International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 13s, 2025
https://theaspd.com/index.php/ijes

Web Page Evolution Monitoring: Algorithms, Tools, And Techniques For Detecting Content Changes

Ramavat Kamal Narendra[1], Prof.(Dr.) Ashutosh A. Abhangi[2]
[1]Computer Science & Engineering, ITM Vocational University, Vadodara, India, kamal.ramavat @gmail.com
[2]*Computer Science & Engineering*, ITM Vocational University, *Vadodara, India*, ashutosha@itmvu.in

Abstract— *In this age of increasing globalization and tough competition, people need to gather knowledge and information for their development in both professional and personal aspects. But the constantly updating data becomes more challenging to manage due to the vast strings of pieces of information that are available. To simplify access, WWW provides a key database resource. It hosts countless web pages, encompassing both static and dynamic content. Content like blog posts, live breaking news, sports scores, and traffic updates falls under dynamic web page, as their information is frequently refreshed. The modifications or changes done could be adding new data, deleting some information, or changing what already exists. Many tools focusing on change detection have been created to track alterations made over time on these web pages. In this paper I review categories along with algorithms and tools used for structured documents that allow tracking relevant changes needed.*

Keywords— *Data management, World Wide Web (WWW), web pages, XML, JSON, static pages, dynamic pages, change detection, tracking alterations, structured documents, algorithms, web monitoring, content updates, patching, querying modified data, CDN systems, website content, website structure, automatic tracking, user notifications.*

1.INTRODUCTION

The volume of web documents is growing rapidly, and the frequency of changes within these documents is rising just as quickly. XML serves as a standard format for sharing information across the web, enabling seamless data exchange [2]. In both industry and academia, XML documents are widely used for storing and transferring data. A critical challenge is ensuring these documents remain unchanged during transfer, whether due to intentional or accidental modifications. This makes effective change detection in XML documents essential to maintain data integrity. Many websites frequently update their content [3], making it essential to monitor and identify these changes. Detecting and analyzing changes in web documents is a promising research area with significant potential. These changes can be evaluated using various factors, including shifts in content, layout alterations, presentational updates, behavioral modifications, and changes resulting from type adjustments. Content changes involve edits to text, such as modifying <rating> scores, <book> names, or <published> years. Layout changes take place when tags are reordered within the document, like swapping the <writer> and <genre> tags. Presentation changes occur when the visual style of book data is altered, for example, through adjustments to font, color, or size. Behavioral changes arise when a data element, such as a <purchase_link> tag, responds unexpectedly. Type changes happen when a tag is replaced by another, making the original tag inaccessible or causing it to behave abnormally. Examine below example considering XML format of a book from library database m as depicted in Figure 1.

```
<library>
  <book name="Introduction to Algorithms">
        <writer>Thomas H. Cormen</writer>
        <published>2009</published>
        <genre>Computer Science</genre>
        <rating>4.8</rating>
        <similar_book name="Algorithms Unlocked" author="Thomas     H. Cormen"/>
        <similar_book name="Data Structures and Algorithms in Python"
 author="Michael T. Goodrich"/>
        <purchase_link>url to online store</purchase_link>
  </book>
</library>

<library>
    <book name="Clean Code">
          <writer>Robert C. Martin</writer>
          <published>2008</published>
          <genre>Software Engineering</genre>
          <rating>4.7</rating>
          <similar_book name="Refactoring" author="Martin Fowler"/>
          <similar_book name="The Pragmatic Programmer" author="Andrew Hunt"/>
          <purchase_link>url to online store</purchase_link>
      </book>
</library>
```

Figure – 1. Library XML Document

To detect changes and identify duplicate in web page and such as structured documents, researchers have developed various change detection algorithms and tools [4]. This paper provides a concise overview of these algorithms, which are designed to recognize operations like insertion, deletion, updating, and repositioning within structured documents. These operations drive the changes observed in web pages. Typically, the first documents are transferred in a DOM based structure. After that depends on the specific use case, an algorithm is employed to detect changes. The method operates by evaluating two distinct versions of a document, identifying discrepancies through the computation of differences between nodes within their individual tree hierarchies.

2. LITERATURE REVIEW BASED ON VARIOUS CHANGE DETECTION ALGORITHMS ON WEB DOCUMENT / XML

Considering the literature review, this paper's author identifies the paper related to the various types of algorithms/tools and techniques that are used for change detection system. Based on this author review the algorithm to identify the glitch of security over the web document for that author observes the parameter of the security concern.

