Developing and Using Bitextor

Miquel Esplà-Gomis

SMART-Select Workshop 'Data Curation' in Luxembourg
November 20, 2018

About me

- Post-doc at Universitat d'Alacant
- Transducens research group: MT and language technologies
- Involved in Bitextor project for the last 10 years (since version 2.0)

1 Bitextor: an introduction

About Bitextor

- Tool to crawl parallel data from a website for any pair of languages:
  - the user provides a list of websites and a language pair
  - Bitextor returns a list of bi-segments in these websites
- Useful if we want to:
  - ... train MT or any other bilingual NLP application
  - ... build a domain-specific parallel corpus
  - ... build a translation memory of our site
  - ... etc.

About Bitextor

- Bitextor is free/open-source (GPLv3 license)
- Mostly implemented in Python and Bash
- Project running since 2003
- Available at `https://github.com/bitextor/bitextor`

A bit of history (I)

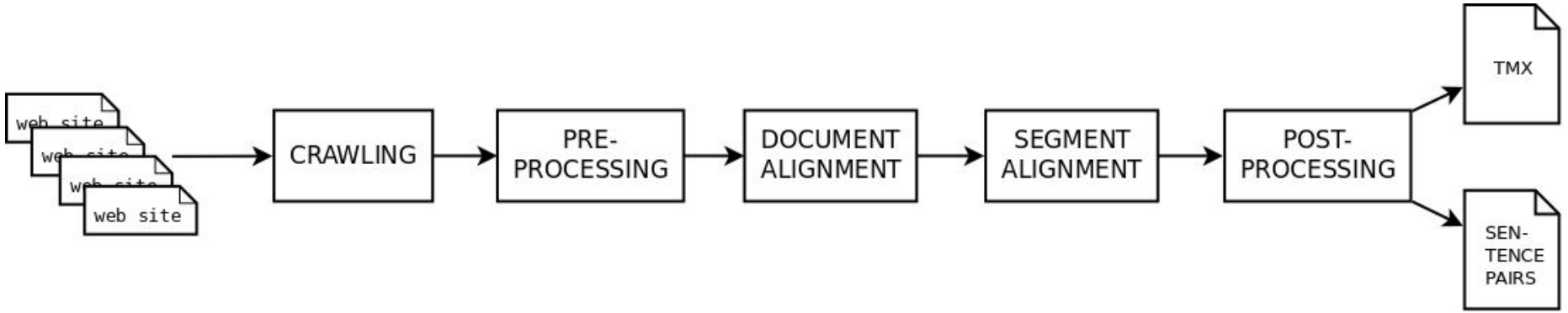
- Version **1.0** in 2003 at Universitat d'Alacant: C++ monolithic implementation
- Version **3.0** in 2010 by Prompsit Language Engineering: reimplementation as a Unix pipeline, modularized and easier to parallelize

A bit of history (II)

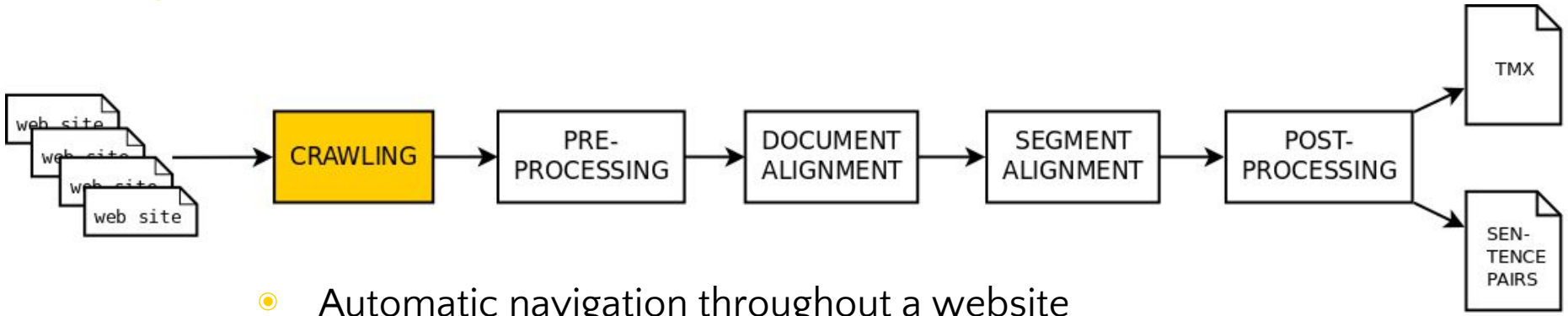
- Version **6.0** (current) by U. Alacant + U. Edinburgh + Prompsit Language Engineering: 1st Paracrawl code release
- Version **7.0** (March 2019):
  - highly scalable pipeline: making the most of any architecture
  - more robust, easy to recover from failures
  - python3 migration
  - adding new tools to the pipeline

