PyMedExt, un couteau suisse pour le traitement des textes médicaux

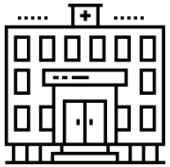
William Digan, Antoine Neuraz, Alice Rogier,
David Baudoin, Nicolas Garcelon, Bastien Rance

Journée AFIA-TLH/ATALA - TAL & IA 2021

jeudi 4 février 2021

L'HÔPITAL, UN MONDE DE LA DONNÉE

Prise en charge des patients



Dossier Patient Informatisé

Entrepôt de Données Cliniques

Entrepôt de données de l'Hôpital Européen Georges Pompidou

Données intégrées depuis 2000
Données pour environs 1 million de patients

Concept	# observations
Items du DPI	176,524,243
Examens de biologie	155,173,140
Codes diagnostics	4,846,602
Prescriptions médicamenteuses	91,251,062

L'HÔPITAL, UN MONDE DE LA DONNÉE

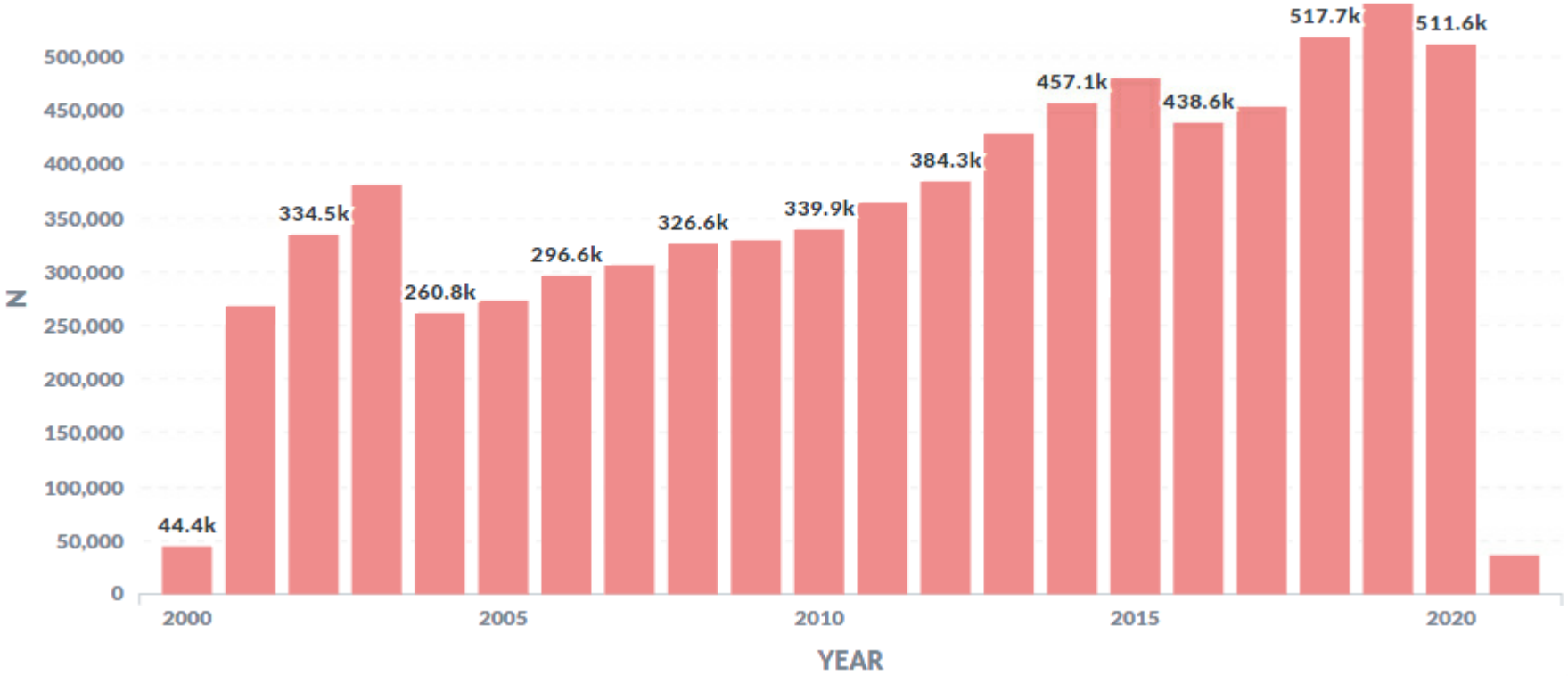
Grande variété de types de documents

Documents
- Comptes rendus de consultation
- Comptes rendus d'hospitalisation
- Comptes rendus opératoires
- Comptes rendus d'imagerie
- Comptes rendus d'anatomo-pathologie
- Lettres
- Ordonnances
- …

Et aussi
- Champs textes du dossier patient
- Transmissions infirmières
- …

Concept	# observations
Item de questionnaires	176,524,243
Biologie	155,173,140
Transmissions infirmières	22,001,856
PMSI	4,846,602
Prescription médicamenteuses	91,251,062
Textes cliniques	5,343,224
Textes complémentaires	1,650,027

