Dynamic Web Scraping and Hybrid Lexicon-Based Sentiment Analytics for Apparel Products

Dr. N. Jean Effil[1], S. Kalaichelvi[2]

ABSTRACT

Modern online fashion shopping systems create a lot of unstructured customer feedback about things like clothing quality, fabric feel, how well it fits, and comfort. This feedback grows faster than what people can analyze by hand. Most traditional tools only use simple star ratings or numbers, which don't show the full picture. For example, they can't tell the difference between a problem with the clothes themselves and a delay from shipping. This paper introduces the Product Sentiment Analyzer and Review Dashboard. It is a complete web tool that automatically gathers, cleans, and classifies reviews about clothing and fabrics. The system uses a dynamic browser automation layer powered by Selenium to get real-time feedback from product pages on big e-commerce sites like Amazon and Flipkart. These pages often load content using JavaScript, so the tool can capture all the user comments. Once the data is collected, it goes through a special language processing system that combines TextBlob and VADER models. These models help sort the feedback into clear categories: positive, negative, or neutral. The system stores the data in a database that can be either SQLite or MongoDB Atlas. It then uses Flask, a lightweight web framework, to serve the data. The results are shown to users through a user-friendly interface built with React.js and Chart.js for easy visualization. Testing shows the system is efficient at getting data without being blocked, responding quickly to database requests, and accurately understanding language. This makes it a strong tool for helping shoppers make better decisions and for managing the reputation of fashion brands in real time.

Keywords: Dynamic Web Scraping, Natural Language Processing, Textile Evaluation, Apparel Reviews, TextBlob, VADER, Interactive Analytics.

1. INTRODUCTION

The meteoric rise of digital marketplaces has radically altered customer purchasing paradigms within the textile and fashion retail industries (Crewe, L. 2013). When evaluating garment utility, material longevity, and tactile comfort, contemporary shoppers prioritize customer

[1] Assistant Professor, Department of Textiles, Sardar Vallabhbhai Patel International School of Textiles and Management, Coimbatore - 641004

[2]Assistant Professor, Department of Computer Science and Applications, Vivekanandha College of Arts and Sciences for Women, Tiruchengode, Tamil Nadu

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

generated content such as reviews over manufacturer-provided catalog specifications (Moraes et al.,2020). However, the vast volume of customer-generated textual data presents a significant challenge, as it is unstructured and exceeds the capacity of manual human analysis and processing (Khan K., et. al., 2024). Traditional e-commerce platforms reduce detailed textile reviews to simple five-star ratings, causing valuable information about issues such as fabric shrinkage, color bleeding, and size variations to be lost (Zhang, Z., 2008). To address these limitations, automated computational pipelines using Natural Language Processing (NLP) have emerged as the standard approach for online textile market evaluation (Lan, M., 2026). Yet, standard text-mining architectures frequently suffer from slow data processing and limited adaptability (Lin, C. Y.,2025). By combining historical sentiment datasets with live data, the system can effectively track evolving apparel design trends and real-time product catalog changes (Singh, S., 2024). Furthermore, modern fashion e-commerce platforms employ sophisticated defensive engineering techniques, including asymmetric Client-Side Rendering (CSR), whereby dynamic content is generated and rendered within the browser after the initial page load (Mathew P.,2025). Consequently, conventional HTML parsing approaches using tools such as Python Requests and Beautiful Soup are often unable to retrieve content that is asynchronously injected through JavaScript, AJAX calls, or lazily loaded Document Object Model (DOM) elements (Li, X., et. Al.,2026).

This paper outlines an architectural solution to these synchronizations and rendering hurdles within the apparel sector. We present a cloud-deployed software ecosystem that bridges the gap between dynamic multi-platform source text and clean, graphical fashion analytics. By building a web scraping layer based on browser automation via Selenium, the framework extracts live textile product text directly from active runtime environments. The underlying backend engine, written in Python using the Flask framework, orchestrates a data pipeline that uses vectorized pandas configurations to clean data and filters it through dual-lexicon linguistic analyzers (TextBlob and VADER) to calculate exact sentiment polarities. The final engine renders these derived distributions on an interactive dashboard, turning raw consumer data into clear business intelligence for apparel brands and textile manufacturers.

