mwetoolkit-lib: Adaptation of the mwetoolkit as a Python Library and an Application to MWE-based Document Clustering

Fernando Rezende Zagatti[*] [‡], **Paulo Augusto de Lima Medeiros**[*], **Esther da Cunha Soares**[*], **Lucas Nildaimon dos Santos Silva**[*] [‡], **Carlos Ramisch**[†], **Livy Real**[*]

[*]americanas s.a
{paulo.medeiros, esther.soares, livy.coelho}@b2wdigital.com

[‡]Federal University of São Carlos
{fernando.zagatti, lucas.nildaimon}@estudante.ufscar.br

[†]Aix Marseille Univ, CNRS, LIS, Marseille, France
carlos.ramisch@lis-lab.fr

Abstract

This paper introduces the mwetoolkit-lib, an adaptation of the mwetoolkit as a python library. The original toolkit performs the extraction and identification of multiword expressions (MWEs) in large text bases through the command line. One of the contributions of our work is the adaptation of the MWE extraction pipeline from the mwetoolkit, allowing its usage in python development environments and integration in larger pipelines. The other contribution is the execution of a pilot experiment aiming to show the impact of MWE discovery in data professionals' work. Thus, we propose a textual clustering experiment in which we compare using single-word and MWE features. This experiment found that the addition of MWE knowledge to the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization altered the word relevance order, improving the linguistic quality of the clusters returned by k-means.

Keywords: Multiword expressions, Python library, Clustering, k-means

1. Introduction

According to the literature, multiword expressions (MWEs) are combinations of two or more words that present some characteristic behavior when occurring together, having a different behavior when compared to the words used individually (such as 'hot dog' and 'human resources'). This difference can be at any given linguistic level(s), including morphology, syntax, semantics and/or pragmatics (Baldwin and Kim, 2010). Moreover, MWEs often present statistical salience with respect to the distributions of the words that compose them. Due to their unpredictable nature, from a computational perspective, it is challenging to know how to deal with such terms, and often they end up generating errors in Natural Language Processing (NLP) tasks. Therefore, in the industrial context, data analysts and scientists need to be able to process such multiword units in order to enhance their analysis and interpretation of textual data.

As explained by Constant et al. (2017) and Watrin and François (2011), MWE processing is essential for several NLP tasks, such as parsing, machine translation, information extraction and retrieval. Also, MWE processing can be divided into MWE discovery and identification (Constant et al., 2017); the former focuses on extracting MWE candidates from corpora and building a lexicon, and the latter targets labelling word combinations as MWEs in context. Although usually explored in academic research contexts, the task of MWE discovery may also turn out relevant in industrial contexts. Thus, tools like the research-oriented `mwetoolkit` (Ramisch, 2014) could be adapted to benefit not only NLP researchers, but also data analysts working on applied text-related problems.

For that reason, we developed a wrapper for the Multiword Expressions toolkit, the `mwetoolkit-lib`[1], aiming at the MWE discovery task. It is a python library which can be seamlessly imported into any external python code, including jupyter notebooks, and it is integrated with `pandas` (Wes McKinney, 2010), a widely used python library for data analysis.

Such as `mwetoolkit` proposes to easily identify MWEs within a given corpus, our library allows its use outside the command lines. As it is a library that can be easily integrated into pipelines, the `mwetoolkit-lib` main target audience is developers and data scientists, but it can still be used by different professionals, such as lexicographers and translators to find terms of interest.

Furthermore, this paper proposes a pilot experiment on how MWE discovery may impact data scientists and analysts' daily work. We observed that a generalist scope encompasses multiple domains that, in turn, have their own specific MWEs. Therefore, it may be that a word combination in one domain is not an MWE in other domains. To avoid potential domain ambiguities and maximize our knowledge and control of the results, we focus on terminological MWEs relevant to our con-

[1] `https://gitlab.com/fernandozagatti/mwetoolkit-lib/`

Proceedings of the 18th Workshop on Multiword Expressions (MWE 2022) n@LREC2022, pages 112–117
Marseille, 25 June 2022. © European Language Resources Association (ELRA), licensed under CC-BY-NC-4.0

text only, delimiting our experiments to texts in the Human Resources (HR) domain.