Statistiques de l'Hôpital Européen Georges Pompidou, AP-HP

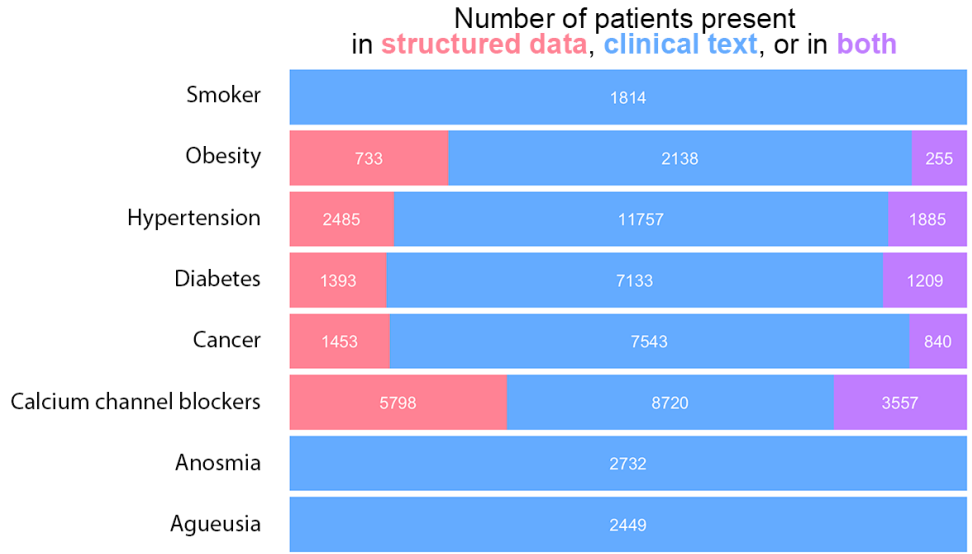


Plus de 1200 nouveaux documents chaque jour

L'importance des Textes Cliniques

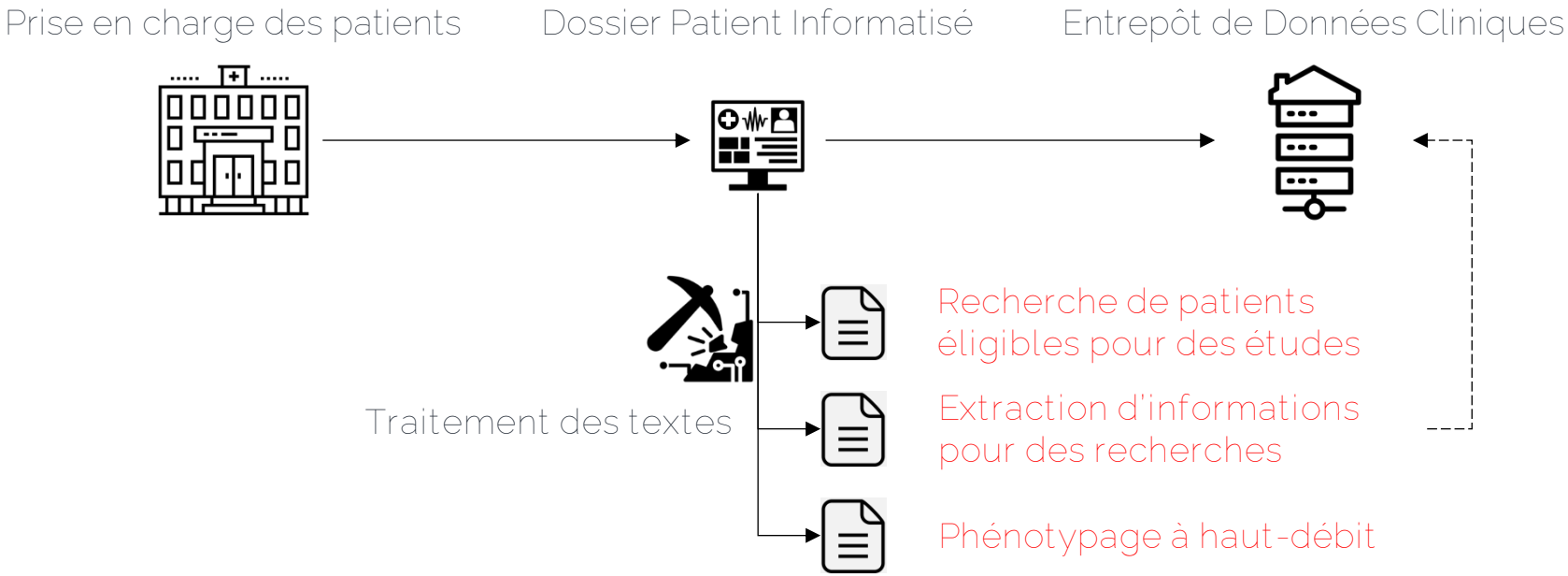
80%

De l'information médicale n'est présente que dans les textes



Neuraz A, Lerner I, Digan W, Paris N, Tsopra R, Rogier A, Baudoin D, Cohen KB, Burgun A, Garcelon N, Rance B; AP-HP/Universities/INSERM COVID-19 Research Collaboration; AP-HP COVID CDR Initiative. Natural Language Processing for Rapid Response to Emergent Diseases: Case