Proceedings of the 12th Web as Corpus Workshop, pages 33–41
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

The ELTE.DH Pilot Corpus – Creating a Handcrafted Gigaword Web Corpus with Metadata

Balázs Indig[1], Árpád Knap[2], Zsófia Sárközi-Lindner[1], Mária Timári[1], Gábor Palkó[1]
[1]Eötvös Loránd University, Centre of Digital Humanities
Múzeum krt. 6-8., H-1088, Budapest, Hungary
[2]Eötvös Loránd University, Faculty of Social Sciences, Research Center for Computational Social Science
Pázmány Péter stny. 1/A, H-1117, Budapest, Hungary
{indig.balazs,lindner.zsofia,zimanyi.maria,palko.gabor}@btk.elte.hu, knap.arpad@tatk.elte.hu

Abstract

In this article, we present the method we used to create a middle-sized corpus using targeted web crawling. Our corpus contains news portal articles along with their metadata, that can be useful for diverse audiences, ranging from digital humanists to NLP users. The method presented in this paper applies rule-based components that allow the curation of the text and the metadata content. The curated data can thereon serve as a reference for various tasks and measurements. We designed our workflow to encourage modification and customisation. Our concept can also be applied to other genres of portals by using the discovered patterns in the architecture of the portals. We found that for a systematic creation or extension of a similar corpus, our method provides superior accuracy and ease of use compared to *The Wayback Machine*, while requiring minimal manpower and computational resources. Reproducing the corpus is possible if changes are introduced to the text-extraction process. The standard TEI format and Schema.org encoded metadata is used for the output format, but we stress that placing the corpus in a digital repository system is recommended in order to be able to define semantic relations between the segments and to add rich annotation.
Keywords: webarchiving, corpus, metadata, trusted digital repository, semantic web, TEI XML, schema.org

Motto:

"It is hard to imagine how one might study the history of the developed world in the late twentieth and early twenty-first century without recourse to the archived web." (J. Winters)

1. Introduction

In the glossary of the handbook entitled *The Digital Humanities*, Gardiner and Musto (2015, 250) define web archiving as "the process of collecting portions of the *World Wide Web* to ensure the information is preserved in an *Archive* for future researchers, historians and the public". It is telling, however, that in the chapter focusing on digital archives as source materials of the present scholarly practices, born-digital archives and web archives are entirely omitted, as the authors solely speak about curated digital collections designed by (digital) archivists for the research community. Web archives are much less organised and curated then digital libraries or databases, and for this reason, are far less usable for (and used by) scholars. If Gardiner and Musto (2015) are right in their choice to emphasise the role of these digital sources in answering present scholarly questions, the fact that web archives do not play a significant role among these sources is a substantial problem for the digital humanities. There are several reasons why web archives are under-represented in the scholarly use of digital sources. The main reason is the lack of high-quality metadata, as source materials must have – among others – a publication date and its authors identified by the archival institution, otherwise, the reference to the material (be it paper-based or *born-digital*) is questionable[1]. The second reason is the uniqueness and authenticity of the records. Web archives usually contain many nearly identical versions of the "same" resource. This problem is exacerbated by the nearly inseparable dirt (recurring boilerplate text) among relevant content. The drawbacks arising from the unstructured nature of a web archive hinder its integration into the network of digital cultural heritage (DCH).

As suggested in (Weber, 2018), the limitations of web archives can be described along two main dimensions: accuracy and completeness. It is very difficult to tell if an archive actually captures *all* the content on the web *accurately* related to a specific topic.

Our method, by using websites' own archives, creates "complete snapshots" of their real content from time to time, which provides real populational data for the portals included in the project. This also means that the ELTE.DH corpus contains all documents from the selected portals' archives which were publicly available at crawling time.

Beyond creating a corpus for NLP applications, our work focuses on providing solutions to the aforementioned issues by developing a trusted digital repository complying with *Linked Open Data technology*. Our goal with this repository is to meet the essential demands of NLP, DCH and other disciplines uniformly.

2. Background

When it comes to crawling, web archiving or corpus creation, there are a number of options. The ISO/TR 14873:2013 standard describes the details of such workflows, however, distinct disciplines have come up with their own solutions ignoring this standard or only partially adhering to it. Holding on to the terminologies of the standard, we have conducted *selective web archiving* that is enriched with more and better metadata compared to *general crawling*. We argue that our method has a smaller footprint while

[1]Winters (2017, 240) deals with the problem of website dates in detail.

remaining easy to manage. This makes the whole workflow sustainable and scalable. In the following sections, we will review the already available tools and formats to place our solution among them.

