IPSJ SIG Technical Report

Vol.2013-DBS-157 No.14
Vol.2013-IFAT-111 No.14
2013/7/23

XML Documents Searching Combining Structure and Keywords Similarities

Apichaya Auvattanasombat[1,2,a)]   Yousuke Watanabe[1,b)]   Haruo Yokota[1,c)]

Abstract: In recent years, XML has been increasingly become an emerging standard and widely used in many applications. For example, office documents which are more and more popular used at this time, are also stored in multiple parts of XML archive formats. It is known that the structure and content of XML files play different roles depending on kind of documents. Therefore, achievement similarity search of an XML file should base on both structure and content. In previous work, LAX+ is an algorithm for reckoning a similarity value from structure and contents of XML files in the office documents. However, since LAX+ used exactly matching method between corresponding leaves, similar words in the leaf-nodes are considered as different. To solve the problem, we propose to combine LAX+ with keyword similarity in leaf-nodes. We use docx, xlsx and pptx file formats as experimental data set. The evaluation shows that our approach can be used to improve the precision and recall.

Keywords: XML Similarity, OOXML, Keyword Similarity, Document Search

1. Introduction

Over the past years, the Extensible Markup Language (XML) has been established as a major means for information management. It has been broadly utilized for complex data representation and many applications, because it can represent different kinds of data from multiple sources. Therefore, more and more information are published and exchanged by XML. Demands for searching XML documents become increasing and complicated.

Moreover, Microsoft Office Document format [1] is also based on XML file format and the amount of office document is still increasing on the Internet. As an application of XML searching, our research group has been developing a system named SOS (Structure-based Office document Search) [12] which finds office documents being similar to a given query document. To provide relevant search results to users, we need a method to calculate similarity of XML trees precisely.

Since, the content and the structure of XML trees play different roles and importance depending on kind and purpose of the document. So, a suitable solution for XML searching should consider both structure and content of XML trees. Hence, we propose the method which combining structure-based similarity with the content-based similarity.

This research can be divided into two parts; first we developed **KLAX**, a method to measure a similarity value between a pair of XML trees. KLAX is an improvement of our previous method LAX+ [12] by use a keyword similarity to match leaf nodes of subtrees in XML trees. Second, we propose **LAX&KEY**, a method to calculate the similarity between XML trees by utilizing both structure and content. LAX&KEY separately computes structure-based similarity from LAX+ and content-based similarity from keywords, then combines with weighting.

We apply KLAX and LAX&KEY to our SOS system instead of LAX+ to improve its precision. In our experimental evaluation, we use docx, xlsx and pptx file formats as experimental data set to investigate effects of our approach.

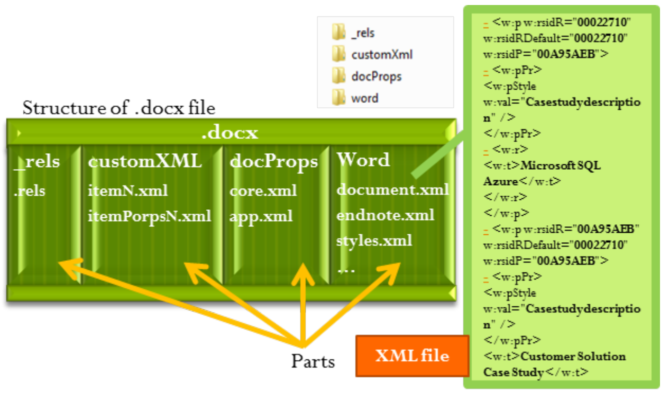
The remaining part of this paper is organized as follows: Section 2 introduces related work. Section 3 explains an OOXML document format. Section 4 overviews Structure-based Office document Search (SOS) and LAX+, and Section 5 explain our proposed methods. Section 6 shows our experimental results. Section 7 concludes this paper and mentions future research issues.

2. Related work

There are several search engines which have difference purposes and processes [2], [3], [4]. Recently, there are many kinds of documents and the quantity is increasing in high rate due to the information period. So, searching and clustering of those documents have become the issue that we need to solve. Since there are many kinds of documents which are depend on their structures and using purposes, the search engine should be different to be proper with them. For XML tree is also need specification for search engine.