A key challenge in analyzing structured documents is detecting changes that occur over time. Researchers have created several change detection algorithms to spot difference from document. Table-1.1 presents the most effective techniques for detecting changes in documents with hierarchical structures. From there one widely recognized method is the diff utility, which has inspired the development of various algorithms [1]. The core concept of these change detection techniques involves representing a structured document as a tree and then comparing the trees of different document version.

Sr.No.	Algorithm / Tools Name	Summery
1	Johnson Algorithm	In the constantly evolving documents of web, It calculates the differences between two web pages by evaluating the weighted impact of adding or removing keywords, content, or paragraphs. This approach was designed to identify updates made to those web pages. If the calculated difference exceeds a predefined threshold, the document may be discarded [6]. However, if the difference falls within an acceptable range, users are notified about the specific changes that occurred.
2	Shingling based algorithm	The shingling technique addresses the problem of detecting web documents that are almost identical. It utilizes a broad approach by forming similarity scores based on n-gram representations extracted from the content. These representations are analyzed using a similarity measure, and when the resulting value surpasses a defined limit, the documents are classified as near to the duplicates. In this process, the algorithm constructs shingle vectors by converting n-grams into 64-bit Rabin fingerprints [5]. The level of similarity between two documents is then assessed through the Jaccard similarity coefficient applied to these vectors.
3	Fuzzy Structure Comparison Technique Algorithm	Based on this Algorithm, as its name suggests, it identifies changes into web based documents by transforming them into structure of tree and comparing the differences between two trees using a fuzzy based structured function as "hash" which applied to their nodes. It filters out irrelevant changes by first establishing which changes are considered significant and which are not. Based on this criterion, it evaluates the computed DOM tree against a threshold value to determine and specify the changes made into document. [8]

4	Proportional Difference Analysis Algorithm	The proportional method expands upon the Johnson algorithm by determining the difference between two documents after initially normalizing their weighted attributes. It does this to generating a signatures for each relevant document parameter and then measuring changes proportionally across all parameters [6]. This signature-based method is widely applied to various web pages. Its systematic approach delivers improved results for detecting and notifying changes.
5	Cosine Algorithm	The Cosine Algorithm is utilized to identify context-driven modifications in web pages. The process starts by estimating the weighted term vectors for each webpage through a heuristic method. These terms, along with their weights, are saved into a file for later use. Afterward, the algorithm merges these vectors to identify the context terms and weights for web pages that have not changed. Finally, it calculates the cosine similarity angle and compares it to the previously recorded value to determine how similar the pages are [7].
6	Algorithm of CX Diff	The method involves to create a XML file with DOM tree in a distributed computing setup. Its main goal of that to track and report the modification based on developer requests. It operates under the assumption that the tree's nodes follow a specific order, although their attributes might not. To detect changes, this algorithm uses the merging of push-pull method by Event-Condition-Action(ECA) to identify and extract the updates [10,28].
7	Algorithm of X-Diff	The Wang and colleagues explored the algorithm of X-Diff, which stand out by an without ordered tree structure, unlike most algorithms that rely on structured documents represented in an ordered format [11]. This method is mainly used for database tools or applications. The algorithm matches node signatures and compares two document versions to identify changes. By calculating the minimum-cost matching, it generates a minimal edit script to address the detected differences [29].
8	Algorithm of Vi-DIFF	Vi-DIFF, as its name implies, is an algorithm designed to detect visual changes on web pages. It can identify modifications in text, images, links, or the overall visual style. The process begins by segmenting web pages into visual semantic blocks. Then, By applying a method of change detection, it compares one or more documents with block nodes [12].
9	Algorithm of LA Diff	LaDiff is a detection of change algorithm developed specifically for tracking modifications in LaTeX based documents. It uses a technique called Fast Match, which is not suitable for handling changes in XML-based structures. Although LaDiff is primarily focused on LaTeX and works with elements such as insertions, deletions, updatation, and movements, its methods are limited to certain LaTeX-specific constructs. As a result, its Fast Match strategy does not translate well to XML document formats. Nonetheless, the core concepts behind LaDiff have influenced the development of several other methods. Fast Match itself is based on identifying the longest common subsequence between nodes, beginning the matching process from the leaf level of the document tree. [13]

