Crawling in Reverse

Lightweight Targeted Crawling of News Portals

Balázs Indig[1,2,*], Tibor Kákonyi[3], Attila Novák[3,4]

[1]Eötvös Loránd University, Centre for Digital Humanities
[2]Hungarian Academy of Sciences, Research Institute for Linguistics
[3]MTA-PPKE Hungarian Language Technology Research Group
[4]Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
*Corresponding author

{indig.balazs@btk.elte.hu,kakonyitibor@gmail.com,novak.attila@itk.ppke.hu}

Based on the identically titled paper [1] presented at the 9th Language & Technology Conference Poznań, Poland in 2019
The improvements that have been made since then are Balázs Indig's work (pending publication)

Table of Contents

1. Motivation
2. The Classic NLP Workflow Including Crawling
3. Let’s Put Crawling in Reverse!
4. Testing the Idea
5. Conclusion

Motivation

Preserving and Using Textual Data

- The classical sources of text are *National Archives*
    - Processing them involves a **lot of manual work** (scanning and OCR)
    - Nowadays, the OCR is done by neural networks very efficiently
    - However, these sources are mostly **not open-access** and their **growth is slow and limited**
- With Web 2.0, a lot of texts are **born-digital**
    - Born-digital materials also need to be preserved
    - They are **more endangered than physically existing materials**
    - Far easier to collect, store and process them (eg. *Common Crawl*, *Internet Archive*)
    - Upcoming EU law allows archiving and using archives for scientific purposes

The Age of Crawlers

What does the **Boss** say?

- The usual **Natural Language Processing (NLP)** workflow: ‘`Get` **`SOME`** `text to work with! The individual content` **`does NOT`** `matter.`’
- The usual **digital humanist** workflow: ‘`Get` **`THAT SPECIFIC`** `text to work with! The individual content` **`does REALLY`** `matter.`’

The Classic NLP Workflow Including Crawling

Crawling for NLP: the Traditional Way

1. Start a webspider to crawl the web, starting from an initial seed (optionally with additional rules)
2. Use some boilerplate removal logic (*heuristics/rule-based*)
3. **Deduplication**
4. Run the NLP pipeline (split to sentences, tokenize, POS-tag, etc.)
5. **Store the corpus**

6. Use the text
7. Discover and fix errors in the pipeline
8. Go to step 1 and start with FRESH/OTHER text

Crawling for NLP: the Traditional Way

Let’s Put Crawling in Reverse!

Crawling for NLP: the Digital Humanist Way

1. Carefully select portals to crawl
2. Study the portal to extract its essential properties

3. Start a webspider to crawl the portal with the gained information (**virtually without duplication**)
4. Store the resulting HTML pages – **these are the primary sources**
5. Use boilerplate removal rules *tailored to the portal*
6. Run the NLP pipeline (split to sentences, tokenize, POS-tag, etc.)

7. Store the corpus elsewhere – **it is automatically reproducible**
8. Use the text
9. Discover and fix errors in the pipeline
10. Go to step 5 and start with **THE VERY SAME** text

The Main Idea

‘If an ARTICLE does not appear in THE (PORTAL’S) ARCHIVE, it does not exist!’ (adapted from Star Wars)

In Technical Terms

The **Two-level Crawling** and **portal-based boilerplate removal**:

- Most (news) portals use **permalinks** to identify articles and use an **article archive** to make the articles searchable
  - The article archive has simple structure and can be crawled easily for the permalinks (**dilemma**: rules or machine-learning?)
- We must only crawl the gathered permalinks
  - Virtually no duplication or junk!
  - **Less noise, reduced load, faster process**
- A specific portal has its unique layout which is the same or very similar for every article
  - Simple, efficient rules to remove boilerplate or targeted machine-learning (**dilemma** again)

In Technical Terms (cont.)

