Preprint from: **https://www.gipp.com/pub/**

M. Beck, M. Schubotz, V. Stange, N. Meuschke, B. Gipp, "Recognize, Annotate, and Visualize Parallel Content Structures in XML Documents", in Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL), 2021.

Recognize, Annotate, and Visualize Parallel Content Structures in XML Documents

Marco Beck
Data & Knowledge Engineering
University of Wuppertal
Wuppertal, Germany
beck@gipplap.org

Moritz Schubotz
Mathematics
FIZ Karlsruhe
Berlin, Germany
moritz.schubotz@fiz-karlsruhe.de

Vincent Stange
OriginStamp AG
Kreuzlingen, Switzerland
vincent.stange@originstamp.com

Norman Meuschke
Data & Knowledge Engineering
University of Wuppertal
Wuppertal, Germany
meuschke@uni-wuppertal.de

Bela Gipp
Data & Knowledge Engineering
University of Wuppertal
Wuppertal, Germany
gipp@uni-wuppertal.de

***Abstract*— We present a four-phase parallel approach for capturing, annotating, and visualizing parallel structures in XML documents. We designed a highlighting strategy that first decomposes XML documents in various data streams, including plain text, formulae, and images. Second, those streams are processed with external algorithms and tools optimized for specific tasks, such as analyzing similarities or differences or differences in the respective formats. Third, we compute comparison metadata such as annotations and highlighting marks. Fourth, the position information is concatenated based on the original XML's computed positions document. Eventually, the resulting comparison can then be visualized or processed further while keeping the reference to the source documents intact. While our algorithm has been developed for visualizing similarities as part of plagiarism detection tasks, we expect that many applications will benefit from a well-designed and integrative method that separates between addressing the match locations and inserting highlight marks. For example, our algorithm can also add comments in XML-unaware plaintext editors. We also treat the edge cases, overlaps as well as multi-match with our approach.**

Keywords*— *XML Documents, parallel structures, compare XML Documents, XML highlighting, XML annotating

I. Introduction

XML is an important and widely used format (e.g., HTML, ODT, and docx are all based on XML) for representing, storing, and exchanging data in the form of documents for numerous use cases. In science, TEI (digital humanities) and JATS (MINT disciplines) are important XML-based document formats. The Text Encoding Initiative (TEI) has also become a de facto standard within the humanities, [1] where it is used, for example, to encode printed works (edition science) or to mark up linguistic information (linguistics) in texts. Examining documents for similarities or differences, storing these corresponding results, and visualizing them for users is expected in document processing. Important use cases would be, for example, tracking changes in collaboratively edited documents (Microsoft Word / Open Office) or the detection of plagiarism in scientific publications. Particularly in the area of academic plagiarism [2], where it goes as far as the highly covert reuse of content, e.g., by paraphrasing or translating of the text, and finally, the reuse of data or reuse of data or ideas without proper attribution [3], investigation of similarities in documents are of elementary importance. Therefore, detecting hidden academic plagiarism in research publications is an urgent problem that concerns many stakeholders, including academic publishers, research institutions, funding agencies, and, of course, other researchers.

Another area of data analyses relates to digital editions, the core area of digital humanities, investigations of similarities, and general comparisons in complex textual and semistructured datasets are helpful for further research questions and reuse [4]. Moreover, (semi-)automatic processing steps (e.g., text recognition in manuscripts and inscriptions, heuristic and inferential statistical detection of structural relationships in and empirical analyses of language and text corpora, etc.) and their systematic evaluation (e.g., image analysis, metadata enrichment, directed information, graphical models, word embeddings, interaction and social networks, etc.) are of elementary importance in digital editions. The approach of Rosselli et al. within the open-source tool EVT - Edition Visualization Technology uses the Digital Vercelli Book as an example to show how digital editions can be searched, explored, and studied [5].

The XML format offers the advantage that hierarchical structures and user-defined tags allow flexible data representation [6]. However, this advantage is challenging when comparing for similarities or differences [7, 8, 9] since

different content in XML documents must be analyzed using different algorithms. A second challenge arises when something within an XML document needs to be annotated, modified, or deleted and the relation to the originals must be preserved. This is the case, for example, when comparing two XML documents for similarities and then highlighting the similarities.

II. Comparison to the State of the Art

Many tools support the highlighting of differences between documents (so-called diff tools). Mostly, these tools focus on comparing native source code and managing code changes. These tools can also be integrated into common source code management systems, for example. As a rule, such tools also make the presupposition that the documents must be essentially similar and therefore not fundamentally different, since the output or the comparison result then also becomes very difficult to interpret. There are three standard layouts for the different tools (uniform, double and triple). Also, the positions of the different text fragments are highlighted in the scrollbars of the individual documents. Figure one shows an example of the double layout by displaying a diff comparison in the tool Code Compare.

