XML Prague 2019, February 8th

Tagdiff: a diffing tool for highlighting differences in the tagging of text-oriented XML documents

Cyril Briquet

cyril.briquet@canopeer.org

Contents

- Text-oriented XML documents and use case
- *diff* vs. *tagdiff* vs. existing GUI-based XML tools
- Description of the algorithm
- Performance
- Conclusions

Structure-oriented vs. text-oriented XML documents

- **structure-oriented XML documents**

  <<In many applications XML documents can be treated as unordered trees – only ancestor relationships are significant, while the left-to-right order among siblings is not significant.>> [WDC03]

  use cases rely on XML documents to store structured data

- **text-oriented XML documents**

  XML is relied upon to (algorithmically) tag sections of text; what's in lateral proximity is important

  use cases include literary texts, linguistic data, news,...

Use case: algorithmic tagging of text-oriented XML documents

- **algorithmic tagging of linguistic data corpus** [BRP10]

- multiple processing steps (sequence of 40 tagging algorithms)

- to validate algorithms, it is useful to visually inspect:

  (1) data before and after applying a given tagging algorithm

  ==> **new tagging correct and complete?**

  (2) data output by a reference version and a new (faster or more readable) version of a tagging algorithm

  ==> **same tagging?**

Command line tool requirements

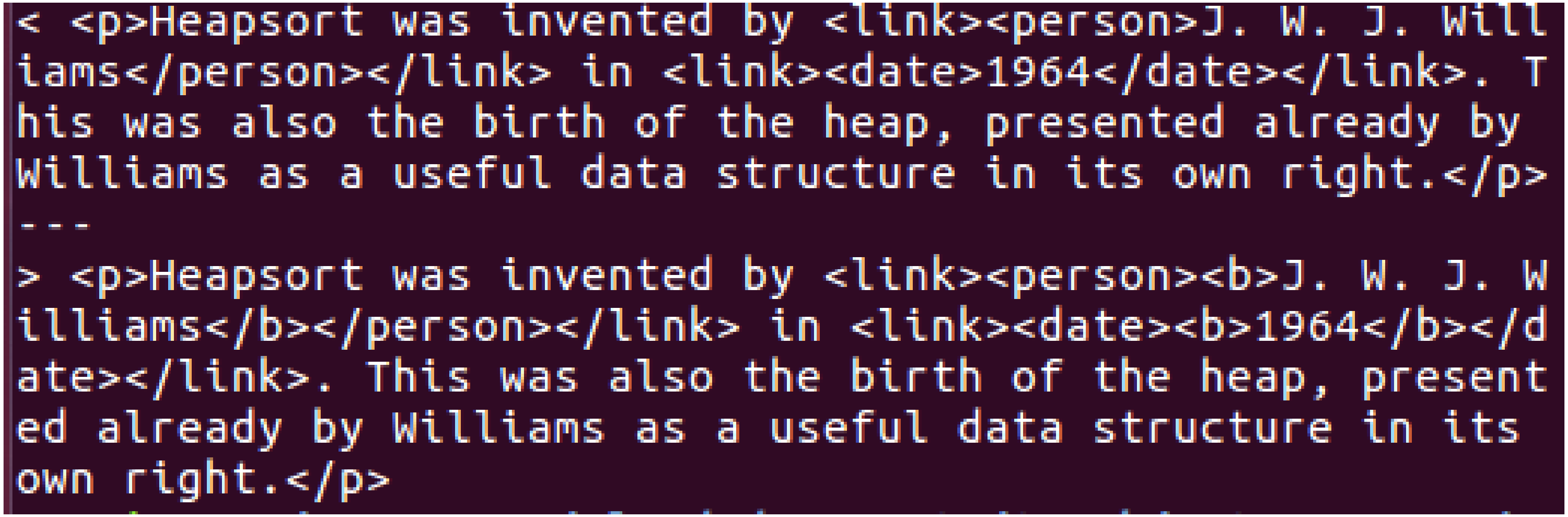
- exact diffing
- no options such as filtering out

  some types of information (whitespace, comments, …)
- not a goal to merge or patch documents
- output easy to visualize
- output easy to process by other command line tools
- no GUI