The pilot experiment consists in analyzing the impact of discovering MWEs as key terms from an HR corpus and clustering the corpus documents using the k-means algorithm (MacQueen and others, 1967) on the extracted MWE features. We choose such task because it is prototypical in the daily work of both data scientists and analysts, who often have to face the lack of annotated data required for supervised learning methods. In addition, there seems to be considerably less literature on MWE-aware applications based on unsupervised methods. Aiming to automatize an applied data processing pipeline using morphosyntactic patterns for MWE discovery and unsupervised techniques to work with corpora, the main contributions of this paper are:

- Development of the `mwetoolkit-lib`, a freely available python library based on the `mwetoolkit`, which ensures larger usability and integration with resources widely used in academia and industry.
- A pilot experiment to check the impact of the use of MWE knowledge (so linguistic/symbolic knowledge) on unsupervised clustering algorithms results.

2. Related work

Usually, state-of-the-art techniques for automatic identification of MWEs use morphosyntactic patterns combining linguistic and statistical information, rarely resorting to explicit representations of the meaning of words (Seretan, 2011; Ramisch, 2014; Constant et al., 2017). The literature is extensive and there are works in different domains and tasks related to MWE, such as discovery, identification and MWE-aware applications. For discovery, also using `mwetoolkit` as the basis of their work, Cordeiro et al. (2016) presented an extension to `mwetoolkit`, named `mwetoolkit+sem`. They add a new metric for MWE discovery/extraction which tries to estimate a combination's compositionality using word embeddings. In general, the score is calculated through the cosine distance between the MWE term and the words that make up the MWE.

Dubremetz and Nivre (2014) used the `mwetoolkit` on a sample of the French Europarl corpus and the French MWE lexicon Delac for training binary classifiers aiming at MWE discovery. They obtained a maximum precision of 74% in a manual evaluation of this classification task, i.e. 74% of the candidate MWEs classified as correct MWEs were indeed MWEs. Also, approximately half of the correctly discovered MWEs were not present in Delac, contributing to the enrichment of the French MWE lexicon.

Unsupervised methods for MWE discovery have been employed in the past, including clustering techniques (Tutubalina, 2015; Chakraborty et al., 2011). Though, MWE discovery supporting unsupervised text analytics remains understudied to the best of our knowledge.

3. The mwetoolkit-lib

Existing tools for MWE discovery propose sub-optimal interfaces for data analysts, specially considering the use of linguistic and domain-specific knowledge. Hence, we aim to integrate the consolidated methodology with these tools daily used by data scientists. Based on the `mwetoolkit`, a robust framework for processing MWEs, it was necessary to adapt the existing code to integrate the methods and commands used in the terminal into any python script, including notebooks, broadly used by data analysts and scientists.

3.1. The mwetoolkit

The `mwetoolkit` (Ramisch, 2014) is a robust toolkit for MWE processing which proposes a command line interface and is organized as a set of python scripts. It allows text preprocessing while supporting different tagger and parser file formats, complex morphosyntactic user-defined pattern searching using multi-level regular expressions, efficient word and n-gram counting, and statistical measures for MWE discovery. In addition, the toolkit has modules for MWE identification based on lexicon matching and on Conditional Random Fields, but these are out of scope given that we focus on MWE extraction.

The MWE discovery task is tackled using the statistical salience that MWEs may have and common morphosyntactic patterns they share. This pipeline is the following: (I) MWE candidates are searched within the corpus' n-grams using the user-defined patterns; (II) the absolute frequency for each candidate is computed; (III) statistical Association Measures (AMs) are computed; and (IV) the discovered candidates are filtered and ranked according to such measures. These steps are detailed below:

I **Pattern searching:** Given a list of morphosyntactic patterns which comprises lemmas, surface forms, POS tags and/or syntactic dependencies, all n-grams that match these patterns are extracted from the input corpus.

II **MWE candidates counting and word indexing:** Occurrences of each MWE candidate and their component words need to be counted in order to compute the final AMs. A suffix array was implemented for word indexing and thus handling this task efficiently.

III **Statistical Association Measures:** Different AMs are computed using both n-gram and component words' counts as input: maximum likelihood estimator, dice's coefficient, pointwise mutual information and student's t-score. Such AMs are key for the lexicometric analysis of the data professional.

IV **Ranking and filtering:** As its name suggests, MWE candidates might not be MWEs. As a post-processing step, filtering such candidates can be

done by using their counts or AMs. Also, candidate ranking using AMs is supported.

