CLPT: A Universal Annotation Scheme and Toolkit for Clinical Language Processing

Saranya Krishnamoorthy **Yanyi Jiang** **William Buchanan**
Ayush Singh **John E. Ortega**
inQbator AI at eviCore Healthcare
Evernorth Health Services
`firstname.lastname@evicore.com`

Abstract

With the abundance of natural language processing (NLP) frameworks and toolkits being used in the clinical arena, a new challenge has arisen – how do technologists collaborate across several projects in an easy way? Private sector companies are usually not willing to share their work due to intellectual property rights and profit-bearing decisions. Therefore, the annotation schemes and toolkits that they use are rarely shared with the wider community. We present the clinical language pipeline toolkit (CLPT) and its corresponding annotation scheme called the CLAO (Clinical Language Annotation Object) with the aim of creating a way to share research results and other efforts through a software solution. The CLAO is a unified annotation scheme for clinical technology processing (CTP) projects that forms part of the CLPT and is more reliable than previous standards such as UIMA, BioC, and cTakes for annotation searches, insertions, and deletions. Additionally, it offers a standardized object that can be exchanged through an API that the authors release publicly for CTP project inclusion.

1 Introduction

With the resurgence of deep learning and neural networks, the interest in using a clinical language framework for classifying clinical text in a digital manner has been heightened in recent years. Several workshops and shared tasks (Harper et al., 2021; Goeuriot et al., 2020; Rumshisky et al., 2020; Wang et al., 2020) have focused on the state-of-the-art approaches and the amount of private enterprises offering clinical solutions backed by machine learning technologies has increased drastically (Parida et al., 2022). Nonetheless, a recent study (Digan et al., 2021) shows that systems like UIMA (Ferrucci and Lally, 2004), CLAMP (Soysal et al., 2018), and cTakes (Savova et al., 2010), despite their age and typical technology stack (Java), are still a standard for clinical language text classification and there are only a few publicly available clinical language frameworks or standardized annotation schemes that provide easy ways to share results and other pertinent information with organizations, private or public. We propose a modern standardized framework that supports collaboration on clinical language research. Here we present the clinical language pipeline toolkit (CLPT), a framework developed with Python designed with software development principles. The CLPT enables researchers and entities to share their project results easily and supports research to be conducted in a fast and reproducible way. The unified annotation scheme for the CLPT is called the clinical language annotation object (CLAO). The CLAO is more reliable for annotation searches, insertions, and deletions than previous standards (e.g. UIMA(Ferrucci and Lally, 2004), cTakes(Savova et al., 2010) and BioC (Comeau et al., 2013)).[1] Additionally, the CLAO can be easily shared and integrated due to its standardization which makes it accessible through an application programming interface (API).

To illustrate the aforementioned concepts which will improve clinical technology processing (CTP) collaboration, we introduce five novel ideas and contributions in this article:

1. A freely available[2] annotation scheme (Clinical Language Annotation Object, CLAO) for CTP projects that can be interchanged between public and private sector organizations through offline and online resources such as APIs or file exchange.

2. A high-level Python framework (Clinical Language Pipeline Toolkit, CLPT) designed purposely in an ambiguous manner with the ob-

[1]The focus of this paper is to introduce the main concepts of the CLPT and the CLAO. We plan to publish efficiency results in a future iteration.

[2]`https://github.com/inQbator-eviCore/clpt`

jective of accepting any input of multiple modal types (i.e., speech, images, text, and more).

3. A novel algorithm for processing the annotation scheme that allows faster annotation inserts, deletes, and searches than previous frameworks.

4. An annotation scheme that can be converted to a linked data format which supports graph analytics on documents.

5. Out-of-the-box support for semantically comparing text in high-dimension spaces for state-of-the-art language models.

In the following sections, we first go through related work on annotation and natural language processing (NLP) tools in Section 2. In Section 3.1, we then describe in detail the CLAO scheme. Next, in Section 3.2, we cover the four CLPT modules for creating a typical CTP pipeline. Lastly, we conclude with the availability and future work.

2 Related Work

Several clinical text processing toolkits and annotations schemes have been introduced in the past but none of them provide the same functionality and efficiency as the CLAO and CLPT. Some widely used NLP tools for clinical text processing include the clinical text analysis and knowledge extraction system (cTAKES) (Savova et al., 2010), BioC (Comeau et al., 2013), Brat Rapid Annotation Tool (BRAT) (Stenetorp et al., 2012), General Architecture for Text Engineering (GATE) (Cunningham et al., 2002), Metamap (Aronson and Lang, 2010), Metamap Lite (Demner-Fushman et al., 2017), clinical language annotation, modeling, and processing (CLAMP) (Soysal et al., 2018) and sciSpaCy (Neumann et al., 2019).