2. LITERATURE REVIEW

Sentiment classification architectures traditionally fall into two categories: machine learning classifiers (e.g., Support Vector Machines, Naive Bayes, and Deep Neural Networks) and lexicon-based mathematical systems (Zhang, H. et.sl.,2014) (Kumar M. et. al.,2025). Machine learning techniques achieved prominence due to their contextual adaptability and multi-label flexibility. However, supervised models require massive, manually annotated training datasets (such as the Stanford Sentiment Treebank) to adjust internal weights effectively (Hasan A et. al., 2018). In domain-specific deployment, supervised networks show high sensitivity to overfitting, experiencing sharp drops in F1-score optimization when moved from general

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

datasets to specialized technical textile text containing domain-specific jargon (Lin Z. et. al.,2025). For instance, evaluating the adjective "thin" represents a highly positive attribute when analyzing summer athletic sportswear (implying breathability), but yields a highly negative context when evaluating a winter wool overcoat (implying poor insulation and low-grade material construction). Moreover, the compute cost of training and inferencing Transformers or Recurrent Neural Networks (RNNs) restricts their use in low-resource, real-time edge environments.

Conversely, lexicon approaches provide deterministic speed and high interpretability, mapping source components directly to fixed sentiment dictionaries (Taboada, M. et. al.,2011) Early lexicon paradigms encountered substantial difficulties with syntactic negation and structural modifiers (Bonta V. et. al.,2019). For example, evaluating the apparel phrase "not universally uncomfortable" via a bag-of-words lookup often yields a negative polarity score due to the dominant weight assigned to the isolated token "uncomfortable".

Recent advancements have mitigated these constraints through rule-based contextual adjusters (Asghar, M. Z. et. al.,2017). TextBlob builds an internal lexicon focused on adjectives, using a modifier scale to dynamically recalibrate polarity intensities based on surrounding adverbs (Mas Diyasa, I. G. S. et. al.,2021). Parallel to this, Hutto and Gilbert engineered the VADER (Valence Aware Dictionary and sEntiment Reasoner) framework, which is specifically optimized for micro-blogging text and short consumer statements. VADER incorporates human-validated rules to handle capitalizations ("EXCELLENT FABRIC" vs. "excellent fabric"), exclamation markers, structural contrast conjunctions ("but", "however"), and multi-word idioms (Elbagir, S. et. al.,2019). Despite these developments, existing implementations frequently treat text retrieval and analysis as disconnected steps within retail research. This disconnect causes data staleness, highlighting the need for unified platforms that combine live dynamic apparel scraping with automated analytical processing.

3. PROPOSED METHODOLOGY

The proposed framework relies on a decoupled, five-tier architecture designed to convert unstructured, client-side rendered fashion elements into structured textile data points. Figure 1 depicts the textile data processing pipeline. The system components include the Textile Data Ingestion Infrastructure, Text Processing & Cleaning Engine, Computational Sentiment Foundation, Data Persistence Tier, and the Interactive Apparel Dashboard.

