

Advanced Information Access to Parliamentary Debates

Maarten Marx*
ISLA, University of Amsterdam
Science Park 107 1098 XG Amsterdam, The Netherlands
maartenmarx@uva.nl

December 11, 2009

Abstract

Parliamentary debates are highly structured transcripts of meetings of politicians in parliament. These debates are an important part of the cultural heritage of many countries; they are often free of copy-right; citizens often have a legal right to inspect them; and several countries make great effort to digitize their entire historical collection and make it available to the general public. This provides many opportunities for the Information Retrieval community.

In this paper, we analyze the structure of parliamentary proceedings and sketch a widely applicable DTD. We show how proceedings in PDF format can be transformed into deeply nested XML.

Having the proceedings in XML makes a wide range of applications possible. We elaborate on five applications: entry point retrieval, advanced content and structure search; automatic creation of tables of contents and hyperlinked navigation menus; graphical result aggregation; large savings on storage space and bandwidth for scanned documents.

1 Introduction

Parliamentary proceedings are an interesting set of data to apply state-of-the-art information retrieval technology. Parliamentary proceedings are written records of parliamentary activities containing a wide range of document types. In English, these proceedings are usually called *Hansard*, after the publisher Thomas Curson Hansard, who first (illegally) printed proceedings of the British parliament. In this paper we only discuss notes of meetings of parliament. As with all meeting notes, these records have the purpose of storing the content of the meeting. They have varying degrees of detail. Currently in most Western democracies it is common to transcribe everything that is being said, keeping the content, but making it grammatically correct and pleasant to read.

We list a number of characteristics which make these documents of special interest to the Information Retrieval community:

- Large historical corpora; For example, in Holland all data starting from 1814 will be available in 2010 (at the time of writing it is available since 1917) [<http://www.statengeneraaldigitaal.nl>]; for the Flemish parliament all data since 1971 is available in PDF [<http://www.vlaamsparlement.be/>]; the British Hansard archives have all parliamentary minutes since 1803 available in XML [<http://hansard.millbanksystems.com>].
- documents contain extensive and consistently applied structure which is rather easy to extract and make explicit;

*Maarten Marx acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

- transcripts of meetings might be accompanied by audio and video recordings, creating interconnected multimedia data [23];
- challenging data integration issues and opportunities [17, 12, 18] both within one country (collections from different periods, in different formats, styles, language, ...), and across countries (cross-lingual IR);
- natural corpus for content and structure queries, combining keyword search with XPath navigation and selection [14, 20];
- natural corpus for search tasks in which the answers do not consist of documents: *expert* or *people search* [4], video search and *entry point retrieval* [24].
- the size of the corpus and of the individual documents calls for result aggregation and summarization [19].

From this list, this paper treats the information extraction, data integration, result summarization and entry-point retrieval aspects.

The paper is organized as follows: Section 2 describes the structure of parliamentary meetings and formalizes it in a DTD. Section 3 describes the Extract-Transform-Load process with which we transform the notes into XML. It also contains an evaluation of the quality of this process. We discuss five applications of structured data in Section 4 and conclude in Section 5. A search engine containing all Dutch parliamentary data from 1984 till May 2008 is built and can be used at <http://www.polidocs.nl>. The corpus of over 80.000 XML files is available from that URL.

1.1 Advanced information access to cultural heritage

Parliamentary proceedings are an important part of each country’s cultural heritage. Because the proceedings are “just” text documents one could expect that modern IR technology is being used to make the proceedings easily accessible to researchers and the general public. Unfortunately this is not the case, at least not in the Netherlands, Belgium, Germany and the UK. In all these countries, the parliaments have created websites which allow the public to search in the proceedings. But, even though these are vertical search engines restricted to documents under control by the owners of the search engine, the quality of the interface and of the search results is much lower than Google.

The most important reason for this low quality is a mismatch between the unit of retrieval used in these systems and the natural answer unit. The unit of retrieval in all these countries is the complete verbatim proceedings of one day. These are very large documents covering a multitude of topics. In the Netherlands an average of 50.000 words are spoken each day. The natural answer unit for a vertical search engine containing meeting notes are agenda-topics, or even more detailed, speeches by members of parliament. Because these much smaller answer units are more likely to be about one topic, they are easier to rank, and easier for users to assess as relevant.

