APPLICATION NOTE

# Facilitating phenotyping from clinical texts: the medkit library

Antoine Neuraz,[1,2,3] Ghislain Vaillant,[1,2] Camila Arias,[1,2] Olivier Birot,[1,2] Kim-Tam Huynh,[1,2] Thibaut Fabacher,[1,2,4] Alice Rogier,[1,2,5] Nicolas Garcelon,[1,2,6] Ivan Lerner,[1,2,5] Bastien Rance[1,2,5] and Adrien Coulet[1,2,*]

[1]Inria Paris, 75013, Paris, France, [2]Centre de Recherche des Cordeliers, Inserm UMR 1138, Université Paris Cité, Sorbonne Université, 75006, Paris, France, [3]Hôpital Necker, Assistance Publique - Hôpitaux de Paris, 75015, Paris, France, [4]University Hospital of Strasbourg, 67000, Strasbourg, France, [5]Hôpital Européen Georges Pompidou, Assistance Publique - Hôpitaux de Paris, 75015, Paris, France and [6]Imagine Institute, Inserm UMR 1163, Université Paris Cité, 75015, Paris, France

[*]Corresponding author. adrien.coulet@inria.fr

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

## Abstract

**Summary:** Phenotyping consists in applying algorithms to identify individuals associated with a specific, potentially complex, trait or condition, typically out of a collection of Electronic Health Records (EHRs). Because a lot of the clinical information of EHRs are lying in texts, phenotyping from text takes an important role in studies that rely on the secondary use of EHRs. However, the heterogeneity and highly specialized aspect of both the content and form of clinical texts makes this task particularly tedious, and is the source of time and cost constraints in observational studies.
**Results:** To facilitate the development, evaluation and reproducibility of phenotyping pipelines, we developed an open-source Python library named `medkit`. It enables composing data processing pipelines made of easy-to-reuse software bricks, named medkit operations. In addition to the core of the library, we share the operations and pipelines we already developed and invite the phenotyping community for their reuse and enrichment.
**Availability and Implementation:** `medkit` is available at `https://github.com/medkit-lib/medkit`.
**Contact:** `medkit-maintainers@inria.fr`.
**Supplementary information:** Documentation, examples and tutorials are available at `https://medkit-lib.org/`.

**Key words:** phenotyping, clinical texts, feature extraction, reproducible computing, open source

## Introduction

The collection at large scale of Electronic Health Records (EHRs) and the constitution of Clinical Data Warehouses (CDW) enable the design of clinical studies relying on a secondary use of healthcare data (Madigan et al., 2014). A substantial part of the necessary information to conduct these studies is available in texts, such as clinical notes, hospitalization or exam reports (Kharrazi et al., 2018). For instance, tasks such as the inclusion / exclusion of patients, and the extraction of outcome variables or covariates often require the consideration of clinical texts.

In biomedical data sciences, the two complementary tasks of either identifying individuals associated with a specific, potentially complex, trait or condition, or listing the traits of an individual are generally named *phenotyping*. And the specific case of phenotyping from clinical text is a continuous challenge for several reasons (Banda et al., 2018). First clinical text is highly specialized as it includes various factors of complexity such as medical entities absent from the general domain, hypotheses, negations, abbreviations, personal information; what motivates the development of dedicated phenotyping tools (Kreimeyer et al., 2017). Besides, many powerful Natural Language Processing (NLP) tools and models are developed and shared for both the general and biomedical texts, making reuse, adaptation and chaining rational approaches in biomedicine. But the highly heterogeneous aspect of clinical texts (*e.g.*, physician *vs.* nurse notes, hospital A *vs.* hospital B notes, French *vs.* English notes) makes the performance of a tool hardly predictable on a new corpus. In addition, clinical texts can barely be shared because of their personal and sensitive aspects. This implies the need for tools that ease the evaluation and adaptation of phenotyping approaches to the various types of texts generated in the large variety of existing clinical settings.

We present here `medkit`, an open-source Python library, that aims primarily at facilitating the reuse, evaluation, adaptation and chaining of NLP tools for the development of reproducible phenotyping pipelines. By extension, medkit enables the extraction of information related to patient care, such as

treatments or procedures, which are not phenotype per se. The rest of this manuscript presents the core elements of the library, develops on its easiness of use and details example pipelines developed with medkit for the extraction of drug treatments from clinical texts. It lists other pipelines already developed and ready for reuse and ends on two particularity added values of the library, which are the support of non-destructive processing and provenance tracing.