3.2. Adaptation to a python library

The code proposed in the `mwetoolkit` for achieving the MWE discovery pipeline is robust and finely organized. Each step is handled by a single script or a pair of scripts: (I) `candidates.py`; (II) `index.py` and `counter.py`; (III) `feat_association.py`; (IV) `sort.py` and `filter.py`. All these scripts make up the internal library `mwetk` which comprises shared functions and classes.

Aiming to adapt this pipeline to the `mwetoolkit-lib`, we first identified the functionalities that should be shared between the proposed library and the aforementioned scripts. Then, the corresponding methods were moved to an internal library `mwetk`, and the scripts were updated accordingly, so that they keep functional after the refactoring.

The main method of the `mwetoolkit-lib`, to be called by the user in a python script, was built inside the `mwetoolkitlib.py` file and must be accessed by calling the `get_candidates_dataframe` method. The idea here is to encapsulate all intermediate steps into a single function, hiding unimportant details about the tool's internal architecture from the users, leaving the pipeline less prone to human errors.

It is necessary to pass two parameters to the method, namely: (1) a corpus file containing the corpus from which the MWEs will be discovered with the POS tags, lemmas and surface forms for each token and (2) a file containing the morphosyntactic patterns that the user wants to extract from the text. Both files can be presented in any format supported by the `mwetoolkit`. This method will return a `pandas` dataframe with MWE candidates and their info. As in the original `mwetoolkit`, candidates are shown in their normalized form (lemmas) alongside with their POS tags, occurrences count and AMs. Ranking and filtering can now be easily done using `pandas` and data can be integrated with other python libraries.

4. Experimental evaluation

For making sure we properly reproduced all the steps of the `mwetoolkit`, we proposed an experimental evaluation considering an industry daily task: creating representative textual datasets using unsupervised techniques. We want to show that `mwetoolkit-lib` does not miss any detail of the `mwetoolkit` and to check how the use of linguistic knowledge through making the textual clustering a MWE-aware task improves the quality of our results.

For this evaluation, we used a private dataset of texts of the HR domain provided by americanas s.a, describing employees activities in Brazilian Portuguese, and containing 20,000 documents.

The first step in the evaluation was to run tests to confirm that the command-line `mwetoolkit` and our python library were extracting the same results. After ensuring that both were extracting the same 8,300 MWEs and generating the same list of candidates, we investigated the impact of MWEs on the k-means clustering algorithm in this data.

4.1. Term Frequency-Inverse Document Frequency

Among different techniques for converting text into numeric vectors, we chose the *Term Frequency-Inverse Document Frequency* (TF-IDF) since this is a straightforward and consolidated technique in the literature. Pimpalkar and Raj (2020) define this technique as a quantitative metric used to determine the relevance of terms in a document. The formulas for calculating the TF-IDF used in this project, taken from scikit-learn[2], are represented by Equations 1 and 2.

$$tfidf(t,d) = tf(t,d) * idf(t) \quad (1)$$

$$idf(t) = log[(1+n)/(1+df(t))] + 1 \quad (2)$$

The following topics define the meaning of each term in Equations 1 and 2:

- **tf-idf(t, d):** "Term Frequency-Inverse Document Frequency" of term "t" in document "d".
- **idf(t):** "Inverse Document Frequency" which measures how common a word is among all documents.
- **tf(t, d):** Computes "term frequency" which is the number of times a word "t" appears in a document "d".
- **n:** Total number of documents available.
- **df(t):** Number of documents in which the term "t" appears.

4.2. K-means clustering

K-means is a clustering algorithm proposed by MacQueen and others (1967). Its main process is the partitioning of its N-dimensional dataset into k distinct groups based on samples. It manages to provide partitions that are reasonably efficient in terms of cluster variation, mainly because it is an unsupervised technique and does not require expert considerations.

As reported by Xiong et al. (2016), after initializing the algorithm and imputing the dataset and the value of k, k samples are randomly selected as centroids, one for each cluster. Then, at each step, the algorithm calculates the distance of the dataset samples from each of the k centroids, assigning the sample to the closest centroid and, once all samples are classified in a cluster, the centroids are recalculated; this process is repeated iteratively until the clusters do not undergo major changes.

[2] `https://scikit-learn.org/`

4.3. The experiment

Since this algorithm does not consider any linguistic feature, we want to test whether imputing the data with MWE analysis would improve the quality of the clusters found by k-means. The pipeline for the experiments, seen in Figure 1, was performed with and without the MWE extraction step.