We have various methods to measure XML similarity values [5], [7], [8], [9], [10]. The important factors that should be considered for the performance of the method are computational cost and precision. Some are popularly used because they are fast and some give high precision. LAX [11] and LAX+ [12] are algo-

1 Tokyo Institute of Technology, 2–12–1 Meguro, Tokyo 152–8552, Japan
2 Chulalongkorn University, 254 Phayathai Road, Pathumwan, Bangkok, 10260, Thailand
a) auvattanasombat.a.aa@m.titech.ac.jp
b) watanabe@de.cs.titech.ac.jp
c) yokota@cs.titech.ac.jp



```
- <w:p w:rsidR="00022710"
w:rsidRDefault="00022710"
w:rsidP="00A95AEB">
- <w:pPr>
<w:pStyle
w:val="Casestudydescriptio
n" />
</w:pPr>
- <w:r>
<w:t>Microsoft SQL
Azure</w:t>
</w:r>
</w:p>
- <w:p w:rsidR="00A95AEB"
w:rsidRDefault="00022710"
w:rsidP="00A95AEB">
- <w:pPr>
<w:pStyle
w:val="Casestudydescriptio
n" />
</w:pPr>
<w:t>Customer Solution
Case Study</w:t>
```

Fig. 1 Structure of OOXML

rithms to calculate similarity values of XML trees by comparing leaf nodes from their subtrees with low computational cost and high precision. Then, SOS [12] is the method for measuring the similarity values for OOXML documents from LAX+. In this paper, we modified LAX+ to improve the precision of the result and also applied our propose methods to SOS.

3. Office Open XML (OOXML or OpenXML)

OOXML file format [1] is a zipped, XML-based file format developed by Microsoft. We can reach internal XML files by extracting the zip archive. This file format can be used to represent electronic office documents. There are formats for word processing documents, spreadsheets and presentations as well as specific formats for material such as mathematical formulae, graphics, bibliographies etc.

OOXML documents are stored in Open Packaging Convention (OPC) packages, which are zip files containing XML and other data, along with a specification of the relationship between them. Depending on the type of the document, the packages have different internal directory structures and names. An OOXML document consists of multiple *Parts*, and each Part consists of one or more XML files. Figure 1 shows a structure of a docx file when it has been extracted.

4. An Overview of Previous Methods

Here, we introduce our previous work [12]. We have been developing SOS system for OOXML similarity search. It has to calculate similarities between sets of XML files in OOXML documents. LAX+ is our previous method to calculate similarity of XML trees. Firstly, we explain LAX+ and SOS in Section 4.1 and Section 4.2, then we describe the problem of LAX+ in Section 4.3.

4.1 LAX+

LAX+ is an algorithm to calculate similarity values between two XML trees. From a pair of XML trees (f_{base}, f_{target}), it returns $S_{xml}(f_{base}, f_{target})$. LAX+ is an symmetric similarity measure, so $S_{xml}(f_{base}, f_{target})$ is equal to $S_{xml}(f_{target}, f_{base})$.

The procedure of LAX+ to calculate a similarity value between a pair of XML trees is as follow:

(1) **Decomposing an XML tree to subtrees by choosing a cutting point**:
A cutting point is a node (element) to divide a XML tree into subtrees. For each node x in the XML tree, we apply the following formula to decide which node is most suitable to be a cutting point.

$$w_x = |children(x)| \times d_x^{\alpha} \quad (1)$$

$|children(x)|$ is the number of x's child nodes, d_x is the maximum depth from node x to leaf nodes, and α is a weight parameter. f_{base} is decomposed into a set of subtrees $T_{base} = \{t_u\}$, and f_{target} is also decomposed into $T_{target} = \{t_v\}$.

(2) **Calculating similarity values between subtrees**:
We make a pair of subtrees (t_u, t_v) from base's subtrees and target's subtrees. Similarity values between subtrees are derived by the following formula.

$$S_{sub}(t_u, t_v) = |Matched\_Leaf(t_u, t_v)| \quad (2)$$

$Matched\_Leaf(t_u, t_v)$ expresses a subset of leaf nodes in t_u, whose element exactly matches one or more leaf nodes in t_v. "Match" means strings (PCDATA) in leaf nodes are completely same.

(3) **Calculating a similarity value between two XML trees by deriving from similarity values of subtree pairs**:
$S_{xml}(f_{base}, f_{target})$ is a LAX+'s similarity value between XML trees f_{base} and f_{target}. It is defined by the following formula.

$$\begin{aligned} & S_{xml}(f_{base}, f_{target}) \\ &= min(\\ & \qquad \frac{\sum_{t_u \in T_{base}} max_{t_v \in T_{target}}(S_{sub}(t_u, t_v))}{|Leaf(f_{base})|}, \\ & \qquad \frac{\sum_{t_v \in T_{target}} max_{t_u \in T_{base}}(S_{sub}(t_u, t_v))}{|Leaf(f_{target})|} \\ & \quad) \end{aligned} \quad (3)$$

4.2 Structure-Based Office Document Search(SOS): Style and Structure Similarity Measure

Since an OOXML document consists of multiple Parts with one or more XML files, comparing only one pair of XML files is not enough. SOS is a method for calculating similarity between sets of XML files from OOXML documents considered hierarchy of parts and XML files.