Advanced information access to meeting notes thus needs to have the structure of the meeting explicitly available. The main contribution of this paper is to show that this structure can be extracted robustly (also from legacy data), and that the availability of the structure makes it possible to create true information access to heritage data.

2 Structure of parliamentary proceedings

Notes of a formal meeting with an agenda (e.g., business meeting, council meeting, meeting of the members of a club, etc) are full of implicit structure and contain many common elements. The notes of meetings with a large historical tradition, like parliamentary debates, are in a uniform format which fluctuates little with time. This makes these notes very well suited for text-mining.

To our knowledge, at the time of writing, there is no generally agreed upon DTD or markup language available for (parliamentary) meeting notes.

Transcripts of a meeting contain three main structural elements:

the topics discussed in the meeting (the agenda);

the speeches held at the meeting: every word that is being said is recorded together with 1) the name of the speaker, 2) her affiliation and 3) in which role or function the person was speaking;

non verbal content or actions These can be:

- list of present and absent members;
- description of actions like *applause by members of the Green Party*;
- description of the outcome of a vote;
- the attribution of reference numbers to actions or topics;
- etc.

The analogy with the structural elements in theatrical drama is striking: scenes, speeches and stage-directions are the theatrical counterparts of the three elements just listed. These are prominent elements in the XML version of Shakespeare's work.¹ The close relation between politics and drama is an emerging theme in political science, see e.g., [13, 11].

These elements are structured as follows:

<code>meeting</code>	→	<code>(topic)+</code>
<code>topic</code>	→	<code>(speech stage-direction)+</code>
<code>speech</code>	→	<code>(p stage-direction)+</code>
<code>p</code>	→	<code>(#PCDATA stage-direction)*</code>
<code>stage-direction</code>	→	<code>(#PCDATA).</code>

All elements contain metadata stored in attributes. The British digitized debates from 1803 to 2004 are available in XML and basically have this structure.

Within the Dutch proceedings however there is an intermediate structural element —the *block*— which distinguishes the theater drama from the political debate. In Dutch parliament, the debate on each topic is organized as follows: each party may hold a speech by a member standing at the central lectern; other members may interrupt this speech; the chairman can always interrupt everyone. Most often, when all parties had their say at the central lectern, a member of government answers all concerns raised while speaking from the government table; again he or she can be interrupted. In most cases this concludes a topic, but variations are possible and do occur, e.g., several members of government speaking or a second round of the whole process.

The *block* is an important debate-structural element because it indicates who is being attacked by the interrupters. Thus for the Dutch situation the DTD becomes

<code>topic</code>	→	<code>(block)+</code>
<code>block</code>	→	<code>(speech stage-direction)+</code>

If this block structure is not present in meeting notes, then each topic contains exactly one block. Thus also such notes fit this DTD.

Note. For presentation purposes, the DTD presented here is the core of the model. The DTD actually in use contains additional elements and attributes for storing all kinds of metadata.

Figure 1 contains a visualization of a one-topic debate using the block structure. It is created with an XSL-stYLESHEET from the XML. Each row stands for one block and each vertically positioned mouth stands for one speech. The size of the mouth is proportional to the length of the

¹<http://metalab.unc.edu/bosak/xml/eg/shaks200.zip> One of the referees pointed out the well-documented DTD for drama which is part of the TEI guidelines for text markup (<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/DR.html>). This DTD is a good starting point for modelling, but for our purposes both too general and too specific.

Debatstijlijn van " Beveiliging Hirsi Ali "

[Uitleg](#)