2 Bitextor pipeline

Bitextor pipeline

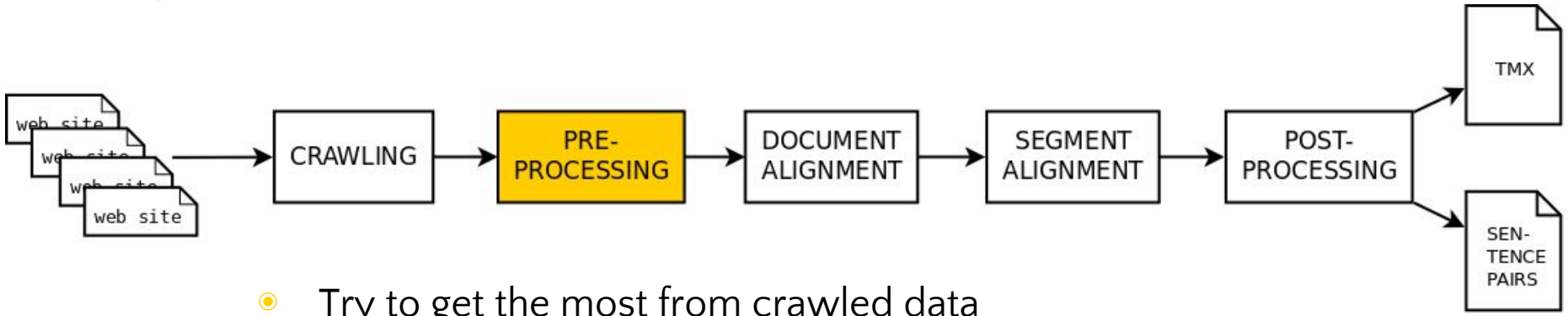


Bitextor pipeline: crawling



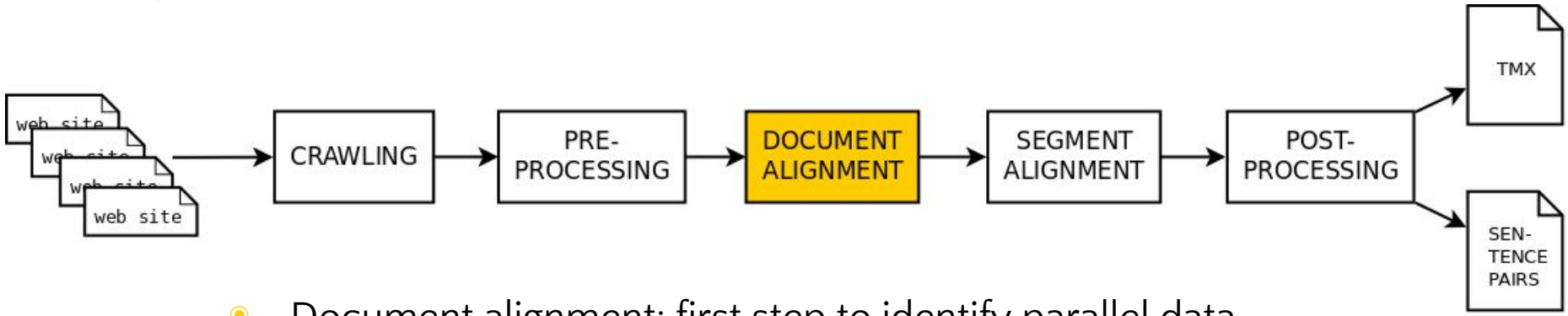
- Automatic navigation throughout a website
- Downloading all documents with useful information (text)
- Usual to limit crawling to time or size (or both)
- Re-crawling same sources allow to get new data
- In Bitextor: Creepy (python library) or HTTrack (external tool)