2.1. Metadata

The standardisation process of web archiving practices, initiated and controlled mainly by national libraries (Oury and Poll, 2013), does not provide comprehensive guidelines to the standardised encoding of the texts extracted from web archiving activity. The situation is much better on the level of metadata. The technical report of *Statistics and Quality Indicators for Web Archiving* stresses the importance of different metadata types for curating web resources[2]: "Long term preservation also includes keeping safe the metadata associated with the resources in the Web Archive, which are critical for supporting collection management, access and preservation activities" (ISO/TC 46/SC 8 N).
The Metadata Encoding and Transmission Standard (METS) distinguishes four metadata types to be used in curated collections sourced from web archiving: (a) Descriptive metadata, (b) Structural metadata, (c) Provenance metadata, (d) Rights metadata. This is the theoretical standpoint, but since the creation of such metadata requires a lot of manual work, it is impossible to find a collection of archived web documents that complies with these requirements on metadata entirely. Therefore, there is virtually no reliable digital cultural heritage source for researchers. In contrast, there are metadata standards which cover fine-grained requirements. The only standard that could gain large-scale adoption is *Dublin Core*, which is not refined enough to comply with the aforementioned standards. Our repository uses Schema.org, a metadata standard we have chosen for several reasons:

- Schema.org is designed explicitly for storing information about web resources
- It has a dynamic, community based development (in contrast with robust standards, such as METS)
- It is increasingly popular on the web, which makes it easy to extract metadata from the HTML source
- It is compatible with semantic web technology (Linked Open Data)
- It has a growing usage in the digital cultural heritage domain (e.g. Europeana)

2.2. Existing Hungarian Corpora

The Szeged corpus is the largest, manually annotated corpus (Vincze et al., 2010) in the Hungarian language containing 1.2 million tokens, KorKorpusz (31,492 tokens) is similar but smaller corpus based on a recent pilot project (Vadász, 2020). The first Hungarian gigaword corpus was the *Hungarian Gigaword Corpus* (Oravecz et al., 2014) with 1,532,933,778 tokens. Both aforementioned corpora contain text only from curated sources (newspapers, literary texts, social media, legal texts, etc.) that are not entirely from the Internet. The first Hungarian web corpus that was created by Kornai and his colleagues (Halácsy et al., 2004) is called the *Hungarian Webcorpus*. It was later superseded by the 1.2 billion token *Pázmány corpus*[3] (Endrédy and Prószéky, 2016) and the 2.5 billion token *HuTenTen corpus* (Jakubíček et al., 2013), two larger corpora entirely from the web. Nowadays, large corpora are utilising the *Common Crawl archive* like the OSCAR corpus (Ortiz Suárez et al., 2019) with 5.16 billion (2.33 billion deduplicated) words in Hungarian. However, the documents presented in these corpora often contain gaps due to deduplication.
All of these corpora – except the ones based on Common Crawl – have the same flaw, namely that after their creation and publication, several errors were discovered in the tools used to create them, and these errors could not be corrected as expected. The reason being that their original source HTML files have been deleted – and these are unavailable in an unmodified form on the web.
Since then, there have been numerous attempts to create web-based corpora, but these were not published and could not arouse public interest, as web corpora and crawling became increasingly common tools. The speciality of the corpus and the method presented in this paper lies in the fact that it unites the experience from the above mentioned corpora into a manually curated gigaword web corpus, which includes metadata and can be built from the source of the downloaded web pages in a reproducible manner.

3. From the web to a corpus

To put our method into a larger perspective, in the following sections we will describe the process of corpus creation in an abstract workflow (see Figure 1.), where the elements have to be further specified by certain design decisions.

3.1. The Web Crawler

Classical web crawling can be characterised by only a few parameters that are set at the start of the process. These parameters include the initial seed of URLs where to start the crawl from, the maximal depth, and the breadth to restrict the crawler's movement. In some cases a set of targeted domains is also specified. Although there are only a few widely used crawler engines, it is hard to characterize these as most web libraries (e.g. `Python requests`, `wget`, etc.) can be used for crawling nowadays and the desired output varies from corpora to "exact offline duplicates" of websites. Here we would like to mention three crawler engines: both *Heritix*[4] and *Apache Nutch* (Laliwala and Shaikh, 2013) are used in the *Internet Archive* and *Common Crawl* projects. The third crawler engine is *Spiderling* (Suchomel and Pomikálek, 2012), which was developed by the authors of Sketch Engine (Kilgarriff et al., 2014). These crawlers are fast, generalised tools, but for targeted or spe-

[2] `http://netpreserve.org/resources/IIPC_project-SO_TR_14873__E__2012-10-02_DRAFT.pdf`

[3] The Pázmány corpus was the first Hungarian corpus which separated edited text (news articles) from unedited text (comments).