Study of Calcium Channel Blockers and Hypertension in the COVID-19 Pandemic. J Med Internet Res. 2020 Aug 14;22(8):e20773. doi: 10.2196/20773. PMID: 32759101; PMCID: PMC7431235.

Le circuit des données



Quelles ressources pour traiter les textes

Nombreuses solutions intégrées (cTakes, CLAMP, GATE…)
- Surtout en anglais
- Souvent liées au formalisme d'annotateurs UIMA

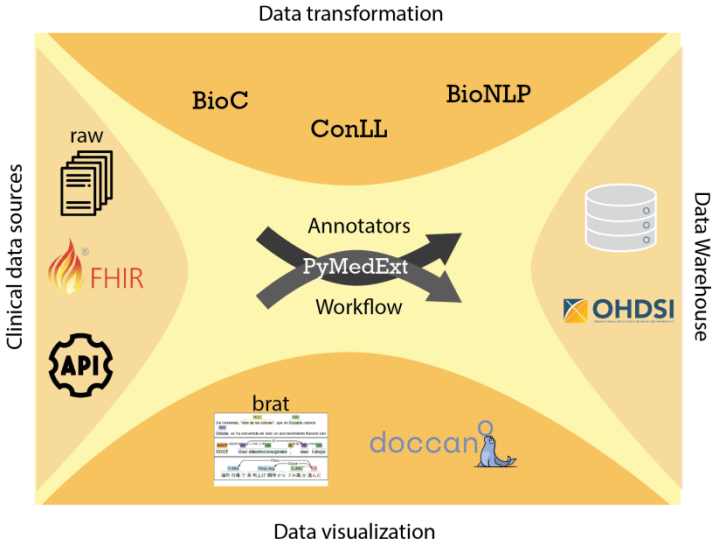
Nombreux outils (Metamap, QuickUMLS, HeidelTime…)
- Grande variété de langages de programmations
- Grande variété de formats d'entrées et sorties