Bitextor pipeline: pre-processing



- Try to get the most from crawled data
- Pre-processing tries to fix what is broken and to extract data about the documents crawled. In Bitextor:
  - HTML normalisation and plain text extraction (html5lib + lxml)
  - fixing character encoding issues (ftfy)
  - boilerplate deletion (boilerpipe)
  - language identification (langid)

Bitextor pipeline: document alignment



- Document alignment: first step to identify parallel data
- Two strategies implemented:
  - Feature extraction + regressor to obtain a confidence score for each document pair
  - Using machine translation and computing TF/IDF (from Edinburgh's Malign parallel data crawler)

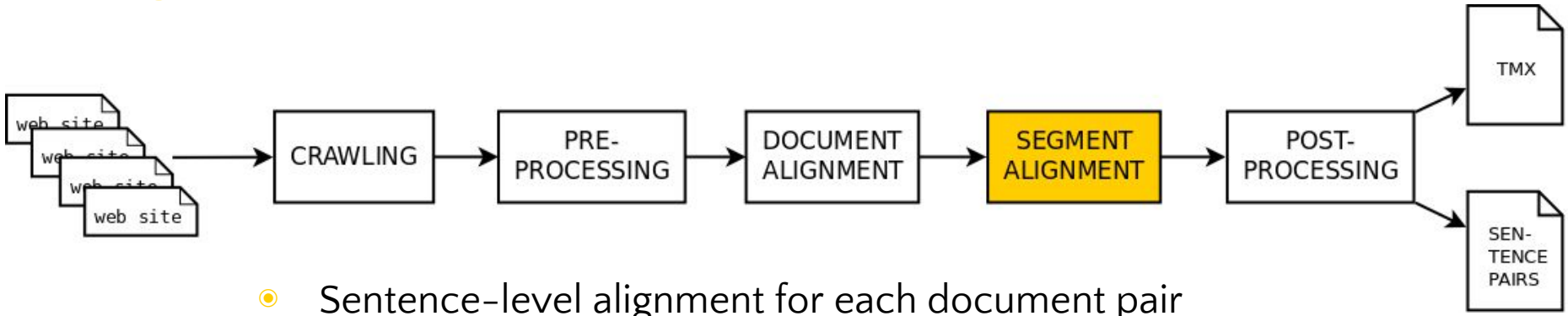
Document alignment: features + regressor

- Features extracted from each document pair:
  - Word overlapping metrics on content (using bilingual lexica)
  - URL similarity
  - Image co-occurrence
  - Mutually linked
  - etc.
- Regressor: needs training data
- Advantages: rather fast, can be run on CPU
- Disadvantages: requires training data, some features are only available on HTML files

Document alignment: MT + TF/IDF

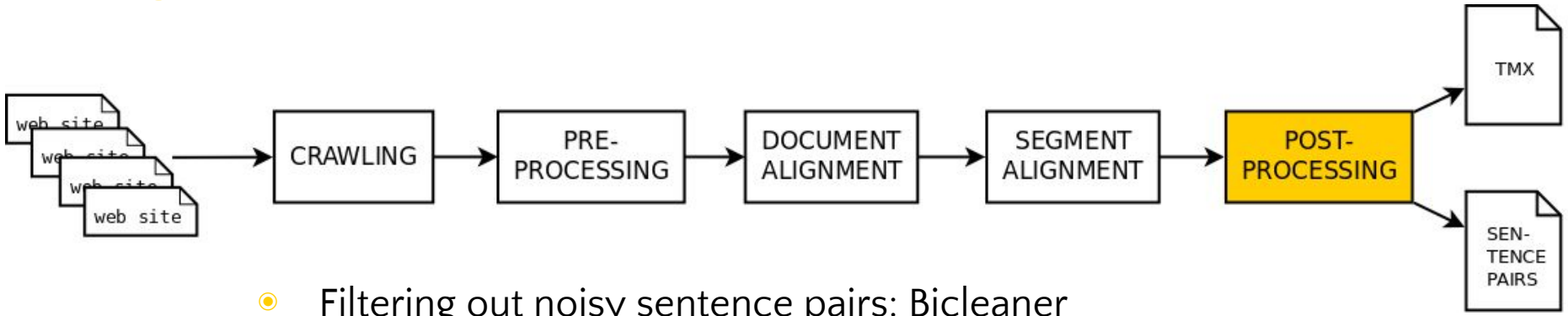
- Text in L1 is machine-translated into L2
- Similarity based on monolingual TF/IDF for each document pair:
  - TF/IDF identify "relevant" words by taking frequent words in document that are infrequent in corpus
- Advantages: no training data needed, works for any text files
- Disadvantages: an MT system is required, depending on the MT system performance can be more time costly