diff

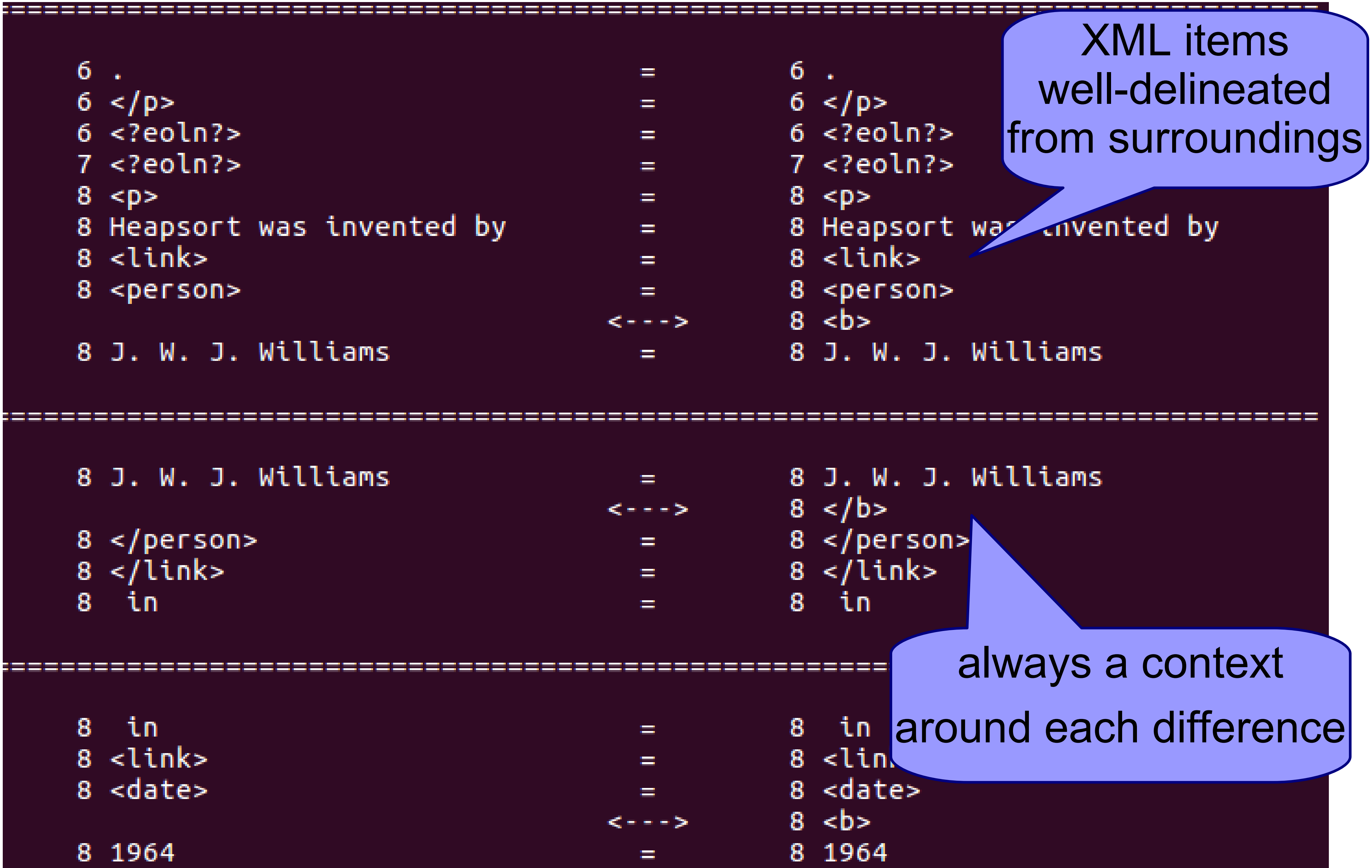
- well-known UNIX command line tool: diff
- based on classic diffing algorithm [Myers86]

  (*An O(ND) Difference Algorithm and Its Variations*)
- focuses on differences between short lines (e.g. source code)
- example: 4 differences in a paragraph, but it's not obvious!



```
< <p>Heapsort was invented by <link><person>J. W. J. Will
iams</person></link> in <link><date>1964</date></link>. T
his was also the birth of the heap, presented already by
Williams as a useful data structure in its own right.</p>
---
> <p>Heapsort was invented by <link><person><b>J. W. J. W
illiams</b></person></link> in <link><date><b>1964</b></d
ate></link>. This was also the birth of the heap, present
ed already by Williams as a useful data structure in its
own right.</p>
```

