08 – Building Language Resources from the Web
IA161 Advanced Techniques of Natural Language Processing

Vít Suchomel

NLP Centre, FI MU, Brno

November 6, 2019
1. Introduction: Web as a Language Resource

2. Efficient Web Crawling

3. Boilerplate Removal

4. De-duplication

5. Plagiarism Detection

6. Task: Plagiarism Detection
Web as a Language Resource – Web as Corpus

Most language phenomena follow the Zipfian distribution.
⇒ The more data the better.

Example: Modifiers 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).
- enTenTen15 (15,7 G words): keynote (3673), opening (684), welcome (413), key (257), major (255), acceptance (233), powerful (229), commencement (226), inspiring (210), inaugural (146).
- 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).
Outline

1. Introduction: Web as a Language Resource
2. Efficient Web Crawling
3. Boilerplate Removal
4. De-duplication
5. Plagiarism Detection
6. Task: Plagiarism Detection
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, broken links checking,
  - Heritrix – general crawler, (Java, multiple threads),
  - 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 → 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

## General unfocused crawling efficiency (Heritrix)

<table>
<thead>
<tr>
<th>Downloaded data size (bytes)</th>
<th>Final data size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yield rate = 0.1</td>
</tr>
<tr>
<td></td>
<td>yield rate = 0.01</td>
</tr>
<tr>
<td></td>
<td>yield rate = 0.001</td>
</tr>
</tbody>
</table>

![Scatter plot showing the relationship between downloaded and final data sizes for different yield rates.](image)
Domain yield ratio optimised efficiency (SpiderLing)
Outline

1. Introduction: Web as a Language Resource
2. Efficient Web Crawling
3. Boilerplate Removal
4. De-duplication
5. Plagiarism Detection
6. Task: Plagiarism Detection
What is boilerplate

- Repeated parts of a web page (not containing a new text) – header, footer, navigation.
- Uninteresting text (too short or not continuous) – 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 importance = node entropy over the whole Site Style Tree.

Context sensitive paragraph classification:

Outline

1. Introduction: Web as a Language Resource
2. Efficient Web Crawling
3. Boilerplate Removal
4. De-duplication
5. Plagiarism Detection
6. Task: Plagiarism Detection
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.
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 $\iff$ at least $p\%$ of n-gram hashes is duplicate (has been observed before).
- Default options: structure = paragraph, $n = 7$, $p = 50$, smoothing.

---

Outline

1. Introduction: Web as a Language Resource
2. Efficient Web Crawling
3. Boilerplate Removal
4. De-duplication
5. Plagiarism Detection
6. Task: Plagiarism Detection
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).

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]

- 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

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 representatives 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 \( X \).

Outline

1. Introduction: Web as a Language Resource
2. Efficient Web Crawling
3. Boilerplate Removal
4. De-duplication
5. Plagiarism Detection
6. Task: Plagiarism Detection
Task: Plagiarists vs. plagiarism detectors

Either:
Create 5 documents (with a similar topic) and 5 plagiarisms of these documents, 10 documents total.  

- 100 words ≤ document length ≤ 500 words
- 20% ≤ plagiarism content ≤ 90%
- POS tagged text:
  - Czech: asteria04:/opt/majka_pipe/majka-czech_v2.sh | cut -f1-3.
  - English: asteria04:/opt/treetagger_pipe/tt-english_v2.1.sh.

For each plagiarism:
1. describe plagiarism technique(s) used
2. which detection methods might be able to reveal it – give reasons
3. which detection methods might not be able to reveal it – give reasons

The minimal homework.

---

4 For the sake of simplicity: A plagiarism cannot have more sources here.
Task: Plagiarists vs. plagiarism detectors

Or:
Select a detection algorithm and implement it in Python.

- Input format: A POS tagged vertical consisting of structures 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).

- Output format: One plagiarism per line: id TAB detected source id TAB real source id. Evaluation line: precision, recall F1 measure.

- ./plagiarism_simple.py < training_data.vert

Your script will be evaluated using data made by others.

- Describe which plagiarism detection technique(s) were implemented.

The right homework if you want to learn something.

---

5 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