## Related work and positioning

The PheKB initiative proposes a collaborative web portal to share phenotyping algorithms in the form of semi-formal flow charts, documenting their steps and chaining (Kirby et al., 2016). In this manner, PheKB helps exchanging and standardizing phenotyping algorithms. However, provided representations are not shared with their computational implementations, which limits their reproducibility and comparison. In addition, algorithmic steps that rely on clinical texts are underspecified, as they usually require an adaptation to the peculiarities of local texts. The OHDSI community offers software tools such as Atlas, which proposes standard and reusable tools for the data analysis of observational studies from EHRs (Schuemie and DeFalco, 2019). Those are developed for steps coming next to the information extraction, once features are structured and normalized. medkit fills this exact step, extracting and normalizing features from unstructured parts of EHRs. The MedCAT library targets this step as well, but only focuses on entity recognition and normalization with the UMLS (Kraljevic et al., 2021). The Gate suite provides a graphical user interface which facilitates sequential application of various preprocessing and NLP tools on texts (Cunningham, 2002). But Gate is mostly adapted to educational or exploratory purposes, because of its limited ability to analyze large corpora and to adapt to novel tools. NLTK (Bird et al., 2009), spaCy (Honnibal and Montani, 2017) and FLAIR (Akbik et al., 2019) are Python libraries dedicated to advanced NLP development, designed for NLP engineers and researchers. Following a different way, medkit aims at being easier to start with, facilitating the reuse and chaining of simple-to-complex NLP tools, such as those developed with the previously cited libraries.

One of the main particularity of medkit is to place a strong emphasis on non-destructive operations, *i.e.*, no information is lost when passing data from one step to another; and on a flexible tracing of data provenance. In this matter, medkit is original and found inspiration in bioinformatic workflow management systems, such as Galaxy and Snakemake (Community, 2022; Mölder et al., 2021), which facilitate reproducibility of bioinformatic pipelines.

## The core components of medkit

For internal data management, medkit represents data with three simple core classes: Documents, Annotations and Attributes. Each of these classes is associated with properties and methods to represent data and metadata of various modalities such as audio or images, even though medkit is primarily designed for text. A Document is the minimal data structure of medkit, which associates an identifier with a set of Annotations; in turn an Annotation associates an identifier, a label and a set of Attributes; lastly Attributes associate an identifier, a label and a value.

For data processing, medkit defines two main classes: Operations and Pipelines. Typically, an operation is taking data as an input, runs a function over these data and returns output data. For instance an Operation can input a Document, perform Named Entity Recognition (NER) and output a set of Annotations associated with the Document. Accordingly, an operation can be the encapsulation of a previously developed tool, or a new piece of software developed in Python using medkit classes. Converters are particular operations for input and output management, which enable the transformation from standard formats such as CSV, JSON, Brat, Docanno annotations, into medkit Documents, Annotations and Attributes, or inversely. Lastly, Pipelines enable to chain Operations within processing workflows.

We refer readers to the medkit documentation for more details on its core components (see Availability for a web link).

## Encapsulate, chain, and reuse operations

Numerous data processing tools exist, in particular in NLP, where pretrained models are routinely shared within libraries or platforms such as spaCy or Hugging Face (Honnibal and Montani, 2017; Wolf et al., 2019). The goal of medkit is to facilitate their reuse, evaluation and chaining. Following are examples of available medkit operations that reuse third-party tools: the Microsoft library named Presidio for text de-identification (Mendels et al., 2018); a date and time matcher from the EDS-NLP lib(Wajsburt et al.); text translator using transformers from the Hugging Face platform. Similarly, medkit operations enable the encapsulation of spaCy modules in particular by input, output and annotation conversion functions.

In addition, we internally developed original operations for: NER; relation extraction; preprocessing; deidentification; evaluation; the pre-annotation of clinical texts to speed-up manual annotation; the detection of negation, hypothesis and antecedents within the context of entities; the fine-tuning of preexisting models; the classification of sentence and documents; the loading of audio patient-caregiver conversations, their diarisation and transcription to text (Nun et al.); and others. We aim at progressively enriching the catalogue of shared tools, thanks to the continuous growth of the community of medkit users and contributors.

The development of medkit pipelines is facilitated by three main elements. First, medkit data format asks for an initial conversion, but avoids further formatting in subsequent treatments. Second, sharing ready-to-use operations and open-source example pipelines speeds up the prototyping of new ones. Third, good practices in software development such as continuous integration, rich documentation facilitate start-off.