We suppose o_q is an OOXML document being a query file of similarity search and o_i ($1 \geq i \geq n$) is an OOXML document in a database. As we describe in Section 3, each OOXML document o_i consists of a set of Parts $P_i = \{p_{ij} | 1 \leq j \leq J_i\}$. And each Parts p_{ij} consists of a set of XML files $F_{ij} = \{f_{ijk} | 1 \leq k \leq K_{ij}\}$. o_q also includes a set of Parts P_q and sets of XML files F_{qj}. The procedure to calculate a similarity value between a pair of OOXML documents in SOS is as follow.

(1) **Calculating similarity values between XML files**:
By matching file names, we make a pair of XML files (f_{qjk}, f_{ijk}) from the j-th Parts of both OOXML documents o_q and o_i. We calculate a similarity value for each pair $S_{xml}(f_{qjk}, f_{ijk})$. The S_{xml} value is derived by LAX+ and our proposed methods. LAX+ is explained in Section 4.1.

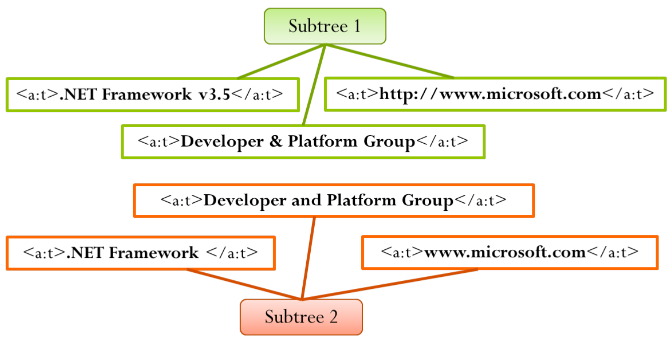

Subtree 1
<a:t>.NET Framework v3.5</a:t>
<a:t>http://www.microsoft.com</a:t>
<a:t>Developer & Platform Group</a:t>
<a:t>Developer and Platform Group</a:t>
<a:t>.NET Framework </a:t>
<a:t>www.microsoft.com</a:t>
Subtree 2

Fig. 2 An example of LAX+'s problem

(2) **Calculating similarity values between Parts by aggregating multiple S_{xml} values:**
$S_{parts}(p_{qj}, p_{ij})$ is a similarity value for the j-th parts by the following formula.

$$S_{part}(p_{qj}, p_{ij}) = \frac{\sum_{f_{qjk} \in F_{qj}, f_{ijk} \in F_{ij}} S_{xml}(f_{qjk}, f_{ijk})}{max(|F_{qj}|, |F_{ij}|)} \quad (4)$$

(3) **Calculating a similarity value between OOXML documents by aggregating multiple S_{parts} values:**
$S_{oo}(o_q, o_i)$ is a similarity value of a pair of OOXML documents (o_q, o_i), and it is obtained by the following formula.

$$S_{oo}(o_q, o_i) = \sum_{p_{qj} \in P_q, p_{ij} \in P_i} \left(\frac{w_j}{\sum_{l=1}^{|P_q|} w_l} \times S_{part}(p_{qj}, p_{ij})\right) \quad (5)$$

When w_j is a weight parameter for j^{th} Parts

4.3 Problem of LAX+ at Leaf Node Similarity Measure

In LAX+, when it compares the leaf node of subtrees-pair, it has used the exactly same matching method between the contents in two leaf nodes as in Formula 2. That means the similarity score between two leaf nodes can be only 0 or 1 even though the content in target leaf node has some common words or almost all are same with the content in base leaf node.