tagdiff
vertical, segmented and typed diffing

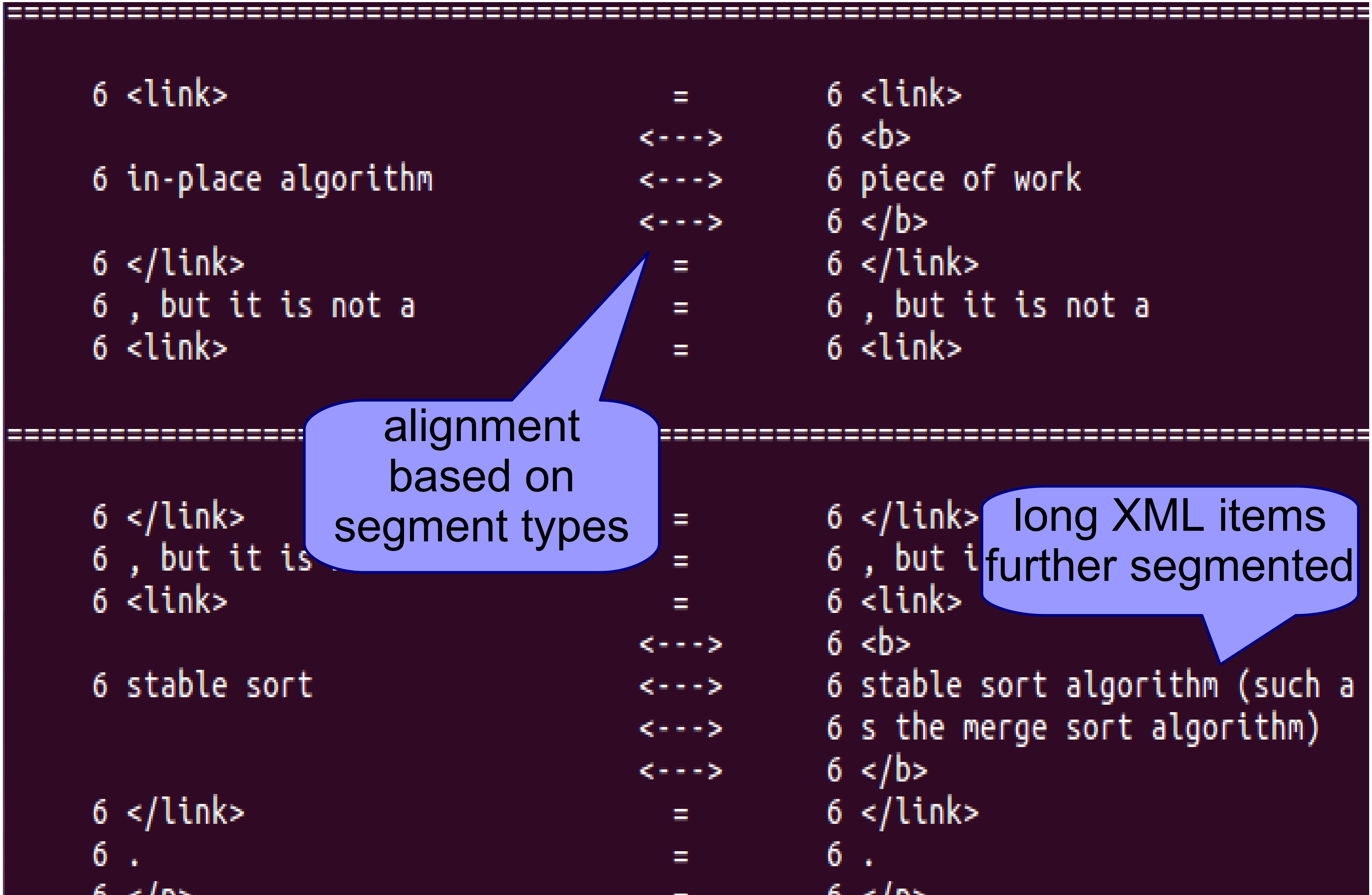


```
================================================================================

    6 .                                        =        6 .
    6 </p>                                     =        6 </p>
    6 <?eoln?>                                 =        6 <?eoln?>
    7 <?eoln?>                                 =        7 <?eoln?>
    8 <p>                                      =        8 <p>
    8 Heapsort was invented by                 =        8 Heapsort was invented by
    8 <link>                                   =        8 <link>
    8 <person>                                 =        8 <person>
                                             <--->      8 <b>
    8 J. W. J. Williams                        =        8 J. W. J. Williams

================================================================================

    8 J. W. J. Williams                        =        8 J. W. J. Williams
                                             <--->      8 </b>
    8 </person>                                =        8 </person>
    8 </link>                                  =        8 </link>
    8  in                                      =        8  in

===========================================================

    8  in                                      =        8  in
    8 <link>                                   =        8 <lin
    8 <date>                                   =        8 <date>
                                             <--->      8 <b>
    8 1964                                     =        8 1964
```