10	Algorithm of MH Differences	MH Differences stands by Meaningful Hierarchical Differences, is the method designed to find significant data change in without ordered hierarchical data structures. Beyond standard operations like inserting, updating, and deleting, it also supports copying and moving entire subtrees. This compares tree transformation into cover problem of edges., which enables the generation of an optimized sequence of changes requires the tree transformation from one to another. It efficient approach focuses on identifying only the most semantically relevant changes [14].
11	XyDiff	The XyDiff algorithm is employed for identifying alterations within XML documents. It operates by performing a comparison between two XML documents. This process initiates from the lowest levels of the document's tree structure, progressively moves to upwards, and entails calculating a unique hash value and a corresponding weight for each individual node. Every node is assigned a distinct identifier. Furthermore, the document's organization adheres to a prefix order facilitated by an XidMap. The algorithm actively seeks out subtrees that are identical. Upon discovering such a match, it then employs specific rules to select relevant nodes. As the algorithm traverses the tree structure in an upward direction, the nodes associated with these nodes are concurrently updated [15].
12	DiffX	DiffX is designed to identify changes between two XML document versions [16]. XML data is first converted into a tree format to start the process, where each node is clearly labeled and categorized by type. The algorithm then searches for the largest portion of the tree that remains consistent between both versions by comparing individual nodes. After this, it performs a series of detailed passes through the entire structure to locate any alterations in node content or layout. These detected changes are recorded in an edit script, which can later be used to update the main document accordingly.
13	MM-XM Diff Algorithm	Sudarshan and colleagues explain that MMDiff is a method for comparing structured documents that can fit entirely in a computer's main memory without needing extra storage [17]. The algorithm operates by creating a matrix through string-edit operations, which it then uses to produce a script outlining the differences. XMDiff, a similar technique, is tailored for analyzing large-scale structured data that cannot be entirely loaded into main memory. Instead, it relies on external storage to manage and compare massive documents efficiently, generating an edit script with the lowest possible cost.
14	Algorithm of KF-Diff+	Xu and colleagues explain KF Diff+ as a streamline-based algorithm that quickly detects changes in both ordered and unordered structured tree models [18]. Unlike traditional methods that rely on tree-to-tree correction, KF-Diff+ compares key trees—labeled trees with unique paths. It identifies multi-instance nodes (nodes with the same parent and label) to remove duplicate paths, calculates a key path, and uses it to match the trees, Calculating the lowest possible effort or resource usage required to identify modifications

Table : 2.1 Algorithm Summery

3. USE CASE FOR THE SYSTEM OF DETECTION FOR CHANGE

This part outlines several key domains where change detection techniques are commonly applied. These algorithms play a crucial role across various fields by identifying modifications in data, structure, or content.

3.1 Monitoring Modifications in Web Content

WebDAV, a protocol built upon HTTP, facilitates remote access and editing of web-based documents. It

provides functionality for users and providers to modify, create or move files on a server, while maintaining metadata such as edit history, contributor details, and grouped resources. Functioning similarly to a network-based file system, it supports version control, file sharing, and locking mechanisms to prevent conflicts during collaborative work.

3.2 Synchronizing Data Across Dynamic Platforms

To keep data aligned in a constantly changing environment, you can use the git-merge tool to blend updates from different versions of documents or web pages. It combines shared changes, and if conflicts arise during the merge, kdiff3 helps resolve them, producing a single document free of duplicates. This approach works for merging two documents and can extend to three [20]. In a 2-way merge, the system tracks changes in documents or websites and integrates them into one file. For a 3-way merge, you maintain the original file alongside two altered versions. [21].

3.3 *Managing Versions of XML-Based Documents*

A document version control system is used to monitor and record every update made to XML files and shared content. It ensures that each change is documented, making it easier for users to track modifications, retrieve earlier versions, and collaborate more efficiently. For example, a news website uses it to monitor updates and notifications sent to its audience. It stores every version of the data sent out and maintains a record of all previous versions. This ensures that any changes or updates to web pages are properly saved and organized. [19]

3.4 Visualize changes

Seeing changes clearly is a key part of any system that tracks updates. By visualizing changes, people can quickly spot what's different on a webpage. Most tools for this show the changes right within the same interface used towards the system of CDN. [23]

3.5 Applying Document Modifications

Change detection methods work by comparing two versions of a document to spot variations. Based on these findings, they generate a series of operations—such as adding, removing, or rearranging elements. When these operations are executed on the original document, it is effectively converted into its updated form.

