Getting our hands dirty with the mwetoolkit

Carlos Ramisch & Silvio Cordeiro

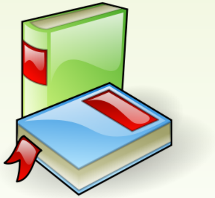
Valletta, Malta - 4th PARSEME meeting - March 19-20, 2015

Outline

1. Warming up
2. The mwetoolkit

Lexical resources

- Essential to any NLP application
- Contain information about the ***lexical units***
- Structured data, more than a list of words
- Dictionaries, terminologies, thesauri, ontologies



How does one build a lexicon? I

The standard approach

- years of work
- dozens of highly trained professionals
- thousands of dollars
- for humans or for machines (or for both)?
- high quality result

How does one build a lexicon? II

The “lazy” approach

- automatically learn lexical information from texts
- language independent
- cheap and dirty
- requires large amounts of text and high computational power
- contains noise and silence

The goal of this tutorial

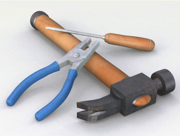
To present a tool that automatically discovers relevant MWEs in corpora, which can in turn help lexicon construction

Automatic MWE lexicon construction

- Idea: capture regularities in word combinations
- Combinations: contiguous or not contiguous
- Number of words (> 2)
- Use of POS and syntax patterns for ***candidate extraction***
- Use of association measures and learning for ***candidate filtering***
- Support and speed up manual lexicographic work

Tools for MWE acquisition

- LocalMaxs – `hlt.di.fct.unl.pt/luis/multiwords/`
- Text::NSP – `search.cpan.org/dist/Text-NSP`
- UCS – `www.collocations.de/software.html`
- jMWE – `projects.csail.mit.edu/jmwe`
- Varro – `sourceforge.net/projects/varro/`
- Terminology extraction tools
- Many more (see WG3 summary)



Limitations

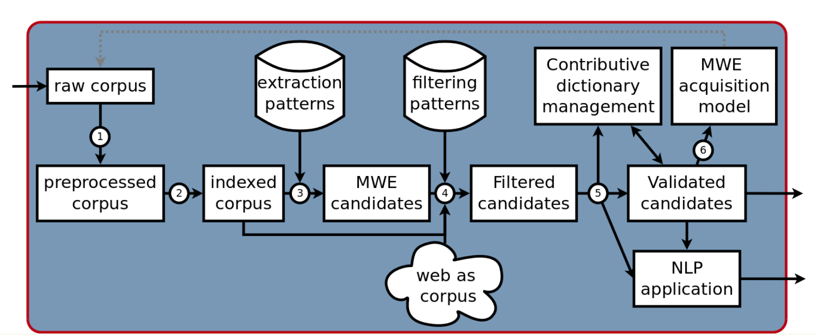
- Focus on part of acquisition pipeline
- Depend on given language, formalism or tool
- Choice a priori of level of analysis
- Lack of integrated and systematic framework

The mwetoolkit

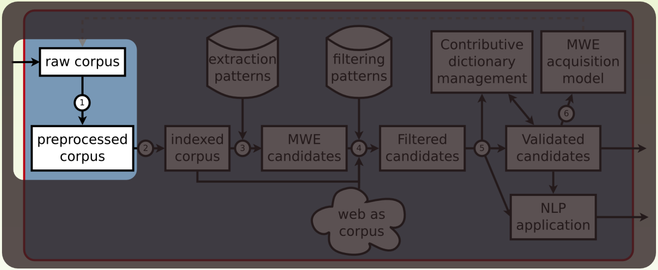
mwetoolkit.sf.net

- Developed since 2009 [Ramisch et al., 2010b, Ramisch et al., 2010a, Araujo et al., 2011, Ramisch, 2015]
- 201 downloads in 2014
- 57 mentions in the ACL anthology