Bitextor pipeline segment alignment



- Sentence-level alignment for each document pair
- Two strategies supported:
  - Based on length and dictionaries (Hunalign)
  - Based on length and MT (Bleualign)
- Document pairs with too few alignments are discarded
- Segment pairs with low confidence score can be discarded

Bitextor pipeline: post-processing



- Filtering out noisy sentence pairs: Bicleaner
- Deduplication of repeated segments
- TMX generation
- In next version: MT extrinsic evaluation

3

Major challenges faced during Bitextor development

Major challenges faced in Bitextor

1. Extensive crawling and noisy data
2. Language independence
3. Evaluation of improvements
4. Legal issues with data obtained

Sources of parallel data and noise (I)

- Bitextor conceived to work on given URLs
- To crawl large amounts of data, sources (websites) need to be identified automatically
- Extensive crawling:
  - crawling entire TLD for some languages (for example .hr or .bs)
  - language statistics files from web corpus (Common Crawl)

Sources of parallel data and noise (II)

- Identification of candidate websites by:
  - Setting a threshold for the amount of bilingual data
  - Running Bitextor and crossing fingers
- Result: noisy data is obtained that needs to be cleaned automatically
- Bicleaner: new Bitextor tool that provides a confidence score per segment pair [https://github.com/bitextor/bicleaner]

Sources of parallel data and noise: Bicleaner

- 1st: Pre-processing with handcrafted straightforward rules
- 2nd: Random forest binary classifier using features:
  - Probabilistic dictionary in both directions
  - Sentence length
  - Average token length
  - Common punctuation and numeric expressions
  - Difference in the number of capitalized tokens

Language independence (I)

- Bitextor uses natural language processing techniques/resources
  - language identification
  - segmentation and tokenisation
  - content similarity techniques (MT + TF/IDF or dictionaries word overlapping metrics)
  - use of probabilistic bilingual dictionaries for filtering

Language independence (II)

- Issues related to language dependence:
  - lack of support of libraries for some languages
  - difficult to find/build some of the resources
  - some techniques difficult to apply for some language pairs

Language independence: considerations

- Some ideas to alleviate these problems:
  - modularity allows easier replacement of tools/libraries
  - use of existing NLP solutions:
    - automatic transliteration,
    - lemmatisation,
    - stemming

Evaluation of a parallel data crawler

- Objective: as much high-quality data as possible in the shortest time
- What should be evaluated?
  - Use of time and storage?
  - Quality of the data generated?
  - Amount of data obtained (recall)?

Quality of the data generated

- Quality depends on the objective of the corpus built
- Two possibilities:
  - intrinsic evaluation (task independent)
  - extrinsic evaluation (task specific)

Quality of the data generated: intrinsic

- How to assess parallel data quality?
  - Human evaluation:
    - straightforward, considered reliable but expensive
    - still depends on subjective opinion
  - Automatic evaluation (cheaper):
    - Translation quality estimation?
    - ELRC automatic quality indicators?

Quality of the data generated: extrinsic

- Train an MT system with our data and measure BLEU on a test set:
  - MT is (to some extent) tolerant to noise
  - More parallel data not always improve BLEU
  - BLEU score itself difficult to interpret

Quality of the data generated: recall

- How to know how much data was crawled from what is available?
  - Balance: higher quality usually implies lower recall
  - Different stages of the pipeline involved: crawling, document alignment, segment alignment, filtering.
  - Each stage can be measured independently, but global evaluation is difficult

Evaluation: summary

- Multifaceted open problem
- Conditioned by the specific problem:
  - What will be our corpus used for?
  - Languages crawled
  - Time constraints
  - Budget

Legal issues: licenses

- Parallel data crawled from the Internet can be:
  - Licensed under standard restrictive copyright
  - Licensed under permissive licenses
  - Not licensed: tacitly equivalent to the most restrictive copyright possible (Berne convention)

Legal issues: distributing data

- How to share crawled data?
  - Distributing without taking into account copyright issues
  - Distributing adding a notice and take-down disclaimer
  - Distributing data on demand
  - Not distributing data at all