3.6 *Improved Query system*

The enhanced query system ensures accurate delivery of current or historical data whenever a user requests it. It maintains a comprehensive record of all modified and unmodified values to support any inquiry. Change detection algorithms track all document updates, while the system also preserves a history of past values for reference. Document versioning is critical for this process. Additionally, users can set up notifications for specific changes, with triggers automatically activating when those updates occur. Continuous monitoring and oversight of documents are essential to optimize the query system's performance [22].

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 13s, 2025
https://theaspd.com/index.php/ijes

Overview of Change Identification Methods and Their Core Characteristics

No.	Name of Algorithm	Textual Modification	Structure Changes	Style Adjustments	Behavioral Changes	Concept Changes	Observation
1	Johnson's Algorithm	Yes	No	Yes	No	No	Measures how much two documents differ by calculating their unique distance.
2	Shingling based Algorithm	Yes	No	No	No	No	Spots duplicate content across websites or documents with ease.
3	Fuzzy Structure Comparison Technique Algorithm	Yes	No	No	No	No	Detects website changes using a flexible tree-based comparison approach.
4	Proportional Difference Analysis Algorithm	Yes	Yes	Yes	No	No	Compares both content and structure to highlight differences between documents.
5	Cosine Algorithm	No	No	No	No	Yes	Evaluates context shifts by comparing angles between content and change terms.
6	Algorithm of CX Diff	Yes	Yes	No	No	No	Pinpoints customized changes within XML documents.
7	Algorithm of X-Diff	Yes	Yes	No	No	No	Compares document tree structures, ignoring their order, to find
8	Algorithm of Vi-Diff	Yes	Yes	No	No	No	Reveals visual and meaningful differences between two HTML page versions.

9	Algorithm of LA-Diff	No	Yes	No	No	No	Examines structural differences in LaTeX documents with a fast matching method.
10	Algorithm of MH Differences	Yes	Yes	No	No	No	Compares hierarchically structured documents in a clear, meaningful way.
11	XyDiff	Yes	Yes	No	No	No	Identifies differences by comparing unique node signatures in documents.
12	DiffX	Yes	Yes	No	No	No	Maps matching sections of document trees for precise, isolated comparisons.
13	MM-XM Diff Algorithm	Yes	Yes	No	No	No	Finds differences in ordered trees using a memory-friendly shortest path method.
14	Algorithm of KF-Diff+	Yes	Yes	No	No	No	Simplifies tree comparisons by focusing on key tree structures.

4. RECENTLY AVAILABLE CHANGE DETECTION AND NOTIFICATION SYSTEM

The initial Change Detection and Notification (CDN) tool emerged from NetMind's innovation in 1996. Over the years, various platforms have been launched, including ChangeDetection in 1999 (later rebranded as Visualping), ChangeDetect in 2002, Google Alerts in 2003, Follow That Page, and Wachete. These systems have significantly advanced over time, featuring enhanced detection accuracy, streamlined web crawling methods, and intuitive notification systems for users. [1,4]. Today's intelligent update tracking systems play a vital role in keeping users aware of modifications made to websites and online documents. These tools automatically monitor content and promptly notify users whenever any changes occur, ensuring that they stay current without needing to manually revisit the sources. These tools alert subscribed users whenever website changes occur. Before sending notifications, factors like the timing, delivery method, and specific content to share must be considered. Today, CDN systems account for additional aspects, such as tracking changes across an entire website, specific sections of a page, or particular file types like PDFs, images, or text, as well as how often to send updates [25]. Various CDN tools are in use, including:

4.1 WebMonotor / Distill - 2013

Launched in 2013, WebMonitor is a smart tool designed to automatically detect changes on websites. It enables users to stay updated with the latest content whenever they choose to review it. By creating an account, individuals can receive real-time alerts via email, text, notifications, or pop-up messages on their mobile phones or other devices. Additionally, the tool keeps

track of different document versions and allows users to access the data in a format that suits their needs.

4.2 DataMiner

DataMiner is a cloud-based monitoring web scraping and automation platform developed by PromptCloud, a data-as-a-service (DaaS) company founded in 2014.[26].

4.3 Visualping (2017)

This is the tool which alerts users to website changes with a single click. You can pick specific parts of a site to monitor, and if anything changes, you'll get an email notification. The tool also offers a visual mode to compare changes side by side. Alternatively, you can track an entire webpage, and any updates will be highlighted for quick and easy viewing.[27]

4.4 Wachete (2014)

It monitors entire webpages or specific content, sending change alerts to users through email or app notifications. It also supports tracking protected pages, requiring users to provide login credentials.