BRAT (Stenetorp et al., 2012) is a web-based annotation tool for defining entities and creating annotations. Annotations created by BRAT are stored in a standoff format. Since BRAT XML output is similar to CLPT output, it can be easily adapted to CLAO by creating an adapted script, unlike outputs from cTakes or UIMA which are CAS files that are serialized using Java-style notation. Though the CLPT implements a similar approach of storing the annotation in a CLAO object, the CLAO's annotation scheme supports faster annotation insertion, deletion, and searching by implementing B-tree for indexing (see 3.1 for details).

cTAKES (Savova et al., 2010) is a clinical information retrieval system that combines rule-based methods and machine learning techniques for clinical narrative processing. It has been shown to work well on clinical notes alone but does not cover a broader set of NLP tasks (Neumann et al., 2019). The CLPT has been designed purposely ambiguous in order to accept multi-modal input and perform several NLP tasks.

GATE (Cunningham et al., 2002), CLAMP (Soysal et al., 2018), and BioC (Comeau et al., 2013) provide multiple tools which can be used for language processing tasks, annotating corpora, and performing evaluation. Yet, all three of them are either based on Java or C++. Additionally, GATE (Cunningham et al., 2002) and CLAMP (Soysal et al., 2018) depend on a framework called the unstructured information management architecture (UIMA) (Ferrucci and Lally, 2004). CLPT makes similar offerings as the three aforementioned frameworks but it uses Python which makes it easier to integrate with other modern deep-learning NLP frameworks such as TensorFlow (Abadi et al., 2016), MedSpacy (Eyre et al., (in press, n.d.) and PyTorch (Paszke et al., 2019).

The National Library of Medicine[3] presented a framework called Metamap (Aronson and Lang, 2010) for mapping biomedical text to unified medical language system (UMLS) concepts. Others (Soysal et al., 2018; Peng et al., 2020a) have found Metamap difficult for building machine learning models and hard to predict long entities due to its dictionary lookup method (Peng et al., 2020a). Previous research (Zhang et al., 2021) argues that neither Metamap nor CLAMP incorporate deep learning models directly. We believe that the CLAO and CLPT address several downfalls by creating an easy-to-use annotation scheme along with the targeted focus on deep learning.

We consider the work on sciSpaCy (Neumann et al., 2019) similar to ours because it was developed in Python and takes into account recent classification techniques in deep learning. However, to our knowledge, sciSpaCy (Neumann et al., 2019) does not support some of the default features found in the CLPT, such as a shareable annotation file that can be serialized to disk and efficient entity lookups as are offered in the CLAO.

[3]`https://www.nlm.nih.gov`

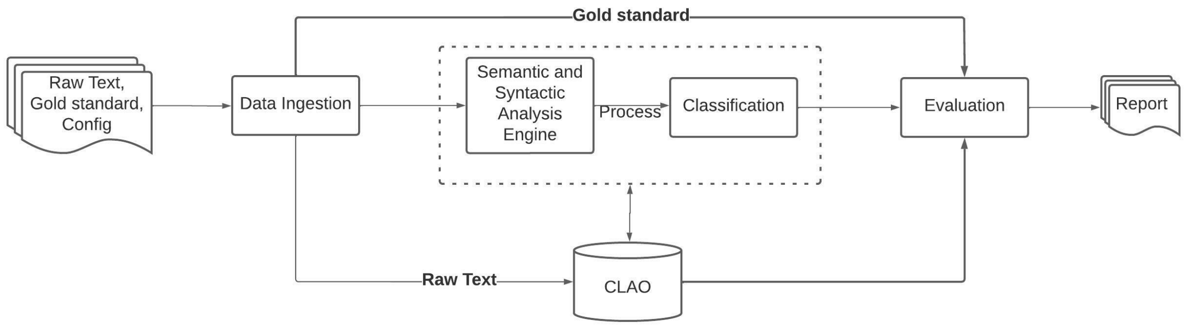

Figure 1: Clinical Language Pipeline Toolkit (CLPT) architecture

3 Methods

3.1 Clinical Language Annotation Object

In this section, we present two core CLAO innovations that provide efficient annotation storage and retrieval. The CLAO receives raw text as input which is cleaned and broken down into minimal units of analysis, expressed in this article as tokens. The CLAO has three main divisions for an annotation: (1) its elements, (2) its attributes and values, and (3) the relations linking the annotation to others (often times for syntactic or semantic representations).

