# Fast and Flexible MWE Candidate Generation with the `mwetoolkit`

**Vitor De Araujo**[♠] **Carlos Ramisch**[♠ ♡] **Aline Villavicencio**[♠]
♠ Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
♡ GETALP – LIG, University of Grenoble, France
{vbuaraujo,ceramisch,avillavicencio}@inf.ufrgs.br

## Abstract

We present an experimental environment for computer-assisted extraction of Multiword Expressions (MWEs) from corpora. Candidate extraction works in two steps: generation and filtering. We focus on recent improvements in the former, for which we increased speed and flexibility. We present examples that show the potential gains for users and applications.

## 1 Project Description

The `mwetoolkit` was presented and demonstrated in Ramisch et al. (2010b) and in Ramisch et al. (2010a), and applied to several languages (Linardaki et al., 2010) and domains (Ramisch et al., 2010c). It is a downloadable open-source[1] set of command-line tools mostly written in Python. Our target users are researchers with a background in computational linguistics. The system performs language- and type-independent candidate extraction in two steps[2]:

1. Candidate generation
   - Pattern matching[3]
   - $n$-gram counting
2. Candidate filtering
   - Thresholds, stopwords and patterns
   - Association measures, classifiers

The main contribution of our tool, rather than a novel approach to MWE extraction, is an environment that systematically integrates the functionalities found in other tools, that is, sophisticated corpus queries like in CQP (Christ, 1994) and Manatee (Rychlý and Smrz, 2004), candidate generation like in Text::NSP (Banerjee and Pedersen, 2003), and filtering like in UCS (Evert, 2004). The pattern matching and $n$-gram counting steps are the focus of the improvements described in this paper.

## 2 An Example

Our toy corpus, consisting of the first 20K sentences of English Europarl v3[4], was POS-tagged and lemmatized using the TreeTagger[5] and converted into XML. [6] As MWEs encompass several phenomena (Sag et al., 2002), we define our target word sequences through the *patterns* shown in figure 1. The first represents sequences with an optional (?) determiner DET, any number (*) of adjectives A and one or more (+) nouns N. This shallow pattern roughly corresponds to noun phrases in English. The second defines expressions in which a repeated noun is linked by a preposition PRP. The `backw` element matches a previous word, in this example the same lemma as the noun identified as `noun1`.

After corpus indexing and $n$-gram pattern matching, the resulting unique candidates are returned. Examples of candidates captured by the first pattern

[1] `sf.net/projects/mwetoolkit`

[2] For details, see previous papers and documentation

[3] The following attributes, if present, are supported for patterns: surface form, lemma, POS, syntactic annotation.

[4] `statmt.org/europarl`

[5] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger`

[6] For large corpora, XML imposes considerable overhead. As corpora do not require the full flexibility of XML, we are currently experimenting with plain-text, which is already in use with the new C indexing routines.



*Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE 2011)*, pages 134–136,
Portland, Oregon, USA, 23 June 2011. 

```
<pat id="1">
   <pat repeat="?"><w pos="DET"/></pat>
   <pat repeat="*"><w pos="A"/></pat>
   <pat repeat="+"><w pos="N"/></pat>
</pat>
<pat id="2">
   <w pos="N" id="noun1"/>
   <w pos="PRP"/>
   <backw lemma="noun1" pos="noun1"/>
</pat>
```

Figure 1: Pattern 1 matches NPs, pattern 2 matches sequences $N_1$ PRP $N_1$.

are *complicated administrative process*, *the clock*, *the War Crimes Tribunal*. The second pattern captures *hand in hand*, *eye to eye*, *word for word*. [7]

## 3 New Features

**Friendlier User Interface** In the previous version, one needed to manually invoke the Python scripts passing the correct options. The current version provides an interactive command-based interface which allows simple commands to be run on data files, while keeping the generation of intermediary files and the pipelining between the different phases of MWE extraction implicit. At the end, a user may want to save the session and restart the work later.[8]

**Regular Expression Support** While in the previous version only wildcard words were possible, now we support all the operators shown in figure 1 plus repetition interval (2,3), multiple choice (`either`) and in-word wildcards like *writ** matching *written*, *writing*, etc. All these extensions allow for much more powerful candidate patterns to be expressed. This means that one can also use syntax annotation if the text is parsed: if two words separated by $n$ words share a syntactic head, they are extracted. Multi-attribute patterns are correctly handled during pattern matching, in spite of individual per-attribute indices. Some scripts may fuse the individual indices on the fly, producing a combined index (e.g. $n$-gram counting).

