05 – Building Language Resources from the Web

IA161 Advanced Techniques of Natural Language Processing

Vít Suchomel

NLP Centre, FI MU, Brno

November 9, 2015

Outline

Web as a Language Resource – Web as Corpus

Most language phenomena follow the Zipfian distribution.
$\Rightarrow$ The more data the better.

Example: Modificators of phrase ,,deliver speech“ (frequency):

- BNC (96 M words): major (8), keynote (6).
- ukWaC (1,32 G words): keynote (125), opening (12), budget (8), wedding (7).
- enTenTen12 (11,2 G words): keynote (813), acceptance (129), major (127), wedding (118), short (101), opening (97), famous (80).
- enClueWeb09 (70,5 G words): keynote (3802), acceptance (1035), opening (589), famous (555), commencement (356), impassioned (335), inaugural (333).

Size is not everything . . .

A significant fraction of all web pages are of poor utility.[1]

Why are qualitative aspects so important?

- Web is the most used data source to obtain enough source texts – 'Web as Corpus'.
- Web is garbage (by definition).
- Garbage as corpus?
- Building corpora from web requires extensive post-processing.

[1] [Manning et al., 2008, Chapter 20]

Issues of Building Language Resources from the Web

Particular tasks:

- Language identification,
- Character encoding detection,
- Efficient web crawling,
- Boilerplate removal,
- De-duplication (removal of identical or nearly identical texts),
- Fighting web spam,
- Authorship recognition & plagiarism detection,
- Storing & indexing of large text collections.

NLPC & Lexical Computing corpus tools: `http://corpus.tools/`

Crawling web corpora in NLPC FI MU

- Prepare language dependent models used in the following steps – training data: Wikipedia or previous version of the corpus.
- Start the crawler (`SpiderLing`).
- Processing and evaluation by the crawler on the fly:
  - encoding detection (`Chared`),
  - language filtering (character trigram model),
  - boilerplate removal (`Justext`),
  - de-duplication of identical documents,
  - evaluation of the yield rate of downloaded web domains.
- Post-processing:
  - de-duplication of near duplicate paragraphs (`Onion`),
  - tokenisation (`Unitok` or a morphological analyser in case of East Asian languages),
  - annotation: morphological/syntactical/semantical level – external tools,
  - encoding and indexing by a corpus manager (`Manatee`/`Bonito`).