The first step leading to the creation of a CLAO (as seen in Figure 1) is the segmentation of textual data into its minimal elements for annotation. Elements and values for the CLAO are extracted from the segments using sentence (or segment) detection and are stored and finally represented in a common annotation structure represented by a XML-based hybrid standoff format (Ide et al., 2017). We chose to represent the CLAO with a generalized representation in order to provide flexibility so that the annotation scheme was not constrained to the use of specific domains or tools. The version of the CLPT presented here supports exporting the CLAO into a JSON format, future iterations will provide a mechanism to allow users to export the CLAO into a JSON-LD format (Cimiano et al., 2020). JSON-LD is a novel contribution because, unlike other frameworks, it allows queries on the CLAO to be data-driven yet graph-based, similar to previous research (Hellmann et al., 2013; Cimiano et al., 2020) on efficiency. This promotes inter-operability and collaboration through a standard. For convenience, we have provided an example of a serialized CLAO in Appendix A.1.

As another novelty of our annotation implementation, the CLAO supports addition, deletion, and update operations along with the enhancement of annotations through the use of what are known as *B-Trees* for indexing (algorithms for processing stored data that are high performing, Johnson and Sasha (1993)). B-Tree indexing within the CLAO is performed at an asymptotic speed of $O(log\, n)$ for operations on CLAO entities – providing for a small storage footprint, easy scaling (without the need for rehashing as in the case of typical hash maps), and optimum segment loading.

The B-tree based algorithm, called a *blist*, used for indexing a CLAO uses an algorithm written by Daniel Stutzbach[4]. It combines a B-tree with an array for searches. In order to qualify that a blist would be the optimum algorithm for indexing a CLAO, we performed two main experiments illustrated in Figures 2 and 3. Both experiments compare the use of a default Python 3 list[5] data structure and the blist from Daniel Stutzbach. Our first experiment consisted in the creation of one-hundred default Python 3 lists and one-hundred blists both containing one million random floating numbers between 0 and 1. The second experiment consisted of random slicing which was done on both data structures (the Python 3 list and the blist)

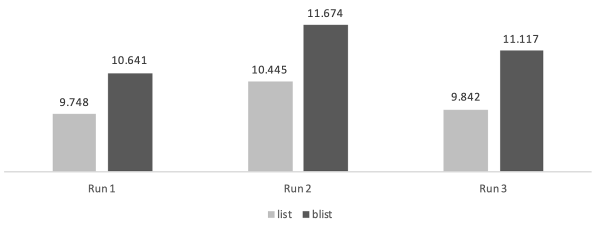

Figure 2: Creation Time Comparison (in Seconds)

[4]https://stutzbachenterprises.com/blist

[5]https://docs.python.org/3/library/stdtypes.html#list

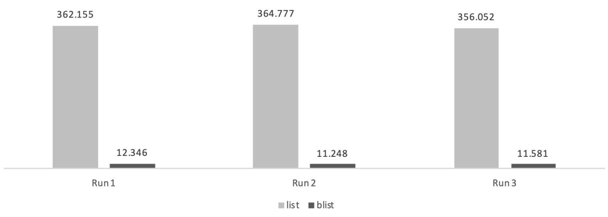

Figure 3: Slicing Time Comparison (in Seconds)

1000 times. The run time for both experiments was recorded and we found that the blist outperformed the Python 3 list as it was approximately 30 times faster thus making it the optimal choice for the CLPT at this time. In future work, we plan on extending the blist algorithm to include an even faster search.

3.2 Clinical Language Pipeline Toolkit

The CLPT is a CTP pipeline meant for exclusive use with the CLAO. We created the CLPT as an easy-to-use first pass for building a CLAO that can then be processed by others. In this short article, we only introduce novel themes along with findings and further plan to extend our work to introduce a larger pipeline backed by a CLAO. The architecture of the CLPT can be considered similar to other architectures like UIMA (Ferrucci and Lally, 2004) and CLAMP (Soysal et al., 2018) in some ways. However, it is our intent to allow further out-of-the-box novel features such as annotations mixed with embeddings. The CLPT, similar to its predecessors, has these four pipelines modules: (1) ingestion, (2) analysis engine, (3) classification, and (4) evaluation as shown in Figure 1. Each module has the option of saving any information to the CLAO as needed, in a repository-like manner. The CLAO is configured via a configuration file that enables any of the four modules, including the analysis and classification components, as explained in the following sections.