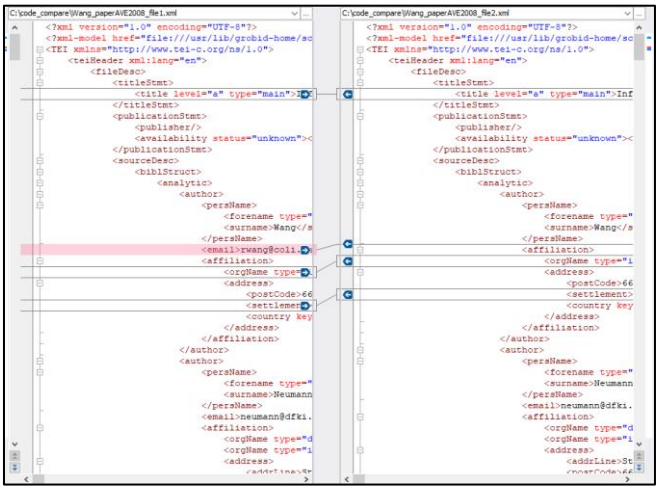

Fig. 1. File comparison and merging using Code Compare by Devart

Also, specialized solutions exist, which compare files in the same format, such as images or plain text files. Still, no tools can highlight similarities of different types (such as image and text), which is an important prerequisite for visualization of multimodal diffs.

However, there are two main differences between different tools to merge different texts in particular source code and highlight a parallel structure in XML documents. First, XML documents contain structural, textual, and information predefined presentation standards such as SVG images or MathML formulae. In case to highlight those non-textual elements or fractions of it, special treatment is necessary. Second, in source code outputs, overlapping highlighting is less problematic. This is because there are only three types of annotations, insert delete, and move. For XML documents, corresponding sections might be determined by several analytics algorithms, and thus the overlapping of highlights is more complex. In particular, the highlighting can span through different formatting instructions.

Third, the analytical tasks are more diverse. While for source code, the task is usually to resolve conflicting change proposals or understand a change in the source code, investigating similarities in documents is more diverse. The overview first paradigm is more important in this context.

III. Our Approach

We present a four-stage approach for identifying, annotating, and highlighting parallel content structures in XML documents. Our approach address scenarios in which two documents shall be analyzed for similar content, specifically similar text, images, and formulae. This scenario is particularly relevant for scientific document processing use cases, such as plagiarism detection or editorial theory.

In the first stage, an XML or HTML document is decomposed into plain text, formulae, and images. These elements are processed with appropriate external algorithms for text similarity or difference analysis. Then, these elements can be concatenated with the positions of the original XML document to form minimal, non-overlapping elements and insert, for example, the type-specific highlight marker. To avoid highlighting constraints, matching groups are introduced, the corresponding tags are moved, and the part to be highlighted is split accordingly. Thus, many applications will benefit from a well-designed method that separates addressing matching locations from inserting highlighting. In our approach, we also consider edge cases, overlapping matches, as well as multi-match. Our approach and tool is a further development of HyPlag [10, 11], which provides a template or command-line tool for extracting different data streams from the XML document, annotating them, and then reassembling the original XML tags with the plain text, images, and formulae using our algorithm.