Firstly, we conducted the textual preprocessing (tokenization, transformation of the text into lowercase, removal of diacritics and stopwords), and vectorization with TF-IDF. Then, the k-means method was applied with 8 clusters. The number of clusters was defined by the Elbow method.[3]

Secondly, the MWE discovery step was inserted before preprocessing and, when tokenization was performed, NLTK's MWETokenizer (Bird et al., 2009) was used to merge the discovered MWEs into single tokens. The morpho-syntactic patterns used by the `mwetoolkit-lib` can be seen in Table 1. These patterns were defined by linguists based on related works such as Boos et al. (2014) and experimental tests within HR domain. Lemmas and POS tags used by the MWE extraction pipeline were computed using the `stanza` library(Qi et al., 2020).

Pattern	Examples
NOUN ADP NOUN	atendimento ao cliente (customer service)
NOUN ADJ ADJ	planejamento orçamentário anual (annual budget planning)
NOUN NOUN ADJ	inglês nível intermediário (intermediate English)
NOUN NOUN NOUN	Supremo Tribunal Federal (Federal Supreme Court)
NOUN ADJ	nota fiscal (invoice)
NOUN NOUN	vale transporte (transportation allowance)

Table 1: Morphosyntactic patterns used for discovery.

4.4. Evaluation results

Using vectorization with TF-IDF, it was possible to extract the degree of relevance of the words and to rank them according to their value. Extracting the top-5 words (Table 2) for vectorization using the knowledge of MWE, the token "atendimento ao cliente" ('customer service') was identified as something very relevant to the text. In the clusters without MWEs, this information was lost and the TF-IDF considered "service" and "customer" as distinct features. It may look very simple, but having this MWE identified, we could obtain a single cluster in which it is very salient, while, in the clusters without MWEs knowledge, the words

[3]This method tests the algorithm with different numbers of clusters in order to identify the optimal value of k.

Rank	With MWE	Without MWE
Top1	atividades	atendimento
Top2	responsavel	responsavel
Top3	principais	atividades
Top4	atendimento	area
Top5	atendimento_ao_cliente	cliente

Table 2: Relevance of words and MWEs by TF-IDF.

"atendimento" and "cliente" appeared in all the other clusters within the 15 most common unigrams.

For clustering, in the first run, without MWE, k-means created a cluster with the word "atendimento" (service) in which it brought texts about services in general, customer service, public service, telephone service, among others. In parallel, when we applied MWE discovery in the pipeline, a cluster was created specifically for the MWE "atendimento_ao_cliente" (customer service) and another for activities and services in general.

Adding the knowledge of MWE, 1282 MWEs appeared among the most frequent terms in the clusters. Without this information, only 48.60% had appeared among the most frequent terms. These MWEs are in the HR domain, such as "producao_de_conteudo" ('content creation') and "fechamento_de_caixa" ('financial close'). With one of the groups being more specifically about 'customer service', we were able to better differentiate the other clusters. In the MWE-aware version of this experiment, we obtain a cluster that deals specifically with financial tasks with terms such as "notas_fiscais" ('invoices'), "controle_de_contas" ('billing control') and "emissao_de_notas" ('invoice issuance') that were not representative in any cluster in the 'flat' version of the experiment.

It is important to emphasize that using unsupervised methods impose some difficulty on having highly trustful evaluation. Thus, our pilot experiment still requires a more in-depth quantitative evaluation in other datasets to observe the real effects of MWE on unsupervised clustering. However, it already showed the usability of `mwetoolkit-lib` and how it was easy to integrate the linguistic knowledge of `mwetoolkit` with other methods in a larger pipeline, bringing up an easy way to have hybrid approaches implemented for textual clustering.

5. Conclusions and future work

We implemented the `mwetoolkit` (Ramisch, 2014) as a python library, aiming to make this MWE module easier for data scientists to use in non-academic R&D contexts. We conducted some experiments to demonstrate the impact of using MWE knowledge in clustering methods and how MWEs extracted by `mwetoolkit-lib` can be used in an unsupervised method.