[4] `https://github.com/internetarchive/heritrix3`

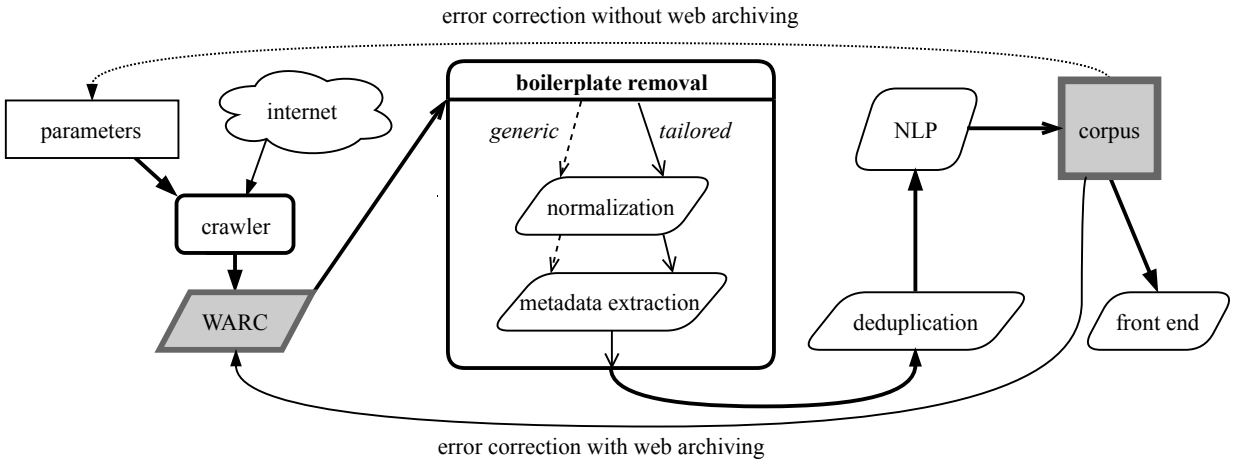

Figure 1: The abstract workflow of web corpus creation. Parallelogram-shaped boxes denote the optional phases, the grey background denotes the produced data.

cialised crawling they became tedious to use. This may explain the numerous different libraries used for crawling.
Nowadays, we do not even have to use a crawler as we can begin the process with *Common Crawl*, *The Internet Archive* and similar resources. In this case, the first step is to clean the actual data, and remove collected garbage. Due to the nature of the internet, there are numerous aggressive SEO traps out in the web that are used to optimise the page rank in search engines that end up in the archive. These traps are placed in such a manner that they can divert crawler bots from their original course when these bots stumble upon them. Such general bots cannot distinguish "normal" pages from these traps, a task that humans are able to carry out in a matter of seconds. Another common problem using these sources is the need for deduplication (see Section 3.3.), which causes the waste of resources on both sides (crawler and deduplicator).
To overcome these problems, Indig et al. (2019) built a new NLP-oriented crawler tool called *Corpus Builder*, which utilises a very different approach from the above: a two-level crawling method. By using targeted crawling of large or medium portals, they claim that with their method, it is possible to crawl only the targeted portals virtually without duplication, with a small footprint and in a sustainable manner. Their main idea is the exploitation of two different levels of recurring patterns ("archives" and "article" pages) that can be discovered by analysing the structure of different web pages.

3.1.1. The Article Crawler

The first and obvious level is the recurring boilerplate content on the actual web pages of a portal. Objects of the same type are usually stored in the database back end and generated on demand into specifically formatted web pages. This is the internal nature of the data. In this paper, we call these pages "articles" regardless of their content type: whether they represent news articles, images with captions in a gallery, product descriptions and customer reviews in a webshop, posts in a forum or blog, etc. The output pages for these content types look the same in narrower time frames for a certain portal, but they can be very different from website to website. These pages are generated on the server-side, so we must collect HTMLs and extract their content. If we collect a mass amount of web pages representing the same objects from the same portal using the same template selectively or classify them after collection, the (uniform) boilerplate can be easily removed with a few simple rules per class, therefore this task does not require complex tools.

3.1.2. The Archive Crawler

The second level arises from the first: how can we systematically collect such web pages on a given portal? The answer is very simple: portals created for human readers are very likely to have some kind of a "table of contents", "product or topic list" or an "article archive", which display all available pages in a structured form. Because objects are traditionally stored in taxonomies – e.g. temporal (years, months) or other feature-based (colour, shape, price, etc.) – that can be enumerated and each object has finite number of possible values. If we enumerate articles for right feature values, we will gather links to all pages of the same layout systematically from the given portal.

3.1.3. The Possible Parameters for Portals

Using the two-step method described above, it is possible to gather a massive number of web pages even from only a small number of portals that will have virtually no duplication and effectively zero garbage pages in contrast to the general crawling methodology. This method has been successfully tested on three Hungarian news portals (Indig et al., 2019), while the further generalisation of the method for the steps following the crawling of different portals with different schemes and layouts requires further elaboration. Indig et al. (2019) assembled the minimal number of parameters that are needed to handle such portals in a unified framework. The major highlights of the configuration are showcased as the following:

- The date of the first and last article, or page number where applicable

- The archive URL format with the placeholders to be substituted
- The function for finding the *next-page URL* for the archive where applicable
- The function to get the article URLs from the archive pages
- Boolean variables to answer the following questions: is the archive paginated, infinite scrolling, or date-based?

One can distinguish between crawl-based (applicable for the current crawl), portal-based (which applies to the crawled portal regardless of crawl settings), and portal-specific configurations. Our method follows the latter direction for crawling. For problems not addressed by Indig et al. (2019) we present our solutions in Section 4.