Proposal: deferred crawling

- Distributing a *stand-off* annotation of the Web (not *what* the content is but *where* it is)
- Legally safe(r) distribution of parallel data
- Implementation requires two tools:
  - Deferred parallel data crawler
  - Reconstructor of parallel data (“re-crawler”)

Stand-off annotation

- For every segment pair, deferred TMX files contain:
  - URL of the original document
  - segments replaced by XPath annotation: `XPath:startOffset:endOffset`
  - Checksum for safety in re-crawling

Original:

```
<tmx version="1.4">
  <header
    creationtool="Bitextor"
    datatype="PlainText" segtype="sentence"
    adminlang="en-us" srclang="en"/>
  <body>
    <tu>
      <tuv xml:lang="ca">
        <seg>Associació Europea per a la Traducció Automàtica</seg>
      </tuv>
      <tuv xml:lang="en">
        <seg>European Association for Machine translation</seg>
      </tuv>

      ...

  </body>
</tmx>
```

Original:

```
    <tuv xml:lang="ca">
     <seg>Associació Europea per a la Traducció Automàtica</seg>
    </tuv>
```

Deferred:

```
    <tuv xml:lang="ca">
     <prop
type="source-document">http://www.dlsi.ua.es/~mespla/programari.html</prop>
     <prop type="deferred-seg">body/div/div[2]/p[5]/a:0-47</prop>
     <prop type="checksum-seg">73999000b9d5beef600a11a7e5c37703</prop>
     <seg></seg>
    </tuv>
```

Original:

```
<tuv xml:lang="ca">
 <seg>Associació Europea per a la Traducció Automàtica</seg>
</tuv>
```

Deferred:

```
<tuv xml:lang="ca">
 <prop type="source-document">http://www.dlsi.ua.es/~mespla/programari.html</prop>
 <prop type="deferred-seg">body/div/div[2]/p[5]/a:0-47</prop>
 <prop type="checksum-seg">73999000b9d5beef600a11a7e5c37703</prop>
 <seg></seg>
</tuv>
```

Original:

```
<tuv xml:lang="ca">
 <seg>Associació Europea per a la Traducció Automàtica</seg>
</tuv>
```

Deferred:

```
<tuv xml:lang="ca">
 <prop
type="source-document">http://www.dlsi.ua.es/~mespla/programari.html</prop>
 <prop type="deferred-seg">body/div/div[2]/p[5]/a:0-47</prop>
 <prop type="checksum-seg">73999000b9d5beef600a11a7e5c37703</prop>
 <seg></seg>
</tuv>
```

Original:

```
<tuv xml:lang="ca">
 <seg>Associació Europea per a la Traducció Automàtica</seg>
</tuv>
```

Deferred:

```
<tuv xml:lang="ca">
 <prop
type="source-document">http://www.dlsi.ua.es/~mespla/programari.html</prop>
 <prop type="deferred-seg">body/div/div[2]/p[5]/a:0-47</prop>
 <prop type="checksum-seg">73999000b9d5beef600a11a7e5c37703</prop>
 <seg></seg>
</tuv>
```

Original:

```
<tuv xml:lang="ca">
 <seg>Associació Europea per a la Traducció Automàtica</seg>
</tuv>
```

Deferred:

```
<tuv xml:lang="ca">
 <prop
type="source-document">http://www.dlsi.ua.es/~mespla/programari.html</prop>
 <prop type="deferred-seg">body/div/div[2]/p[5]/a:0-47</prop>
 <prop type="checksum-seg">73999000b9d5beef600a11a7e5c37703</prop>
 <seg></seg>
</tuv>
```

4 Concluding remarks

Concluding remarks

- Bitextor is a free/open-source tool to build parallel data from multilingual websites
- Multiple tools and strategies implemented
- Release of version 7.0 coming soon with new features and better user experience

Concluding remarks

- What we have learned working on Bitextor:
  - Try to keep the tool as modular as possible
  - Do not forget that other languages exist (even though we are working on specific ones)
  - Filtering is crucial and allow to crawl more data
  - Carefully choose your evaluation method
  - Try to choose a legally-safe strategy to share or corpora

Thanks!

Any questions ?

- You can contact me at: mespla@dlsi.ua.es
- Find Bitextor at:

  `https://github.com/bitextor/bitextor/`