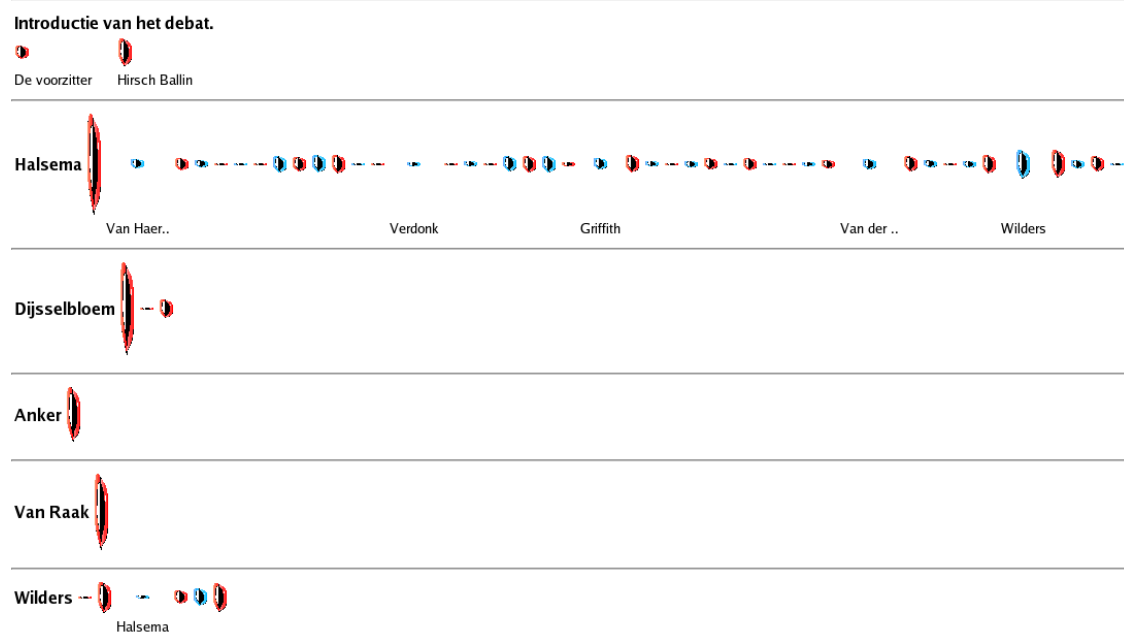


Figure 1: High-level visualization of the first part of the debate on the protection of Hirsi-Ali. Original available at <http://www.geencommentaar.nl/parlando/index.php?action=doc&filename=HAN8183A16>. The first speaker on the lectern is *Halsema* who is interrupted by *Van Haersma Buma*, *Verdonk*, *Griffith*, *Van der Stai* and *Wilders*, in that order. The name of the speaker is only shown at her first interruption.

speech measured in number of words. The speaker on the central lectern has the red mouth, the interrupters have a blue mouth. Interruptions by the chairman are not shown.

We end this section with two more observations on interesting structure in debates, also visible in Figure 1:

1. Blocks consist either of one uninterrupted speech or they have the form $(\text{red}, \text{blue})^+, \text{red}$, that is a sequence of pairs of speeches by the central speaker and an interrupter ended by the central speaker.
2. Zooming in on a block, if *A* is the speaker at the lectern and *B, C, D* are the ones interrupting *A*, then blocks very often look like $(\text{AB})^+(\text{AC})^+(\text{AD})^+\text{A}$, i.e., a sequence of small conversations with different members with *A* having the last word.

Debates in the Dutch parliament are governed by a set of written regulations and a set of unwritten codes. Both the observations above are instantiations of unwritten codes. The first observation restates the rule that the speaker at the lectern always has the last word. The second observation corresponds to the rule that a member of parliament may only once make a block of interruptions of a(nother) member at the central lectern. See [26] for these rules. Another rule is that someone may only interrupt another 3 times in a row. So according to these unwritten codes the second regular expression should be $(\text{AB})\{1, 3\}(\text{AC})\{1, 3\}(\text{AD})\{1, 3\}\text{A}$ and none of *B, C, D* should be equal.

Formalizing these written and unwritten rules in terms of regular expressions, and using them