3.2.1 Ingestion

The CLPT is designed to be multi-modal, able to accept any form of input such as text, speech, video or images. At this point, we have experimented with text only and left other modalities for future work. The ingestion process is similar to other pipelines in that an object is considered for and serialized to the CLAO format. One main difference between the CLPT and other toolkits is that the CLPT was purposely created with high abstraction and is able to model any type of data. Figure 4 provides an example of the ingestion process which, similar to (Ferrucci and Lally, 2004), uses a document reader (called Document Collector), to read in data. Additionally, users have the option to pass in a configuration file (`.yml` format) designed to allow high-level control as to which modules to use. Nonetheless, there is also a "default" pipeline configuration which requires no intervention. The ingestion module handles the initial creation of the CLAO and passes the CLAO on for further processing to the analysis engine.

3.2.2 Analysis Engine

Our deep learning contribution is based on adding embeddings to the CLAO. Since embeddings are a key difference between the CLPT and other toolkits, we cover them here in further detail. Our embeddings can be used as part of creating a model for processing or loading a pre-trained model. Given that the majority of modern work on clinical NLP uses deep learning and/or embeddings, we felt it necessary to promote their inclusion in the CLPT. Our novel technique of storing embeddings by use of the CLAO has not been performed in the past. Additionally, we provide sub-word embedding combined with hashing trick for efficiency (Bojanowski et al., 2017) which are able to handle out-of-vocabulary (OOV) words. Embeddings are stored in CLAO objects efficiently, allowing comparison between tokens and spans of tokens. This is done by assigning a vector to each token or spans of tokens where the CLAO returns an average of all of the embeddings within it. Furthermore, CLPT offers a configuration mechanism for changing this span embedding method of calculation. Allowing for this flexibility can be considered a novel approach as it allows users to easily test various embedding types for experiments.

3.2.3 Classification

The classification module extracts knowledge from the CLAO by retrieving information from the upstream CLPT component(s). In this module, machine learning and other techniques (e.g., heuristics) are applied to further augment annotations for classification tasks before evaluation. Some of the major components to be released in the CLPT (See Appendix Figure 6), for the classification module are: (1) *acronym expansion* (similar to CARD (Wu et al., 2017)); (2) *mention detection* split into two phases, first a step to identify the mentions and then a step to group them together; (3) *fact extraction* to extract clinical concepts from the mentions which help to better disambiguate clinical notes and provide fact-based evidence for classification; (4) *relationship extraction* further expansion of mention detection to allow linking entities and the creation of a knowledge graph – to be presented as future work.

3.2.4 Evaluation

The CLPT provides an evaluation module (shown in Figure 7) as a separate module rather than the addition of classification or other processing techniques. The aim is to allow several forms of evaluation while, at a minimum, providing the baseline measurements such as precision, recall, F1-score, and accuracy. The baseline evaluation can be extended to cover any other common metrics but at this time we leave that for future work. The evaluation module takes two inputs: a CLAO and a gold standard. The CLAO is what will allow us to compare against the gold standard and both are required.

4 Concluding remarks and future work

We have introduced a novel and efficient toolkit for creating CTP pipelines with several new contributions. The thought has been to make a centralized format for exchanging information amongst entities, albeit academic or private. This will allow entities to compare and contrast results by comparing CLAOs that adhere to a standardized guideline. We contribute this to the public community as a way to use a more updated framework for modern CTP techniques. It is our thought that the CLPT can increase productivity and the exchange of information. The current implementation of the CLPT and the CLAO is in its infancy; the plan is to develop more functionality such as multi-modal inputs, the creation of a knowledge graph, and improved evaluation methods.

Additionally, we plan on extending the current implementation which performs classification using public machine learning models and heuristics by training models with the CLPT. Once those models have been trained, we also plan on adding the capability for fine-tuning those models for several clinical tasks able to handle diverse NLP problems like seminal work (Peng et al., 2020b) has done.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Alan R Aronson and François-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Philipp Cimiano, Christian Chiarcos, John P McCrae, and Jorge Gracia. 2020. Linguistic linked open data cloud. In *Linguistic Linked Data*, pages 29–41. Springer.

Donald C Comeau, Rezarta Islamaj Doğan, Paolo Ciccarese, Kevin Bretonnel Cohen, Martin Krallinger, Florian Leitner, Zhiyong Lu, Yifan Peng, Fabio Rinaldi, Manabu Torii, et al. 2013. Bioc: a minimalist approach to interoperability for biomedical text processing. *Database*, 2013.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175.