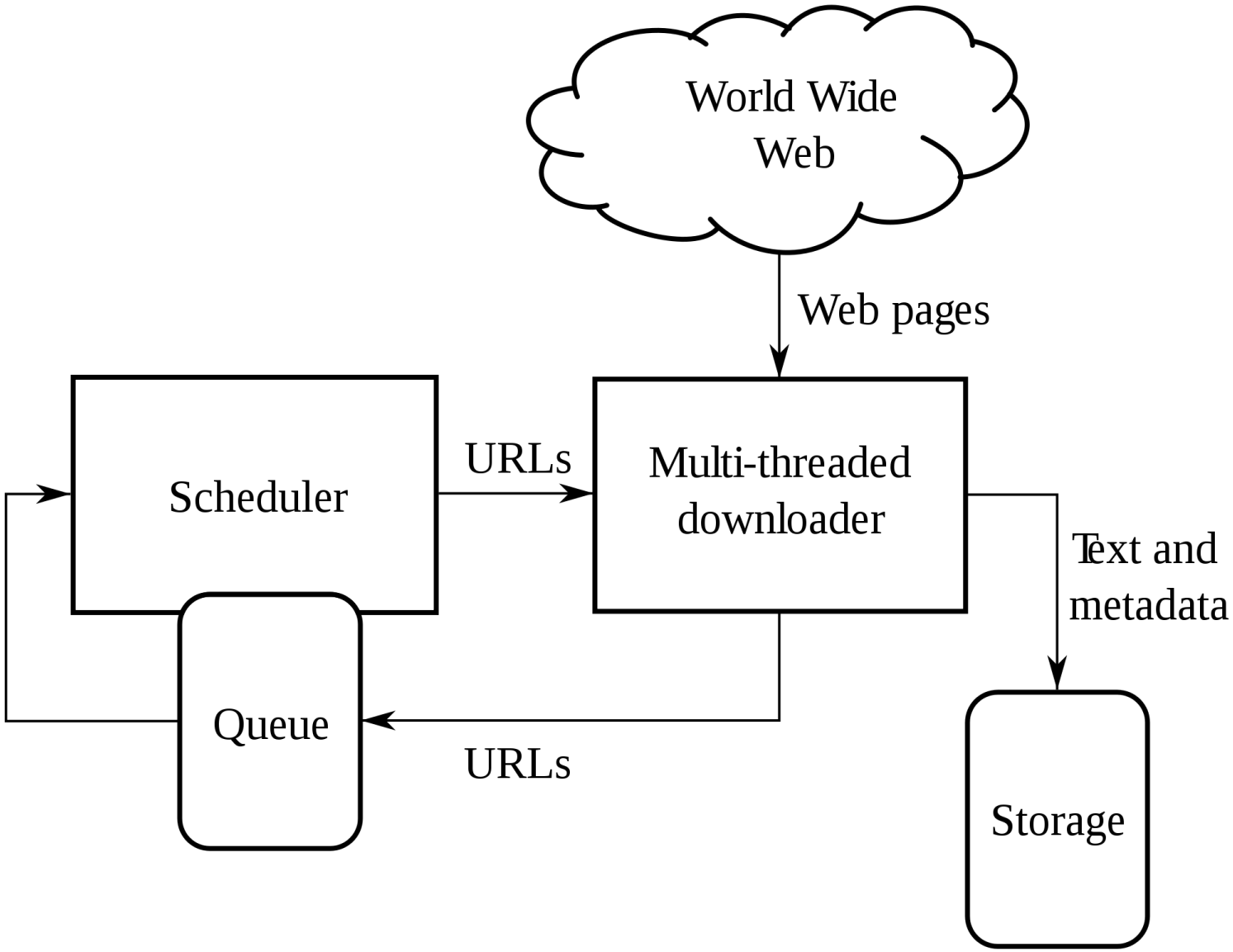
Web crawler

- Traverses the internet (graph of pages and links).
- Downloads documents (content & meta information).
- Stores documents (or their parts) in various formats for further use.
- Crawlers for various purposes:
  - GoogleBot – web indexing,
  - Linkcrawler – links, borken links checking,
  - Heritrix – general crawler, (Java, multiple treads),
  - SpiderLing – text corpora, (Python, multiple sockets).

Features a crawler should provide [Manning et al., 2008, Chapter 20]

- **Distributed:** Executable in a distributed fashion across multiple machines.
- **Scalable:** Scaling up the crawl rate by adding extra machines and bandwidth.
- **Performance and efficiency:** Efficient use of system resources (processor, memory, storage and network bandwidth).
- **Quality:** Biased towards fetching "useful" pages first.
- **Freshness:** Operate in continuous mode: obtain fresh copies of previously fetched pages, i.e. with a frequency that approximates the rate of change of that page. *Search engine crawler* $\rightarrow$ *the index contains a fairly current representation of each indexed web page.*
- **Extensible:** Cope with new data formats, new fetch protocols, various data processing needs. Modular architecture.