## Example pipelines

As an illustration, we describe two medkit pipelines in Figure 1 for the extraction of drug treatment from clinical texts. The first, in black, aims at comparing the performances of two NER tools named Drug NER 1 and 2, which are dictionary-based and Transformer-based methods respectively. Considering that Drug NER 2 obtained the best performances, the second pipeline is designed to use only the latter to extract the mentions of drug treatments from new texts. Both pipelines share three steps of preprocessing: convertion of raw texts into medkit Documents, sentence splitting and de-identification.

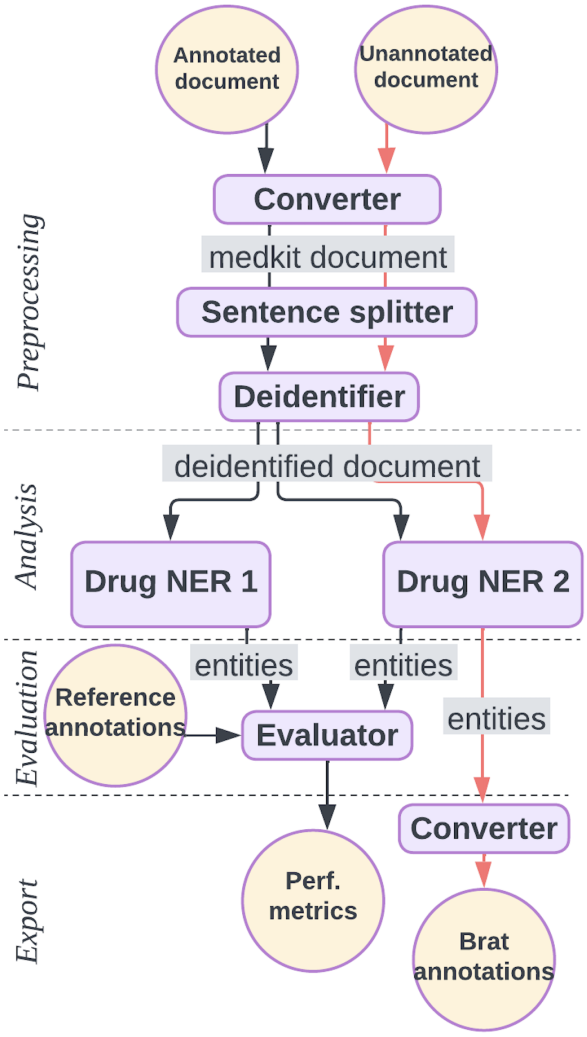



**Fig. 1.** Example medkit pipelines. The black pipeline converts raw texts to the medkit format, deidentifies them, recognizes drug entities with two distinct tools and compute performances for comparison. The orange pipeline, performs the same preprocessing tasks, recognizes drugs with only Drug NER 2 and outputs annotations in the Brat format.

```python
from medkit.core import DocPipeline
from medkit.core import Pipeline
from medkit.core import PipelineStep
from pathlib import Path
from medkit.core.text import TextDocument
#loading documents
docs = TextDocument.from_dir(path=Path("./working_dir"), pattern='*.txt', encoding='utf-8')
#pipeline definition (operations have been defined earlier. See full notebook)
my_pipeline = Pipeline(steps=[
    PipelineStep(sentence_tokenizer, input_keys=["full_text"], output_keys=["sentence"]),
    PipelineStep(deidentifier, input_keys=["sentence"], output_keys=["deided_sentence"]),
    PipelineStep(bert_matcher, input_keys=["deided_sentence"], output_keys=["drug_ner2_entities"])],
                 input_keys=["full_text"], output_keys=["drug_ner2_entities"])
#execution of the pipeline on each document
doc_pipeline = DocPipeline(pipeline=my_pipeline)
doc_pipeline.run(docs)
```

**Fig. 2.** Snippet of the implementation of the orange pipeline of Fig. 1.

The first pipeline evaluates the two tools on the basis of reference annotations saved as Brat format, whereas the second pipeline annotates new documents with drugs and produces output annotations in Brat format. A snippet of code for the medkit implementation of the second pipeline is shown in Fig. 2. The full implementation of the two pipelines is available at `https://medkit-lib.org/cookbook/drug_ner_eval/` .