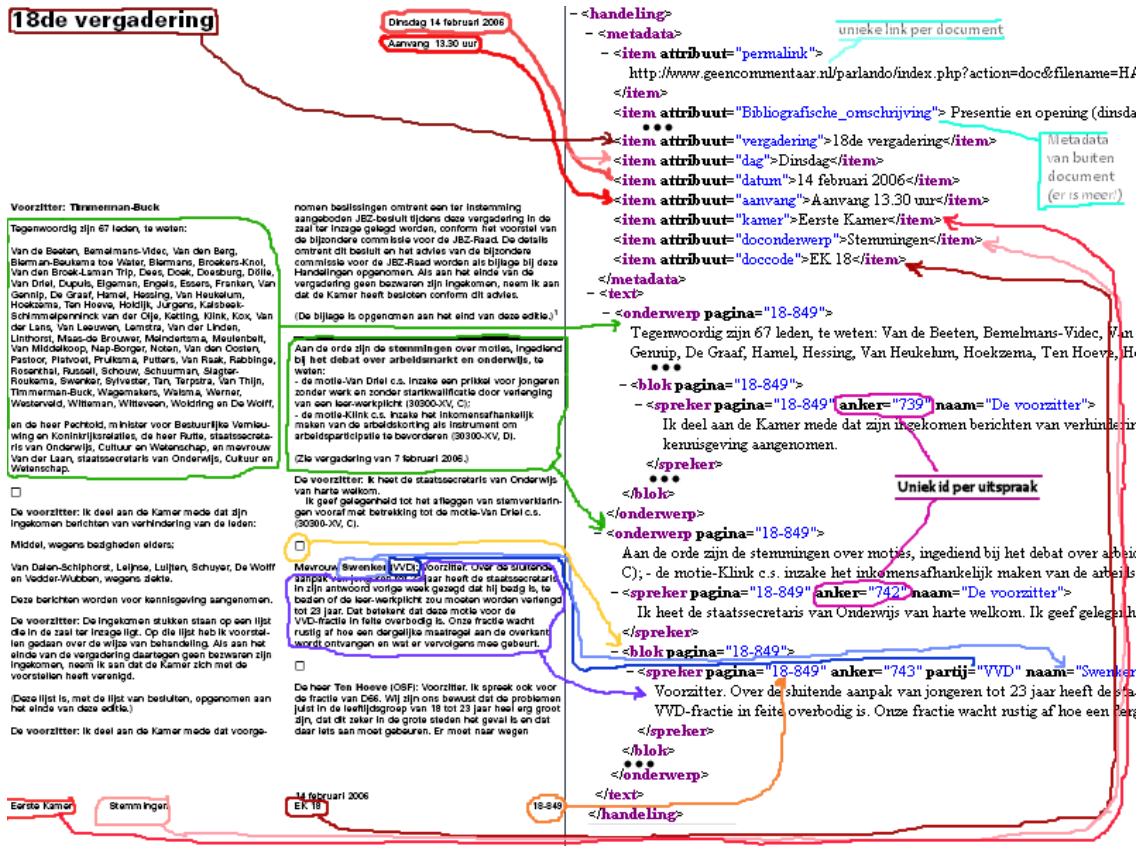


Figure 2: Example of the mapping from the description of a debate in PDF to the version in XML. Note how the start of a new block is indicated by a \square (mapping indicated in yellow).

to find *violations* is an interesting future direction of research.²

This internal structure of blocks can be used to create high-level overviews of debates that show who interrupts whom and which can be used for navigation. We present an example in Section 4.3. The regular expression which best fits or describes a block can be obtained by the algorithm which induces DTD's from a set of example XML-files described in [6].

3 Extracting structure: from flat PDF to deep XML

All documents contain structure. Often we can easily recognize titles, paragraphs and page numbers on the basis of their layout (e.g., position or size) and their content (e.g., a number) for example. This structure is often not explicit in digital form, especially not when the document is scanned and OCR'd. If the structure of documents in a corpus is standardized to some degree, the structure can be made explicit in digital form automatically [7].

The Dutch parliamentary proceedings shows this kind of standardization of document structure. All elements of a proceeding, for example topics and speakers, are represented in their own distinct way. This enables the automatic recognition of these elements. We programmed the text and structure extraction as an Extract-Transfer-Load process [21] consisting of eight steps.

Figure 2 gives a good indication of the mappings created in the transformation process.

²We have found such violations with Dutch members of parliament who have a new debating style like Wilders and Verdonk.

First we extract the text from the PDF using the open source program `pdftohtml`³ with the `-xml` option. This yields an XML file with, for each line of text, four coordinates which indicate the bounding box of that text. Multiple columns are detected and preserved. Some font and layout information is preserved but not all. The XML structure is simple and flat:

```

root   → (page)*
page   → (text)*
text   → (#PCDATA,b,i,span)*

```

The second step involves cleaning the XML output from `pdftohtml`. We make sure that the file is well-formed XML and we solve problems with diacritics and character encodings. In this step, we also fix the most common and recurring OCR errors. We found one specific error quite often: a space inserted before the last letter of a word. For example, the token `wij` is OCR'd as `wi j`. We repair this mistake with a regular expression.