Acquisition pipeline



Preprocessing



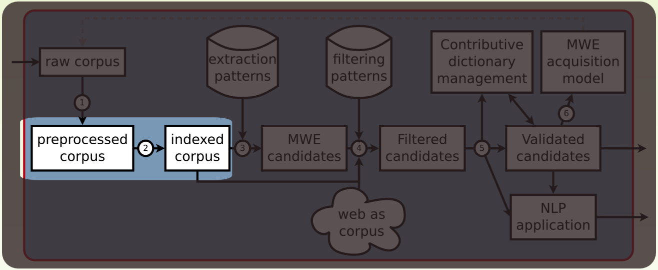
Preprocessing (external)

External tools + import

1. Tokenisation
2. Lemmatisation
3. POS tagging
4. Dependency parsing

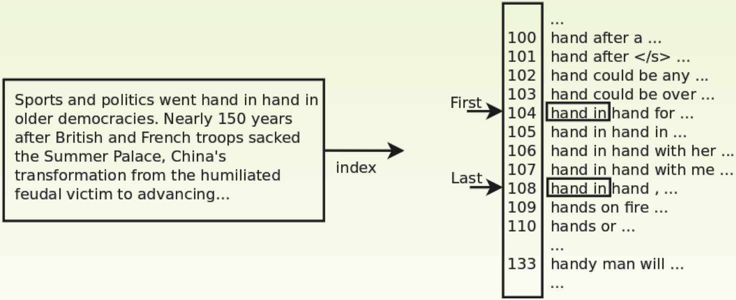
NEW: support several file formats for corpora

2. Indexing


raw corpus
extraction patterns
filtering patterns
Contributive dictionary management
MWE acquisition model
1
preprocessed corpus
2
indexed corpus
3
MWE candidates
4
Filtered candidates
5
Validated candidates
6
web as corpus
NLP application

Indexing

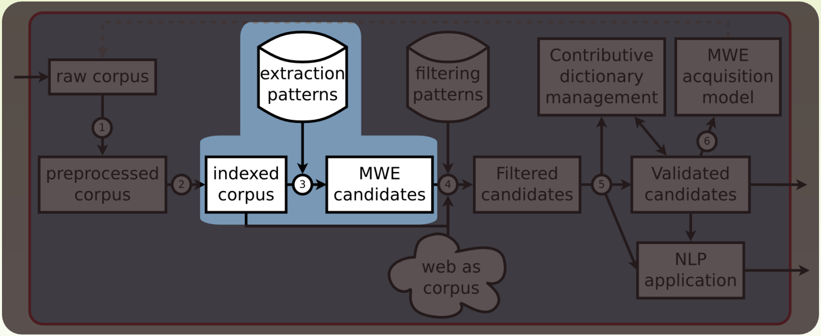
- Suffix array



Exercise: indexing

1. Open the source `ted-en-sample.conll`. Do you know this format? What information does each column contain?
2. Compile the fast C indexer
   ```
   cd mwetoolkit
   make
   cd ..
   ```
3. Run the command below to index the corpus
   ```
   python mwetoolkit/bin/index.py -v -i index/ted \
       ted-en-sample.conll
   ```

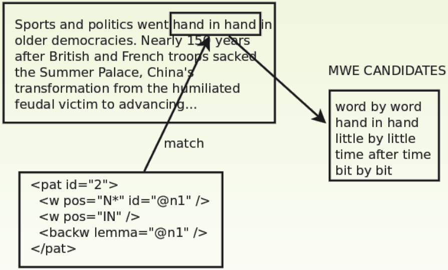
Candidate extraction



Candidate extraction

- *Inputs:* indexed corpus, extraction patterns
- *Outputs:* MWE candidates

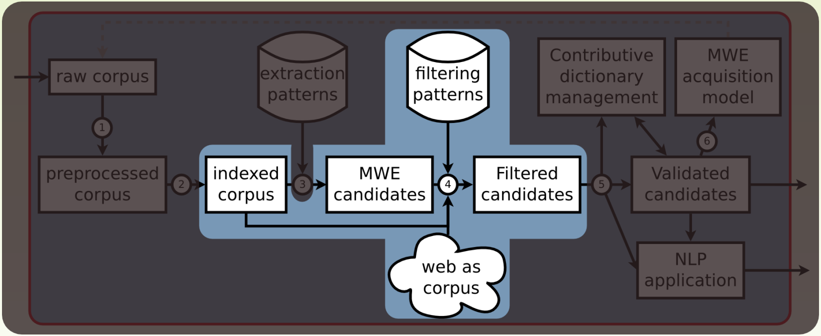
- RegExp pattern
- Multilevel
- Back reference
- Wildcard
- NEW: negation, match length