Des solutions de partage de *pipelines* et d'intégration

Reproductibilité en Traitement des langues
PyMedExt une proposition pour facilité la reproductibilité

Un couteau suisse pour le traitement des langues cliniques

- Un format de représentation des documents et des annotations (extension de BioC)
- Des convertisseurs de formats
- Des exports vers des formats standards ou outils d'annotation
- Un format de représentation d'annotateurs
- Un moteur de pipelines simples



Travail démarré pendant la crise Covid. Collaboration de toute l'équipe (PI Antoine Neuraz)

- Utilisé pour l'annotation des comptes rendus cliniques à l'AP-HP pendant la première vague
- Outils d'échange dans l'équipe

https://github.com/equipe22/pymedext_core

Le format PyMedExt

Document

Annotation
- ID
- Type
- Valeur
- Source
- SourceID
- Span(début,fin)

Le format PyMedExt

Document

Annotation
- ID
- Type
- Valeur
- Source
- SourceID
- Span(début,fin)

Types :
raw_text, types libres

PyMedExt – Représentation d'un document et d'annotation

Le format PyMedExt

Document

Annotation
- ID
- Type
- Valeur
- Source
- SourceID
- Span(début,fin)

Une annotation plus qu'un token (e.g. une zone de texte)

Annotation
- ID
- Type
- Valeur
- Source
- SourceID
- Span(début,fin)

Hiérarchie d'annotations

Philosophie de PyMedExt pour les annotateurs :

Structure simple
- Héritage d'une classe python
- Deux fonctions à compléter

Deux possibilités pour la création de nouveaux annotateurs PyMedExt :
- Développement en python natif
- Encapsulation de codes existants

Ajouter un annotateur en UIMA

1 Type System holds CAS Feature Structure representation(s)

```xml
<typeSystemDescription xmlns="http://uima.apache.org/resourceSpecifier">
    <name>SethTypeSystem</name>
    <description>Type System Definition for SETH</description>
    <vendor>Apache Software Foundation</vendor>
    <types>
        <typeDescription>
            <name>de.dfki.lt.fimda.fimda.MutationAnnotation</name>
            <description>Mutation annotation created with SETH</description>
            <supertypeName>uima.tcas.Annotation</supertypeName>
            <features>
                <featureDescription>
                    <name>mtType</name>
                    <description>mutation type</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
                <featureDescription>
                    <name>wtResidue</name>
                    <description>wtResidue</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
                <featureDescription>
                    <name>mtResidue</name>
                    <description>mtResidue</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
                <featureDescription>
                    <name>mtPosition</name>
                    <description>mutation position</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
                <featureDescription>
                    <name>anTool</name>
                    <description>annotation tool</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
                <featureDescription>
                    <name>hgvs</name>
                    <description>hgvs</description>
                    <rangeTypeName>uima.cas.String</rangeTypeName>
                </featureDescription>
            </features>
        </typeDescription>
    </types>
</typeSystemDescription>
```

2 The Analysis Engine Descriptor

```xml
<!--  Descriptor for the MutationAnnotator. -->
<analysisEngineDescription xmlns="http://uima.apache.org/resourceSpecifier">
    <frameworkImplementation>org.apache.uima.java</frameworkImplementation>
    <primitive>true</primitive>
    <annotatorImplementationName>
        de.dfki.lt.fimda.fimda.MutationAnnotator
    </annotatorImplementationName>
    <analysisEngineMetaData>
        <name>SETH Mutation Annotator</name>
        <description>A mutation annotator using SETH (SNP Extraction Tool for Human Variations).</description>
        <version>1.0</version>
        <vendor>Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)</vendor>
        <typeSystemDescription>
            <imports>
                <import location="SethTypeSystem.xml"/>
            </imports>
        </typeSystemDescription>
        <capabilities>
            <capability>
                <inputs />
                <outputs>
                    <type allAnnotatorFeatures ="true">de.dfki.lt.fimda.fimda.MutationAnnotation</type>
                </outputs>
            </capability>
        </capabilities>
    </analysisEngineMetaData>
</analysisEngineDescription>
```