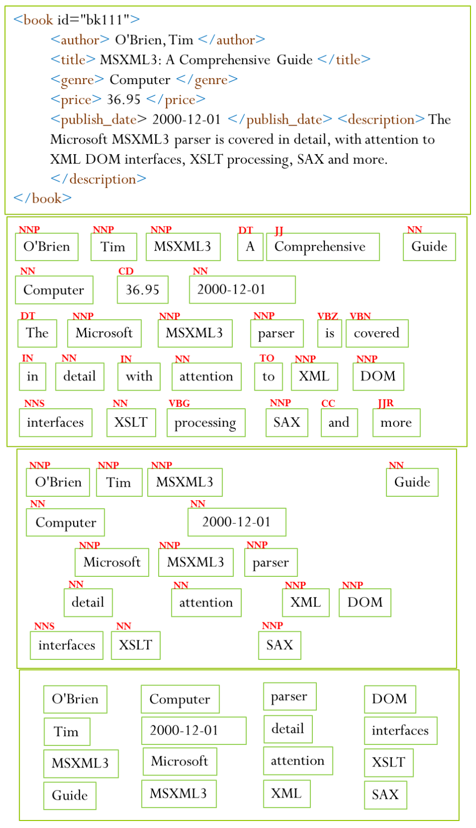
For instance, as in Figure 2 if using LAX+ to compare these two subtrees, the overall score will be 0. In spite of, the content in leaf nodes are very close to each other. This example shows that LAX+'s score is insufficient in some cases.

5. KLAX and LAX&KEY (Proposed methods)

To solve the problem introduced in Section 4.3, we used keyword similarity method for comparing leaf nodes of subtree. We propose **KLAX** (Keyword LAX+) extending the LAX+ process in Section 5.2. Moreover, we also propose another way to indicate the similarity between XML trees. **LAX&KEY** is described in Section 5.3.

5.1 Keyword Extraction and keyword-based similarity

Figure 3 illustrates how to extract keywords. First, we choose keywords by assort only the content of the file. The content means the text that readers (human) can read when open an OOXML documents excluding tags and attributes of each tag. Then, separate the content into words and roughly assign Part of speech (POS) [17], [18] for each word. After that, pick only "Noun" words (including proper noun and others that cannot determine its POS) to be represent keywords for each files. Last, comparing each file (or leaf node) using these keywords to measure the similarity. The similarity score is defined by the following formula.



```
<book id="bk111">
    <author> O'Brien, Tim </author>
    <title> MSXML3: A Comprehensive Guide </title>
    <genre> Computer </genre>
    <price> 36.95 </price>
    <publish_date> 2000-12-01 </publish_date> <description> The
    Microsoft MSXML3 parser is covered in detail, with attention to
    XML DOM interfaces, XSLT processing, SAX and more.
    </description>
</book>
```

Fig. 3 Procedure of how to choose keywords

$$S_{keyword}(f_{base}, f_{target}) = \frac{|MatchKeyword(f_{base}, f_{target})|}{max(|K_{base}|, |K_{target}|)} \quad (6)$$

◇ $|MatchKeyword(f_{base}, f_{target})|$ is the number of keywords from base file's content that matching with the keywords in target file's content.
◇ $|K_{base}|$ is the number of keywords in base file's content.
◇ $|K_{target}|$ is the number of keywords in target file's content.

5.2 KLAX

KLAX is an algorithm to measure the XML similarity which is extended from LAX+ by utilizing keyword similarity to measure the leaf nodes of subtrees similarity instead of the rigid matching of LAX+. When we compute a similarity value between leaf nodes, we considered tags, attributes and attribute's values separately from the content. First, we compare tags, attributes and attribute's values of leaf-node-pair. If they are all same it get first γ score, then compare content with keyword-based similarity by Formula 6. So, the overall score for KLAX is derived by following formula.

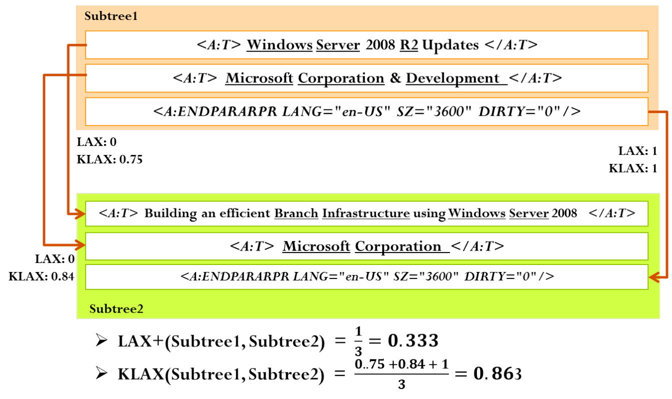

Subtree1
<A:T> Windows Server 2008 R2 Updates </A:T>
<A:T> Microsoft Corporation & Development </A:T>
<A:ENDPARARPR LANG="en-US" SZ="3600" DIRTY="0"/>
LAX: 0
KLAX: 0.75
LAX: 1
KLAX: 1
<A:T> Building an efficient Branch Infrastructure using Windows Server 2008 </A:T>
<A:T> Microsoft Corporation </A:T>
<A:ENDPARARPR LANG="en-US" SZ="3600" DIRTY="0"/>
LAX: 0
KLAX: 0.84
Subtree2
LAX+(Subtree1, Subtree2) = 1/3 = 0.333
KLAX(Subtree1, Subtree2) = (0..75 + 0.84 + 1)/3 = 0.863