Basic crawler design

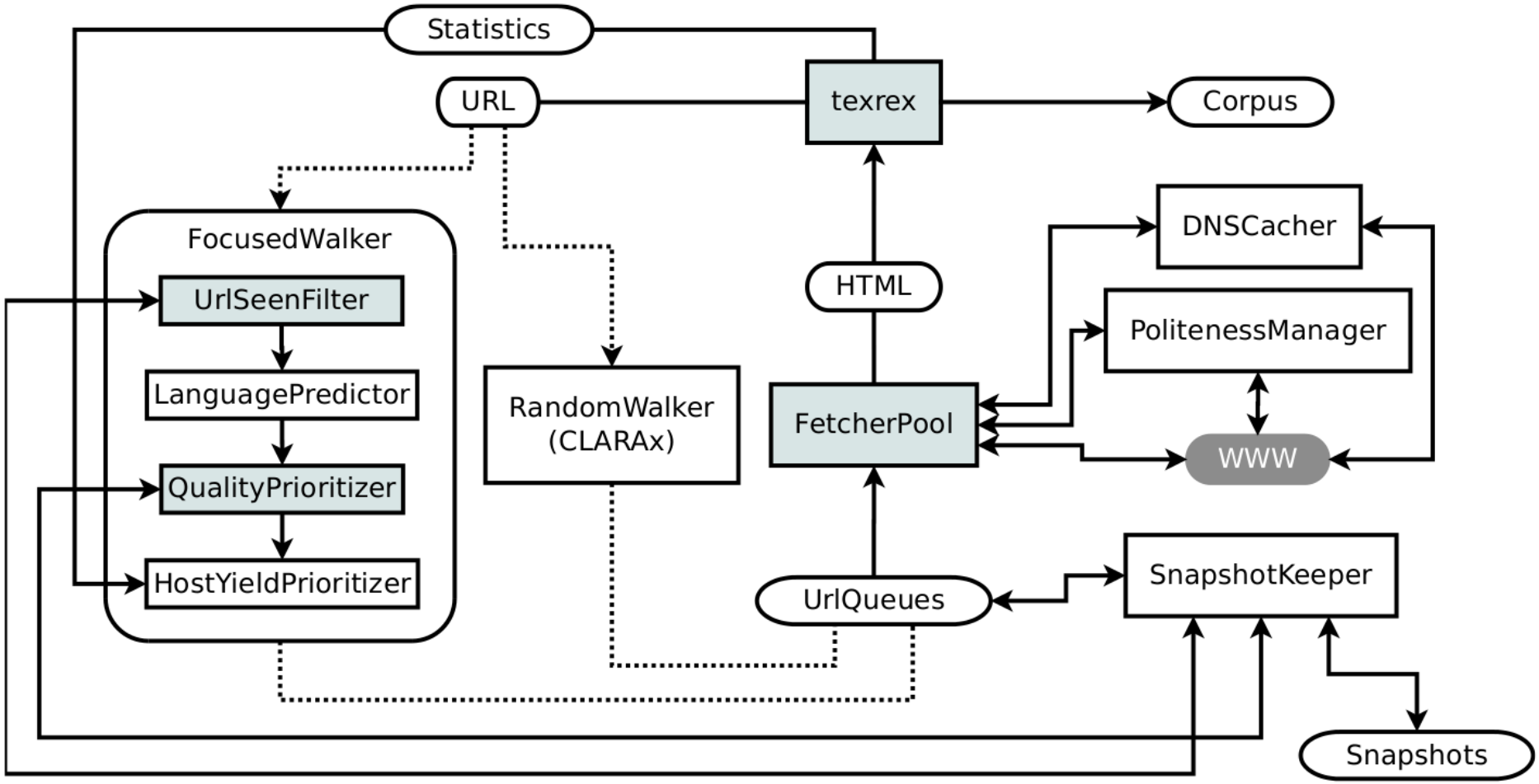


Source: http://en.wikipedia.org/wiki/Web_crawler

Advanced crawler implementation details

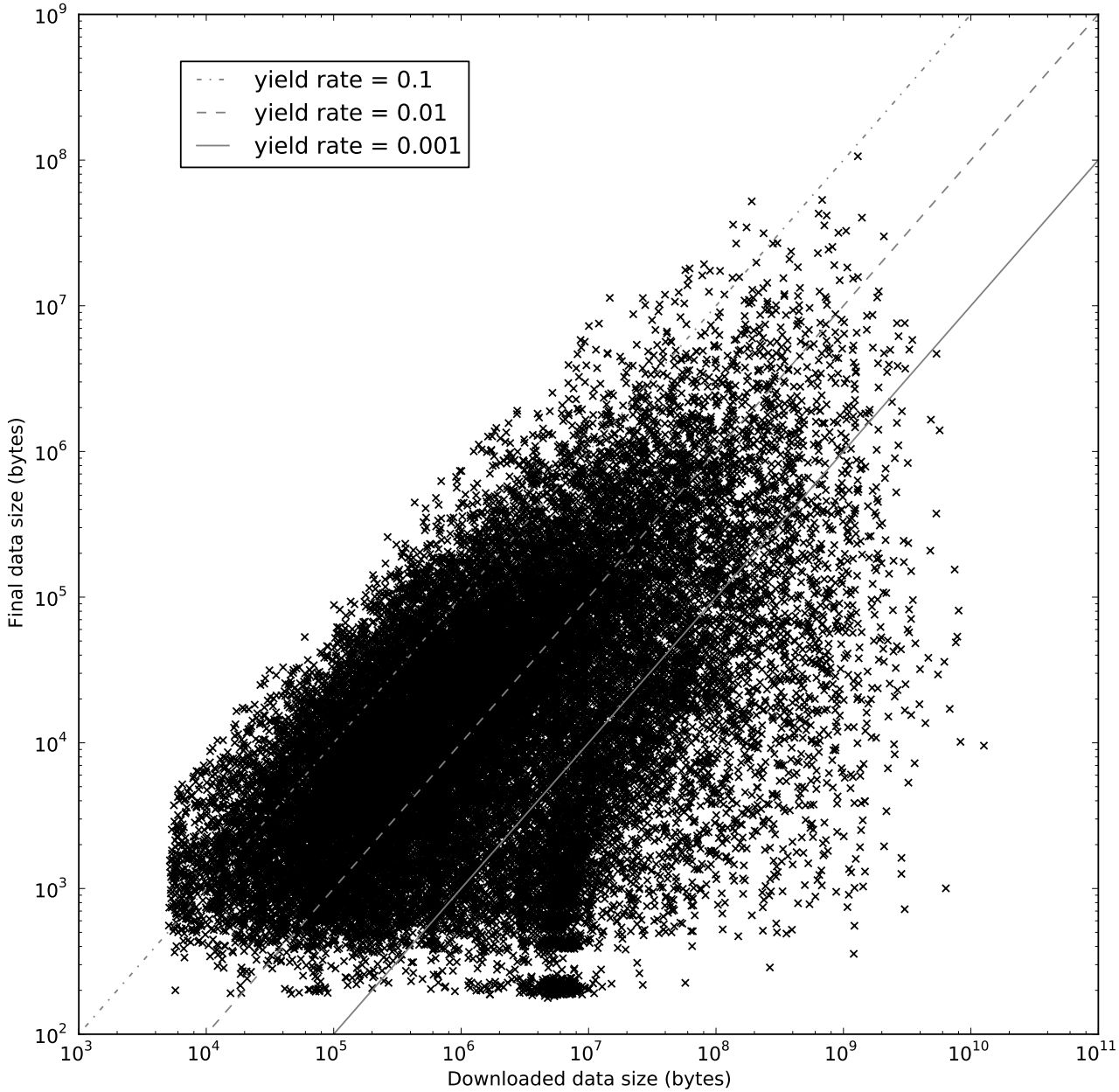
- Distributed vs. extensible.
- Multi-threaded vs. multi-socketed.
- Web traversal policy:
  - depth vs. breadth,
  - domain selection,
  - domain distance,
  - focused crawling (topic oriented) vs. general crawling,
  - yield ratio.