## Available pipelines

We implemented and share pipelines for: the phenotyping of chemotherapy toxicities, and their grades (Rogier et al., 2022); the phenotyping of rheumatoid arthritis in French clinical reports (Fabacher et al., 2023b); the phenotyping of COVID-19 and the comparisons of pipelines relying either on the English *vs.* French UMLS (Neuraz et al., 2023); the benchmarking of NER approaches on three clinical case corpora, comparing dictionary-based, transformer and generative approaches (Hubert et al., 2024); the detection of text duplications in collections of clinical texts (Fabacher et al., 2023a). A last example of pipeline chains the recognition of drug, dates with a sentence classifier to detect treatment start and stop (Pohyer et al., 2024).

We refer the reader to the tutorial and cookbook sections of the medkit documentation for a list of available operations and examples of pipelines (see Availability for a web link).

## Non-destructive processing and provenance

The medkit library features two noteworthy functionalities: non-destructive processing and flexible provenance tracing. Non-destructive processing ensures that no information is lost when passing from one operation to the next. This is of interest to get back on the raw text, after this one underwent various transformation steps, such as deidentification or character replacements. Non-destructive processing is enabled by the propagation of original spans through successive operations. We note that this functionality might be lost in the case of external and noncompliant tools encapsulated in medkit operations.

Provenance tracing consists in recording provenance data, *i.e.*, meta-data documenting where data come from and how it was transformed. medkit implements this tracing by generating provenance data using the PROV-O standard ontology (Lebo et al., 2013). This tracing is flexible in the sense that users can set the level of verbosity and details they want to trace about the previous operations and states, in order to avoid generating large amounts of provenance data when those are unnecessary.

The unique combination of non-destructive processing and provenance tracing improves the explainability and reproducibility of results of pipelines of various level of complexity. These functionalities, along with its open-source nature and its focus on interoperability with existing libraries, pipelines and models, make it well aligned with the FAIR principles for research software (Barker et al., 2022).

## Availability

medkit is hosted at `https://github.com/medkit-lib/medkit`, and released under an MIT license. Its documentation, with examples and tutorials, is hosted at `https://medkit-lib.org/`.

## Conclusion and perspectives

medkit is an open source library for the composition of data processing pipelines made of easy-to-reuse software bricks, which aim at facilitating phenotyping from clinical texts. In addition to the core of the library, we share many of these bricks and examples of pipelines, and invite the phenotyping community for their reuse and enrichment.

So far, medkit enables linear execution of pipelines over a set of documents. Whereas it is simple to distribute the execution

of pipelines by splitting a large corpus in subsets, parallelization within pipelines is not supported yet, but is planned for the future. We would like to grow the community of users of medkit, first by developing a searchable library of available operations, by enriching this library and enabling users to share their own pipelines. Pipelines may be showcased in a gallery of examples to inspire and facilitate reuse. Next developments will concern operations for the generation of features that are compliant with the OMOP Common Data Model, and operations that facilitate the use of large language models and prompting.

## Competing interests

No competing interest to declare.

## Author contributions statement

G.V., C.A., O.B. and K.T.H. designed and implemented the library and reviewed the manuscript. T.F. and A.R. implemented pipelines and reviewed the manuscript. A.N., N.G., I.L., B.R. and A.C. obtained the funding, supervised the design and development, wrote and reviewed the manuscript.

## Acknowledgments

Authors thank users of medkit, L.-A. Guiottel, M. Hassani, S. Cossin, T. Hubert, V. Pohyer for their insightful inputs. This work was supported by Inria, Inria Paris; and the ANR under the France 2030 program [ANR-22-PESN-0007].

## References

A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019 (Demonstrations)*, pages 54–59, 2019.

J. M. Banda, M. Seneviratne, T. Hernandez, and N. H. Shah. Advances in electronic phenotyping: from rule-based definitions to machine learning models. *Annu. Rev. Biomed. Data Sci.*, 1:53–68, 2018. doi:10.1146/annurev-biodatasci-080917-013315.

M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, and T. Honeyman. Introducing the FAIR Principles for research software. *Sci. Data*, 9(1):622, 2022. ISSN 2052-4463. doi:10.1038/s41597-022-01710-x.

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

T. G. Community. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses. *Nucleic Acids Res.*, 50(W1):W345–W351, 2022. ISSN 0305-1048. doi:10.1093/nar/gkac247.

H. Cunningham. GATE: A framework and graphical development environment for robust nlp tools and applications. In *ACL 2002*, pages 168–175, 2002.

T. Fabacher, O. Birot, C. Arias-Villamil, K.-T. Huynh, A. Neuraz, and B. Rance. Détection de zones dupliquées dans des comptes rendus médicaux. In *Actes de la journée d'étude sur la Similarité entre Patients, SimPa 2023*, 2023a.