Dina Demner-Fushman, Willie J Rogers, and Alan R Aronson. 2017. Metamap lite: an evaluation of a new java implementation of metamap. *Journal of the American Medical Informatics Association*, 24(4):841–844.

William Digan, Aurélie Névéol, Antoine Neuraz, Maxime Wack, David Baudoin, Anita Burgun, and Bastien Rance. 2021. Can reproducibility be improved in clinical natural language processing? a study of 7 clinical nlp suites. *Journal of the American Medical Informatics Association*, 28(3):504–515.

Hannah Eyre, Alec B Chapman, Kelly S Peterson, Jianlin Shi, Patrick R Alba, Makoto M Jones, Tamara L Box, Scott L DuVall, and Olga V Patterson. (in press, n.d.). Launching into clinical space with medspacy: a new clinical text processing toolkit in python. In *AMIA Annual Symposium Proceedings 2021*.

David Ferrucci and Adam Lally. 2004. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Lorraine Goeuriot, Hanna Suominen, Liadh Kelly, Antonio Miranda-Escalada, Martin Krallinger, Zhengyang Liu, Gabriella Pasi, Gabriela Gonzalez Saez, Marco

Viviani, and Chenchen Xu. 2020. Overview of the clef ehealth evaluation lab 2020. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 255–271. Springer.

Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr, and Paul Groth. 2021. Semeval-2021 task 8: Measeval–extracting counts and measurements and their related contexts. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 306–316.

Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. 2013. Integrating nlp using linked data. In *International semantic web conference*, pages 98–113. Springer.

Nancy Ide, Christian Chiarcos, Manfred Stede, and Steve Cassidy. 2017. Designing annotation schemes: From model to representation. In *Handbook of linguistic annotation*, pages 73–111. Springer.

Theodore Johnson and Dennis Sasha. 1993. The performance of current b-tree algorithms. *ACM Transactions on Database Systems (TODS)*, 18(1):51–101.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*.

Prasanta Kumar Parida, Lingraj Dora, Monorama Swain, Sanjay Agrawal, and Rutuparna Panda. 2022. Data science methodologies in smart healthcare: a review. *Health and Technology*, pages 1–16.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Jacqueline Peng, Mengge Zhao, James Havrilla, Cong Liu, Chunhua Weng, Whitney Guthrie, Robert Schultz, Kai Wang, and Yunyun Zhou. 2020a. Natural language processing (nlp) tools in extracting biomedical concepts from research articles: a case study on autism spectrum disorder. *BMC Medical Informatics and Decision Making*, 20(11):1–9.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020b. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

Anna Rumshisky, Kirk Roberts, Steven Bethard, and Tristan Naumann, editors. 2020. *Proceedings of the 3rd Clinical Natural Language Processing Workshop*. Association for Computational Linguistics, Online.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Ergin Soysal, Jingqi Wang, Min Jiang, Yonghui Wu, Serguei Pakhomov, Hongfang Liu, and Hua Xu. 2018. Clamp–a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association*, 25(3):331–336.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Yanshan Wang, Sunyang Fu, Feichen Shen, Sam Henry, Ozlem Uzuner, and Hongfang Liu. 2020. The 2019 n2c2/ohnlp track on clinical semantic textual similarity: Overview. *JMIR Med Inform*, 8(11):e23375.

Yonghui Wu, Joshua C Denny, S Trent Rosenbloom, Randolph A Miller, Dario A Giuse, Lulu Wang, Carmelo Blanquicett, Ergin Soysal, Jun Xu, and Hua Xu. 2017. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (card). *Journal of the American Medical Informatics Association*, 24(e1):e79–e86.

Yuhao Zhang, Yuhui Zhang, Peng Qi, Christopher D Manning, and Curtis P Langlotz. 2021. Biomedical and clinical english model packages for the stanza python nlp library. *Journal of the American Medical Informatics Association*, 28(9):1892–1899.