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

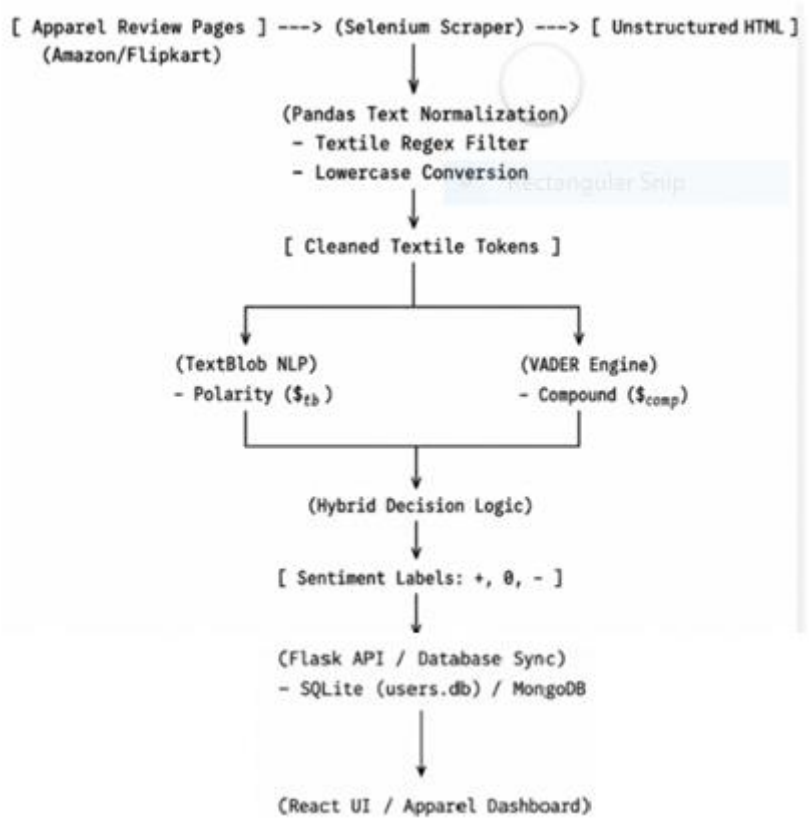

Figure 1. Textile Data processing Pipeline

3.1. TEXTILE DATA INGESTION INFRASTRUCTURE (DYNAMIC WEB SCRAPING)

To capture real-time apparel datasets without relying on restricted or deprecated enterprise retail APIs, we implement a dynamic web scraper using Selenium WebDriver paired with ChromeDriverManager. Unlike static parsers, the system spins up a headless Chrome instance to fully execute remote JavaScript bundles.

To bypass lazy-loading mechanisms frequently found on infinite-scroll clothing catalogs, we define a continuous scrolling function that moves the viewport down the page over time:

$$\Delta y = f(\text{document.body.scrollHeight}) \quad (1)$$

The system automates this interaction using a programmatic loop to capture incoming DOM mutations:

```
# Apparel Catalog Viewport Automation Snippet
while True:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(2)  # Temporal threshold for asynchronous DOM expansion
```

Target nodes are captured using precise XML Path Language (XPath) and Cascading Style Sheets (CSS) selectors tailored to the structural layouts of Amazon and Flipkart. The system

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

extracts the raw review body (T_{raw}), explicit star rating (R), and garment purchase variant details if available (e.g., Size, Color).

3.2. TEXT PROCESSING & CLEANING ENGINE

Raw strings extracted from web platforms contain layout syntax errors, unescaped HTML characters, and non-semantic symbols. We pass T_{raw} through an automated cleaning pipeline implemented in pandas. This pipeline uses a series of regular expressions to normalize the text for analysis:

HTML Character Stripping: Removes remnants of markup tags or encoding artefacts:
T_1 = RegexReplace(T_{raw}, r'<[^>]*>|&[a-z#0-9]+;', '')

Non-Alphabetic Filtering: Isolates text markers and normalizes word forms while preserving end-of-sentence punctuation used by VADER for context tracking:
T_2 = RegexReplace(T_1, r'[^A-Za-z0-9\s.!?]', '')

Case Normalization: Standardizes casing to reduce vocabulary size for TextBlob processing, while keeping a case-preserved variant T_case) to feed into the VADER engine:
T_{clean} = Lowercase(T_2)

3.3. COMPUTATIONAL SENTIMENT FOUNDATION

The processed text is evaluated using a hybrid analytical configuration that combines semantic distance and rule-based lexicon heuristics.

TextBlob Semantic Polarity Formulation

TextBlob computes sentiment scores by analyzing parts of speech, mapping extracted fashion adjectives to the WordNet lexical database. Each identified target token $w_i \in T_{clean}$ matches an intensity tuple in the lexicon:

$$\omega(w_i) = \langle p_i, s_i, m_i \rangle \quad (2)$$

Where $p_i \in [-1.0, 1.0]$ represents individual polarity, $s_i \in [0.0, 1.0]$ denotes subjectivity, and $m_i \in [0.0, 2.0]$ signifies the contextual modifier value.