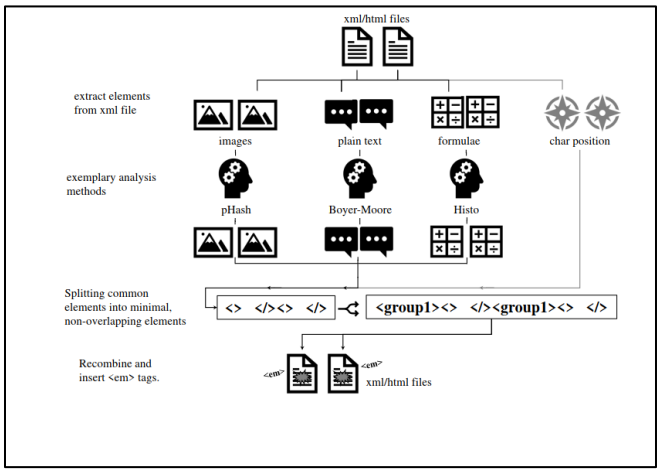

Fig. 2. Overview of highlighting parallel structures in XML documents

Phase 1 Decomposition of the elements of the XML document and notation of the respective positions from the XML document

```
    <text xml:lang="en">
        <body>
            <div>
                <head n="1">Introduction and Related Work</head>

                <p>Answer Validation is an important step for Question Answering (QA)
systems, which aims to validate the answers extracted from natural language texts, and
select the most proper answers for the final output.</p>

                <p>Using Recognizing Textual Entailment (RTE-1 – <ref type="bibr"
target="#b2">Dagan et al., 2006</ref>; RTE-2 – <ref type="bibr" target="#b0">Bar-Haim et
al., 2006</ref>) to do answer validation has shown a great success (<ref type="bibr"
target="#b8">Peñas et al., 2007</ref>). We also developed our own RTE system and
participated in AVE2007. The RTE system proposed a new sentence representation extracted
from the dependency structure, and utilized the <b>Subsequence Kernel method</b> (<ref
type="bibr" target="#b1">Bunescu and Mooney, 2006</ref>) to perform machine learning. We
have achieved fairly high results on both the RTE-2 data set (<ref type="bibr"
target="#b9">Wang and Neumann, 2007a</ref>) and the RTE-3 data set (<ref type="bibr"
target="#b10">Wang and Neumann, 2007b</ref>), especially on Information Extraction (IE)
and QA pairs. However, on the AVE data sets, we still found much space for the
improvement. Therefore, based on the system we developed last year, our motivation this
year is to see whether using extra information, e.g. <b>namedentity (NE)
recognition</b>, <b>question analysis<b>, etc., can make further improvement on the
```

Fig. 3. Extract from the input XML document

The XML/HTML document is decomposed into plain text, mathematical expressions, and images in the preprocessing phase. In the next step, all XML/HTML formatting instructions, links, and tags are removed. However, we note and store the resulting plaintext string positions where each XML/HTML tag was removed during this extraction. After doing this for both XML/HTML documents, we have two plaintext files and a list of the tags removed from these text files and their location in the original XML document.

```
                Answer Validation is an important step for Question Answering (QA)
systems, which aims to validate the answers extracted from natural language texts, and
select the most proper answers for the final output.

                Using Recognizing Textual Entailment (RTE-1 –  ; RTE-2 –  ) to do
answer validation has shown a great success ( ). We also developed our own RTE system
and participated in AVE2015. The RTE system proposed a new sentence representation
extracted from the dependency structure, and utilized the USP method ( ) to perform
machine learning. We have achieved fairly high results on both the RTE-2 data set ( )
and the RTE-3 data set ( ), especially on Information Extraction (IE) and QA pairs.
However, on the AVE data sets, we still found much space for the improvement. Therefore,
based on the system we developed last year, our motivation this year is to see whether
using extra information, e.g. namedentity (NE) recognition, question analysis, etc., can
```

Fig. 4. Extract from extracted plain text

Phase 2 Find common elements in the documents and identify changed positions.

In this second phase, the plaintext can then be identified using a standard text similarity or difference algorithm (e.g., Encoplot [12], Boyer-Moore [13]), and the formulae and images can be identified using appropriate algorithms using specialized partially XML unaware tools (e.g., also Frequency Histograms of Mathematical Identifiers (Histo), Longest Common Subsequence of Identifiers (LCIS), [14] Perceptual hashing (pHash), Positional text matching, [10]Bibliographic Coupling (BC), Longest Common Citation Sequence (LCCS), Citation Chunking (CC)[[15], [16]]) are processed. Alternatively, characters or words can be added, modified, or deleted manually in the extracted plain text. After processing the respective elements with external algorithms or manual modifications, we evaluate the returned positions' changes to the input strings' integer position. Using our position list generated in the first phase, we can relate these matches to the positions of the respective XML/HTML tags in the original documents.

```
{0=357, 3=367, 8=386, 15=430, 34=462, 41=512, 45=529, 52=557, 61=581, 165=701,
172=730, 179=759, 188=775, 195=804, 204=820, 211=838, 220=863, 239=899, 248=923,
253=944, 260=963, 265=983, 270=1002, 277=1045, 287=1068, 294=1107, 304=1130,
311=1171, 318=1191, 325=1229, 333=1250, 340=1296, 350=1319, 357=1346, 366=1393,
377=1413, 387=1433, 396=1455, 403=1483, 410=1503, 419=1527, 460=1584, 467=1605,
474=1627, 483=1667, 494=1684, 507=1706, 513=1722, 526=1748, 539=1775, 550=1793,
561=1840, 572=1889, 581=1908, 588=1931, 595=1955, 679=2045, 686=2066, 695=2100,
722=2134, 843=2293, 860=2328, 867=2350, 874=2383, 883=2400, 890=2416, 899=2431,
903=2442, 910=2460, 917=2475, 919=2486, 926=2500, 927=2509, 934=2523, 936=2533,
943=2547, 945=2557, 952=2577, 961=2607, 996=2664, 1005=2689, 1009=2710, 1018=2812,
1029=2834, 1256=3073, 1265=3092, 1272=3113, 1277=3133, 1280=3144, 1283=3153,
1288=3161, 1894=3800, 1895=3808, 1896=3842, 1897=3850, 2156=4113, 2161=4121,
```