3 Generate Java classes

```xml
<dependency>
   <groupId>org.apache.uima</groupId>
   <artifactId>jcasgen-maven-plugin</artifactId>
   <version>3.0.0</version>
</dependency>
```

4 The Annotator Class

5 Calling the Annotator

```java
public class FIMDA {

    private AnalysisEngine ae;

    FIMDA() throws IOException, InvalidXMLException, ResourceInitializationException {
        //get Resource Specifier from XML file
        XMLInputSource in;
        URL url = getClass().getResource("/resources/desc/MutationAnnotator.xml");
        // important: use URL to create the XMLInputSource, otherwise it wont work from the jar package
        in = new XMLInputSource(url);
        ResourceSpecifier specifier = UIMAFramework.getXMLParser().parseResourceSpecifier(in);

        //create AE here
        ae = UIMAFramework.produceAnalysisEngine(specifier);
    }
```

Crédit : https://www.fosteropenscience.eu/learning/how-to-wrap-your-java-nlp-tool-in-an-uima-component/#/id/5af19b09a8216e322d135c55

Annotateurs PyMedExt

1

Définition de l'annotateur
(héritage de la classe annotateur)
Implémentation de la fonction *annotate_function*

```
class findMatches(annotators.Annotator):
    """
    Annotator based on linux grep to search regext from a source file
    """
    def __init__(self, key_input, key_output, ID, findValues ):
        """FIXME! initialize the annotator

        :param key_input: input ['raw_text']
        :param key_output: Annotation type here "Liposarcom.V0"
        :param ID: regex_fast.version
        :param findValues: "list of value to identify in the text"
        :returns:
        :rtype:

        """
        super().__init__(key_input, key_output, ID)
        self.findValues=findValues
```

```
def annotate_function(self, _input):
""" main annotation function
:param _input: in this case raw_text
:returns: a list of annotations
:rtype:
"""
logger.debug(_input)
inp = self.get_key_input(_input,0)[0]
annotationsList=[]
for thisValue in self.findValues:
    #result = [i.start() for i in re.finditer(thisValue, inp.value.lower())]
    for i in re.finditer(thisValue, inp.value.lower()):
        matchPos=i.start()
        if matchPos is not []:
            logger.debug("ok go in loop")
            logger.debug(matchPos)
            ID = str(uuid.uuid1())
            annotationsList.append(annotators.Annotation(type= self.key_output,
                                                value=thisValue, #thisMatch,
                                                span=(int(matchPos), int(matchPos)+len(thisValue)),
                                                source=self.ID,
                                                isEntity=True,
                                                ID=ID,
                                                source_ID = inp.ID))
        logger.debug(annotationsList)
return(annotationsList)
```

Didacticiel : https://github.com/equipe22/pymedext_core

ANNOTATEURS PYMEDEXT

1

Définition de l'annotateur
(héritage de la classe annotateur)
Implémentation de la fonction *annotate_function*

```
class findMatches(annotators.Annotator):
    """
    Annotator based on linux grep to search regext from a source file
    """
    def __init__(self, key_input, key_output, ID, findValues ):
        """FIXME! initialize the annotator

        :param key_input: input ['raw_text']
        :param key_output: Annotation type here "Liposarcom.V0"
        :param ID: regex_fast.version
        :param findValues: "list of value to identify in the text"
        :returns:
        :rtype:

        """
        super().__init__(key_input, key_output, ID)
        self.findValues=findValues
```

```
def annotate_function(self, _input):
""" main annotation function
:param _input: in this case raw_text
:returns: a list of annotations
:rtype:
"""
logger.debug(_input)
inp = self.get_key_input(_input,0)[0]
annotationsList=[]
for thisValue in self.findValues:
    #result = [i.start() for i in re.finditer(thisValue, inp.value.lower())]
    for i in re.finditer(thisValue, inp.value.lower()):
        matchPos=i.start()
        if matchPos is not []:
            logger.debug("ok go in loop")
            logger.debug(matchPos)
            ID = str(uuid.uuid1())
            annotationsList.append(annotators.Annotation(type= self.key_output,
                                                value=thisValue, #thisMatch,
                                                span=(int(matchPos), int(matchPos)+len(thisValue)),
                                                source=self.ID,
                                                isEntity=True,
                                                ID=ID,
                                                source_ID = inp.ID))
        logger.debug(annotationsList)
return(annotationsList)
```