XML items well-delineated from surroundings

always a context around each difference

tagdiff
vertical, segmented and typed diffing



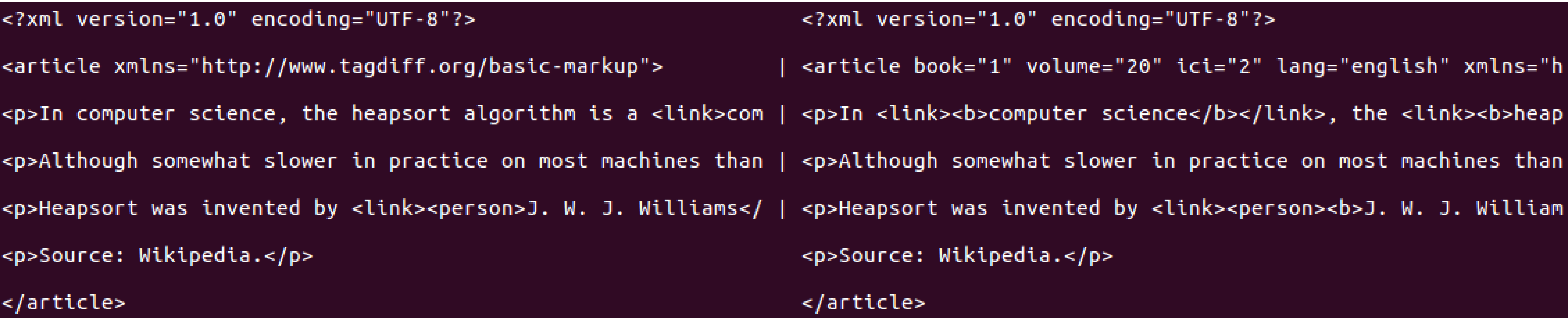
```
==========================================================================================

    6 <link>                                  =        6 <link>
                                            <--->      6 <b>
    6 in-place algorithm                    <--->      6 piece of work
                                            <--->      6 </b>
    6 </link>                                 =        6 </link>
    6 , but it is not a                       =        6 , but it is not a
    6 <link>                                  =        6 <link>


==========================================================================================

    6 </link>                                 =        6 </link>
    6 , but it is                             =        6 , but i
    6 <link>                                  =        6 <link>
                                            <--->      6 <b>
    6 stable sort                           <--->      6 stable sort algorithm (such a
                                            <--->      6 s the merge sort algorithm)
                                            <--->      6 </b>
    6 </link>                                 =        6 </link>
    6 .                                       =        6 .
    6 </p>                                    =        6 </p>
```