Fig. 4 Comparison between LAX+ and KLAX to calculate the similarity between leaf nodes of subtrees

$$S_{K\_leaf}(l_u, l_v) = \begin{cases} 0 & \text{if tags are not same} \\ \gamma + ((1-\gamma) \cdot S_{keyword}(l_u, l_v)) & \text{otherwise} \end{cases} \quad (7)$$

We have used $\gamma = 0.5$ in this research. When (l_u, l_v) is leaf-node-pairs on a subtree-pair (t_u, t_v).

Therefore, in KLAX Formula 2 in the second step of LAX+ is replaced by the following formula.

$$S_{sub}(t_u, t_v) = \frac{\sum_{l_u \in Leaf(t_u)} max_{l_v \in Leaf(t_v)}(S_{K\_leaf}(l_u, l_v))}{min(|Leaf(t_u)|, |Leaf(t_v)|)} \quad (8)$$

When $|Leaf(t_u)|$ and $|Leaf(t_v)|$ stand for the number of leaf nodes in subtree t_u and t_v respectively.

Figure 4 shows the difference between using LAX+ and KLAX when they calculate the similarity between Subtree1 and Subtree2. The remaining part to compute the similarity between XML files is same as LAX+.

5.3 LAX&KEY

Due to the different roles between XML structure and the content of files, the document similarity should be a combination of the structure similarity value and the content similarity values. LAX&KEY calculates LAX+ similarity and keyword similarity separately. Then, it combines two scores together in order to reckon the overall similarity score. The similarity score of LAX&KEY is defined by following formula.

$$S_{LAX\&KEY}(f_{base}, f_{target}) = \beta \cdot S_{xml}(f_{base}, f_{target}) + (1-\beta) \cdot S_{keyword}(f_{base}, f_{target}) \quad (9)$$

- ◇ $S_{xml}(f_{base}, f_{target})$ is the LAX+ score between base file(f_{base}) and target file(f_{target})
- ◇ $S_{keyword}(f_{base}, f_{target})$ is the keyword similarity score between base file(f_{base}) and target file(f_{target})
- ◇ β is a weight parameter to control influences of LAX+ and keyword similarity. ($\beta \in (0, 1)$)

In additional, we also used this purpose with SOS to find the similarity between OOXML documents and the similarity score for SOS with LAX&KEY is derived by combining the SOS score with the keyword-based similarity score which is shown in Formula 6. We have join both score by using Formula 9 above.

6. Experiment Evaluation

In this section, we held two experiments and for each can be separated into two parts. First experiment, we compare LAX+ with our new proposed methods to show that our proposals give better result in precision and recall for calculation similarity of single pair of XML files. Second, we compare the original SOS (using LAX+) with improved SOS (using KLAX and LAX&KEY) to investigate the precision and recall of measuring a similarity between OOXML documents (multiple sets of XML files). Figure 5 is the experiment data for our experiments below.

Experiment	pptx files		docx files		xlsx files	
	Number of files	Number of subgroups	Number of files	Number of subgroups	Number of files	Number of subgroups
6.1.1 LAX+ with KLAX	258	53	215	22	-	-
6.1.2 LAX+ with LAX&KEY	285	59	215	25	-	-
6.2.1 SOS with KLAX	285	53	215	22	111	32
6.2.2 SOS with LAX&KEY	285	59	215	25	107	28

Fig. 5 Experiment Data

6.1 Comparison of LAX+ with KLAX and LAX&KEY

In this experiment, we extracted the XML files from pptx files and docx files, for pptx files we used "slide1.xml" which includes style information and content data of the first slide and docx files we used "document.xml" which is the main part of any word documents to calculate the similarity value between XML files for each kind of documents. Therefore, the procedure of this experiment is as follows.

(1) From a set of pptx and docx files, we choose one file as a query file o_q. Then, the remaining files are target of similarity search.

(2) For each query file and target files, we choose "slide1.xml" for pptx files and "document.xml" for docx files. Then, we calculate a similarity value between XML files.

(3) We rank the target files based on its similarity values and chooses the file whose similarity values are greater than the threshold θ. Then, we compare the result with an answer set corresponding to the query file, and get precision and recall; precision and recall are defined as follows.

$$Precision = \frac{Number\_Retrieved\_Relevant}{Number\_Total\_Retrieved} \quad (10)$$

$$Recall = \frac{Number\_Retrieved\_Relevant}{Number\_Possible\_Relevant} \quad (11)$$

(4) We iterate step 1, 2, 3 for each file in an experimental data to compute average values of precision and recall.