The values indicating the position and size of the bounding box of the text are normalized in the third step because they vary depending on the device used.

In the fourth step, we analyze the document's layout. The margins, the number of columns and the header and footer are detected and marked. For this, we use the position of the text elements and the (deducted) position of the whitespace. Using this information, we sort the text of the body (i.e., not including the header or the footer) in reading order in the fifth step.

During the sixth and seventh step, different markers are placed in the text to signal the start of different elements in the document. In the sixth step we place markers indicating the position of text elements in the document. We place markers on places where paragraphs begin (they are indented), where there is whitespace and where a new column starts. This information is used in the seventh step where semantic markers are placed based on the content of the document. In the seventh step we use regular expressions to recognize standardized structure. The start of a statement for example, is represented as follows:

Mevrouw **Swenker** (VVD):

This adheres to the following structure: title, last name, party name within parenthesis, colon. This is then followed by the statement this person made. The start of a statement can only begin after a whiteline marker or the start of a new column. We convert this to the following semantic marker

```
<speechstart speaker='Swenker' party='VVD' .../>
```

with the `...` containing additional information.

The output of this step is an XML document that is flat and contains markers that mark the beginning of structural elements, but not the end. In the last step we determine the scope of paragraphs, speeches, blocks, topics and other structural elements. We simply replace the markers by XML tags that enclose the entire element and nest the elements appropriately. This is done by performing a cascade of groupings starting with the elements that need to be most deeply nested: the paragraphs `p`. XSLT 2.0 has a very useful command for this task: `xs1:for-each-group`. This command, new in XSLT 2.0, replaces the so-called Muenchian method which was often used in version 1.0 of XSLT [16].

The result is an XML file with the same text as the original document but with explicit structure. The file is valid with respect to the DTD described in the previous section.

3.1 Evaluation

To deal with OCR-errors in the input data, the patterns which trigger the placing of markers have to be more flexible than indicated above. This introduces errors in the transformation. [10] contains an evaluation of the quality of the transformation process. Table 1 from [10] shows the percentage of correctly marked elements for seven of the most important structural features.

³<http://pdftohtml.sourceforge.net/>

Feature	Score	comments
Topics	77,8%	All recognized, but in 22.2% included too much text
Blocks	100%	
Speakers	88.7%	Caused by OCR errors
Paragraphs	93.5%	
Header	91.5%	Caused by OCR errors
Footer	92.5%	Caused by OCR errors
Stage directions	73.5%	

Table 1: Percentage of correctly extracted structural elements; evaluation done on two complete days of proceedings (50 pages).

4 Applications of the XML structure

We describe five applications of the XML structure. None of these applications is possible when working with the unstructured text data. The applications are entry point retrieval and the use of permalinks, complex content and structure queries, automatic creation of tables of contents and navigation menus, graphical result aggregation, and savings on bandwidth.

4.1 Entry point retrieval and permalinks

The most natural answer unit in a retrieval system for parliamentary debates is the speech. The result page after a keyword query then will be a ranked list of items consisting of the name of the speaker, her party, a photo of the speaker, the date of the speech, a relevant text snippet of the speech, a hyperlink which points to the anchor attached to the speech within a debate, and a hyperlink to the original PDF source. This is how it works in the UK on the site <http://www.theyworkforyou.com>, on the site of the European Parliament, and also in the retrieval engine that we built for the Dutch data <http://www.polidocs.nl>.

Though natural, this notion of answer is by no means standard for parliamentary retrieval systems. The search systems of the German and Flemish parliaments return the proceedings of one day. These can be PDF files with two columns of up to a 100 pages. In the Netherlands, the situation is even more complex:

- proceedings before 1995 are available at <http://www.statengeneraaldigitaal.nl/>. The answer unit is the proceedings of a complete meeting;
- proceedings after 1995 are available at <http://parlando.sdu.nl/cgi/login/anonymous>. The answer unit roughly corresponds to one topic. It is indeed roughly as topics almost never start at the top of a page nor finish at the bottom of a page, and the PDF documents at Parlando are divided into overlapping sets of pages;
- preliminary proceedings are available at <http://www.tweedekamer.nl/>. Search is not really possible on this site. Preliminary proceedings are available in HTML which is shown together with a navigation menu which contains the same topic–block–speech hierarchy as described in Section 2.