3.2. Boilerplate Removal

There are a lot of pages that present the same type of objects or articles surrounded by the same boilerplate content (i.e. scheme or template) – menu, advertisements, etc. in a portal. In most cases, this boilerplate content can be characterised by not having multiple paragraphs with many lines of text, but containing short texts, as well as many links and pictures (Pomikálek, 2011). The process of boilerplate removal can be broken down into two steps presented in the following sections.

3.2.1. Normalisation

By normalisation we mean the reformatting of visual elements into a simpler visual annotation (e.g. the elements of Markdown language or XML tags) to create a common ground for the text of different portals in the resulting corpus. Normalisation is not a trivial task: most tools extract paragraphs as *plain text*, however, *visual formatting elements are essential* for humans and may also help the machine reader, therefore these elements should be kept and standardised.

3.2.2. Metadata extraction

Curated metadata is the cornerstone of proper (web) archiving. It can be regarded as gold standard labels for each document, which can later be utilised for training or benchmarking ML algorithms (i.e. authorship attribution, keyword extraction, topic modelling, etc.). There are automatic tools for extracting metadata from the crawled web pages such as the *Web Curator Tool*[5] or *Apache Tika*[6]. These tools extract standards compliant descriptive metadata automatically from the crawled web pages, but they are very complex and it is difficult to understand and improve their method for the specific portals. Moreover, they are plagued with the same problems as other boilerplate removal tools (see Section 3.2.3.): their heuristics and output formats are wired in by design and it is very hard to change these without major conflicts.

When the these programs yield deficient output for the targeted portals – for example due to the lack of knowledge about the typographical rules of the language, or when the output is missing some important variables, – it is inevitable to implement a custom metadata extractor methodology. We decided to use this method to allow future modifications, to be able to compare results with the presented generic tools (see Section 3.2.3.), and also to demonstrate how easily our method can be implemented. Our findings will be described in Section 4..

3.2.3. Existing Tools and Techniques

As web page layouts, coding styles, and HTML standards differ throughout the portals and were used differently over the years, the boilerplate removal task is mostly solved by clever heuristics, which makes it hard for the users to create general measurements and comparisons between them. It is also hard to set their parameters, fix, extend or modify their functionality. Some tools are designed to remove boilerplate from a single page, while others use multiple pages of the same layout to delete recurring elements (Endrédy and Novák, 2013). In this paper, we could not survey all the available methods, therefore we are comparing *JusText* (Pomikálek, 2011), a tool created directly for NLP-centric web crawling and *Newspaper3k* (Ou-Yang, 2013), created especially for news portal crawling. Both modules are still popular and widely-used because of their simplicity and effectiveness. They both remove formatting and yield plain text paragraphs, but the latter tool supports extracting metadata from web pages and has other features to simplify the crawling process for beginners.

We followed the route marked by Indig et al. (2019) and created our own handcrafted boilerplate removal rules. At first we found ourselves in a dilemma about choosing between regular expressions and HTML parsers. Regular expressions are simple, and it is also easier to train machines to create them, while HTML parsers are easier to create, read and maintain for humans, but are harder to automate. As some of the portals recently added to the corpus have very complex layouts, it is not feasible to further extend the number of portals using regular expressions. For example, it may be impossible or become very unpractical to encode various attributes and their combinations (which might be in arbitrary order due to the structure of HTML).

We compared the aforementioned methods on our gold standard data set[7]. This measurement is presented in Section 5., followed by other details of our method.

3.3. Deduplication and NLP

Sometimes the exact same content – available on several domains – can be stored in the web archive multiple times, but, of course, we need one intstance only. There are great tools for deduplication (like Onion (Pomikálek, 2011)), but their waste of valuable resources, such as disk space and network bandwidth is not ideal. When using targeted crawling, such as Indig et al. (2019), we can select only those distinct URLs which are needed and so bypass the process of deduplication.

[5] https://webcuratortool.readthedocs.io/

[6] http://tika.apache.org/

[7] Some elements were kept or thrown away by design decision that may not match with the compared tools or future use cases. However, we support the change of these decisions by the user.

The main problem with deduplication – besides wasting resources – is that some parts of a document or the whole document may become missing because it had been recognised and deleted as a duplicate. This undermines the completeness of the crawling which is the the first priority for some user groups (e.g. humanists and sociologists). The publicly available corpora that were created for NLP purposes have further disabilities: their sentences are scrambled to avoid the infringement of copyright laws. This makes the analysis of full documents – an emergent trend – impossible. The role – and legal privilege – of national libraries is to preserve documents in its entirety, even for born-digital materials. This role can be fulfilled with our method, in contrast to the traditional ones.

Different levels of NLP annotation can optionally be applied before or between the deduplication with the plethora of available tools. Until recently, texts have been stored only after this step in some format, however, the increasing performance of NLP tools makes it advisable to store crawled content also in raw format (e.g. WARC files) to be able to correct errors found in the processing pipeline. This is mainly important to humanists, sociologists and other scholars outside NLP where the specific text is the subject of analysis, in contrast to NLP, where only the amount of text matters.