A Appendix

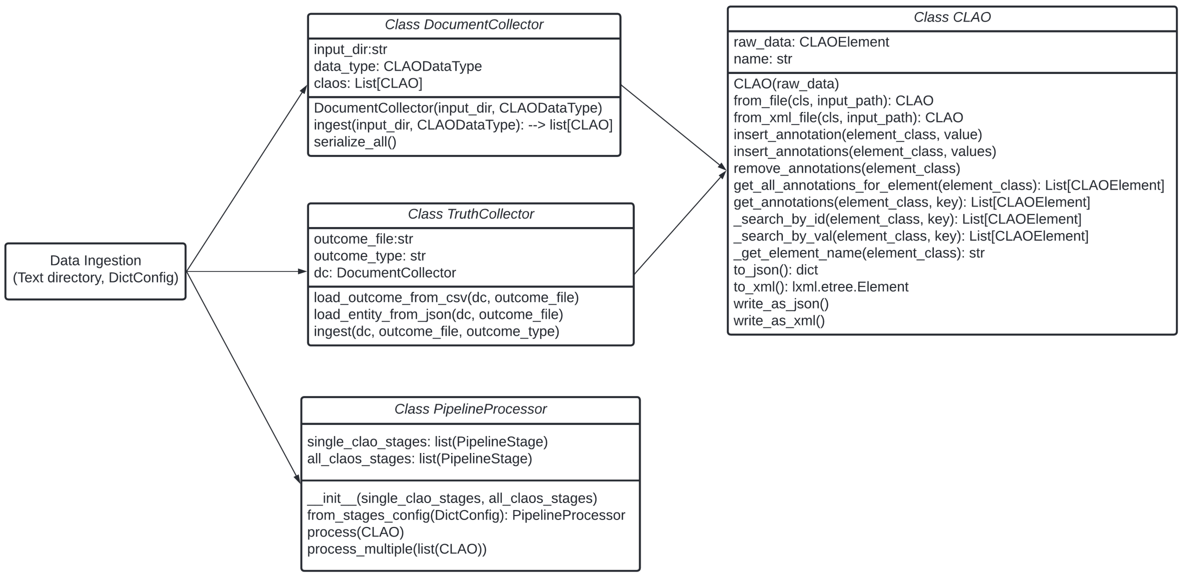
Data Ingestion
(Text directory, DictConfig)
Class DocumentCollector
input_dir:str
data_type: CLAODataType
claos: List[CLAO]
DocumentCollector(input_dir, CLAODataType)
ingest(input_dir, CLAODataType): --> list[CLAO]
serialize_all()
Class TruthCollector
outcome_file:str
outcome_type: str
dc: DocumentCollector
load_outcome_from_csv(dc, outcome_file)
load_entity_from_json(dc, outcome_file)
ingest(dc, outcome_file, outcome_type)
Class PipelineProcessor
single_clao_stages: list(PipelineStage)
all_claos_stages: list(PipelineStage)
__init__(single_clao_stages, all_claos_stages)
from_stages_config(DictConfig): PipelineProcessor
process(CLAO)
process_multiple(list(CLAO))
Class CLAO
raw_data: CLAOElement
name: str
CLAO(raw_data)
from_file(cls, input_path): CLAO
from_xml_file(cls, input_path): CLAO
insert_annotation(element_class, value)
insert_annotations(element_class, values)
remove_annotations(element_class)
get_all_annotations_for_element(element_class): List[CLAOElement]
get_annotations(element_class, key): List[CLAOElement]
_search_by_id(element_class, key): List[CLAOElement]
_search_by_val(element_class, key): List[CLAOElement]
_get_element_name(element_class): str
to_json(): dict
to_xml(): lxml.etree.Element
write_as_json()
write_as_xml()

Figure 4: The data ingestion module. It is used to ingest data and create an initial clinical language annotation object (CLAO) which can include text or other types (in future iterations).