6.1.1 LAX+ vs. KLAX

We show a precision-recall graph of LAX+ and KLAX in Figure 6 for pptx file type and Figure 7 for docx file type. The horizontal axis expresses recall values and the vertical axis expresses precision values. From the result, KLAX is better that LAX+ in precision at the same recall value. Since KLAX used the keyword-based similarity to match leaf nodes between subtrees which is more flexible than LAX+'s, the similarity scores of

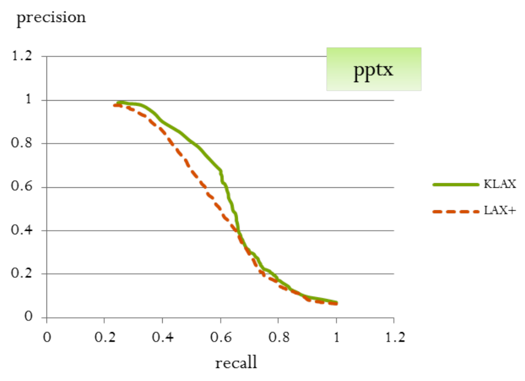


Fig. 6 Precision-recall graph of LAX+ and KLAX on pptx files

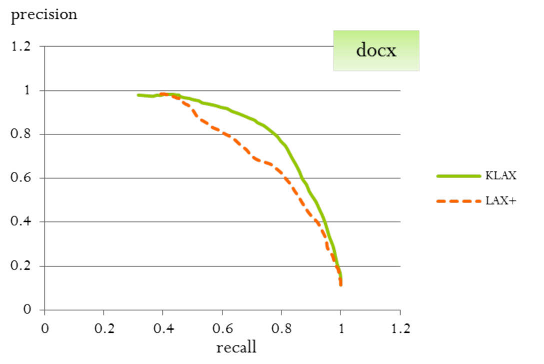


Fig. 7 Precision-recall graph of LAX+ and KLAX on docx files

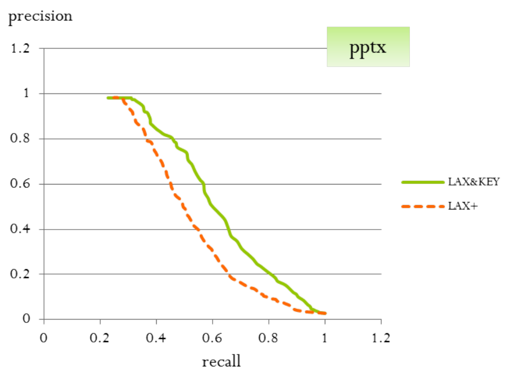


Fig. 8 Precision-recall graph of LAX+ and LAX&KEY on pptx files

KLAX are higher in most cases especially for alike files. Thus, the similarity score by KLAX is more than LAX+ for each in case of similar files pair.

6.1.2 LAX+ vs. LAX&KEY

In this experiment, we have varied the β values from 0.01 to 0.99 to explore the properly β that gives the best result in precision and recall for each document type which are 0.50 and 0.35 for pptx and docx respectively. We show a precision-recall graph of LAX+ and LAX&KEY in Figure 8 for pptx file type and Figure 9 for docx file type. From above result, LAX&KEY is better that LAX+ in precision at the same recall value. Due to the fact that XML file has two considerable parts which are structure and content of file, consideration of both should get more accuracy in result.