Didacticiel : https://github.com/equipe22/pymedext_core

Annotateurs PyMedExt

1

```
class findMatches(annotators.Annotator):
    """
    Annotator based on linux grep to search regext from a source file
    """
    def __init__(self, key_input, key_output, ID, findValues ):
        """FIXME! initialize the annotator

        :param key_input: input ['raw_text']
        :param key_output: Annotation type here "Liposarcom.V0"
        :param ID: regex_fast.version
        :param findValues: "list of value to identify in the text"
        :returns:
        :rtype:

        """
        super().__init__(key_input, key_output, ID)
        self.findValues=findValues
```

```
def annotate_function(self, _input):
""" main annotation function
:param _input: in this case raw_text
:returns: a list of annotations
:rtype:
"""
logger.debug(_input)
inp = self.get_key_input(_input,0)[0]
annotationsList=[]
for thisValue in self.findValues:
    #result = [i.start() for i in re.finditer(thisValue, inp.value.lower())]
    for i in re.finditer(thisValue, inp.value.lower()):
        matchPos=i.start()
        if matchPos is not []:
            logger.debug("ok go in loop")
            logger.debug(matchPos)
            ID = str(uuid.uuid1())
            annotationsList.append(annotators.Annotation(type= self.key_output,
                                        value=thisValue, #thisMatch,
                                        span=(int(matchPos), int(matchPos)+len(thisValue)),
                                        source=self.ID,
                                        isEntity=True,
                                        ID=ID,
                                        source_ID = inp.ID))
        logger.debug(annotationsList)
return(annotationsList)
```

2

Appel de l'annotateur

```
demoAnnotator = findMatches(key_input = ['raw_text'],
                    key_output = 'Liposarcom.V0',
                    ID = "demoreiter", findValues = ["liposarcome"])

# add all your annotators in a list
annotatorsList =[demoAnnotator]
# annotate your document
LetterPyMedExt.annotate(annotatorsList)
```

Pipelines d'annotation simples

La gestion de pipelines complexes sera faite en utilisant des outils dédiés hors de PyMedExt

Didacticiel : https://github.com/equipeEZR/pymedext_core

Exemple d'annotateurs au format PyMedExt

Développement en python natif

- Regex
- Médicaments
- Necker/Imagine contexte

Encapsulation de codes existants

- QuickUMLS
- HeidelTime
- Romedi

Prochainement rendus disponibles
https://github.com/equipe22/pymedext_core

Conversion de formats

Conversions

- Des convertisseurs de formats
- Des exports vers des formats standards ou outils d'annotation

Entrées

PyMedExt

Text

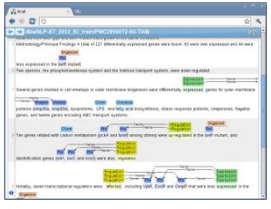
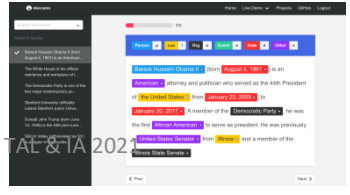
FHIR

BioC

Conversion de formats

Conversions

- Des convertisseurs de formats
- Des exports vers des formats standards ou outils d'annotation

Entrées		Sorties		
PyMedExt		PyMedExt-json		
Text				
FHIR	Standards informatique médicale	OMOP	OHDSI	
BioC				
	Annotation Visualisation	BRAT		
		Doccano	doccano	

