for deep spiking neural networks: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. doi: 10.1109/TNNLS.2023.3263008.

[18] K. Sharifani and M. Amini, "Machine Learning and Deep Learning: A Review of Methods and Applications," *World Inf. Technol. Eng. J.*, vol. 10, no. 7, pp. 3897–3904, 2023. [Online]. Available: https://ssrn.com/abstract=4458723

[19] V. Singrodia, A. Mitra, and S. Paul, "A Review on Web Scraping and its Applications," in *2019 Int. Conf. Comput. Commun. Informatics (ICCCI)*, Coimbatore, India, pp. 1–6. doi: 10.1109/ICCCI.2019.8821809.

[20] T. Vogels, O. Ganea, and C. Eickhoff, "Web2Text: Deep structured boilerplate removal," in *Lecture Notes in Computer Science*, pp. 167–179, 2018. doi: 10.1007/978-3-319-76941-7_13.

[21] X. Wen and W. Li, "Time Series Prediction Based on LSTM-Attention-LSTM Model," *IEEE Access*, vol. 11, pp. 48322–48331, 2023. doi: 10.1109/ACCESS.2023.3276628.