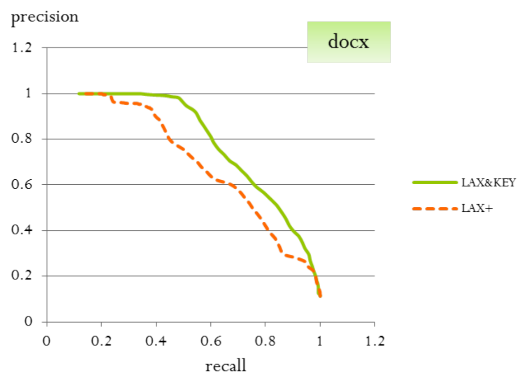


Fig. 9 Precision-recall graph of LAX+ and LAX&KEY on docx files

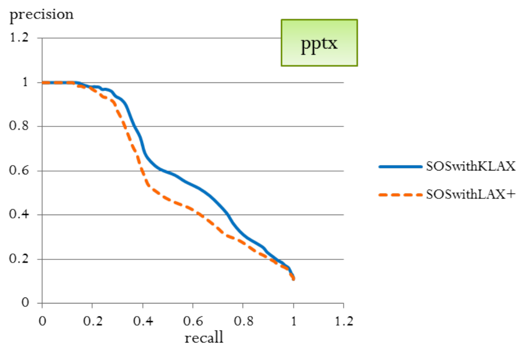


Fig. 10 Precision-recall graph of SOS and KLAX on pptx files

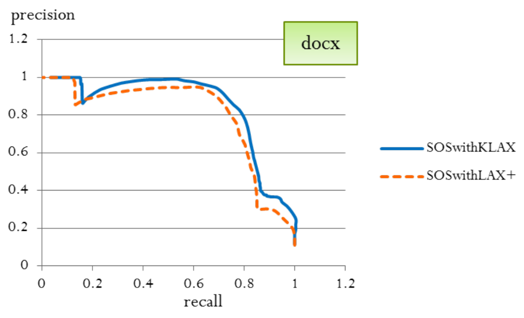


Fig. 11 Precision-recall graph of SOS and KLAX on docx files

6.2 Applying KLAX and LAX&KEY to SOS

6.2.1 SOS with KLAX

In this experiment, we compare SOS using LAX+ and SOS using KLAX to verify whether KLAX can be improves SOS in precision and recall. We show precision-recall graphs of SOS with LAX+ and SOS with KLAX for pptx, docx and xlsx file type in Figure 10, 11, 12 respectively. From above result, SOS with KLAX is better than SOS with LAX+ in precision at the same recall value for pptx and docx file type. In case of xlsx files, the keyword similarity might not very significant to calculate the similarity value between them. Due to the purpose of xlsx files which often include many numbers or IDs and almost not composed of words or keywords, keyword-based similarity is not much influential in this document type.

6.2.2 SOS with LAX&KEY

In this experiment, we also varied the β values from 0.01 to 0.99 to explore the properly β that gives the best result in preci-

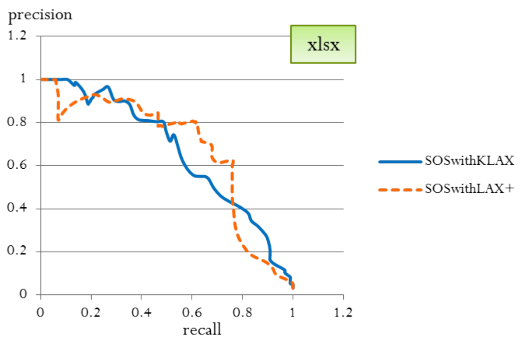


Fig. 12 Precision-recall graph of SOS and KLAX on xlsx files

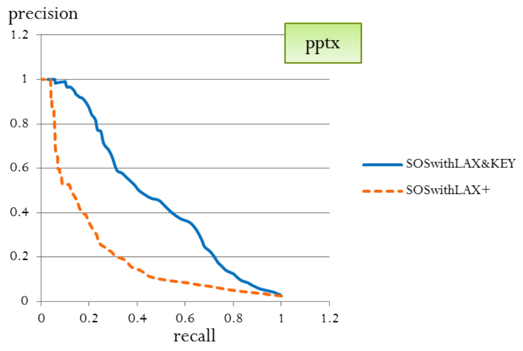


Fig. 13 Precision-recall graph of SOS and LAX&KEY on pptx files

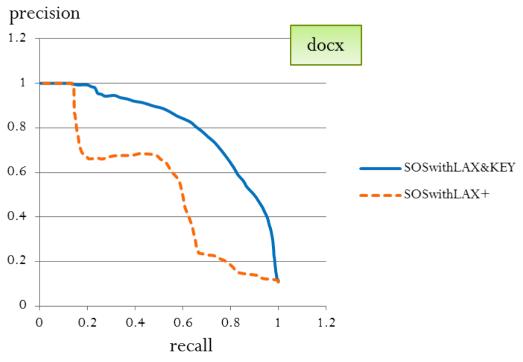


Fig. 14 Precision-recall graph of SOS and LAX&KEY on docx files

sion and recall for each document type which are 0.50, 0.40 and 0.50 for pptx, docx and xlsx respectively. We show a precision-recall graph of SOS with LAX+ and SOS with LAX&KEY in for pptx, docx and xlsx file type in Figure 13, 14, 15 respectively. From the result graph, SOS with LAX&KEY is better than SOS with LAX+ in precision at the same recall value for all type of OOXML documents in the experiment.