alignment based on segment types

long XML items further segmented

diff with -y flag

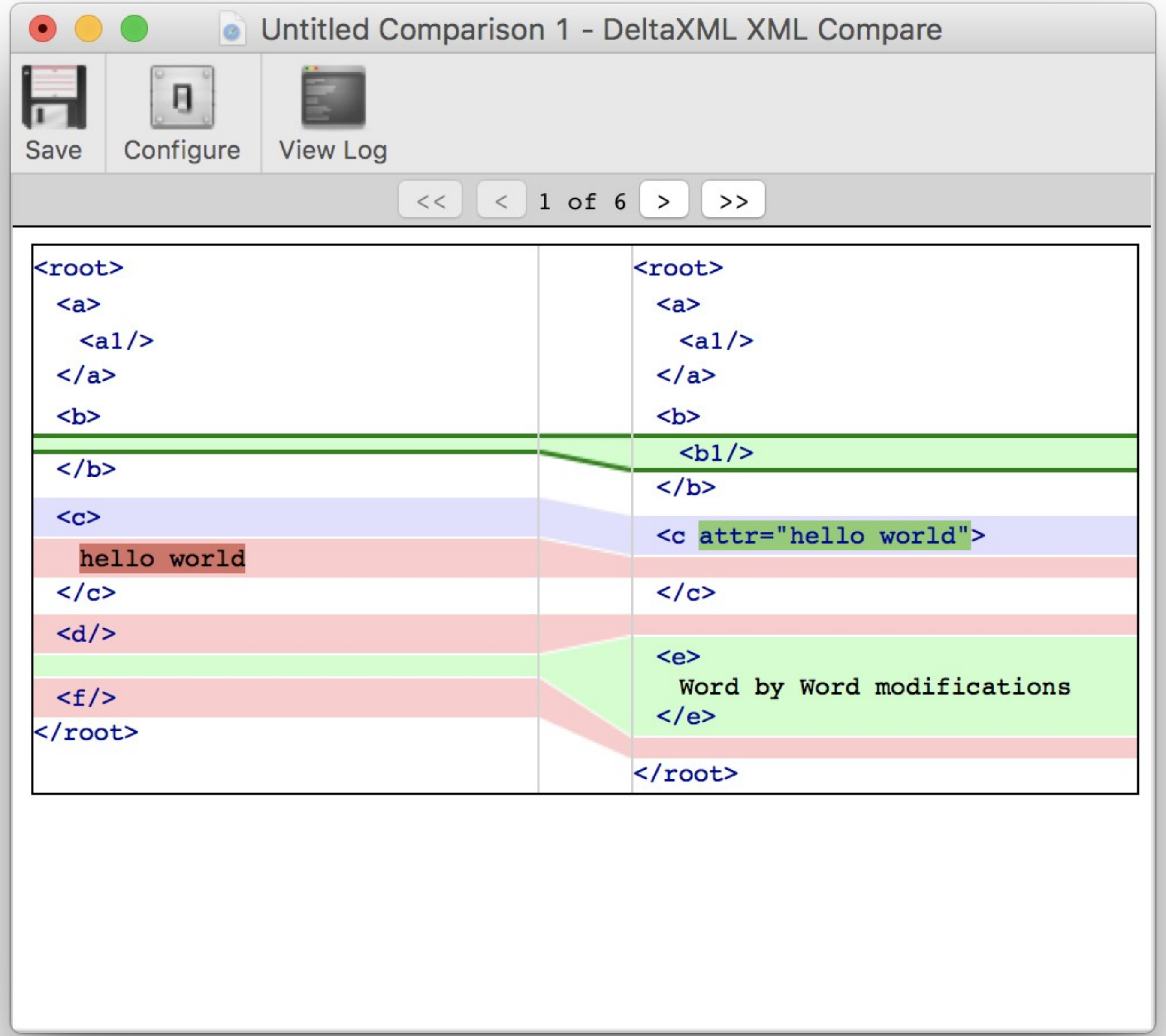
- diff -y: output in two columns, but
  - * no segmentation of long lines (only truncation)
  - * all the contents displayed, no specific contextualization



```
<?xml version="1.0" encoding="UTF-8"?>                          <?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://www.tagdiff.org/basic-markup">         | <article book="1" volume="20" ici="2" lang="english" xmlns="h
<p>In computer science, the heapsort algorithm is a <link>com | <p>In <link><b>computer science</b></link>, the <link><b>heap
<p>Although somewhat slower in practice on most machines than | <p>Although somewhat slower in practice on most machines than
<p>Heapsort was invented by <link><person>J. W. J. Williams</ | <p>Heapsort was invented by <link><person><b>J. W. J. William
<p>Source: Wikipedia.</p>                                       <p>Source: Wikipedia.</p>
</article>                                                      </article>
```