Fig. 5. Extract from the record of the positions of the XML-tags

Phase 3 Splitting common elements into minimal, non-overlapping elements.

However, we need to ensure that no XML tags are in the matched span to implement highlighting correctly. Otherwise, the highlighting could interfere with other formatting instructions, such as *bold*. Therefore, we introduce matching groups and split the range to be highlighted into as many fractions as leaf nodes are affected in the XML/HTML document and assign the same group tag to the splits. We repeat this method for all algorithms and all content types to be compared. In addition to the split tag problem, we now face the challenge of overlapping groups in leaf elements. To avoid this, we split the groups into non-overlapping highlights and adjust the mathematical group information accordingly. Note that this means that identical highlights are merged into one. Also, contextual information, such as highlighting or additional comments, is stored in the group and not in the highlighting itself.

Phase 4 Recombine match groups and insert highlight tags.

Based on the group information, we now add the highlighting tags to the sheet elements. The highlighting depends on the content type of the sheet. We keep an expandable list that assigns a highlighting method to each sheet type. For example, for the HTML tags head, title, base, link, meta, style, body, article, header, footer, div, figure, data, ruby, span, we use the em-tag for highlighting.

```
    <text xml:lang="en">
        <body>
            <div>
                <head n="1">Introduction and Related Work</head>

                <p>Answer Validation is an important step for Question Answering (QA)
systems, which aims to validate the answers extracted from natural language texts, and
select the most proper answers for the final output.</p>

                <p>Using Recognizing Textual Entailment (RTE-1 – <ref type="bibr"
target="#b2">Dagan et al., 2006</ref>; RTE-2 – <ref type="bibr" target="#b0">Bar-Haim et
al., 2006</ref>) to do answer validation has shown a great success (<ref type="bibr"
target="#b8">Peñas et al., 2007</ref>). We also developed our own RTE system and
participated in <em>AVE2007</em>. The RTE system proposed a new sentence representation
extracted from the dependency structure, and utilized the <b><em>Subsequence</em> Kernel
method</b> (<ref type="bibr" target="#b1">Bunescu and Mooney, 2006</ref>) to perform
machine learning. We have achieved fairly high results on both the RTE-2 data set (<ref
type="bibr" target="#b9">Wang and Neumann, 2007a</ref>) and the RTE-3 data set (<ref
type="bibr" target="#b10">Wang and Neumann, 2007b</ref>), especially on Information
Extraction (IE) and QA pairs. However, on the AVE data sets, we still found much space
for the improvement. Therefore, based on the system we developed last year, our
motivation this year is to see whether using extra information, e.g. <b>namedentity (NE)
recognition</b>, <b>question analysis<b>, etc., can make further improvement on the
```

Fig. 6. Extract from the composite output XML document

IV. Discussion and Future Work

Our results show that it is possible to extract the plain text, random images, and mathematical expressions separately from an XML document. Then, these separate types can be compared using various external tools, and the comparison results can be merged back with reference to the original document. In addition to pure comparison, this makes it possible to modify the text in human-readable form, delete and add characters, and then insert the XML tags and formatting instructions into this modified pure text. In addition, our suggested approach allows for a much better and more informed analysis of XML documents. Analysis can now be done at the level of individual elements, such as plain text, images, or mathematical expressions.

One challenge in extracting and compiling the plain text and XML tags is dealing with whitespace. By default, no whitespace is inserted between text and the respective XML

tags in an XML document. So if we remove the XML tags as part of the suggested approach, there will be no whitespace between each word in the plain text output file.

The following example

<postCode>66123</postCode><settlement>Saarbrücken</settlement><country key="DE">Germany</country>

illustrates that as soon as the XML tags are removed, the plain text is written together as in a continuous text.