Exercise: candidate extraction

1. What does the file `pat_nn.xml` describe?
2. Extract nominal candidates using the pattern file
```
python mwetoolkit/bin/candidates.py -p pat_nn.xml -S \
    -v --from=BinaryIndex index/ted.info > cand.xml
```
3. How many candidates matched the pattern?
```
python mwetoolkit/bin/wc.py cand.xml
```
4. Count the candidates in the corpus
```
python mwetoolkit/bin/counter.py -v \
    -i index/ted.info cand.xml > cand-count.xml
```

Candidate filtering



Association measures I

- Compare expected count $E(w_1^n)$ and observed count $c(w_1^n)$

$$E(w_1^n) = \frac{c(w_1)c(w_2)\dots c(w_n)}{N^{n-1}}$$

- Some popular association measures

$$\text{t-score} = \frac{c(w_1^n) - E(w_1^n)}{\sqrt{c(w_1^n)}}$$

$$\text{pmi} = \log_2 \frac{c(w_1^n)}{E(w_1^n)}$$

$$\text{dice} = \frac{n \times c(w_1^n)}{\sum_{i=1}^{n} c(w_i)}$$

- Use of thresholds to remove noise

Association measures II

Contingency tables

	w_2	$\neg w_2$	
w_1	$c(w_1 w_2)$	$c(w_1 \neg w_2)$ $= c(w_1) - c(w_1 w_2)$	$c(w_1)$
$\neg w_1$	$c(\neg w_1 w_2)$ $= c(w_2) - c(w_1 w_2)$	$c(\neg w_1 \neg w_2)$ $= N - c(w_1) - c(w_2) + c(w_1 w_2)$	$c(\neg w_1)$ $= N - c(w_1)$
	$c(w_2)$	$c(\neg w_2)$ $= N - c(w_2)$	N

$$\text{LL} = \sum_{w_i w_j} \log \left[\frac{c(w_i w_j)}{E(w_i w_j)} \right]^{c(w_i w_j)}$$

Stefan Evert's website `http://www.collocations.de`

Exercise: candidate filtering I

1. Filter out candidates occurring once in the corpus

```
python mwetoolkit/bin/filter.py -v \
    -t ted:2 cand-count.xml > cand-count-f1.xml
```

2. How many candidates left in the file?

```
python mwetoolkit/bin/wc.py cand-count-f1.xml
```

3. Calculate the standard association measures

```
python mwetoolkit/bin/feat_association.py -v \
    cand-count-f1.xml > cand-feat.xml
```

4. Keep the top 1000 candidates according to `ll_ted`

```
python mwetoolkit/bin/sort.py -v \
    -f ll_ted cand-feat.xml |
    python mwetoolkit/bin/head.py -v  \
    -n 1000 > cand-feat-ft.xml
```

5. What other information could be used to filter?

NEW: Corpus annotation

1. Use information about source sentence generated by candidates.py (expressive regexp)
2. Project a lexicon on a corpus (independent resources)