4.5 ChangeTower (2017) – Monitor Website

Most popular a cloud based tool that automatically monitors websites. It offers a full suite of modern CDN features, including visual, code, and text comparison, snapshots, and alerts. Its robust monitoring system detects any website changes, such as modifications to keywords, images, visuals, or sentences, based on user specifications. Notifications are delivered through ChangeTower's feed.

4.6 Pagescreen (2018)

It is a widely used tool that combines multiple features into one platform. It is a data log management tool which saved web pages, making it valuable related to business or other market, thanks to its powerful change detection technology. The tool allows users to set up custom alerts, receiving notifications via email or Webhooks. With its Webhooks and Pagescreen features, users can seamlessly integrate the tool into their existing workflows.

5. DISCUSSION

Detecting changes in structured documents allows users to review and decide on modifications, store multiple versions of large datasets efficiently, and track updates to website content. Modern web pages, with their dynamic, user-tailored designs, pose challenges for algorithms trying to spot these changes. Focusing solely on static content isn't enough, as it rarely updates, while dynamic content shifts often and needs constant monitoring. The rise of multimedia and visual effects further complicates detecting changes in user interfaces. Surveys show that both new and older algorithms tend to prioritize content changes over

other types of updates. These algorithms mainly inform users about changes in advance, but they also support tasks like analyzing visual updates, applying patches, querying only modified data, and monitoring content shifts.

6. CONCLUSION

This paper explores a range of well-known change detection algorithms and where they're typically used. By laying out a table to compare these algorithms, it's clear that CDN systems are good at spotting changes in website content and structure, but there's still a lot of ground to cover when it comes to detecting shifts in presentation, behavior, type, or context. Developing tools to tackle these gaps would make it easier for users to decide whether to embrace updates or stick with older versions. The paper also sheds light on features in modern commercial tools, which alert users to changes in web documents based on their specific needs, using various algorithms to track updates automatically and keep users informed.

Acknowledgement

REFERENCES

[1] Mallawaarachchi, Lakmal Meegahapola, Roshan Madhushanka, Eranga Heshan, Dulani Meedeniya, and Sampath Jayarathna. 2020. Change Detection and Notification of Web Pages: A Survey. ACM Comput. Surv. 53, 1, Article 15 (January 2021), 35 pages. https://doi.org/10.1145/3369876
[2] G. Cobena, S. Abiteboul and A. Marian, "Detecting changes in XML documents," Proceedings 18th International Conference on Data Engineering, San Jose, CA, USA, 2002, pp. 41-52, doi: 10.1109/ICDE.2002.994696.
[3] Kyong-Ho Lee, Yoon-Chul Choy and Sung-Bae Cho, "An efficient algorithm to compute differences between structured documents," in IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 8, pp. 965-979, Aug. 2004, doi: 10.1109/TKDE.2004.19.
[4] Filip Radlinski, Paul N. Bennett, and Emine Yilmaz. 2011. Detecting duplicate web documents using clickthrough data. In Proceedings of the fourth ACM international conference on Web search and data mining (WSDM '11). Association for Computing Machinery, New York, NY, USA, 147–156. https://doi.org/10.1145/1935826.1935859
[5] Buttler, David. *A short survey of document structure similarity algorithms*. No. UCRL-CONF-202728. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2004
[6] Luis Francisco-Revilla, Frank Shipman, Richard Furuta, Unmil Karadkar, and Avital Arora. 2001. Managing change on the web. In Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries (JCDL '01). Association for Computing Machinery, New York, NY, USA, 67–76. https://doi.org/10.1145/379437.379973
[7] Zubin Dalal, Suvendu Dash, Pratik Dave, Luis Francisco-Revilla, Richard Furuta, Unmil Karadkar, and Frank Shipman. 2004. Managing distributed collections: evaluating web page changes, movement, and replacement. In Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries (JCDL '04). Association for Computing Machinery, New York, NY, USA, 160–168. https://doi.org/10.1145/996350.99638
[8] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. 2014. Relevant change detection: a framework for the precise extraction of