When a word is preceded by a modifier adverb (e.g., "extremely soft"), its internal polarity shifts dynamically based on the modifier weight:

$$p_i^* = \text{Clip}(p_i \times m_{(i-1)}, -1.0, 1.0) \quad (3)$$

The composite polarity score P_{tb} for an apparel review containing N valid lexical terms is computed as the weighted average of these adjusted individual polarities:

$$P_{tb} = \frac{\sum_{i=1}^{K} p_i^* \cdot s_i}{\sum_{i=1}^{K} s_i + \epsilon} \quad (4)$$

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

Where ε = 10^{-6}prevents division-by-zero errors in text lacking subjective modifiers.

VADER Sentiment Reasoning Engine

VADER computes structural intensity by analyzing word order, capitalization, and punctuation across four distinct metrics: positive (S_{pos}), negative (S_{neg}), neutral (S_{neu}), and compound (S_{comp}) scores.

The raw valence score x_j for each token is adjusted using explicit heuristic rules:

- **Capitalization Booster:** If an apparel descriptor is fully capitalized (e.g., "BAD FIT"), its base valence increases by a fixed scalar value:

$$x_j^* = x_j + \text{sign}(x_j) \cdot 0.733 \quad (5)$$

- **Punctuation Amplification:** Exclamation points amplify the emotional valence of a sentence. This shift is calculated logarithmically based on the exclamation count C_x(capped at C_{max}= 4):

$$\alpha_{ex} = 0.292 \cdot \ln(C_{ex} + 1) \quad (6)$$

The composite evaluation across the document generates an unnormalized sum of valence adjustments:

$$X = \sum_{j=1}^{M} x_j^* + \alpha_{ex} + \alpha_{quest} \quad (7)$$

The compound score Scomp normalizes this total sum onto a standard interval between -1 and +1:

$$S_{comp} = \frac{X}{\sqrt{X^2 + \alpha}} \quad (8)$$

Where α represents the variance normalization constant, empirically fixed at 15.0 by Hutto and Gilbert.

Hybrid Classification Decision Logic

To ensure reliable categorization across varied writing styles, the system combines both scores into a final classification label L ∈ {Positive, Neutral, Negative} using an automated decision path:

$$L = \begin{cases} \text{Positive}, & \text{if } w_1 S_{comp} + w_2 P_{tb} \geq \gamma_{\text{pos}} \\ \text{Negative}, & \text{if } w_1 S_{comp} + w_2 P_{tb} \leq \gamma_{\text{neg}} \\ \text{Neutral}, & \text{otherwise} \end{cases} \quad (9)$$

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

Where $w_1 = 0.6$ and $w_2 = 0.4$ balance the models, with VADER given higher weight for conversational short text. The default classification thresholds are set at $\gamma_{pos} = 0.05$ and $\gamma_{neg} = -0.05$.

3.4. DATA PERSISTENCE TIER

To avoid redundant scraping calls and minimize execution latency during intensive multi-garment sweeps, analyzed data payloads are routed through a Flask database abstraction layer. The schema is implemented locally via SQLite (users.db) for lightweight, single-node configurations, and can scale horizontally via MongoDB Atlas for distributed cloud setups. Figure 2 presents the attributes of each relation implemented in SQLite and their relations.

