Nous utilisons plusieurs fonctions pour récupérer le contenu des corpus et traiter le texte. Nous utilisons la bibliothèque NLTK.

Nous « tokenisons » le texte et nous supprimons les mots qui ne sont pas « alphabétiques » afin de supprimer les titres de tableaux, les années, mots incorrects, etc…

Nous appliquons ensuite un lemmatizer sur les mots, puis nous supprimons les « stop words ». Ensuite nous utiliser la classe PorterStemmer() pour stem notre liste de mots. Nous retournons ensuite une liste sans duplicatas dans un dictionnaire (grâce à la clé de la classe dictionnaire).

Nous avons deux fonctions nous permettant de récupérer le contenu des corpus (utilisant des fonctions différentes également) :

- getStemWords() permet de récupérer le dictionnaire de tous les corpus mélangés, afin d'obtenir la liste réel de tous les mots présents dans nos corpus.
- getStemWordsByCorpus() nous permet de récupérer 10 listes, une pour chaque corpus, afin de fabriquer nos index plus tard.

Avec ces deux listes ainsi obtenus, nous pouvons avec la fonction getIncidenceMatrix() générer la matrice d'incidence.

La fonction getReversedIndex() nous permet de récupérer l'index inversé pour chaque corpus dans le dossier « corpus ».

```
getStemWords().index("develop")
300
getReversedIndex()[300]
['develop', 0, 1, 3, 4, 5, 8, 9]
getReversedIndex()[301]
['receiv', 0, 1, 2, 3, 4, 5, 7, 8]
```

La fonction booleanRequest() nous permet ensuite de faire des requêtes utilisant l'index / la matrice d'incidence. Nous utilisons de la récursivité et les requêtes fonctionnent entièrement.

```
------------
Request : disease AND severe AND pneumonia

['diseas', 'and', 'sever', 'and', 'pneumonia']
[[0, 1, 2, 3, 4, 5, 6, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 8, 9], 'and', [0, 2, 6, 8]]
[[0, 1, 2, 3, 4, 5, 6, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 8, 9], 'and', [0, 2, 6, 8]]
[0, 2, 6, 8]
------------
Request : antibody AND plasma AND (cells OR receptors)

['antibodi', 'and', 'plasma', 'and', '(', 'cell', 'or', 'receptor', ')']
[[0, 1, 2, 3, 4, 6, 8, 9], 'and', [0, 1, 3, 4, 8], 'and', '(', [0, 1, 3, 4, 8, 9], 'or', [0, 2, 4, 8, 9], ')']
[[0, 1, 2, 3, 4, 6, 8, 9], 'and', [0, 1, 3, 4, 8], 'and', [0, 1, 3, 4, 8, 9, 0, 2, 4, 8, 9]]
[0, 1, 3, 4, 8]
------------
Request : antimalarial drugs OR antiviral agents OR immunomodulators

['antimalari', 'drug', 'or', 'antivir', 'agent', 'or', 'immunomodul']
[[0, 4], 'and', [0, 1, 3, 4, 6, 8], 'or', [0, 2, 3, 4, 8], 'and', [0, 1, 2, 3, 4], 'or', [3]]
[[0, 4], 'and', [0, 1, 3, 4, 6, 8], 'or', [0, 2, 3, 4, 8], 'and', [0, 1, 2, 3, 4], 'or', [3]]
[0, 4, 2, 3]
------------
Request : NOT plasma AND risk of infection AND restrictions

['not', 'plasma', 'and', 'risk', 'of', 'infect', 'and', 'restrict']
['not', [0, 1, 3, 4, 8], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 6, 7, 8, 9], 'and', [0, 6, 7, 8, 9]]
['not', [0, 1, 3, 4, 8], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 6, 7, 8, 9], 'and', [0, 6, 7, 8, 9]]
[6, 7, 9]
------------
Request : (older adults AND antibodies) AND NOT (genomes OR variant)

['(', 'older', 'adult', 'and', 'antibodi', ')', 'and', 'not', '(', 'genom', 'or', 'variant', ')']
['(', [0, 1, 2, 3, 4, 5, 9], 'and', [0, 1, 5, 8, 9], 'and', [0, 1, 2, 3, 4, 6, 8, 9], ')', 'and', 'not', '(', [0], 'or', [0, 4], ')']
[[0, 1, 9], 'and', 'not', [0, 0, 4]]
[1, 9]
```

Les requêtes plus complexes sans logiques booléennes fonctionnent également :

```
------------
Request : efficacy and safety of the treatments

['efficaci', 'and', 'safeti', 'of', 'the', 'treatment']
[[1, 2, 3, 4, 5, 9], 'and', [0, 1, 2, 3, 4], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 6, 8, 9]]
[[1, 2, 3, 4, 5, 9], 'and', [0, 1, 2, 3, 4], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 'and', [0, 1, 2, 3, 4, 6, 8, 9]]
[1, 2, 3, 4]
```