The adoption of hybrid approaches (such as MWE + clustering) brings advantages to the automatizing meth-

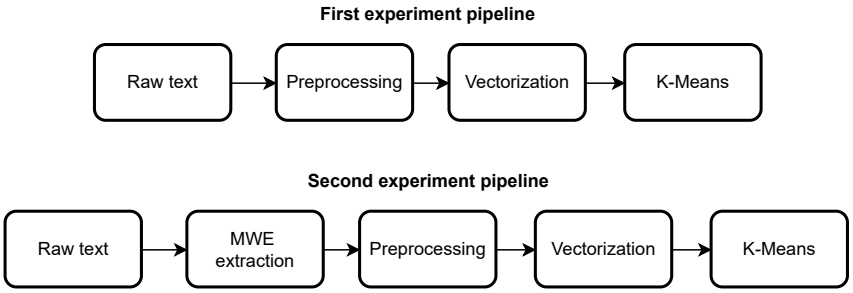

Figure 1: Difference between the pipeline of experiment 1 (top, no MWEs) and experiment 2 (bottom, with MWEs)

ods, in a way that the data does not need any previous human annotation to be used. We do believe that the future of NLP is based on bringing together linguistics/logic knowledge within the big data knowledge we can access together, and making these sources of information dialogue with each other.

The use of hybrid methods with MWEs can also bring domain knowledge that is implicit in the data. This knowledge can be extracted more easily when applying the techniques together with human experts to analyze the results individually.

As future work, extensions in both `mwetoolkit-lib` and experiments can be explored. The `mwetoolkit-lib` can benefit from the implementation of the MWE identification pipeline from the `mwetoolkit`, thus allowing training, evaluating and execution of the labelling of MWEs in running text. Furthermore, benefiting from the rich python environment, different Machine Learning algorithms can be used to tackle this new task by integrating the `mwetoolkit-lib` with other python libraries such as `scikit-learn` (Pedregosa et al., 2011) and `keras` (Chollet and others, 2015).

Concerning the experiments, we would like to carry out clustering using other algorithms (such as MiniBatch k-Means or HDBSCAN) and in new datasets, ensuring that the linguistic quality improvement we found generalizes over other architectures and domains.

6. Acknowledgements

This research was supported by americanas s.a. We thank our colleagues Helena de Medeiros Caseli, Diego Furtado Silva, Daniel Lucrédio, Lucas Cardoso Silva and Bruno Silva Sette from Federal University of São Carlos who provided insights and expertise that assisted this research. This work has been funded by the French Agence Nationale pour la Recherche, through the SELEXINI project (ANR-21-CE23-0033-01), and is part of the UFSCar extension project "Dos dados ao conhecimento: extração e representação de informação no domínio do e-commerce" (#23112.000186/2020-97)

7. Bibliographical References

Baldwin, T. and Kim, S. N. (2010). Multiword expressions. *Handbook of natural language processing*, 2:267–292.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Boos, R., Prestes, K., and Villavicencio, A. (2014). Identification of multiword expressions in the brwac. In *LREC*, pages 728–735.

Chakraborty, T., Das, D., and Bandyopadhyay, S. (2011). Semantic clustering: an attempt to identify multiword expressions in Bengali. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 8–13, Portland, Oregon, USA, June. Association for Computational Linguistics.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Constant, M., Eryiğit, G., Monti, J., van der Plas, L., Ramisch, C., Rosner, M., and Todirascu, A. (2017). Survey: Multiword expression processing: A Survey. *Computational Linguistics*, 43(4):837–892, December.

Cordeiro, S., Ramisch, C., and Villavicencio, A. (2016). mwetoolkit+sem: Integrating word embeddings in the mwetoolkit for semantic MWE processing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1221–1225, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Dubremetz, M. and Nivre, J. (2014). Extraction of nominal multiword expressions in french. In *MWE@EACL*.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pimpalkar, A. P. and Raj, R. J. R. (2020). Influence of pre-processing strategies on the performance of ml classifiers exploiting tf-idf and bow features. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 9(2):49–68.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Ramisch, C. (2014). *Multiword expressions acquisition: A generic and open framework*. Springer.

Seretan, V. (2011). *Syntax-based collocation extraction*, volume 44. Springer Science & Business Media.

Tutubalina, E. (2015). Clustering-based approach to multiword expression extraction and ranking. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 39–43, Denver, Colorado, June. Association for Computational Linguistics.

Watrin, P. and François, T. (2011). An n-gram frequency database reference to handle MWE extraction in NLP applications. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 83–91, Portland, Oregon, USA, June. Association for Computational Linguistics.

Wes McKinney. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt et al., editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Xiong, C., Hua, Z., Lv, K., and Li, X. (2016). An improved k-means text clustering algorithm by optimizing initial cluster centers. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 265–268. IEEE.