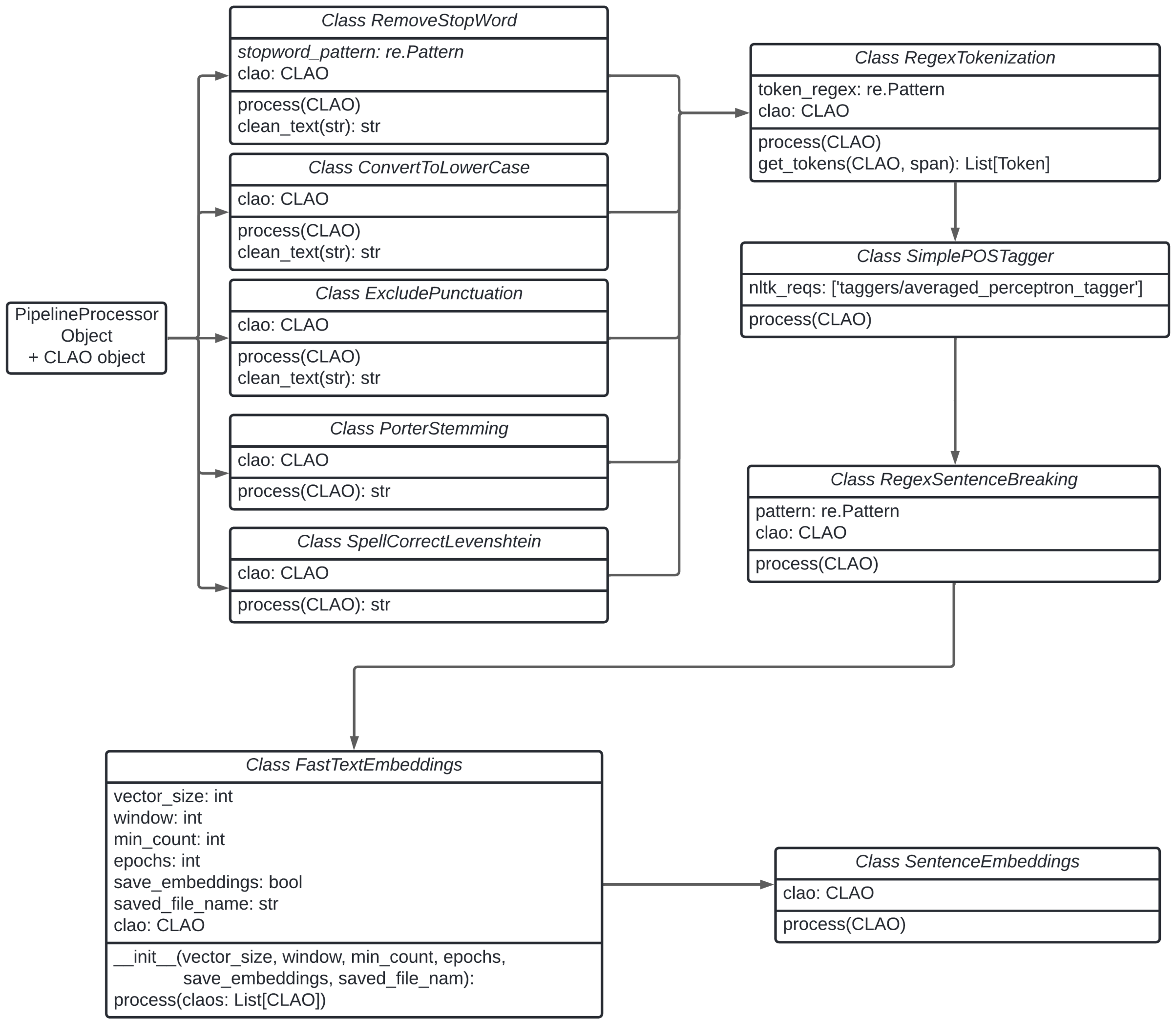

Figure 5: The analysis engine module. Each class has a method named *process()* that pre-processes and stores information from and to a clinical language annotation object (CLAO) during each stage.

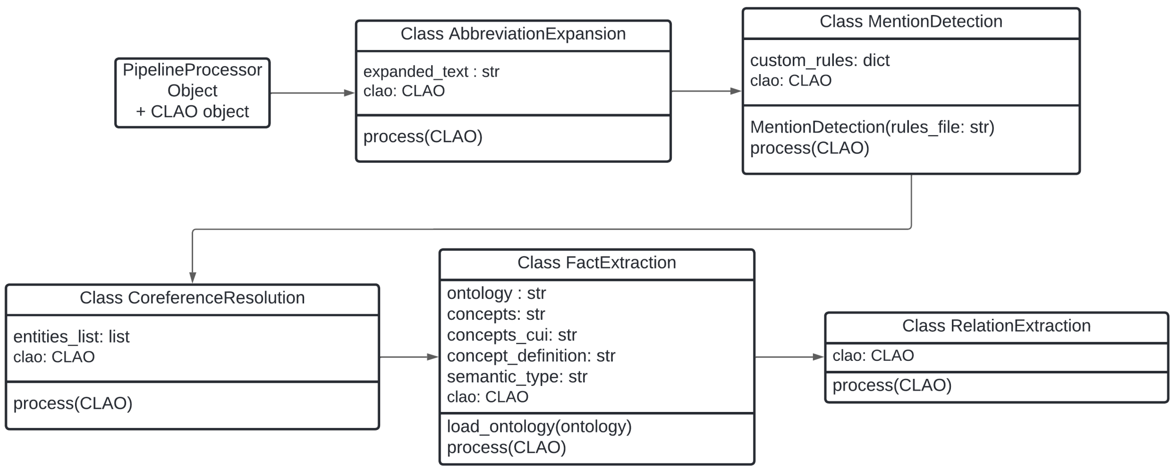

Figure 6: The classification module. This module is used to process and classify input from a clinical language annotation object (CLAO) in turn adding new information to it.