3.4. The Final Format and Front End

The process of creating the output from the HTML files can be split into four steps for easier maintainability:

- *Simplification of HTML* by finding the tightest bounding HTML tag of the whole text content and decomposing unneeded subtrees[8]
- *Extraction of paragraphs and metadata* from the HTML tree keeping only specific – intended – formatting
- *Rewriting elements to a unified format* by standardising site-specific formatting
- *Writing the output file according to the expected format*. In this step, the fields get their final place and canonical names

The first three steps contain well-defined portal specific instructions, while the fourth is only dependent on the output format, which – as it is totally separated from the others – can comply with the actual purpose and front end in the future. Some user groups have special requirements, such as full documents and metadata, while others only require the raw text. Nonetheless, both requirements can be achieved at the same time.

In the field of NLP, three main use cases exist. To search patterns in large corpora, the classic vertical format used primarily by the Sketch Engine (Kilgarriff et al., 2014) is recommended. If the aim is to process the corpus with a wide variety of standard NLP tools, the CoNLL-U format[9] is adequate. If the goal is to put documents to a full text search engine or into a language model, it is necessary to comply with the input expectation of such software, which is usually raw text.

In the field of digital humanities, – especially in philology,– the XML document markup language and the Text Encoding Initiative (TEI) recommendation have become dominant over the decades (Schreibman et al., 2008). TEI makes the versioning and annotation of the enriched articles possible in an easy and reliable way, and it is also capable of storing metadata and the body of the document structurally in one file. This format satisfies NLP users as well, while opening the resulting corpus for other audiences including librarians, humanists and sociologists. TEI also allows the verification of the authenticity of the source text by the metadata and increases the reproducibility of research which has an increasing importance in the 'distant reading' paradigm (Da, 2019). Text can be converted to a simpler form corresponding to the actual use case, while keeping the master copy untouched, in a similarly to how it is done with images by resizing and cropping them on demand dynamically.

4. Method

We examined several Hungarian news portals and increased the number of examined portals to six, compared to the three portals examined by Indig et al. (2019) in order to test how the presented method can be applied to portals of different structures. First, we selected mainstream Hungarian news portals, because these contain a vast number of articles. As a secondary priority, we included portals that are special from the perspective of used web technology and architecture. We wanted to reach a milestone, where adding new portals and maintaining the existing ones is a routine task that can be handled by as little manpower as possible. In this section, we describe the main highlights of our crawling method compared to (Indig et al., 2019) (for further comparisons see Section 3.)

4.1. HTML Parsers vs. Regular Expressions

We decided to change the regular expressions used in *Corpusbuilder* (Indig et al., 2019) for Python functions, which use an HTML parser to handle the input as an HTML tree. Using HTML trees enabled us both to simplify many regular expression patterns and to support many different layouts. With this change, the accuracy of extracting article URLs from the page archives has dramatically increased, as we found that on some portals different columns may be hosted on different domains, or – while using the same site template – they may not match the expressions written for extracting URLs. This can be recognised by tree searching expressions more easily than with regular expressions. This, of course, sacrifices speed for clarity and precision, but saves us from HTML fragments slipping through regular expressions.

[8] There are three classes of decomposing rules: a) general rules used for every portal, b) "must-have" portal-specific rules, c) rules which follow certain design decisions about the data to be extracted.

[9] `https://universaldependencies.org/format.html`

4.2. The Refined Archive Crawler

The *date-based pagination handling logic* (Indig et al., 2019) was separated from other pagination methods, as it allows sorting and can be used to filter crawling by specific date intervals, since we found that date-based pagination can be and is combined freely with the other methods. We also introduced support for open (date) intervals.

Our other significant change was in handling *infinite scrolling*[10] and *active archives*[11] together in an easy-to-understand form by extracting the page URLs before determining the URL of the next archive page. We have broken down the possible patterns of finding the next archive page URL to the following cases:

- There is no pagination at all
- There is a *next page link* which we need to use
- There is *infinite scrolling*: we use the page number from the base value to "infinity" where no more article URLs are detected
- There is page numbering: we use the page number from the base value to a portal-specific maximum
- There is page numbering, but we expect the archive to expand during crawling (can be identified by finding known article URLs during crawling)

By using these features, all examined portals could be handled, therefore we narrowed down our experiments to six portals that showcase all of the described features, and allows them to be thoroughly tested.

4.3. Advanced Metadata Extraction

Metadata can be extracted from multiple sources from an article page. We identified and handled the following:

- Date, title and column are frequently encoded *in the URLs*
- HTML meta tag properties which can be encoded according to various conventions (like Dublin Core, Schema.org, etc.) that are mainly included for Search Engine Optimization (SEO) purposes
- The increasingly popular JSON-LD, storing properties that were previously stored as meta tags, but in a more structured form
- From the content of the HTML itself, where it is included to be displayed for the user

There are several portals that use more than one of the above sources of metadata. We also found examples where different sources yielded contradicting results or missing values, these are probably due to bugs in the websites' engines. Older articles tend to have more of these errors as they were probably converted from a previous layout and the conversion introduced such errors[12]. Some portals partially or fully generate metadata dynamically by using JavaScript and non-standard data-sources. This practically makes it impossible to extract such metadata with traditional tools and forces us to use a portal-specific solution.