[7] Currently only contiguous $n$-grams can be captured; non-contiguous extraction (e.g., verb-noun pairs, with intervening material, not part of the expression) is planned.

[8] Although it is not a graphical interface some users request, it is far easier to use than the previous version.

**Faster processing** Candidate generation was not able to deal with large corpora such as Europarl and the BNC. The first optimization concerns pattern matching: instead of using the XML corpus and external matching procedures, now we match candidates using Python's builtin regular expressions directly on the corpus index. On a small corpus the current implementation takes about 72% the original time to perform pattern-based generation. On the BNC, extraction of the two example patterns shown before took about 4.5 hours and 1 hour, respectively. The second optimization concerns the creation of the index. The previous script allowed a static index to be created from the XML corpus, but it was not scalable. Thus, we have rewritten index routines in C. We still assume that the index must fit in main memory, but the new routines provide faster indexing with reasonable memory consumption, proportional to the corpus size. These scripts are still experimental and need extensive testing. With the C index routines, indexing the BNC corpus took about 5 minutes per attribute on a 3GB RAM computer.

## 4 Future Improvements

Additionally to evaluation on several tasks and languages, we intend to develop several improvements to the tool. First, we would like to rewrite the pattern matching routines in C to speed the process up and reduce memory consumption. Second, we would like to test several heuristics to handle nested candidates (current strategy returns all possible matches). Third, we would like to perform more tests on using regular expressions to extract candidates based on their syntax annotation. Fourth, we would like to improve candidate filtering (not emphasized in this paper) by testing new association measures, filters, context-based measures, etc. Last but most important, we are planning a new release version and therefore we need extensive testing and documentation.

## References

Satanjeev Banerjee and Ted Pedersen. 2003. The design, implementation, and use of the Ngram Statistic Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Com-*

*putational Linguistics*, pages 370–381, Mexico City, Mexico, Feb.

Oli Christ. 1994. A modular and flexible architecture for an integrated corpus query system. In *COMPLEX 1994*, Budapest, Hungary.

Stefan Evert. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart, Stuttgart, Germany.

Evita Linardaki, Carlos Ramisch, Aline Villavicencio, and Aggeliki Fotopoulou. 2010. Towards the construction of language resources for greek multiword expressions: Extraction and evaluation. In Stelios Piperidis, Milena Slavcheva, and Cristina Vertan, editors, *Proc. of the LREC Workshop on Exploitation of multilingual resources and tools for Central and (South) Eastern European Languages*, pages 31–40, Valetta, Malta. May.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010a. Multiword expressions in the wild? the mwetoolkit comes in handy. In *Proc. of the 23rd COLING (COLING 2010) — Demonstrations*, pages 57–60, Beijing, China, Aug. The Coling 2010 Organizing Committee.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010b. mwetoolkit: a framework for multiword expression identification. In *Proc. of the Seventh LREC (LREC 2010)*, Malta, May. ELRA.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010c. Web-based and combined language models: a case study on noun compound identification. In *Proc. of the 23rd COLING (COLING 2010)*, pages 1041–1049, Beijing, China, Aug. The Coling 2010 Organizing Committee.

Pavel Rychlý and Pavel Smrz. 2004. Manatee, bonito and word sketches for czech. In *Proceedings of the Second International Conference on Corpus Linguisitcs*, pages 124–131, Saint-Petersburg, Russia.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd CICLing (CICLing-2002)*, volume 2276/2010 of *LNCS*, pages 1–15, Mexico City, Mexico, Feb. Springer.