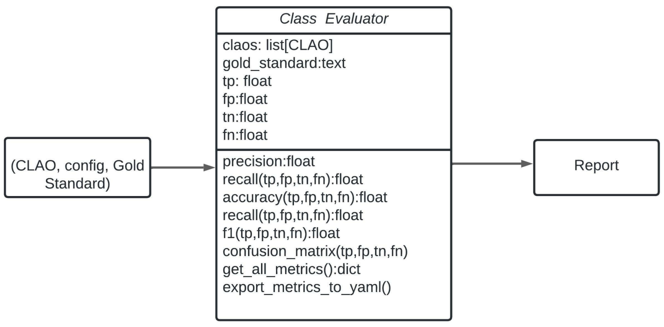

Figure 7: The evaluation module. A module that uses a clinical language annotation object (CLAO) and a gold standard to provide evaluation output in a report format.

A.1 Sample annotation

```
<?xml version='1.0' encoding='UTF−8'?>
<annotation>
    <text start="0" end="41" description="raw_text">Patient has type ii dm. This is not good.</text>
    <sentence id="0" start="0" end="23">
        <entity id="0" start="12" end="22" entity_group="0" token_ids="[2, 5)" type="MENTION" confidence="1"
            label="PROBLEM">Type II Diabetes Mellitus</entity>
        <token id="0" start="0" end="7" pos="NN" stem="patient" embedding_id="0">Patient</token>
        <token id="1" start="8" end="11" pos="VBZ" stem="ha" embedding_id="1">has</token>
        <token id="2" start="12" end="16" pos="VBN" stem="type" embedding_id="2">type</token>
        <token id="3" start="17" end="19" pos="JJ" stem="ii" embedding_id="3">ii</token>
        <token id="4" start="20" end="22" pos="NN" stem="dm" embedding_id="4">dm</token>
        <token id="5" start="22" end="23" pos="." stem="." embedding_id="5">.</token>
    </sentence>
    <sentence id="1" start="24" end="41">
        <token id="6" start="24" end="28" pos="DT" stem="thi" embedding_id="6">This</token>
        <token id="7" start="29" end="31" pos="VBZ" stem="is" embedding_id="7">is</token>
        <token id="8" start="32" end="35" pos="RB" stem="not" embedding_id="8">not</token>
        <token id="9" start="36" end="40" pos="JJ" stem="good" embedding_id="9">good</token>
        <token id="10" start="40" end="41" pos="." stem="." embedding_id="5">.</token>
    </sentence>
    <embedding id="0">[−0.0021704417, −0.010320467, −4.0913405e−06, −0.026113503, 0.003324223]</
        embedding>
    <embedding id="1">[0.03536414, −0.066816024, 0.018991465, 0.03511271, −0.02413405]</embedding>
    <embedding id="2">[−0.04219764, 0.051192448, 0.053828064, 0.013828199, −0.024849724]</embedding>
    <embedding id="3">[−0.011548042, −0.056690447, 0.0042386726, 0.013731264, −0.042996213]</
        embedding>
    <embedding id="4">[−0.015310202, −0.06731376, −0.023788698, −0.070030175, 0.0918083]</embedding>
    <embedding id="5">[−0.07549597, −0.034822427, −0.048076335, 0.05481594, −0.04260452]</embedding>
    <embedding id="6">[−0.08328381, 0.042492405, 0.026664842, 0.000608474, −0.023121541]</embedding>
    <embedding id="7">[−0.095420435, −0.043184925, 0.05082492, −0.015773036, −0.037915066]</embedding
        >
    <embedding id="8">[0.01620562, 0.030467993, −0.0037846065, 0.009880951, 0.0008572937]</embedding>
    <embedding id="9">[0.10948994, 0.040386822, 0.030505553, −0.03049627, 0.04858529]</embedding>
    <entity_group id="0" entity_type="MENTION">Type II Diabetes Mellitus</entity_group>
    <actual_label>0</actual_label>
    <probability>0.67</probability>
    <predicted_label>0</predicted_label>
</annotation>
```

Figure 8: A sample CLAO file comprising of two sentences in a single paragraph.