During the transformation from PDF to XML we add a unique anchor ID to every speech. The ID is structured as a *Digital Object Identifier* (DOI) [8]. This ID thus is like a Uniform Resource Identifier (URI) and constitutes a unique permanent reference to each speech, in line with the recommendations on publishing government data by the W3C [3, 5].

The permanent hyperlinks (permalinks) for each speech made in parliament have many applications besides making entry point retrieval possible. Examples are easy referencing in emails, weblogs and even scientific papers. Permalinks also stimulate third party development of websites (such as mashups) based on this data [2].

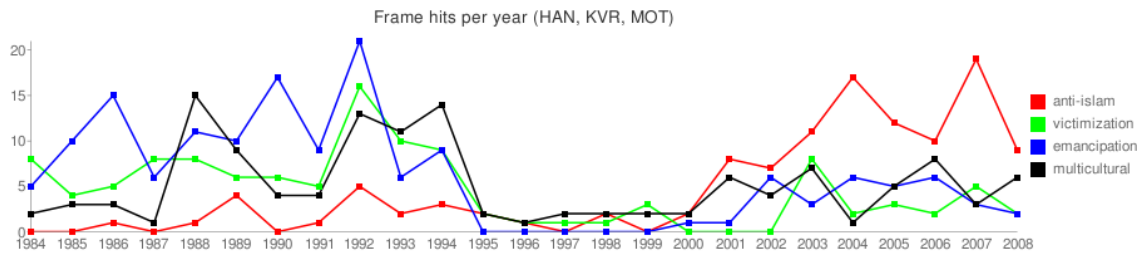


Figure 3: Google chart created by an XQuery run over the XML dataset. The XQuery and the code to produce the figure can be found in [1].

4.2 Complex content and structure queries

The explicit XML structure allows one to formulate information needs using natural XPath, XQuery, XSLT or NEXI [14, 20] expressions. We illustrate this with some examples:

- *give speeches about Islam from debates about immigration* can be formulated as the NEXI query

```
//topic[about(.,immigration')]/  
speech[about(.,'islam')].
```
- *give all speakers who interrupted Geert Wilders during the Islam debate* can be formulated in XPath 1.0 as `//topic[@title='islam']//block[@speaker='Wilders']`

```
//speech[@speaker != 'Wilders']/@speaker.
```
- *give a list of these speakers together with their number of interruptions ordered by that number* is expressed in XQuery or XSLT using the structure of the XPath expression from the last bullet and the `fn:count()` function.
- *Create a cross table of speakers at the lectern and their interrupters and list the number of interruptions in each data cel* is a typical task for XSLT. The result for the *Algemene Beschouwingen* on September 17 2008, containing 624 speeches in one debate, is reproduced in Table 2.

Information science students find it easier to formulate such complex queries in XSLT or XQuery directly on the original XML files than to state them in SQL on a relational representation of a debate [1]. [1] contains the XQueries needed to answer the rather complex temporal analysis questions from [22]. The XQueries output a csv-file with all hits on a query and the code to produce a Google-chart, as in Figure 3.

4.3 Automatic creation of tables of contents and navigation menus

The notes of a one day meeting of Parliament are long, averaging 50.000 words and typically between 50 and 100 pages two column PDF. In Belgium, Germany and The Netherlands, these are the documents returned to users. Unfortunately these documents do not contain a table of contents listing the topics discussed in a meeting. But even if tables of content were available they would be of little help when browsing these documents on a computer because they do not contain hyperlinks.

Since the topics are explicit elements in the XML version of the data it is straightforward to generate a hyperlinked table of contents automatically for each document. This can be done with XSLT. Alternatively one can create table of contents which also summarize part of the debate structure. In fact the debate timelines in Figure 1 are navigation menus: each mouth contains a hyperlink to exactly that part of the proceedings which record the speech represented by the mouth. This is possible because we created a permalink for every speech. These navigation menus are devices to provide focussed access to XML data [9, 25].