4.4. Converting HTML to the Output Format

To handle millions of pages without reading them – through "distant reading" –, we invented utilities to examine, analyse and normalise the tags and the scheme used by a portal, and then freely convert it to the new and customisable output format. We started with cutting the HTML to the relevant part, as mentioned in Section 3.4..

The first utility function helps to *filter* out tags that do not contain any text. Next, we introduced placeholders to *simplify* some elements (e.g. links). The second function aids in *simplifying* the tags by manually selecting groups that belong to the same class (e.g. formatting, embedded content, etc.), but are specialised to the portal's scheme.

This method is quite effective even without portal-specific parameters. Table 1 shows how the number of tags (from one of the examined domains) is reduced after using these tools allowing further fine-grained modifications in an iterative manner.

	No. of tags	%
all tags	33,466	100
text containing tags	18,517	55
after simplify tags	359	10
relevant tags	267	7

Table 1: Illustration of how the number of tags to be analysed manually decreases in magnitude.

4.4.1. The Tree Representation

Possible layouts for all URLs of a domain were described with the help of a *tree-representation*: the subtrees of the contents' tightest bounding HTML tag for all pages were merged, counting the frequencies of each element and the cumulative length of their immediate text. It was also marked if a specific tag had no child elements in the tree. The resulting frequency distribution allows efficient examination and handling of subtrees for all URLs at once.

In order to be able to make decisions concerning the remaining tags, we built a *tag dictionary*. To each tag (or simplified tag), we assigned the average length of the contained text, the average number of descendants, and the average length of the immediate text supplemented with a sample of occurrences (URLs). This dictionary was augmented with the operation to perform at each occurrence of that specific tag. As we formalised the operators, their execution was made by the code automatically generated from the dictionary. These steps can be iterated to gain more insight on the portal's scheme and finally arrive to the desired form.

[10] A technique used to dynamically add new content to the page when the user scrolls down.

[11] If new elements are added to the archive during crawling, the list of articles will be divided to pages in a way that their content URLs will appear on different pages than as expected. This makes it impossible to handle archive pages' URLs as permalinks.

[12] This can be solved by crawling articles as soon as possible after their publication.

4.4.2. Rewriting Rules and Transformation Methods

When standardising and rewriting elements, we found the following operators useful:

- decomposing (deleting the tag with its contents, e.g. advertisements, boilerplate)
- unwrapping (deleting the tag, keeping its contents, e.g. text anchors)
- unwrapping all descendants (simplifying a block)
- rewriting tags context-free
- splitting tags to super-subordinate pairs (e.g. when the content and formatting properties are in the same tag)
- rewriting tags context-specific (special blocks)

These operators can be applied sequentially in the proper order for every URL. We narrowed down the various layouts (e.g. left, right, top block) into a few, portal independent types of blocks that we intended to keep. The *context-specific rules* mark the root tag for each block we found, so their subtrees can be handled by independent dictionaries in the same way. The analysis of the visual layout of the examined portals shows that there are no blocks embedded into other blocks. This property allows us to rely on the described two-level transformation with a low number of distinct tag dictionaries modified by the defined operators.

To conclude, normalising the tags and then rewriting them to the final schema are independent steps which can be achieved with successive approximation in an iterative manner. This allows us fine-grained control to change design decisions or customise the output (TEI XML in our case) easily at all times.

5. Evaluation

We ran our crawling on a low-end desktop machine (Intel i3, 4 GB RAM) for 30 days on a 100 MB/s connection (with rate-limiting to avoid the hammering of the remote servers) using circa 100 GB of disk space to demonstrate the effectivity of the method presentes here. It is not possible to compare this method's crawling performance to other general crawler engines mentioned earlier, as the workflow and methodology differ significantly (see Section 3.1.). It is possible, however, to compare the crawling accuracy to the most widely used archiving practice: the Internet Archive (see Section 5.2.). It is also possible to compare our site-specific rule-based boilerplate removal and metadata extractor functions to the mainstream crawling methods (see Section 3.2.3.).

The goal of the compared tools and their design differs significantly so the way how to make an objective comparison was not at all obvious. When comparing our method with the aforementioned tools, we strived to highlight performance differences due to design, while separating them from the strengths and weaknesses stemming from the methods themselves.