DeltaXML XML Compare

https://docs.deltaxml.com/xml-compare/current/docs/gui-help/

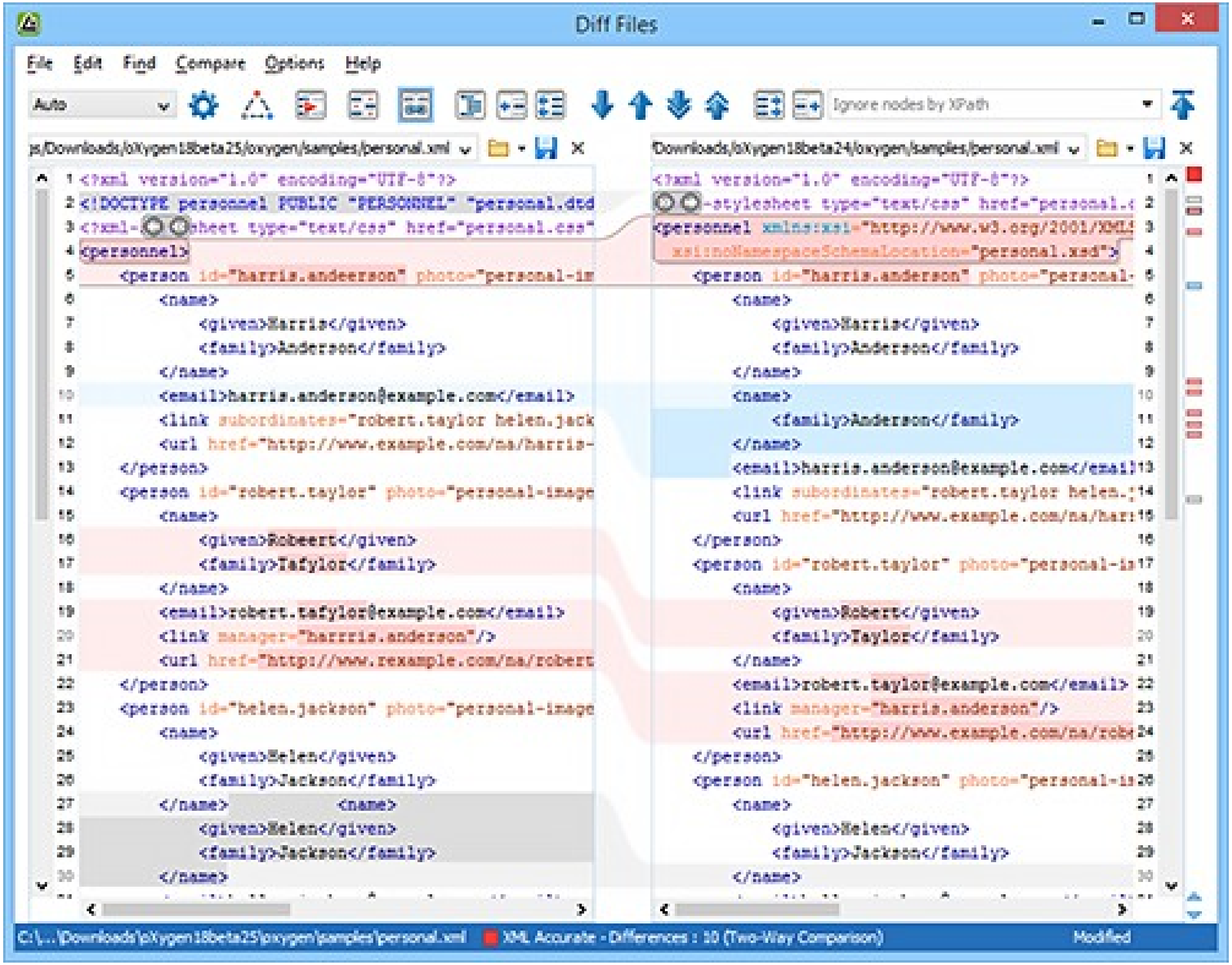


```
<root>
  <a>
    <a1/>
  </a>
  <b>
  </b>
  <c>
    hello world
  </c>
  <d/>
  <f/>
</root>
```

```
<root>
  <a>
    <a1/>
  </a>
  <b>
    <b1/>
  </b>
  <c attr="hello world">
  </c>
  <e>
    Word by Word modifications
  </e>
</root>
```

Oxygen XML Editor

https://www.oxygenxml.com/files_compare_img.html



js/Downloads/oXygen18beta25/oxygen/samples/personal.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personnel PUBLIC "PERSONNEL" "personal.dtd
<?xml-stylesheet type="text/css" href="personal.css"
<personnel>
    <person id="harris.andeerson" photo="personal-im
        <name>
            <given>Harris</given>
            <family>Anderson</family>
        </name>
        <email>harris.anderson@example.com</email>
        <link subordinates="robert.taylor helen.jack
        <url href="http://www.example.com/na/harris-
    </person>
    <person id="robert.taylor" photo="personal-image
        <name>
            <given>Robeert</given>
            <family>Tafylor</family>
        </name>
        <email>robert.tafylor@example.com</email>
        <link manager="harrris.anderson"/>
        <url href="http://www.rexample.com/na/robert
    </person>
    <person id="helen.jackson" photo="personal-image
        <name>
            <given>Helen</given>
            <family>Jackson</family>
        </name>       <name>
            <given>Helen</given>
            <family>Jackson</family>
        </name>
```

Downloads/oXygen18beta24/oxygen/samples/personal.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="personal.c
<personnel xmlns:xsi="http://www.w3.org/2001/XMLS
  xsi:noNamespaceSchemaLocation="personal.xsd">
    <person id="harris.anderson" photo="personal-
        <name>
            <given>Harris</given>
            <family>Anderson</family>
        </name>
        <name>
            <family>Anderson</family>
        </name>
        <email>harris.anderson@example.com</email
        <link subordinates="robert.taylor helen.j
        <url href="http://www.example.com/na/harr
    </person>
    <person id="robert.taylor" photo="personal-im
        <name>
            <given>Robert</given>
            <family>Taylor</family>
        </name>
        <email>robert.taylor@example.com</email>
        <link manager="harris.anderson"/>
        <url href="http://www.example.com/na/robe
    </person>
    <person id="helen.jackson" photo="personal-im
        <name>
            <given>Helen</given>
            <family>Jackson</family>
        </name>
```