66123SaarbrückenGermany

Of course, space can also be inserted or output during extraction, but then the calculated positions would not be correct when the changed text and the XML tags are later assembled. It might be helpful to note the position where the space is inserted during extraction so that this position can be determined again during composition. The position of the XML tags is finally corrected for the inserted spaces. Even if the algorithm is implemented that appropriate blanks are inserted, the problem can occur if the XML tag is set within a word that thereby the word is pulled apart based on the blank.

Another challenge is when words are inserted in plain text at the edge of an XML tag that the algorithm so far cannot detect in which XML tag the new or changed word should be inserted. Further considerations are required at these points.

V. Conclusion

We have presented a proposal for capturing, annotating, and visualizing parallel structures in XML documents. An XML document can first be decomposed into plain text, formulas, and images and processed and analyzed with external algorithms for text similarity or difference analysis. After processing the elements with external algorithms and inserting annotations, such as plain text or highlight marks, these elements can be concatenated based on the positions of the original XML document. To this end, we have also proposed a solution approach for highlighting constraints, edge cases, overlaps, and multiple matches. Our standalone software package, we have created can be integrated into various applications to split XML documents into different data streams for analysis, for example, and then reassemble the modified data streams into one XML document using the calculated positions in case of changes or highlighting.

Our Code is available as open-source at:

https://github.com/ag-gipp/parallelXmlHighlighting

Acknowledgment

This work was partially supported by the German Research Foundation (DFG) grant no. GI 1259/3-1.

References

[1] M. L. Jockers and R. Thalken, Text Analysis with R: For Students of Literature (= Quantitative Methods in the Humanities and Social Sciences), Cham: Springer International Publishing, 2020.

[2] T. Fishman, ""We know it when we see it"? is not good enough: towarda standard definition of plagiarism that transcends theft, fraud, and copyright. I," *Proc. Asia Pacific Conf. on Educational Integrity.*, 2009.

[3] D. Weber-Wulf, False Feathers: A Perspective on Academic Plagiarism, Berlin Heidelberg: Springer-Verlag, 2014.

[4] A. Busch, Visualisierung textgenetischer Phänomene in digitalen Editionen, A. Bosse and W. Fanta, Eds., Berlin: De Gruyter, 2019.

[5] R. Rosselli Del Turco, G. Buomprisco, C. Di Pietro, J. Kenny, R. Masotti and J. Pugliese, "Edition Visualization Technology: A Simple Tool to Visualize TEI-based Digital Editions," *TEI - Journal of the Text Encoding*, vol. Issue 8, 2014.

[6] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," 26 11 2008. [Online]. Available: https://www.w3.org/TR/xml/. [Accessed 01 04 2021].

[7] G. Barabucci, "diffi: diff improved; a preview," *DocEng '18: Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1-4, 2018.

[8] A. Oliveira, G. Tessarolli, G. Ghiotto, B. Pinto, F. Campello, M. Marques, C. Oliveira, I. Rodrigues, M. Kalinowski, U. Souza, L. Murta and V. Braganholo, "An efficient similarity-based approach for compari," *Information Systems*, no. 78, pp. 40-57, 2018.

[9] C. Thao and E. V. Munson, "Using versioned tree data structure, change detection and node identity for three-way XML merging," *DocEng '10: Proceedings of the 10th ACM symposium on Document engineering*, pp. 77-86, 2010.

[10] N. Meuschke, C. Gondek, D. Seebacher, C. Breitinger, D. Keim and B. Gipp, "An Adaptive Image-based Plagiarism Detection Approach," *Proc. ACM/IEEE Joint Conf. on Digital Libraries (JCDL).*, 2018.

[11] Data & Knowledge Engineering Group of Prof. Bela Gipp, "HyPlag," 2021. [Online]. Available: https://www.hyplag.org. [Accessed 25 03 2021].

[12] C. Grozea, C. Gehl and M. Popescu, ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection, Proc. PAN Workshop, 2009.

[13] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, 1977.

[14] N. Meuschke, M. Schubotz, F. Hamborg, T. Skopal and B. Gipp, "Analyzing Mathematical Content to Detect Academic Plagiarism," *Proc. Conf. on Inform. and Knowl. Manage. (CIKM)*, 2017.

[15] B. Gipp, Citation-based Plagiarism Detection - Detecting Disguised and Cross-language Plagiarism using Citation Pattern Analysis, Springer, 2014.

[16] B. Gipp and N. Meuschke, "Citation Pattern Matching Algorithms for Citation-based Plagiarism Detection: Greedy Citation Tiling, Citation Chunking and Longest Common Citation Sequence," *Proc. ACM Symp. on Doc. Eng*, 2011.