T. Fabacher, E.-A. Sauleau, N. Leclerc, H. Bergier, J.-E. Gottenberg, A. Coulet, and A. Névéol. Evaluating the Portability of Rheumatoid Arthritis Phenotyping Algorithms: case study on French EHRs. *Stud Health Technol Inform.*, (302):768–772, 2023b. doi:10.3233/SHTI230263.

M. Honnibal and I. Montani. spaCy2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To Appear. 2017.

T. Hubert, G. Vaillant, O. Birot, C. Arias, A. Neuraz, and A. Coulet. Comparing NER approaches on French clinical text, with easy-to-reuse pipelines. *Stud Health Technol Inform.*, 316:272–276, 2024. doi:10.3233/SHTI240396.

H. Kharrazi, L. J. Anzaldi, L. Hernandez, A. Davison, C. M. Boyd, B. Leff, J. Kimura, and J. P. Weiner. The value of unstructured electronic health record data in geriatric syndrome case identification. *J. Am. Geriatr. Soc.*, 66(8): 1499–1507, 2018. doi:10.1111/jgs.15411.

J. C. Kirby, P. Speltz, L. V. Rasmussen, M. Basford, O. Gottesman, P. L. Peissig, J. A. Pacheco, G. Tromp, J. Pathak, D. S. Carrell, et al. PheKB: a catalog and workflow for creating electronic phenotype algorithms for transportability. *J. Am. Med. Inform. Assoc.*, 23(6): 1046–1052, 2016. doi:10.1093/jamia/ocv202.

Z. Kraljevic, T. Searle, A. Shek, L. Roguski, K. Noor, D. Bean, A. Mascio, L. Zhu, A. A. Folarin, A. Roberts, R. Bendayan, M. P. Richardson, R. Stewart, A. D. Shah, W. K. Wong, Z. Ibrahim, J. T. Teo, and R. J. B. Dobson. Multi-domain clinical natural language processing with MedCAT: The medical concept annotation toolkit. *Artif. Intell. Med.*, 117:102083, 2021. ISSN 0933-3657. doi:10.1016/j.artmed.2021.102083.

K. Kreimeyer, M. Foster, A. Pandey, N. Arya, G. Halford, S. F. Jones, R. Forshee, M. Walderhaug, and T. Botsis. Natural language processing systems for capturing and standardizing unstructured clinical information: A systematic review. *J. Biomed. Inform.*, 73:14–29, 2017. ISSN 1532-0464. doi:10.1016/j.jbi.2017.07.012.

T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao. PROV-O: The PROV Ontology. *W3C*, 30, 2013.

D. Madigan, P. E. Stang, J. A. Berlin, M. Schuemie, J. M. Overhage, M. A. Suchard, B. Dumouchel, A. G. Hartzema, and P. B. Ryan. A systematic statistical approach to evaluating evidence from observational studies. *Annu. Rev. Stat. Appl.*, 1(1):11–39, 2014. doi:10.1146/annurev-statistics-022513-115645.

O. Mendels, C. Peled, N. Vaisman Levy, T. Rosenthal, L. Lahiani, et al. Microsoft Presidio: Context aware, pluggable and customizable pii anonymization service for text and images, 2018.

F. Mölder, K. P. Jablonski, B. Letcher, M. B. Hall, C. H. Tomkins-Tinch, V. Sochat, J. Forster, S. Lee, S. O. Twardziok, A. Kanitz, et al. Sustainable data analysis with snakemake. *F1000Research*, 10, 2021.

A. Neuraz, I. Lerner, O. Birot, C. Arias, L. Han, C. L. Bonzel, T. Cai, K. T. Huynh, and A. Coulet. TAXN: Translate Align Extract Normalize, a multilingual extraction tool for clinical texts. *Stud Health Technol Inform.*, (310):649–653, 2023. doi:10.3233/SHTI231045.

A. Nun, B. Olivier, G. Gaël, I. Lerner, and L. Frédéric. Simsamu-a french medical dispatch dialog open dataset. *Preprint*. doi:10.2139/ssrn.4869223.

V. Pohyer, E. Fabre, S. Oudard, L. Fournier, and B. Rance. Fake it till you predict it: data augmentation strategies to detect initiation and termination of oncology treatment.

*arXiv*, 2024. doi:10.48550/arXiv.2410.10271.

