201811330 🧑‍💻 اويس الحمود 🧑‍💻 جامعة القلمون

In [1]:

```
# define our Boolean operators in set
boolean_operators = {'AND', 'OR', 'AND_NOT','OR_NOT'}
```

In [2]:

```
# implement some helper functions below
# list of terms
def get_terms(data):
    terms=[ ]
    for doc in data:
        for term in data[doc].split() :
            terms.append(term)
    return terms
```

In [3]:

```
#define a function which return list of unique terms
def get_unique_terms(terms):
    unique_terms=[]
    for d in terms :
        if d not in unique_terms:
            unique_terms.append(d)
    return unique_terms
```

In [4]:

```
# define a function which return document collection terms
def get_document_collection_terms(data):
    docs_collection={}
    for doc in data:
        if doc not in boolean_operators :
            docs_collection[doc]=get_unique_terms(data[doc].split())
    return docs_collection
```

In [5]:

```
def display_dict(collection):
    for doc in collection:
        print(doc,' : ',collection[doc])
```

In [6]:

```
#Next, we will implement a function to build a term-document incidence matrix
#this function takes the collection of documents in a form of dictionary as an input
def get_incidence_matrix(collection):
    ## list of terms from the data file collection
    terms=get_terms(collection)
    #list of unique terms
    unique_terms=get_unique_terms(terms)
    #Document collection terms
    docs_terms=get_document_collection_terms(collection)
    #TermDocumentIncidenceMatrix
    term_docs_matrix={}
    for term in unique_terms:
        vector=[]
        for c in docs_terms:
            if term in docs_terms[c]:
                vector.append(1)
            else :
                vector.append(0)
        term_docs_matrix[term]=vector
    return term_docs_matrix
```

In [7]:

```
#this function takes a term and a terms-document incidence matrix and returns the incide
#this function just for explanation and display purposes
def get_incidence_vector(term,incidence_matrix):
    try:
        return incidence_matrix[term]
    except:
        return "term not found"
```

In [8]:

```
# First, we need to implement our function that splits the query to differentiate
# between the terms and the boolean operators  and considers only boolean operators
# defined previously and the terms available in our collection.
# input : Query
# output : List of terms of a given query which match with the terms in our collection
# and the our defined boolean operators
def query_filter(query,collection):
    terms= get_terms(collection)
    unique_terms=get_unique_terms(terms)
    query_terms=[]
    splitted_query=query.split()
    for temp in splitted_query:
        if temp in unique_terms or temp.upper() in boolean_operators:
            query_terms.append(temp)
    return query_terms
```

In [9]:

```python
# input : Boolean Operator ,Next term Incedence Vector ,Previous term Incedence Vector
def boolean_operator_processing(bool_operator,prevV,nextV):
    if bool_operator == "AND":
        return [a & b for a, b in zip(prevV, nextV)]
    elif bool_operator=="OR" :
        return [a | b for a, b in zip(prevV, nextV)]
    elif bool_operator == "AND_NOT":
        return [a & (1-b) for a, b in zip(prevV, nextV)]
    elif bool_operator == "OR_NOT":
        return [a | (1-b) for a, b in zip(prevV, nextV)]
```

In [10]:

```python
def boolean_retrieval(query,collection):
    print('MY Collection : ')
    display_dict(collection)
    print('MY Query : ',query)
    incidence_matrix=get_incidence_matrix(collection=collection)
    query_terms=query_filter(collection=collection,query=query)
    print('query terms : ',query_terms)
    res=[ 1 for i in collection]
    op='init'
    print('Vector terms of query : ')
    for term in query_terms:
        termU=term.upper()
        if termU in boolean_operators:
            op=termU
        else:
            vec=get_incidence_vector(term,incidence_matrix)
            print(term,' : ',vec)
            if op=='init':
                res=boolean_operator_processing('AND',res,vec)
            else:
                res=boolean_operator_processing(op,res,vec)
    result_collection={}
    i=0
    for key in collection:
        if res[i]==1:
            result_collection[key]=collection[key]
        i+=1
    print('Result Collection : ')
    display_dict(result_collection)
```

define collection from 5 document for test and simple query

In [11]:

```
d0 ='هذا هو الدرس الأول من مقرر استرجاع المعلومات'
d1 ='المقرر موجه لطلاب جامعة القلمون كلية الهندسة قسم تقانة المعلومات'
d2 ='هو المقرر رمز ENIT5517 '
d3 ='نأمل ان نقدم لكم الفائدة المرجوة'
d4 ='نرحب بأسئلتكم و استفساراتكم '
collection = {
"doc0": d0,
"doc1": d1,
"doc2": d2,
"doc3": d3,
"doc4": d4
}
boolean_retrieval(query='مقرر OR المعلومات OR استرجاع',collection=collection)
```

```
MY Collection :
doc0  :  هذا هو الدرس الأول من مقرر استرجاع المعلومات
doc1  :  المقرر موجه لطلاب جامعة القلمون كلية الهندسة قسم تقانة المعلومات
doc2  :  هو المقرر رمز ENIT5517
doc3  :  نأمل ان نقدم لكم الفائدة المرجوة
doc4  :  نرحب بأسئلتكم و استفساراتكم
MY Query :  مقرر OR المعلومات OR استرجاع
query terms :  ['مقرر', 'OR', 'المعلومات', 'OR', 'استرجاع']
Vector terms of query :
[0 ,0 ,0 ,0 ,1]  :  مقرر
[0 ,0 ,0 ,1 ,1]  :  المعلومات
[0 ,0 ,0 ,0 ,1]  :  استرجاع
Result Collection :
doc0  :  هذا هو الدرس الأول من مقرر استرجاع المعلومات
doc1  :  المقرر موجه لطلاب جامعة القلمون كلية الهندسة قسم تقانة المعلومات
```

In [12]:

```
d0 ='He likes to wink , he likes to drink'
d1 ='He likes to drink and drink and drink'
d2 ='The thing he likes to drink is ink'
d3 ='The ink he likes to drink is pink'
d4 ='He likes to wink and drink pink ink'
collection = {
"d0": d0,
"d1": d1,
"d2": d2,
"d3": d3,
"d4": d4
}
boolean_retrieval(query='wink OR pink',collection=collection)
```

```
MY Collection :
d0   :  He likes to wink , he likes to drink
d1   :  He likes to drink and drink and drink
d2   :  The thing he likes to drink is ink
d3   :  The ink he likes to drink is pink
d4   :  He likes to wink and drink pink ink
MY Query :  wink OR pink
query terms :  ['wink', 'OR', 'pink']
Vector terms of query :
wink  :  [1, 0, 0, 0, 1]
pink  :  [0, 0, 0, 1, 1]
Result Collection :
d0   :  He likes to wink , he likes to drink
d3   :  The ink he likes to drink is pink
d4   :  He likes to wink and drink pink ink
```