Op de spreekstoel	Achter de interruptiemicrofoon											Voorzitter	Totaal
	Kant	Van Geel	Rutte	Hamer	Wilders	Slob	Halsema	Pechtold	Thieme	Van der Vlies	Verdonk		
Kant	14	5	-	10	3	-	1	3	-	-	-	3	39
Van Geel	18	28	10	-	4	-	8	10	8	4	4	14	108
Rutte	-	14	35	17	-	9	13	-	4	-	-	9	101
Hamer	24	-	4	46	-	-	6	20	-	2	7	11	120
Wilders	-	5	-	3	11	2	11	6	-	-	-	7	46
Slob	-	-	-	-	-	5	-	10	-	-	4	6	25
Halsema	-	-	-	-	-	2	1	2	-	-	-	3	8
Pechtold	-	-	-	4	-	5	3	8	-	-	-	7	27
Thieme	-	-	-	-	-	-	1	-	-	-	-	-	1
Van der Vlies	-	-	-	-	-	-	-	1	3	2	-	3	9
Verdonk	-	-	-	-	-	-	-	3	-	-	3	2	8
Totaal	56	52	49	80	18	23	44	63	15	8	18	65	492

Table 2: Who attacks whom in the debate *Algemene Beschouwingen* on September 17 2008. Speakers at the lectern are listed in the first column; their attackers on the top row. The numbers in the cells indicate how often the person on the x-axis interrupted the speech by the person on the y-axis. The numbers on the diagonal (in gray) are the number of answers to interruptions given by the speaker on the lectern. Source: <http://staff.science.uva.nl/~marx/politicalmashup/AB2008/DebatstructuurAB2008.html>.

4.4 Result aggregation

Result aggregation and aggregated search [19] are important when there are many relevant hits and when hits themselves Parliamentary proceedings are large complex documents. Even one topic can be quite long. For instance, the meeting of September 18, 2008 took the whole day, consisted of 624 speeches with a total of 74.068 words, all within one topic. Figure 1 provides one possible way of visually summarizing this data. Table 4 gives a higher level summary, indicating who interrupted whom and how often. This table can also be presented graphically in the form of a social network, as is done in [15].

4.5 Savings on bandwidth

The Dutch parliamentary data from before 1995 was only available in printed form. Within the StatenGeneraalDigitaal project of the Dutch Royal Library this data is scanned and OCR-ed, resulting in complex PDF documents consisting of facsimile images of every page, the OCR-ed text and a mapping from each word to its position on every page.⁴

Such files can be enormous in size. For instance, the proceedings on <http://resolver.kb.nl/resolve?urn=sgd:mpeg21:19851986:0000761> are 72 pages PDF. The size of this file is 24 Megabyte. The same proceedings in XML is less than .5Mb, 2% of the original size. Downloading such large files takes too much time, especially in a search-and-browse use case in which many candidates must be quickly assessed and discarded. Apart from that, standard PDF readers like Acrobat become inconveniently slow.

What is a good solution to this problem? One could serve the XML file in conjunction with a stylesheet that provides a good layout. But for cultural heritage data, the look-and-feel is often important. Thus one could try to reconstruct the original data but now in digital format.

In [10] we have shown that reconstructing PDF documents from scanned and OCR'd data is feasible and leads to size-reductions of two orders of magnitude. The quality of the recreated PDF files is very good, and in several aspects better than the original. The most important gain of this exercise is the reduction in download time from unacceptably slow to instantaneous.

⁴See <http://www.statengeneraaldigitaal.nl/backgrounds.html> for extensive information on the digitization process (in Dutch).

We believe that the facsimiles need to be available as the ultimate source but that in a search and browse interaction process with the data the alternative, much smaller, version based on the XML is preferable. Users get results faster, they get clean hyperlinked files, and they use much less bandwidth. Once a user knows exactly which document she wants to consult, the large facsimile PDF can be downloaded.

5 Conclusions

We have shown that text extraction from Parliamentary proceedings based on regular expressions and XSLT is feasible, scalable, possible on both digital and scanned data, and leads to numerous benefits.

We stress that this extraction process is transparent, repeatable and independent of any software or hardware because we only use declarative programming languages with a well described semantics. This means that when the extraction scripts (which are themselves XML files, since it is XSLT) together with a copy of the XSLT reference [16] are stored together with the original digitized data in a safe place, it is in principle always possible to recreate the XML versions we have described here.