modified and novel web-based content as a filtering technique for analysis engines. In Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion). Association for Computing Machinery, New York, NY, USA, 595–598. https://doi.org/10.1145/2567948.2578039
[9] Cobéna, Grégory, Talel Abdessalem, and Yassine Hinnach. "Etude comparative sur la détection de changements en XML." *Delta* 10.100 (2002): 1000.
[10] S. Chakravarthy and S. C. H. Hara, "Automating Change Detection and Notification of Web Pages (Invited Paper)," 17th International Workshop on Database and Expert Systems Applications (DEXA'06), Krakow, Poland, 2006, pp. 465-469, doi: 10.1109/DEXA.2006.34.
[11] Wang, Y., D. DeWitt, and J. Y. Cai. "X-Diff: A Fast Change Detection Algortihm for XML Documents." *Proc. of 19th IEEE Intl. Conf. on Data Engineering*. 2003.
[12] Pehlivan, Z., Ben-Saad, M., Gançarski, S. (2010). Vi-DIFF: Understanding Web Pages Changes. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds) Database and Expert Systems Applications. DEXA 2010. Lecture Notes in Computer Science, vol 6261. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15364-8_1
[13] Sebastian Rönnau, Geraint Philipp, and Uwe M. Borghoff. 2009. Efficient change control of XML documents. In Proceedings of the 9th ACM symposium on Document engineering (DocEng '09). Association for Computing Machinery, New York, NY, USA, 3–12. https://doi.org/10.1145/1600193.1600197
[14] Hannes Dohrn and Dirk Riehle. 2014. Fine-grained change detection in structured text documents. In Proceedings of the 2014 ACM symposium on Document engineering (DocEng '14). Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/2644866.2644880
[15] Peters, Luuk. "Change detection in XML trees: a survey." In *3rd Twente Student Conference on IT*, vol. 7, no. 2.1, p. 2. 2005.
[16] Al-Ekram, Raihan, Archana Adma, and Olga Baysal. "diffX: an algorithm to detect changes in multi-version XML documents." In *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research*, pp. 1-11. 2005.
[17] Chawathe SS. Comparing hierarchical data in external memory. InVLDB 1999 Sep 7 (Vol. 99, pp. 90-101).
[18] Xu, H., Wu, Q., Wang, H., Yang, G., Jia, Y. (2002). KF-Diff+: Highly Efficient Change Detection Algorithm for XML Documents. In: Meersman, R., Tari, Z. (eds) On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE. OTM 2002. Lecture Notes in Computer Science, vol 2519. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36124-3_80
[19] S. Berlik and M. Fathi, "Differences of Structured Documents - Improving their Quality," 2007 IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, USA, 2007, pp. 486-491, doi: 10.1109/IRI.2007.4296667.
[20] Thao, C., Munson, E.V. Using versioned trees, change detection and node identity for three-way XML merging. *SICS Softw.-Inensiv. Cyber-Phys. Syst.* **34**, 3–16 (2019). https://doi.org/10.1007/s00450-013-0253-5
[21] Julia Rubin and Marsha Chechik. 2013. N-way model merging. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013). Association for Computing Machinery, New York, NY, USA, 301–311. https://doi.org/10.1145/2491411.2491446
[22] Tchendji, Maurice Tchoupé, Lionel Tadonfouet, and Thomas Tébougang Tchendji. "A tree pattern matching algorithm for XML queries with structural preferences." *arXiv preprint arXiv:1906.03053* (2019).
[23] A. Oliveira, Oliveira, Tessarolli A, Gabriel G, Gleiph P, Bruno C, Fernando M, Oliveira M, Rodrigues C, Igor K, Souza M, Murta U, Braganholo L and Vanessa 2018 An efficient similarity-based approach for comparing XML documents *Information System* **78** pp 40–57.
[24] Hyungik Oh, L. Jalali and R. Jain, "An intelligent notification system using context from real-time personal activity monitoring," 2015 IEEE International Conference on Multimedia and Expo (ICME), Turin, Italy, 2015, pp. 1-6, doi: 10.1109/ICME.2015.7177508.
[25] fSharma Chakravarthy, Jyoti Jacob, Naveen Pandrangi, and Anoop Sanka. "Webvigil: An approach to just-in-time information propagation in large network-centric environments." *Proc. of 2nd Int. Workshop on Web Dynamics in Conjunction with Eleventh Int. World Wide Web Conference*. 2002.
[26] https://dataminer.io/how-it-works
[27] Visualping. 2017. Visualping: #1 Website change detection, monitoring and alerts. VisualPing. Retrieved November 9, 2019 from https://visualping.io/
[28] Levene, Mark, et al. "Web dynamics, structure, and page quality." *Web dynamics: Adapting to change in content, size, topology and use* (2004): 93-109.
[29] Bhat, S.Y., Jan, F., & Abulaish, M. (Eds.). (2026). Community Structure Analysis from Social Networks (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781003515890