C:\...\Downloads\oXygen18beta25\oxygen\samples\personal.xml 🟥 XML Accurate - Differences : 10 (Two-Way Comparison) Modified

tagdiff algorithm

- main idea:

segment the XML documents into **small typed segments** that are **easy to align, to compare, and to visualize**

- three main phases:

1) **diffing the raw text versions** of the XML documents
2) **XML parsing** and **segmenting** the XML documents
3) **aligning** sequences of the (differing) typed segments

- implementation: **Java 8, ~5000 lines of code**

  (+ several libs: raw text diffing algorithm, XML data model)

Algorithm (1)
Diffing of the raw text versions

- **classic diffing algorithm** [Myers86]:

  identifies which sections of the **raw text versions**

  of two XML documents are **equal**, and which are **differing**

- measured performance slow for large number of differences

  => **optimization required**

Algorithm (1 continued)
Optimization of the diffing algorithm

- **optimization: split** the two XML documents **into short sections** with limited number of differences so that the [Myers86] diffing algorithm performance remains good

- **splitting points?**

  * **must match** in the two documents

  * default splitting points: paragraph boundaries, but can be **specified by the user as a regexp**

- example:

  document 1: ...|<p>|...|</p>|...|<p>|...|</p>|...|<p>|...|</p>|...

  document 2: ...|<p>|...|</p>|...|<p>|...|</p>|...|<p>|...|</p>|…

Algorithm (2.1)
XML parsing

- **XML parsing** (SAX, but could be DOM)

- **no schema required** and **no validation performed**

  => enables to find **differences in non-valid documents**

- **chunk-based** (non-DOM) **data model** (previous work [BRP10])

  * 1 opening, empty, or closing tag ==> 1 XML chunk

  * end of line ‘\n’ ==> processing instruction ==> 1 XML chunk

|<?xml version="1.0" encoding="UTF-8"?>**|**<?eoln?>**|**<article book="1" ici="2" lang="english" volume="20">**|**<?eoln?>**|**<?eoln?>**|**<p>**|**In computer science, ...**|**…

Algorithm (2.2) Segmentation

- 1 XML chunk + 1 diffing type + 1 offset == **1 typed segment**

- 8 type values
  (given by **classic diffing algorithm of the previous phase**):
  equal text, equal tag, equal PI, equal comment,
  differing text, differing tag, differing PI, differing comment

- offsets in the **equal-data version** of the XML documents
  (typed segments of a differing type don't have an offset)

Algorithm (2.2 continued) Segmentation

|<?xml version="1.0" encoding=| equal processing instruction

|"UTF-8"?>| equal processing instruction

|<?eoln?>| equal processing instruction

|<article book="1" ici="2" lan| differing tag

|g="english" volume="20">| differing tag

|<?eoln?>| equal processing instruction

|<?eoln?>| equal processing instruction

|<p>| equal tag

…

“long” segments are further segmented

(max column width, e.g. 29 for 80 chars terminal)

Algorithm (3.1)
Alignment of equal data

- **neighboring differing** segments are **grouped** together
- sequences of equal data aligned

  based on their **offsets in the equal-data version**
- **alternation** of equal-data and differing-data sequences

<table>
<tr><th>document 1 sequences</th><th>document 2 sequences</th></tr>
<tr><td>seq i: |&lt;link&gt;|</td><td>seq j: |&lt;link&gt;|</td></tr>
<tr><td>seq i+1: |in-place algorithm|</td><td>seq j+1: |&lt;b&gt;|
|piece of work|
|&lt;/b&gt;|</td></tr>
<tr><td>seq i+2: |&lt;/link&gt;|</td><td>seq j+2: |&lt;/link&gt;|</td></tr>
</table>

Algorithm (3.2)
Alignment of differing data

- differing-data sequences still need alignment with their counterparts

without alignment |in-place algorithm| would be misaligned with |<b>|

document 1 sequences		document 2 sequences
|<link>|	=	|<link>|
|in-place algorithm|	?	|<b>|
(gap)	?	|piece of work|
(gap)	?	|</b>|
|</link>|	=	|</link>|

Algorithm (3.2 continued) Alignment of differing data: optimization algorithm

- **combinatorial alignment problem** solved (many times) for each matching pair of (rather short) differing sequences
- systematic recursive **enumeration of all possible alignments** (with pruning of unpromising solutions to boost performance)
- optimization (minimization) algorithm: the **alignment with the lowest “cost” is selected**

cost of 2 typed segments of same type and equal data
< cost of 2 typed segments of same type and differing data
< cost of 1 typed segment matched with 1 gap
< cost of 2 typed segments of different types

Performance (small test corpus)