Several parliaments are digitizing their complete historical data. We are aware of efforts in the UK, Ireland, Australia, and the Belgium Parliaments. Our DTD is general enough to fit all these proceedings. This opens the possibility of creating a huge integrated multi-lingual XML repository of parliamentary proceedings. Such a repository will facilitate comparative parliamentary (historical) research.

References

- [1] L. Afanasiev and M. Marx. Operationalization of policy framing questions on parliamentary data with XQuery. <http://politicalmashup.nl/framing-questions-on-polidocs-data/>, 2009.
- [2] E. Aimeur, G. Brassard, and S. Paquet. Personal knowledge publishing: fostering interdisciplinary communication. *Intelligent Systems, IEEE*, 20(2):46–53, 2005.
- [3] J. Alonso et al. Improving access to government through better use of the web. W3C Interest Group Note 12 May 2009 <http://www.w3.org/TR/egov-improving/>, May 2009.
- [4] K. Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, June 2008.
- [5] D. Bennet and A. Harvey. Publishing open government data (W3C Working Draft 8 September 2009). <http://www.w3.org/TR/gov-data/>, 2009.
- [6] G. J. Bex, W. Gelade, F. Neven, and S. Vansummeren. Learning deterministic regular expressions for the inference of schemas from xml data. In *Proceedings WWW '08*, pages 825–834, 2008.
- [7] A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Managing information extraction: state of the art and research directions. In *Proceedings SIGMOD '06*, pages 799–800, 2006.
- [8] DOI. *The DOI Handbook*. International DOI Foundation, 2006. Available at <http://dx.doi.org/10.1000/186>.
- [9] N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors. *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007*, volume 4862 of LNCS. Springer, 2008.

- [10] T. Gielissen and M. Marx. Digital Weight Watching: Recreation of scanned documents. In *Proceedings Third Workshop on Analytics for Noisy Unstructured Text Data (AND 2009)*, pages 25–31, 2009.
- [11] M. Hajer. Setting the stage, a dramaturgy of policy deliberation. *Administration & Society*, 36(6):624–647, 2005.
- [12] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proceedings VLDB '06*, pages 9–16, 2006.
- [13] R. Hariman. *Political style. The artistry of power*. University of Chicago Press, 1995.
- [14] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Articulating information needs in XML query languages. *ACM Trans. Inf. Syst.*, 24(4):407–436, 2006.
- [15] R. Kaptein, M. Marx, and J. Kamps. Who said what to whom? Capturing the structure of debates. In *Proceedings SIGIR '09*, pages 831–832, 2009.
- [16] M. Kay. *XSLT 2.0 3rd edition Programmer's Reference*. Wrox, 2004.
- [17] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS*, pages 233–246, 2002.
- [18] A. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings VLDB '96*, pages 251–262, 1996.
- [19] V. Murdock and M. Lalmas. Workshop on aggregated search. *SIGIR Forum*, 42(2):80–83, 2008.
- [20] R. A. O’Keefe and A. Trotman. The Simplest Query Language That Could Possibly Work. In *Proceedings of the 2nd INEX Workshop*, 2004.
- [21] E. Rahm and H. Do. Data cleaning: Problems and current approaches. *IEEE Techn. Bulletin on Data Engineering*, 23(4), 2000.
- [22] C. Roggeband and R. Vliegthart. Divergent framing: The public debate on migration in the Dutch parliament and media, 1995-2004. *West European Politics*, 30(3):524–548, May 2007.
- [23] J. Seaton. The Scottish Parliament and e-democracy. *Aslib Proceedings: New Information Perspectives*, 57(4):333–337, 2005.
- [24] B. Sigurbjörnsson. *Focused information access using XML element retrieval*. PhD thesis, University of Amsterdam, 2006.
- [25] A. Trotman, S. Geva, and J. Kamps. Report on the sigir 2007 workshop on focused retrieval. *SIGIR Forum*, 41(2):97–103, 2007.
- [26] C. van Baalen and A. Bos. In vergadering bijeen. Rituelen, symbolen, tradities en gebruiken in de Tweede Kamer. In *Jaarboek Parlementaire Geschiedenis 2008*. Boom, 2008.