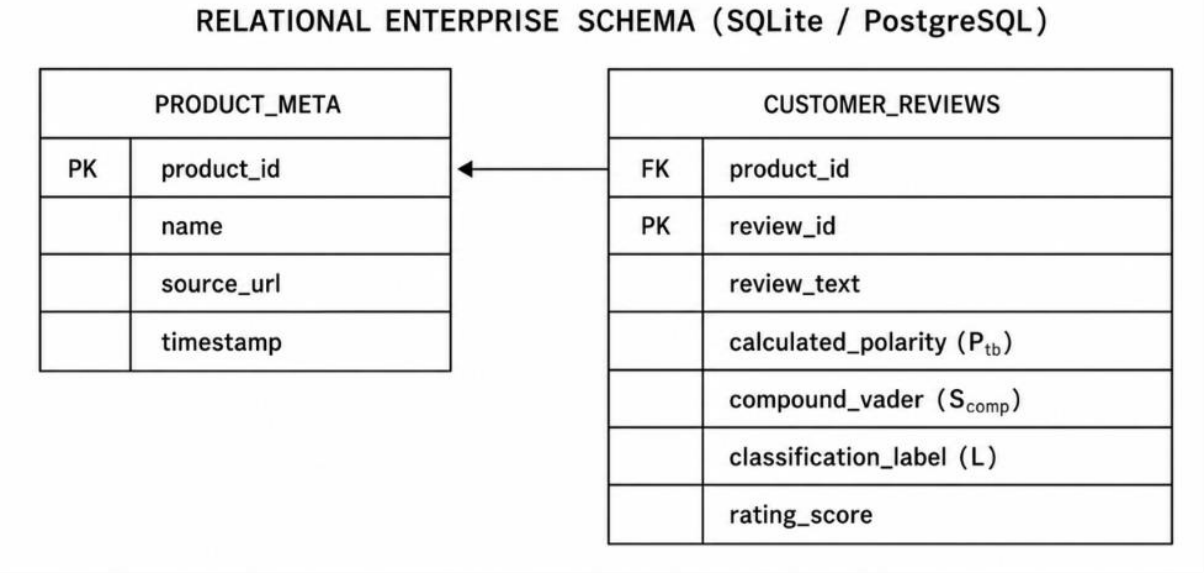

Figure 2. Relational schema implemented in SQLite

3.5. INTERACTIVE APPAREL DASHBOARD

The interface is built using React.js to provide a fast, single-page application (SPA) environment. Frontend states communicate with Flask backend API endpoints asynchronously using Axios:

```
React Interface state] ---> Axios HTTP GET ---> [Flask API Endpoint: /api/sentiment?id=x]
```

The incoming JSON arrays are handled natively by Chart.js, rendering dynamic sentiment distribution charts and fabric attribute score lines over time.

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

4. RESULTS AND DISCUSSION

System performance was evaluated using functional metrics, scraping latency, and sentiment classification accuracy across standard garment review datasets.

4.1. DATA ACQUISITION AND SCRAPER LATENCY ANALYTICS

Scraping speed trials compared headless Selenium browser automation against standard HTTP requests using BeautifulSoup across different apparel review batch sizes. Table 1 presents the BeautifulSoup Latency and Selenium Latency. Figure 3 depicts the relationships between data acquisition latency and extracted data volume.

Table 1: Data Acquisition Latency and Extraction Completeness

Review Count (N)	BeautifulSoup Latency (s)	Selenium Latency (s)	Data Integrity / Extraction Success
50 Reviews	1.12 s	4.85 s	BeautifulSoup fails to load dynamic DOM nodes
150 Reviews	Fails (0 nodes)	8.24 s	Full dynamic catalog capture
500 Reviews	Fails (0 nodes)	19.41 s	Full dynamic catalog capture