A. Rogier, A. Coulet, and B. Rance. Using an ontological representation of chemotherapy toxicities for guiding information extraction and integration from EHRs. *Stud Health Technol Inform.*, (290):91–95, 2022. doi:10.3233/SHTI220038.

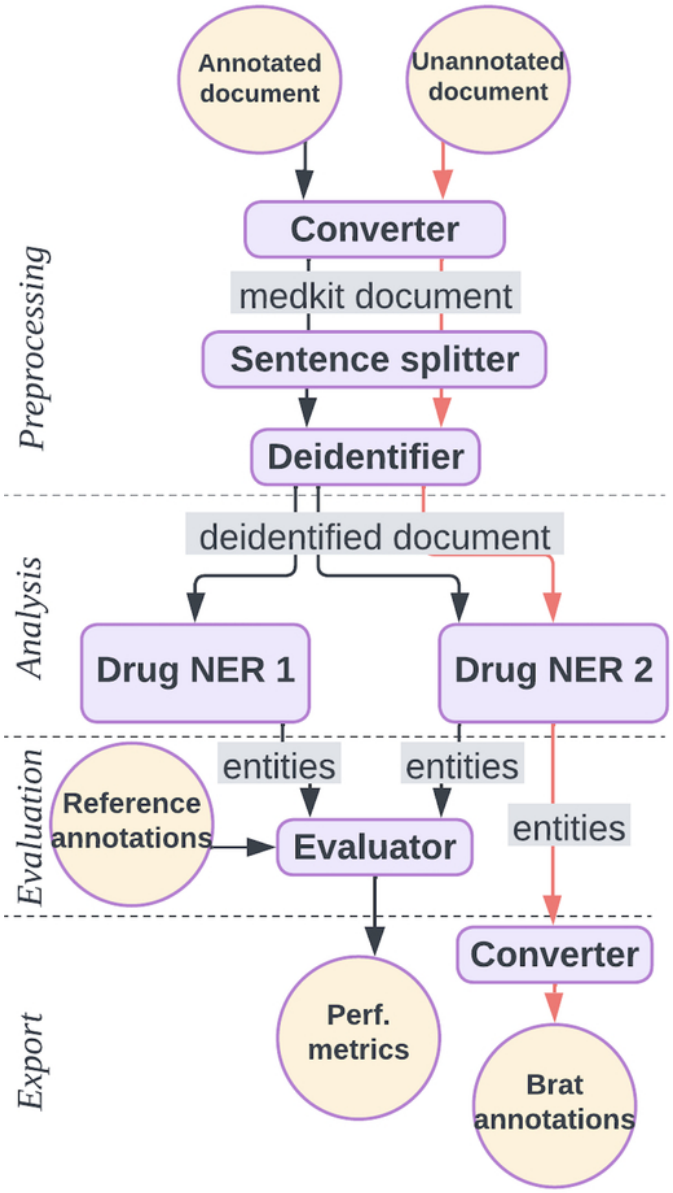
M. Schuemie and F. DeFalco. OHDSI Analytics Tools. In *The Book of OHDSI: Observational Health Data Sciences and Informatics*, chapter 8. OHDSI, 2019. ISBN 9781088855195.

P. Wajsburt, T. Petit-Jean, B. Dura, A. Cohen, C. Jean, and R. Bey. EDS-NLP: efficient information extraction from French clinical notes. *Zenodo*. doi:10.5281/zenodo.6424993.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv*, 2019. doi:10.48550/arXiv.1910.03771.

```python
from medkit.core import DocPipeline
from medkit.core import Pipeline
from medkit.core import PipelineStep
from pathlib import Path
from medkit.core.text import TextDocument
#loading documents
docs = TextDocument.from_dir(path=Path("./working_dir"), pattern='*.txt', encoding='utf-8')
#pipeline definition (operations have been defined earlier. See full notebook)
my_pipeline = Pipeline(steps=[
    PipelineStep(sentence_tokenizer, input_keys=["full_text"], output_keys=["sentence"]),
    PipelineStep(deidentifier, input_keys=["sentence"], output_keys=["deided_sentence"]),
    PipelineStep(bert_matcher, input_keys=["deided_sentence"], output_keys=["drug_ner2_entities"])],
                      input_keys=["full_text"], output_keys=["drug_ner2_entities"])
#execution of the pipeline on each document
doc_pipeline = DocPipeline(pipeline=my_pipeline)
doc_pipeline.run(docs)
```

143x46mm (300 x 300 DPI)




22x39mm (600 x 600 DPI)