The details:

- We use a subset of **the ISO standard WARC archive format** for the crawled webpages **(request, response record pairs)** and *reuse them as cache* when needed
    - **Everything is reproducible** in the pipeline from here on (We only need to have the archive and know the exact versions of the programs used)
- We tailored the crawling and the boilerplate removal to the selected portals
    - As layout changes are infrequent, **it can collect new materials on a daily basis**
    - In an *easy-to-adjust framework*
- We can supervise and adjust the rules and add new portals if needed

Testing the Idea

The Task and the Resources

The Task:

- From five (structurally) quite different Hungarian news portals
- Extract text with metadata: *Author*, *Publication date*, *Title*, *Lead*, *Specified keywords*, *Text*
- Be **precise and sustainable**, runtime is secondary
- Reuse existing tools when possible!

The Resources:

- One low-end office machine (4 GB RAM, Intel i3 with 4 cores)
- 100 Mb/s uplink

Programs Compared, Problems Found

Crawlers:

- The existing crawlers were too different to compare
- However, we compared one portal with the crawl made by the National Széchényi Library
  - The result was about 1,000 vs. 52,850 articles for our method

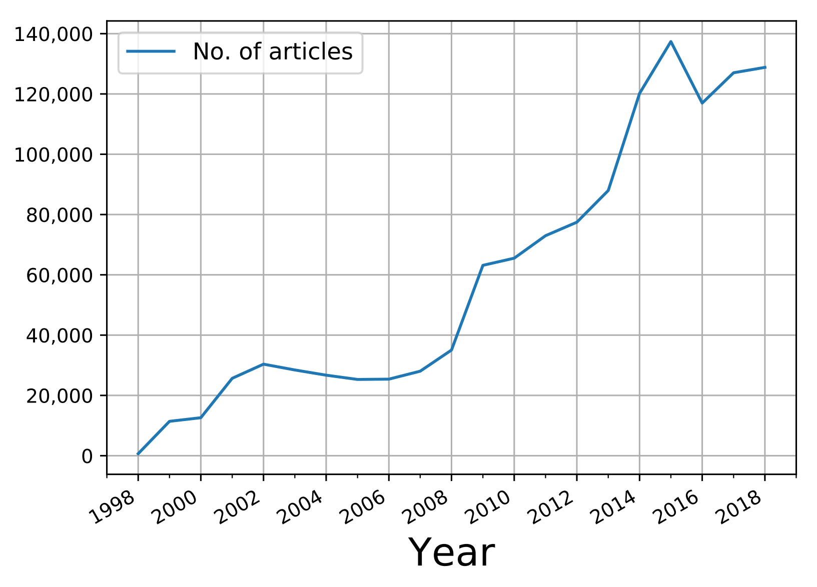
Boilerplate removal tools (JusText [3], Newspaper3k [2], our rules):

- **All methods are rule-based** and hard to compare
- Our method is specialised in the examined portals
- The two other methods are general and **built as a monoliths**
- Most existing tools can not (properly) extract metadata
- Existing tools have limited support for the Hungarian typography

Results

- **Regular Expressions** < existing programs < hand-crafted rules that **meet our requirements**
    - Now we use **HTML parsers** instead of REs (hard to automatise)
    - On the portals' article archives it was a great success!
- Numbers are growing, but new problems come to surface
- The first comparison with other archiving techniques is very promising, but there are more to come
- We clearly need more portals, more comparisons, more time to standardise the workflow

The annual distribution of 1,247,082 Articles (5 News Portals)



Conclusion

Conclusion

- In **10 days** with a low-end computer (due to rate limiting)
- Less than 100 GB space required (no garbage, just HTMLs)
- About **half billion** tokens estimated and growing
- Sustainable, **low load on both sides**
- Reproducible, improvable, extendable
- **Groundbreaking** work for later studies
    - Topic modeling, Stylometry analysis (with the available metadata)
    - Temporal (socio-)linguistic analysis (with the publication time)
    - Future machine-learning-based improvement of the workflow
    - Extending the set of targeted portals
- Future work:
    - Standardised workflow and TEI output
    - More comparisons in every possible way
    - A semantic searching service on the crawled material

References i

B. Indig, T. Kákonyi, and A. Novák.
Crawling in reverse – lightweight targeted crawling of news portals.
In M. 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 2019. Wydawnictwo Nauka i Innowacje.

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

References ii

J. Pomikálek.
Removing boilerplate and duplicate content from web corpora.
PhD thesis, Masaryk university, Faculty of informatics, Brno, Czech Republic, 2011.