Conversion de formats

Conversions

- Des convertisseurs de formats
- Des exports vers des formats standards ou outils d'annotation

Entrées

PyMedExt

Text

FHIR

BioC

Sorties

PyMedExt-json

Standards informatique médicale

OMOP

Annotation Visualisation

BRAT

Doccano

En cours de développement

BioC

CoNLL

FHIR

Export vers BRAT

1

```
class findMatches(annotators.Annotator):
    """
    Annotator based on linux grep to search regext from a source file
    """
    def __init__(self, key_input, key_output, ID, findValues ):
        """FIXME! initialize the annotator

        :param key_input: input ['raw_text']
        :param key_output: Annotation type here "Liposarcom.V0"
        :param ID: regex_fast.version
        :param findValues: "list of value to identify in the text"
        :returns:
        :rtype:

        """
        super().__init__(key_input, key_output, ID)
        self.findValues=findValues
```

```
def annotate_function(self, _input):
""" main annotation function
:param _input: in this case raw_text
:returns: a list of annotations
:rtype:
"""
logger.debug(_input)
inp = self.get_key_input(_input,0)[0]
annotationsList=[]
for thisValue in self.findValues:
    #result = [i.start() for i in re.finditer(thisValue, inp.value.lower())]
    for i in re.finditer(thisValue, inp.value.lower()):
        matchPos=i.start()
        if matchPos is not []:
            logger.debug("ok go in loop")
            logger.debug(matchPos)
            ID = str(uuid.uuid1())
            annotationsList.append(annotators.Annotation(type= self.key_output,
                                        value=thisValue, #thisMatch,
                                        span=(int(matchPos), int(matchPos)+len(thisValue)),
                                        source=self.ID,
                                        isEntity=True,
                                        ID=ID,
                                        source_ID = inp.ID))
        logger.debug(annotationsList)
return(annotationsList)
```

2

Appel de l'annotateur

```
demoAnnotator = findMatches(key_input = ['raw_text'],
                    key_output = 'Liposarcom.V0',
                    ID = "demoreiter", findValues = ["liposarcome"])

# add all your annotators in a list
annotatorsList =[demoAnnotator]
# annotate your document
LetterPyMedExt.annotate(annotatorsList)
```

```
pymedext.brat.savetobrat(LetterPyMedExt,path)
```

Didacticiel : https://github.com/equipeZZZ/pymedext_core

Conversion de formats – en ligne de commande

```
pymedext -i demo.txt --itype txt -otype pymedext

pymedext -i MEDLINE_train_bioc --itype biocjson -otype pymedext
```

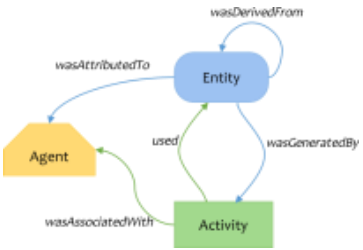
Format explicite

spaCy

spaCy
Large communauté d'utilisateurs
Optimisation code Python et Cython

Tokenisation (Hypothèse forte)

Travail en cours pour :
- intégrer des annotateurs spaCy en PyMedExt
- Rendre compatible les modèles de documents



Reproductibilité
- PROV-O ontologie pour l'expression de la provenance
- Travail sur une containerisation des outils externes et de PyMedExt

Conclusion

Projet récent

- Utilisation dans les hôpitaux Necker/Institut Imagine et HEGP
- Simplifie le partage de ressources
- Prise en main rapide

Encore du travail nécessaire, notamment pour éviter la redondance avec d'autres projets

Ressources distribuées sur le github du projet

Les testeuses et testeurs, contributrices et contributeurs *sont les bienvenues* !

PyMedExt en action :

- Prochaine présentation avec Antoine Neuraz !

https://github.com/equipe22/pymedext_core

Crédits Images

From the Noun Project

- data warehouse by remmachenasreddine
- Mining by Firza Alamsyah
- text file by Wing
- Electronic Health Record by Bold Yellow
- extraction by Eucalyp
- Hospital by ProSymbols