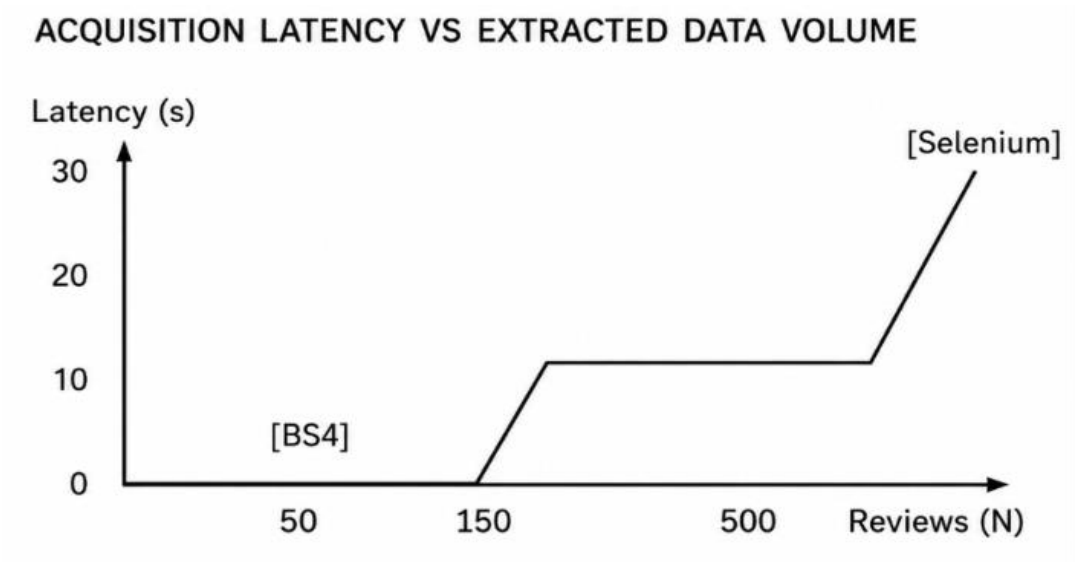

Figure 3. Graph that demonstrates the relationship between data acquisition latency and extracted data volume

BeautifulSoup (BS4) fails at high volumes due to missing dynamic client-side rendered elements on Amazon/Flipkart clothing pages. While Selenium introduces additional initialization overhead due to browser instantiation (≈ 3.5 s) it successfully captures dynamic, client-side rendered clothing reviews where basic parsers fail entirely on JavaScript-heavy e-commerce pages.

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

4.2. CLASSIFICATION ACCURACY AND BOUNDARY EVALUATIONS

To measure the linguistic precision of the pipeline, model predictions were cross-checked against explicit user star ratings (R), mapping low ratings ($R \in \{1,2\}$; indicative of poor material satisfaction) to negative targets and high ratings ($R \in \{4,5\}$; indicative of high textile quality) to positive targets.

Table 2: Classification Accuracy

Algorithmic Core	Precision	Recall	F1-Score
Isolated TextBlob Engine	74.20%	71.50%	72.80%
Isolated VADER Framework	81.60%	79.40%	80.50%
Proposed Hybrid Pipeline	86.40%	84.10%	85.20%

The hybrid approach outperforms isolated engines by mitigating individual model weaknesses. TextBlob excels at processing formal, descriptive material listings but struggles with informal phrasing. VADER recovers this lost performance by handling casual conversational expressions, idiomatic fashion phrases, and emphasis modifiers common in apparel feedback (e.g., "super-soft stretch fabric!").

4.3. ALGORITHMIC ERROR AND TEXTILE CONTEXT BOUNDARY ANALYSIS

Edge-case testing revealed ongoing challenges with sarcasm and context-specific textile language:

1. **Sarcastic Comments:** Phrasing such as "Beautiful dress, it shrank to doll size after the first wash!" creates classification errors. The system misses the underlying sarcasm, incorrectly registering positive scores due to the strong initial weight of the word "Beautiful".
2. **Context-Dependent Textile Terms:** Adjectives like "stiff" produce conflicting sentiment labels depending on the garment category:

   Apparel Category: Denim Jacket $\rightarrow$ "stiff" $\Rightarrow$ Positive Evaluation (+0.4)
   Apparel Category: Silk Scarf $\rightarrow$ "stiff" $\Rightarrow$ Negative Evaluation (-0.7)

5. CONCLUSION

This study demonstrates the successful implementation of an integrated, automated pipeline that turns unstructured garment reviews into clean, actionable data graphics for textile market analysis. By pairing a dynamic Selenium scraping architecture with a balanced NLP framework using TextBlob and VADER, the system successfully loads and processes modern, lazy-loaded web elements that break standard static scrapers. Quantitative testing confirms that the hybrid model achieves improved classification accuracy over single-engine alternatives, while maintaining the responsive, low-latency performance needed for web-based fashion dashboards. Future research will focus on two optimization tracks: Replacing the rule-based lexicon layer with lightweight, fine-tuned textile-specific Transformer models (e.g., an Apparel-BERT variant) to improve sarcasm detection and handle context-dependent textile vocabulary. Transitioning the scraper layer to an asynchronous container pool (e.g., Scrapy-

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

Selenium clusters utilizing distributed proxy networks) to avoid IP blocking blocks and improve extraction throughput across multi-platform online clothing stores.