Exercise: annotate the corpus I

1. Project the candidates on the corpus using source information

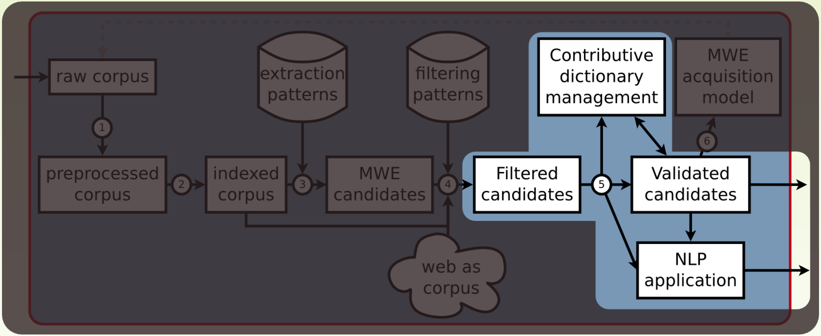
```
python mwetoolkit/bin/annotate_mwe.py \
    --detector=Source -c cand-feat-ft.xml \
    --to PlainCorpus ted-en-sample.conll \
    > ted-en-sample-mwe.conll
```

2. Look for underscore "_" in the resulting file.
3. Try again using `ContiguousLemma` as detector. What changes?
4. Try also options `-g` and `--filter`, and HTML output

```
python mwetoolkit/bin/annotate_mwe.py \
    --detector=Source -c cand-feat-ft.xml \
    --to HTML --filter-and-annot \
    ted-en-sample.conll > ted-mwe.html
```

5. What happens for nested MWEs?

Validation, evaluation, application



Evaluation of MWE acquisition

1. What are the acquisition goals (that is, the target applications) of the resulting MWEs?
2. What is the nature of the evaluation measures that we intend to use?
3. What is the cost of the resources (dictionaries, reference lists, human experts) required for the desired evaluation?
4. How ambiguous are the target MWE types?

Acquisition context

Generalisation of evaluation results depends on parameters of acquisition context

- Characteristics of target MWEs
  - Type
  - Language
  - Domain
- Characteristics of corpora
  - Size
  - Nature
  - Level of analysis
- Existing resources

Exercise: evaluation

1. Export the candidates to CSV format
```
python mwetoolkit/bin/convert.py --from XML \
    --to CSV cand-feat-ft.xml > cand-feat-ft.csv
```
2. Open the file using a spreadsheet processor
```
libreoffice cand-feat-ft.csv
```
Note: Don't forget to set TAB as separator
3. Sort the data according to different association measures. Which one is best?
4. Sort the data in descending order of `t_ted` and annotate the first 20 candidates as true/false MWEs
5. Compare your results with your neighbours. Do you agree? Why?
6. Homework: repeat the extraction using `pat_open.xml`
NEW: try `--id-order=base:collocate` in `candidates.py`

Acknowledgements

Aline Villavicencio, Magali Sanches Duran, Evita Linardaki, Vitor de Araujo, Sandra Castellanos, CAMELEON & AIM-WEST projects

Merci beaucoup !

Muito obrigado!

Thank you!

Getting our hands dirty with the mwetoolkit

mwetoolkit team (mwetoolkit@gmail.com)

References I

Araujo, V. D., Ramisch, C., and Villavicencio, A. (2011).
Fast and flexible MWE candidate generation with the mwetoolkit.
In Kordoni, V., Ramisch, C., and Villavicencio, A., editors, *Proc. of the ALC Workshop on MWEs: from Parsing and Generation to the Real World (MWE 2011)*, pages 134–136, Portland, OR, USA. ACL.

Ramisch, C. (2015).
Multiword Expressions Acquisition: A Generic and Open Framework, volume XIV of *Theory and Applications of Natural Language Processing*.
Springer.

Ramisch, C., Villavicencio, A., and Boitet, C. (2010a).
Multiword expressions in the wild? the mwetoolkit comes in handy.
In Liu, Y. and Liu, T., editors, *Proc. of the 23rd COLING (COLING 2010) — Demonstrations*, pages 57–60, Beijing, China. The Coling 2010 Organizing Committee.

Ramisch, C., Villavicencio, A., and Boitet, C. (2010b).
mwetoolkit: a framework for multiword expression identification.
In *Proc. of the Seventh LREC (LREC 2010)*, pages 662–669, Valetta, Malta. ELRA.