- a pair of small XML documents:

  14 lines and about 2 kB each, 26 differences

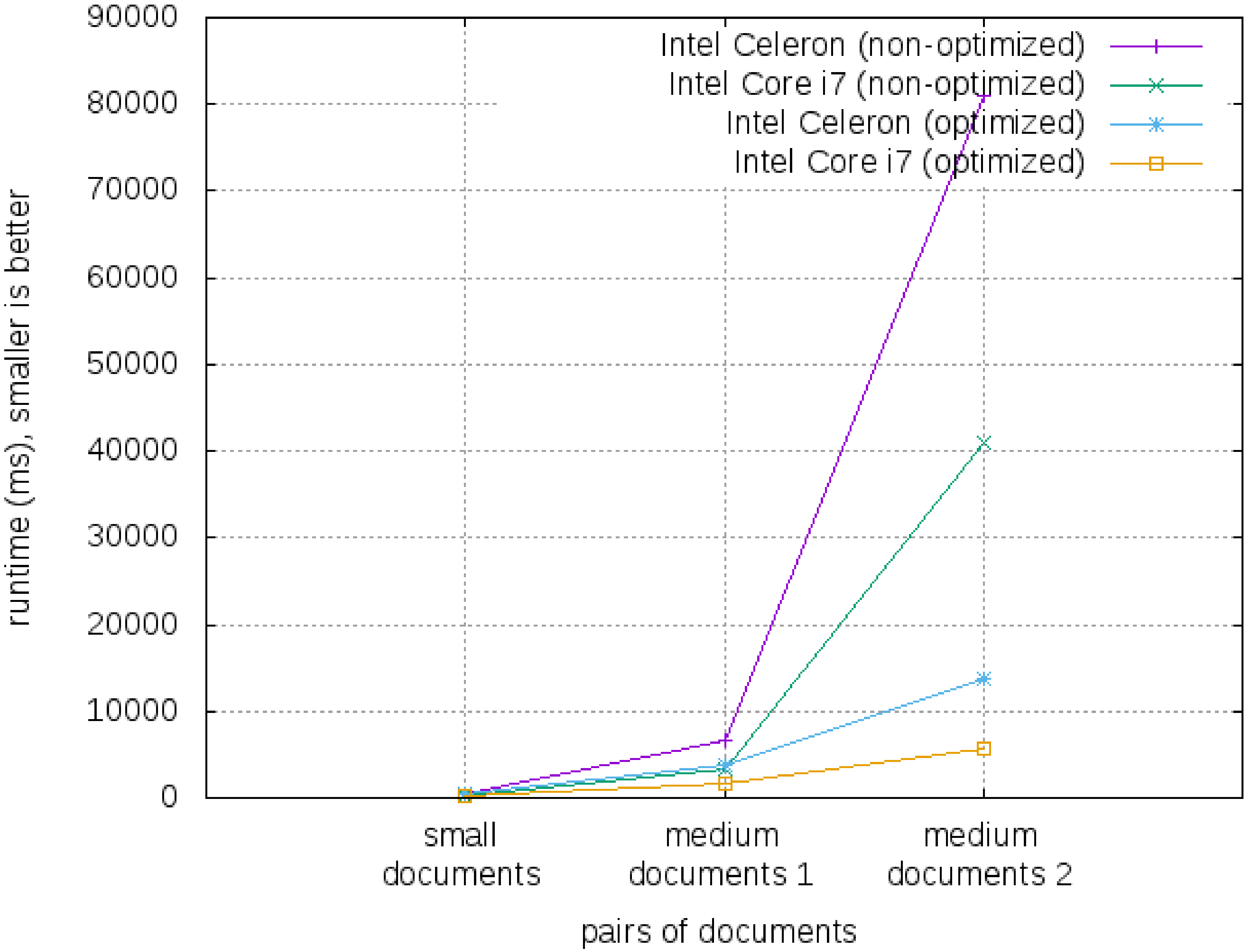
- a pair of medium XML documents:

  ~1000 lines (x70),115 kB each, 743 differences

- a 2nd pair of medium XML documents:

  ~4000 lines (x4), 500 kB each, 3638 differences

Performance (runtimes)



Conclusions

- ***tagdiff***:

  a command line tool for diffing **text-oriented XML documents**

  (**no schema required**, no XML validation performed);

  visualization is vertical, segmented and typed

- (optimized) use of **classic diffing algorithm** [Myers86]

- **alignment done by an optimization algorithm**

  (which we think could be integrated into existing XML tools)

  applied to many small sequences of **typed segments**

  that are easy to **align, compare** and **visualize**

Thank You !