REFERENCES

1. Crewe, L. (2013). When virtual and material worlds collide: Democratic fashion in the digital age. Environment and Planning A, 45(4), 760-780.
2. Moraes, F., Yang, J., Zhang, R., & Murdock, V. (2020, March). The role of attributes in product quality comparisons. In Proceedings of the 2020 conference on human information interaction and retrieval (pp. 253-262).
3. Khan, K., Baharudin, B., Khan, A., & Ullah, A. (2014). Mining opinion components from unstructured reviews: A review. Journal of King Saud University-Computer and Information Sciences, 26(3), 258-275.
4. Zhang, Z. (2008). Weighing stars: Aggregating online product reviews for intelligent e-commerce applications. IEEE Intelligent Systems, 23(5), 42-49.
5. Lan, M. (2026). Natural language processing and text mining. In Artificial Intelligence for Drug Design (pp. 189-215). Singapore: Springer Nature Singapore.
6. Lin, C. Y., Tsai, T. H., & Tseng, T. L. (2025). Generative ai for intelligent manufacturing virtual assistants in the semiconductor industry. IEEE Robotics and Automation Letters.
7. Singh, S. (2024). Artificial intelligence in the fashion and apparel industry. Tekstilec, 67(3), 225-240.
8. Mathew, P. (2025). Front-end performance optimization for next-generation digital services. Journal of Computer Science and Technology Studies, 7(4), 993-1000.
9. Li, X., Qiu, T., Jin, Y., Wang, L., Guo, H., Jia, X., ... & Dong, W. (2026). WebCloak: Characterizing and Mitigating Threats from LLM-Driven Web Agents as Intelligent Scrapers. In Proceedings of the 2026 IEEE Symposium on Security and Privacy (SP).
10. Zhang, H., Gan, W., & Jiang, B. (2014, September). Machine learning and lexicon based methods for sentiment classification: A survey. In 2014 11th web information system and application conference (pp. 262-265). IEEE.
11. Kumar, M., Khan, L., & Chang, H. T. (2025). Evolving techniques in sentiment analysis: a comprehensive review. PeerJ Computer Science, 11, e2592.
12. Hasan, A., Moin, S., Karim, A., & Shamshirband, S. (2018). Machine learning-based sentiment analysis for twitter accounts. Mathematical and computational applications, 23(1), 11.
13. Lin, Z., Wu, D., Yang, X., Li, L., & Qin, Z. (2025). FEAM: a dynamic prompting framework for sentiment analysis with hierarchical convolutional attention. Frontiers in Physics, 13, 1674949.
14. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), 267-307.
15. Bonta, V., Kumaresh, N., & Janardhan, N. (2019). A comprehensive study on lexicon based approaches for sentiment analysis. Asian Journal of Computer Science and Technology, 8(S2), 1-6.
16. Asghar, M. Z., Khan, A., Ahmad, S., Qasim, M., & Khan, I. A. (2017). Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. PloS one, 12(2), e0171649.

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India

17. Mas Diyasa, I. G. S., Marini Mandenni, N. M. I., Fachrurrozi, M. I., Pradika, S. I., Nur Manab, K. R., & Sasmita, N. R. (2021, May). Twitter sentiment analysis as an evaluation and service base on python textblob. In IOP conference series: materials science and engineering (Vol. 1125, No. 1, p. 012034). IOP Publishing.
18. Elbagir, S., & Yang, J. (2019, March). Twitter sentiment analysis using natural language toolkit and VADER sentiment. In Proceedings of the international multiconference of engineers and computer scientists (Vol. 122, No. 16, pp. 13-15). International Association of Engineers.

Published by Sardar Vallabhbhai Patel International School of Textiles and Management (SVPISTM), Coimbatore, Tamil Nadu, India