5.1. The data set

We extracted a total of 2,227,180 articles from six Hungarian news portals, this signifies 984 million words (without tokenisation) of extracted text without metadata from November 1998 until September 2019. We visualised the annual distribution of articles to see the estimated growth in the number of articles and the expected number of articles per year (see Figure 2). The figure shows a clearly growing tendency in the number of articles published on the crawled portals during the last twenty years – except 2019, which does not qualify as a full year at the time of measurement. In the case of the six portals, this means that more than 200 articles have been published on average every day in the recent years. These numbers tells us that by adding new portals the quantity of the crawled articles and the volume of the corpus can be increased quickly and easily with low human resource investment and a lightweight technical infrastructure.

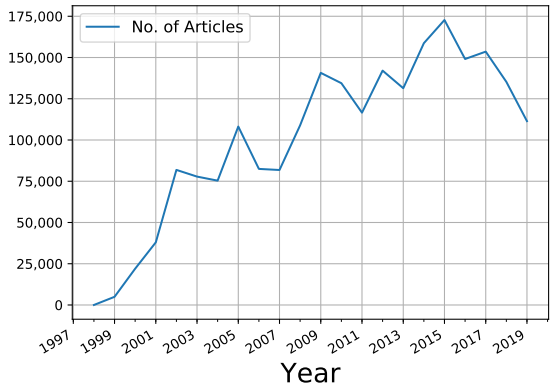

Figure 2: The annual distribution of 2,227,180 articles from six portals from November 1998 to September 2019. The number of articles per year is increasing. The decrease at 2019 is due to the fact that it is not a full year at the time of measurement.

In Table 2, we can see the performance of the boilerplate removal tools in different scenarios. We examined *JusText* and *Newspaper3k* on the full HTML code, the article body and constrained to the original and the cleaned up paragraphs. We wanted to check whether an educated initial guess (on the text's location) helps these programs or not. As the former package does not extract metadata separately, we present numbers with metadata and provide the number of words without metadata in brackets. The numbers have some small differences that suggests that a more detailed evaluation of the content is needed. We also compared the actual values of the extracted metadata (author, publication date, title) in terms of precision and recall for *Newspaper3k* (see Table 3). Our educated initial guess does not help metadata extraction, but for the text extraction it has a potential because it rules out unwanted content in one step. It is clear from these that our method is superior to the compared ones, however, a content-based comparison of the extracted paragraphs is needed in order to be able to evaluate the mentioned methods objectively. We argue that if full articles are chosen, the precision provided by

our method is needed to ensure that the right amount and quality of texts can be extracted with the compared methods.

	Full HTML	Article Body	Paragraphs	
			orig.	clean
All Text	12,99	1,757	1,085	982
Justext	1,157	1,020	919	918
Newspaper3k	992 (963)	974 (970)	919	917
Our Method	1,028 (984)			

Table 2: The extracted text from different parts of the HTML with different tools in million words. *Newspaper3k* and our method is displayed with and without metadata.

	Full HTML	Article Body
Newspaper3k (precision)	0.77	0.69
Newspaper3k (recall)	0.52	0.26

Table 3: The content-wise comparison of metadata (author, title, publication date) extracted by *Newspaper3k* and our method (=1.0).

5.2. Crawling Compared to Archive.org

We compared our results of the six crawled news portals to the *Internet Archive* as the "standard" source of web archiving. We evaluated whether the same set of URLs could be acquired using the *Internet Archive*, and also compared the number of crawled articles by portals with data downloaded from *The Wayback Machine*.

In the following step, based on the mime type attribute, we removed all URLs from the *Internet Archive* data sets that represent content other than articles (e.g. images, scripts, etc.). Using the status code variable, we omitted all URLs that were not successfully downloaded for some reason (e.g. 404 errors and redirections). From our crawl we selected the timestamp of the last article downloaded for each domain, and removed all URLs from the *Internet Archive* data that were crawled after that date.

At this point, we still had hundreds of thousands of URLs in the *Internet Archive* data sets that represented e.g. certain taxonomy pages (date, category, author, search, etc.) or any kind of content other than single articles. Thus, we introduced a domain-level cleaning function for each crawled website, in order to remove all URLs representing content other than articles. This proved to be a difficult, time-consuming, iterative task, as in case of some websites, the URL structure changed multiple times over the years, making it nearly impossible to retrospectively identify URLs that certainly lead to articles. This is one important aspect why our method is much easier to use (even retrospectively), when the goal is to produce a clean corpus, without duplicated content. In the case of several websites, the URL structure was not logically constructed (e.g. tag archives have the same URL structure as articles; randomly generated version numbers appear at the end of some of the URLs, but not all of them; etc.), therefore in some cases, we had to restrict the comparison to certain columns of the portal, as it was very difficult to clean the data sets in a more generalised way.

Our next step was to normalise all URLs in both crawls. We removed http, https, www from the beginning, and port numbers (e.g. ":80") and parameters from the end of the URL strings. Using these normalised URLs, we created two dictionaries to store the URLs themselves and their slugs – the last part of the URL (after the last /) – for each portal. For some portals the URLs could not be used for a valid comparison, because the URL structure has changed over time, but not the slug, therefore – in these cases – we used the slug for our comparison.