Focused crawler design

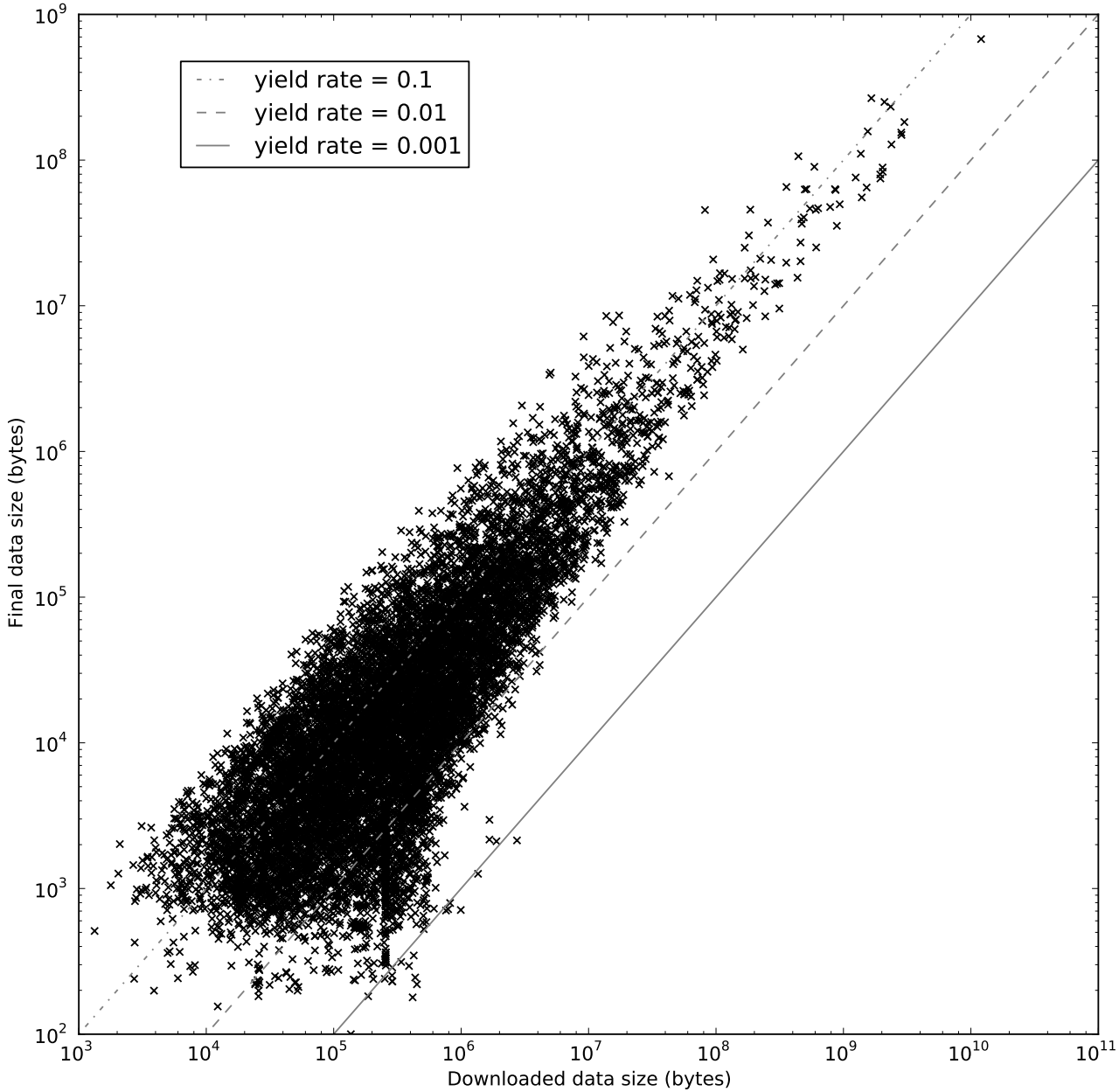


Source: Roland Schafer, Adrien Barbaresi, Felix Bildhauer. Focused Web Corpus Crawling. 9th Web as Corpus Workshop, 2014.

General unfocused crawling efficiency (Heritrix)