7. Conclusion and Future Work

This paper describes the problem of LAX+ to calculate similarity value between XML trees which is the leaf node matching method is rigid, and affects the similarity value to be low. Therefore, we solved the problem with a keyword similarity method called KLAX which extends LAX+ advantages and also propose LAX&KEY to be an alternative way to calculate the similarity values of XML files and OOXML documents. After that, we applied the KLAX to SOS instead of LAX+, and combining LAX&KEY with SOS in our experiments. SOS with KLAX gives slightly bettet result especially for xlsx files while

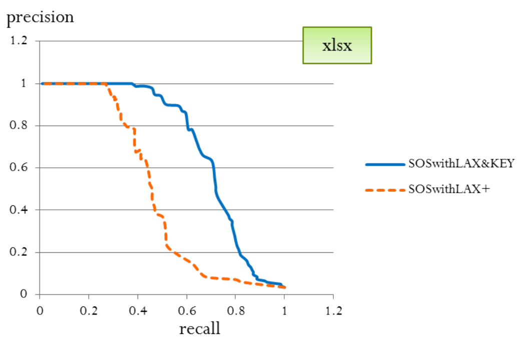


Fig. 15 Precision-recall graph of SOS and LAX&KEY on xlsx files

LAX&KEY gives obviously better result than LAX+. Since, the result of xlsx files is a bit ambiguously, we plan to do further experiments for evaluating KLAX and LAX&KEY for OOXML similarity by using different weight for each Part of OOXML documents. Moreover, the keyword-based similarity also can be improved such as POS tagger, procedure to choosing the words and others.

References

[1] ECMA-376 4th edition,
http://www.ecma-international.org/publications/standards/Ecma-376.htm
[2] Windows Search,
http://www.microsoft.com/japan/windows/desktopsearch/default.mspx
[3] Mac OS X Spotlight,
http://support.apple.com/kb/HT2531
[4] “What file types can Google index?”,
http://support.google.com/webmasters/bin/answer.py?answer=35287
[5] J.Tekli, R.Chbeir and K.Yetongnon, “An overview on XML similarity: Background, current trends and future directions”, Computer Science Review, Vol. 3, No. 3, pp. 151–173, 2009.
[6] J. Pokorny, J. Vávra and V. Snásel, “A Renewed Matrix Model for XML Data”, Proc. International Conference on Intelligent Systems Design and Applications (ISDA2008), pp. 549–556, 2008.
[7] K. C. Tai, “The Tree-to-Tree Correction Problem”, Journal of the Association for Computing Machinery, Vol. 26, No. 3, pp. 422–433, 1979.
[8] E. D. Demaine, S. Mozes, B. Rossman and O. Weimann, “An Optimal Decomposition Algorithm for Tree Edit Distance”, ACM Trans. Algorithms, Vol. 6, No. 1, pp.2:1–2:19, 2009.
[9] D. Buttler, “A Short Survey of Document Structure Similarity Algorithms”, Proc. International Conference on Internal Computing, pp.3–9, 2004.
[10] S. Helmer, “Measuring the Structural Similarity of Semistructured Documents Using Entropy”, Proc. International Conference on Very Large Databases (VLDB2007), pp.1022–1032, 2007.
[11] W. Liang and H. Yokota, “LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration”, Proc. BNCOD, Springer LNCS 3567, pp. 82–97, 2005.
[12] Y. Watanabe, H. Kamigaito and H. Yokota, “Similarity Search for Office XML Documents Based on Style and Structure Data”, International Journal of Web Information Systems, Emerald Group Publishing, Vol. 9, Issue 2, pp. 100–116, June, 2013.
[13] W. Viyanon, S. K. Madria and S. S. Bhowmick, “XML data integration based on content and structure similarity using keys”, Proc. On the Move to Meaningful Internet Systems (OTM2008), 2008.
[14] T. Tran, R. Nayak and P. Bruza, “Combining Structure and Content Similarities for XML Documents Clustering”, Proc. Australasian Data Mining Conference (AusDM’ 08), 2008.
[15] L. Zhang, Z. Li, Q. Chen and N. Li, “Structure and Content Similarity for Clustering XML Documents”, Proc. WAIM 2010 Workshops, LNCS 6185, pp. 116–124, 2010.
[16] J. Tekli, R. Chbeir and K. Yetongnon, “A Hybrid Approach for XML Similarily” Proc. SOFSEM 2007, LNCS 4362, pp. 738–795, 2007.
[17] Stanford Log-linear Part-Of-Speech Tagger.
http://nlp.stanford.edu/software/tagger.shtml
[18] Part Of Speech Tagging PHP/ir.
http://phpir.com/part-of-speech-tagging