With the steps described above, we reduced the number of *Archive.org* URLs from 8.9 million to only 1.2 million for the six crawled portals. After removing entries with wrong status codes 75.4%, after mime-type-based cleaning 53.7% of the URLs remained. While only 0.7% of URLs were removed in the date-based cleaning phase, after running website-specific cleaning functions and compiling the final list of URLs, just 13.5% of the initial number of URLs remained. We found that 846,343 articles are present both in our crawl and in the *Internet Archive's* data, while 1,082,484 articles are only present in the `ELTE.DH` corpus. A further 315,649 articles are only found in *Archive.org's* data. More work is needed in order to eliminate all possible bad URLs, however, it is safe to say that by using our crawler it is easier to achieve the same results than finding and downloading all relevant content from *Archive.org*.

6. Conclusion and Future Work

We have demonstrated that by using a low-end machine – which has similar computational power as our smartphones, the storage capacity of our pendrives nowadays – and minimal manpower it is possible to create a gold-standard quality gigaword corpus with metadata which suits many audiences at the same time[13]. As the presented work was only a pilot study to design and stabilise the workflow on many candidate pages, we plan to apply this methodology on several more websites, and start serving requests on site-specific crawling to provide data for research in multiple disciplines in a future version of this corpus.

In conjunction with the previously outlined plans, we intend to support national libraries with our research as they are responsible of keeping the data of our present for the future researchers who can thus provide objective and balanced research. One obvious step in this direction is to conduct research on how to keep the authenticity of web archives and how to eliminate the risks of tampering, retroactive modification and deletion of content which undermine scholarly credibility. We plan to utilise digital fingerprinting, signatures and blockchain technology on downloaded documents in order to keep them safe, while making them available for the widest possible audience.

[13]The software is published under the *GNU Lesser General Public License v3.0* at `https://github.com/ELTE-DH/WebArticleCurator` and `https://doi.org/10.5281/zenodo.3755323`

7. Bibliographical References

Da, N. Z. (2019). The computational case against computational literary studies. *Critical inquiry*, 45(3):601–639.

Endrédy, I. and Novák, A. (2013). More effective boilerplate removal-the goldminer algorithm. *Polibits*, 1(48):79–83.

Gardiner, E. and Musto, R. G. (2015). *The Digital Humanities: A Primer for Students and Scholars*. Cambridge University Press.

Indig, B., Kákonyi, T., and Novák, A. (2019). Crawling in reverse – lightweight targeted crawling of news portals. In Marek Kubis, editor, *Proceedings of the 9th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 81–87, Poznań, Poland, may. Wydawnictwo Nauka i Innowacje.

Ortiz Suárez, P. J., Sagot, B., and Romary, L. (2019). Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In Piotr Bański, et al., editors, *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom, July. Leibniz-Institut für Deutsche Sprache.

Ou-Yang, L. (2013). Newspaper3k: Article scraping and curation. `https://github.com/codelucas/newspaper`.

Oury, C. and Poll, R. (2013). Counting the uncountable: statistics for web archives. *Performance Measurement and Metrics*, 14(2):132–141.

Pomikálek, J. (2011). *Removing boilerplate and duplicate content from web corpora*. Ph.D. thesis, Masaryk university, Faculty of informatics, Brno, Czech Republic.

Schreibman, S., Siemens, R., and Unsworth, J. (2008). *A companion to digital humanities*. John Wiley & Sons.

Weber, M. S. (2018). Methods and approaches to using web archives in computational communication research. *Communication Methods and Measures*, 12(2-3):200–215.

Winters, J., (2017). *Coda: Web archives for humanities research – some reflections*, pages 238–248. UCL Press.

8. Language Resource References

Endrédy, I. and Prószéky, G. (2016). A pázmány korpusz. *Nyelvtudományi Közlemények*, 112:191–205.

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., and Trón, V. (2004). Creating open language resources for Hungarian. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlỳ, P., and Suchomel, V. (2013). The tenten corpus family. In *7th International Corpus Linguistics Conference CL*, pages 125–127.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (2014). The sketch engine: ten years on. *Lexicography*, pages 7–36.

Laliwala, Z. and Shaikh, A. (2013). *Web Crawling and Data Mining with Apache Nutch*. Packt Publishing.

Oravecz, C., Váradi, T., and Sass, B. (2014). The Hungarian Gigaword corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1719–1723, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Suchomel, V. and Pomikálek, J. (2012). Efficient web crawling for large text corpora. In Adam Kilgarriff et al., editors, *Proceedings of the seventh Web as Corpus Workshop (WAC7)*, pages 39–43, Lyon.

Vadász, N. (2020). KorKorpusz: kézzel annotált, többrétegű pilotkorpusz építése. In Gábor Berend, et al., editors, *XVI. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2020)*, pages 141–154, Szeged. Szegedi Tudományegyetem, TTIK, Informatikai Intézet.

Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., and Csirik, J. (2010). Hungarian Dependency Treebank. In *Proceedings of LREC 2010*, Valletta, Malta, May. ELRA.