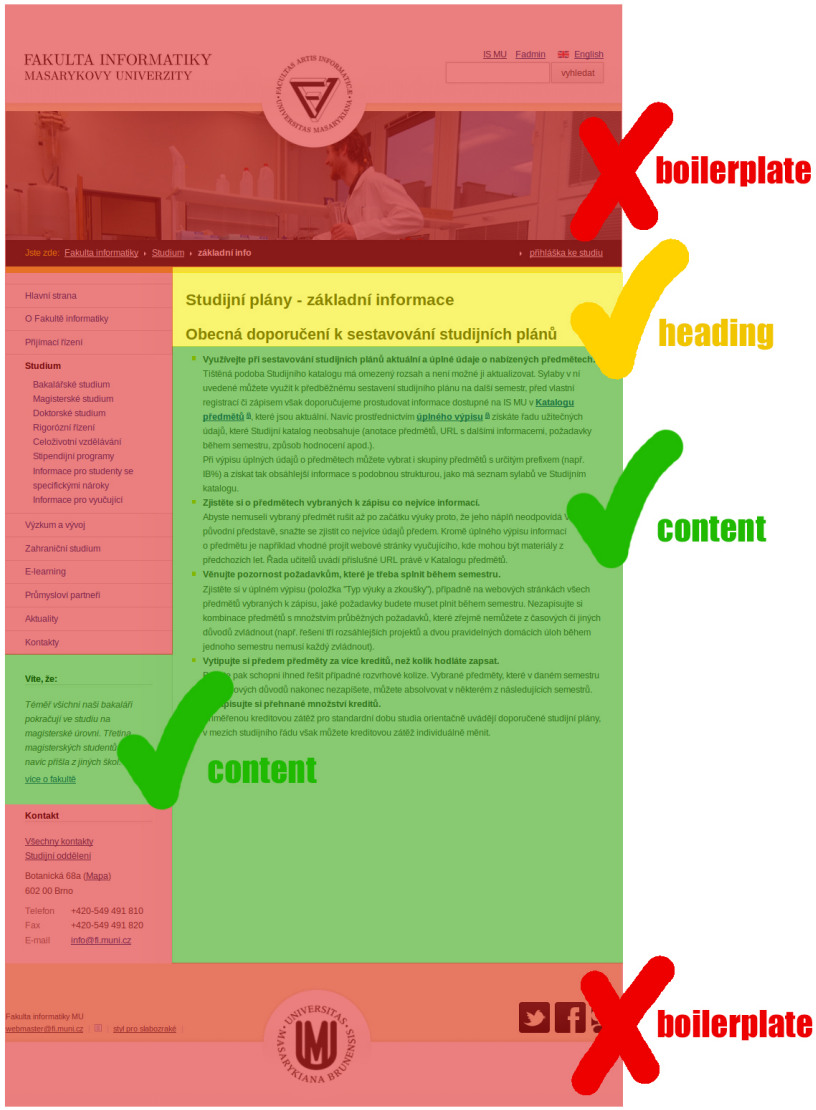
Domain yield ratio optimised efficiency (SpiderLing)



What is boilerplate

- Repeated parts of a web page (not containing a new text) – header, footer, navigation.
- Uninteresting text (too short or not conntinuous) – advertisement, lists of items, article previews.
- Hard to recognise: discussions.

What is boilerplate



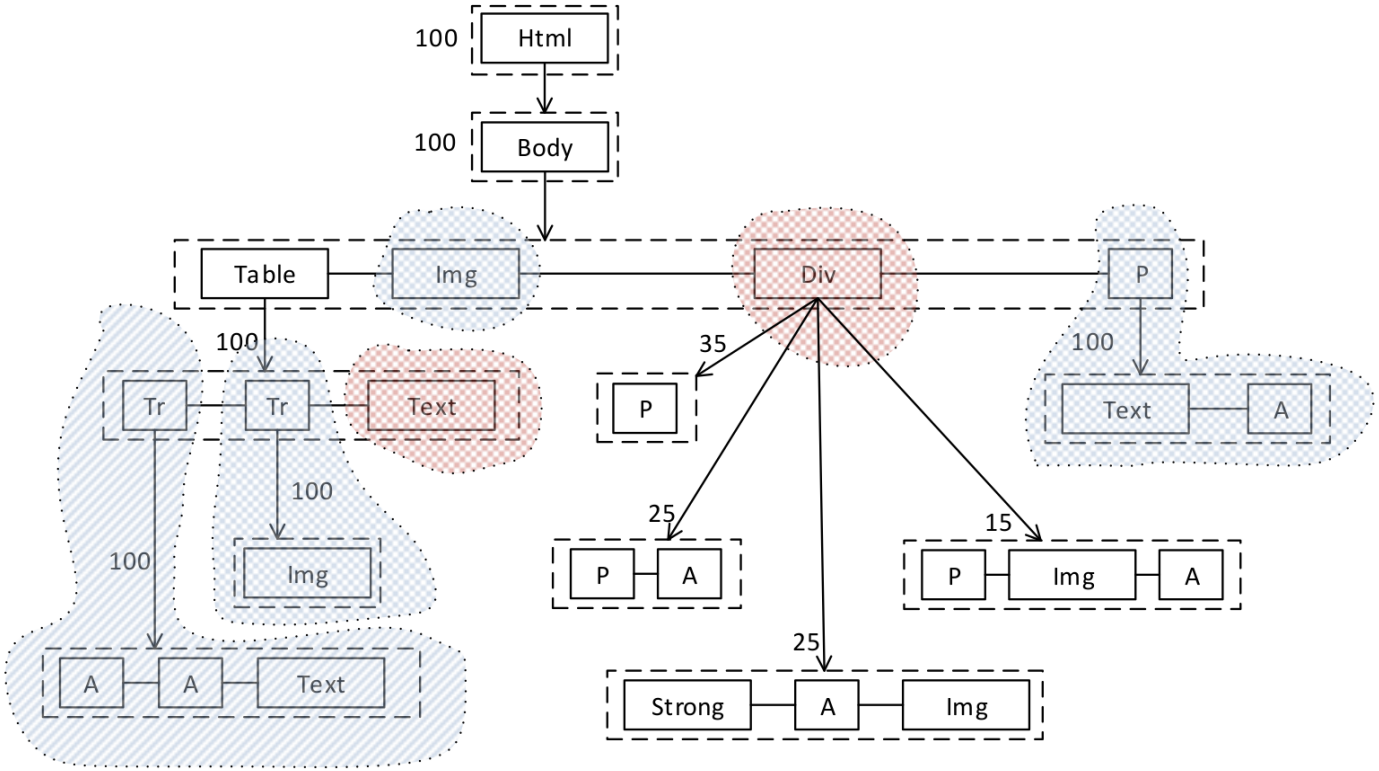
Source: http://corpus.tools/attachment/wiki/Justext/Algorithm/cs_classification_example.png

Boilerplate removal approaches

- Machine learning (SVM, CRF, neural networks, n-gram models):
  - Annotated web pages required for training.
  - Victor (CRF),
  - Ncleaner (n-grams).
- Heuristics:
  - Rules for including/excluding sections of text.
  - BTE (tag density),
  - Boilerpipe (link/text ratio),
  - jusText (link/text ratio, frequent words, context sensitive – smoothing).

Site Style Tree [Yi et al., 2003]

- Represents both layout and content of a web page.
- Node importnace = node entropy over the whole Site Style Tree.



Source: Ján Švec: Inteligentní detekování struktury webu, p. 32. Online: http://is.muni.cz/th/420072/fi_m/.

jusText

Context sensitive paragraph classification:

Demo: `http://nlp.fi.muni.cz/projects/justext/`

De-duplication

- Quite straightforward for full duplicates.
- What about similar documents?
- People copy just parts of the document: original vs. copy
- Or copy and modify: original vs. modified
- Or copy and extend: original vs. extended

N-gram shingling algorithm [Manning et al., 2008, Chapter 19]

- 'Shingles' of length of n words.
- N-grams represented by hashes.
- Effective implementation: random permutations of hashes.

onion – One INstance ONly[2]

Algorithm inspired by Broder's shingling algorithm:

- Make n-grams of words for every structure,
- every n-gram is represented by its hash,
- the current structure is a duplicate $\Leftrightarrow$ at least p % of n-gram hashes is duplicate (has been observed before).
- Default options: structure = paragraph, n = 7, p = 50, smoothing.

[2] Pomikálek, Jan. Removing boilerplate and duplicate content from web corpora. PhD thesis, Masaryk university, 2011.

Main and related tasks in plagiarism detection

- **Plagiarism detection:** Given a document, identify all plagiarized sources and boundaries of re-used passages.
- **Author identification:** Given a document, identify its author.
- **Author profiling:** Given a document, extract information about the author (e.g. gender, age).

Stamatatos et al. Overview of the pan/clef 2015 evaluation lab. In Experimental IR Meets Multilinguality, Multimodality, and Interaction, pages 518–538. Springer. 2015.

External vs. Intrinsic plagiarism detection [Potthast et al., 2010]

External plagiarism detection

Given a set of suspicious documents and a set of source documents the task is to find all text passages in the suspicious documents which have been plagiarized and the corresponding text passages in the source documents.

Intrinsic plagiarism detection

Given a set of suspicious documents the task is to identify all plagiarized text passages, e.g., by detecting writing style breaches. The comparison of a suspicious document with other documents is not allowed in this task.

Plagiarism techniques [Potthast et al., 2015]

- Manual paraphrasing = human retelling. Similar to copywriting.
- Random text operations. Random shuffling, insertion, replacement, or removal of characters, phrases or sentences. Replacement of characters with look-alike UTF characters.
- Semantic word variation. Random replacement words with synonyms, antonyms, hyponyms, or hypernyms.
- Part-of-speech-preserving word shuffling. Shuffling of phrases while maintaining the original POS sequence.
- Improvement of previous synthetic techniques: Insertions, replacements and variations may be obtained from context documents.
- Machine translation, cyclic translation. Automatic translation of a text passage from one language via a sequence of other languages to the original language.
- Summarization. Summaries of long text passages.
- Improvement of machine translation and summarization techniques: Manually corrected output.

Basic techniques for revealing similar documents[3]

Bag of words

Full fingerprint methods

Overlapping substrings of length k in words from the beginning of the document.

Selective Fingerprint methods

Non-overlapping substrings of length k in words from the beginning of the document.

Rarest-in-document

All substrings are sorted according to their document frequency, then the rarest are selected as representants of the document.

Selected Anchors

The document is reduced to pre-selected short chunks of characters.

Symmetric Similarity measure

$SS(X, Y) = \frac{|d(X) \cap d(Y)|}{|d(X) \cup d(Y)|}$ where $d(X)$ is a set of fingerprints of X.

[3] According to HaCohen-Kerner at al. Detection of simple plagiarism in computer science papers. In Proceedings of the 23rd International Conference on Computational Linguistics, pp. 421-429. Association for Computational Linguistics, 2010.

Task: Plagiators vs. plagiarism detectors

1. Create 5 documents (with a similar topic) and 5 plagiarisms of these documents, 10 documents total. 100 words $\leq$ document lenght $\leq$ 500 words. 20 % $\leq$ plagiarism content $\leq$ 75 % (100 % if done well).
2. Select detection algorithm and implement it in Python. At least 1 own plagiarism must be detected, at least 1 must be not detected by your own script.
3. Input format: POS tagged vertical consisting of 10 sctructures `doc` with attributes `author`, `id`, `class`, `source`. Pair author, id is unique. Class is "original" or "plagiarism". Source is the id of the source (in case of plagiarism) or own id (in case of original).[4]
   - Czech: `alba:/opt/majka/majka-desamb-czech.sh | cut -f1-3`.
   - English: `alba:/opt/TreeTagger/tools/tt-english_v2.sh | awk 'print $1"\t"$3"\t"$2'`.
4. Output format: One plagiarism per line: id `TAB` detected source id `TAB` real source id. Evaluation line: precision, recall F1 measure.
5. Your script will be evaluated using data made by others.

[4] For the sake of simplicity: A plagiarism cannot have more sources here.

Task: Input data example

```
<doc author="Já První" id="1" class="original" source="1">
<s>
Dnes     dnes     k6eAd1
je       být      k5eAaImIp3nS
pěkný    pěkný    k2eAgInSc4d1    pěkný
den      den      k1gInSc4        den
<g/>
!        !        k?
</s>
</doc>
<doc author="Já První" id="2" class="plagiarism" source="1">
<s>
Dnes     dnes     k6eAd1
je       být      k5eAaImIp3nS
ale      ale      k9
pěkný    pěkný    k2eAgInSc4d1    pěkný
den      den      k1gInSc4        den
<g/>
!        !        k?
</s>
</doc>
```

Task: Output example

```
2 1 1
1.00 1.00 1.00
```

References I


Manning, C. D., Raghavan, P., Schütze, H., et al. (2008).
Introduction to information retrieval, volume 1.
Cambridge university press.


Potthast, M., Hagen, M., Göring, S., Rosso, P., and Stein, B. (2015).
Towards data submissions for shared tasks: first experiences for the task of text alignment.
Working Notes Papers of the CLEF, pages 1613–0073.


Potthast, M., Stein, B., Barrón-Cedeño, A., and Rosso, P. (2010).
An Evaluation Framework for Plagiarism Detection.
In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China. Association for Computational Linguistics.

References II


Yi, L., Liu, B., and Li, X. (2003).
Eliminating noisy information in web pages for data mining.
In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305. ACM.