In [1]:

```python
import ast
import pandas as pd
import numpy as np
import csv
import json
import os
import pickle
```

Importing basic patent data

here some raw data: this is the official Patents database in the US.

In [2]:

```python
ipcr = pd.read_csv('raw_data/ipcr.tsv', sep='\t')
```

```
/Users/quentinciccarone/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3071: DtypeWarning
: Columns (4) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:

```python
ipcr.head()
```

Out[3]:

	uuid	patent_id	classification_level	section	ipc_class	subclass	main_group	subgroup	symbol_position	classification_value	classific
0	00005z3qh82fwpo5r1oupwpr3	6864832	NaN	G	01	S	013	/42	NaN	NaN	
1	0000662nssr53hdo3lp92sz26	9954111	A	H	01	L	27	1156	L	I	
2	00008u9j3g8oivqtuc1dqayb1	10048897	A	G	06	F	12	891	L	I	
3	00008v5gnw215cdjozwehxqky	10694566	A	H	4	W	4	0	F	I	
4	0000hj3ytmy8g9l2qa5x1hta5	D409748	NaN	D	24	04	NaN	NaN	NaN	NaN	

here a dataset allowing to make the link between the name of the startup and the id.

In [4]:

```python
assignee = pd.read_csv('raw_data/rawassignee.tsv', sep='\t')
```

In [5]:

```python
assignee.head()
```

Out[5]:

	uuid	patent_id	assignee_id	rawlocation_id	type	name_first	name_last	organization	sequence
0	0000c0e5-81d5-4a02-bb9e-d5c3ef596900	4488683	org_zzDG6gSOdiYZdFsxQuQR	b44f6bf0-1f14-4b25-9ab6-06945ff1e8e1	3.0	NaN	NaN	Metal Works Ramat David	0
1	0000p94wkezw94s8cz7dbxlvz	5856666	org_fBtpUrdoVp5Lzvqma3Lv	orskbf54s58e97lkmw8na5rpx	2.0	NaN	NaN	U.S. Philips Corporation	0
2	00013vk881wap9u4mbo7lwwhp	5204210	org_uBBq49OpEQSGb2SJJBBT	mue862v5lcjdhzqqk86ei75kj	2.0	NaN	NaN	Xerox Corporation	0
3	000192sn2u10kzpikl4s7h3r0	5302149	org_ObwJtJBwY2t1dFv6oAAj	o1h9dqdv0yq7dt1b1vmrcal9h	3.0	NaN	NaN	Commonwealth Scientific & Industrial Research ...	1
4	0001ce9b-3c22-4f3a-b621-6c4e3ed85792	D397841	org_q4Ruupmeytq4y41ZiXdM	83c2755a-df62-4f4f-8509-43b9dcfeb038	3.0	NaN	NaN	adidas, AG	0

Canonical dataset

Here is simply the list of patents, with the id of the firm that put it in place, as well as the "action date" of the patent.

In [8]:

```python
assignee_0 = assignee[['patent_id','assignee_id','organization']]

# list of organisation_id/true_name_organization
assignee_1 = assignee[['assignee_id','organization']]
assignee_1['organization'] = assignee_1['organization'].str.upper()

# list of  patent_id / organisation_id
assignee_2 = assignee[['patent_id','assignee_id']]
```

```
<ipython-input-8-8782019bf8df>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  assignee_1['organization'] = assignee_1['organization'].str.upper()
```

In [9]:

```python
ipcr_1 = ipcr[['patent_id','ipc_version_indicator']]

# conversion en date
ipcr_1['ipc_version_indicator']=pd.to_datetime(ipcr_1['ipc_version_indicator'],format='%Y-%m-%d', errors='ignore')

# filtrage pour enlever les patent qui n'ont pas de date
ipcr_1= ipcr_1[ipcr_1['ipc_version_indicator']>pd.datetime(1800,1,1)]

ipcr_1 = ipcr_1.set_index('patent_id')

# filtrage qui permet d'enlever les patents en double
ipcr_1 = ipcr_1[~ipcr_1.index.duplicated(keep='first')]
```

```
<ipython-input-9-8d40f726771a>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  ipcr_1['ipc_version_indicator']=pd.to_datetime(ipcr_1['ipc_version_indicator'],format='%Y-%m-%d', errors='ignore')
<ipython-input-9-8d40f726771a>:7: FutureWarning: The pandas.datetime class is deprecated and will be removed from pan
das in a future version. Import from datetime module instead.
  ipcr_1= ipcr_1[ipcr_1['ipc_version_indicator']>pd.datetime(1800,1,1)]
```

Here is the first reference dataset (only to name the companies, we use the organization_id for the code to avoid errors as much as possible).

In [17]:

```python
# canonique_1 = assignee_1
# pickle.dump( canonique_1, open( "canonique_1.p", "wb" ) )
```

In [16]:

```python
with open("canonique_1.p", "rb") as cano1:
    canonique_1 = pickle.load(cano1)
canonique_1.head()
```

Out[16]:

	assignee_id	organization
0	org_zzDG6gSOdiYZdFsxQuQR	METAL WORKS RAMAT DAVID
1	org_fBtpUrdoVp5Lzvqma3Lv	U.S. PHILIPS CORPORATION
2	org_uBBq49OpEQSGb2SJJBBT	XEROX CORPORATION
3	org_ObwJtJBwY2t1dFv6oAAj	COMMONWEALTH SCIENTIFIC & INDUSTRIAL RESEARCH ...
4	org_q4Ruupmeytq4y41ZiXdM	ADIDAS, AG

Here is the second dataset of reference, which allows to link the id_patent, their action_date and the id_organization

In [15]:

```python
# canonique_2 = pd.merge(ipcr_1,assignee_2, how='inner',on = 'patent_id')
# pickle.dump( canonique_2, open( "canonique_2.p", "wb" ) )
```

In [18]:

```python
with open("canonique_2.p", "rb") as cano2:
    canonique_2 = pickle.load(cano2)
canonique_2.head()
```

Out[18]:

	patent_id	ipc_version_indicator	assignee_id
0	9954111	2017-01-01	org_wYgU2Jhd2MTGAna87vpi
1	10048897	2016-01-01	org_ONzMjdbZXiKfw4L0cXl6
2	10694566	2018-01-01	org_dfvulWENawcU6lTd1Z3w
3	7645556	2006-01-01	org_pCbqlmAg8wlWzoi18lTD
4	8524174	2006-01-01	org_7P7texQL0Q4WvpEEvDg8

We can see here that according to our dataset, the company with the most patents is IBM with 97 202 patents.

In [19]:

```python
canonique_2.groupby(by='assignee_id').count().sort_values(by=['patent_id'],ascending = False).head()
```

Out[19]:

	patent_id	ipc_version_indicator
assignee_id		
org_ONzMjdbZXiKfw4L0cXl6	97202	97202
org_pCbqlmAg8wlWzoi18lTD	68914	68914
org_eAKK85fawH0NS7AdXOig	44778	44778
org_g8U335TH48QmGJOIQnNI	32299	32299
org_OrmhECOcsM3rq5b7Pxfe	31143	31143

In [20]:

```python
canonique_1[canonique_1['assignee_id']=='org_ONzMjdbZXiKfw4L0cXl6'].head()
```

Out[20]:

	assignee_id	organization
81	org_ONzMjdbZXiKfw4L0cXl6	INTERNATIONAL BUSINESS MACHINES CORPORATION
124	org_ONzMjdbZXiKfw4L0cXl6	INTERNATIONAL BUSINESS MACHINES CORPORATION
127	org_ONzMjdbZXiKfw4L0cXl6	INTERNATIONAL BUSINESS MACHINES CORPORATION
166	org_ONzMjdbZXiKfw4L0cXl6	INTERNATIONAL BUSINESS MACHINES CORPORATION
171	org_ONzMjdbZXiKfw4L0cXl6	INTERNATIONAL BUSINESS MACHINES CORPORATION

Importing M&A of startups data

The idea is to see, empirically, the importance of patents in Tech M&A processes. source: crunchbase.

In [10]:

```python
crunchbase_1 = pd.read_csv("/Volumes/Samsung_T5/patent_similarity_data/CRUNCHBASE_EXPORT/acquisitions_private_hardware
_us.csv")
```

Here, a small script that gives the number of mergers and acquisitions between a private company that is being acquired in the US and another company, both companies having patents in the US. Here, for every 100 acquisitions, 31 acquisitions were made between two US companies, with the target being a private unlisted company, both companies having at least one patent each.

In [11]:

```python
n = 0
firm_list = []
for k in range(100):
    acquiree = list(crunchbase_1['Acquiree Name'])[k]
    acquirer = list(crunchbase_1['Acquirer Name'])[k]

    has_aquiree_patent = 0
    has_aquirer_patent = 0

    has_aquiree_patent = sum(set(assignee_1['organization'].str.contains(acquiree.upper(), regex=True, na=False)))
    has_aquirer_patent = sum(set(assignee_1['organization'].str.contains(acquirer.upper(), regex=True, na=False)))

    value = min(has_aquiree_patent,has_aquirer_patent)
    n += value
    if value ==1:
        firm_list.append([acquiree.upper(),acquirer.upper()])
```

```
/Users/quentinciccarone/opt/anaconda3/lib/python3.8/site-packages/pandas/core/strings.py:1954: UserWarning: This patt
ern has match groups. To actually get the groups, use str.extract.
  return func(self, *args, **kwargs)
```

Of the last 100 startup M&A deals reported on Crunchbase at that time, 31 were between two companies each having at least 1 patent :

In [12]:

```python
n
```

Out[12]: 31

The list of the deals, with the target on the left and the company acquiring the target on the right:

In [13]:

```python
firm_list
```

Out[13]:

```
[['CGTECH', 'SANDVIK COROMANT'],
 ['KASTEN', 'VEEAM SOFTWARE'],
 ['BOX ROBOTICS', 'SEEGRID'],
 ['BOLT SOFTWARE', 'ECI SOFTWARE SOLUTIONS'],
 ['NEVION', 'SONY'],
 ['ARCHER SOFTWARE', 'CPRIME'],
 ['AUTOCRIB', 'SNAP-ON'],
 ['PACIFIC STAR COMMUNICATIONS', 'CURTISS-WRIGHT'],
 ['CLOUDISTICS', 'FUNGIBLE'],
 ['ZENIMAX', 'MICROSOFT'],
 ['CRADLEPOINT', 'ERICSSON'],
 ['PORTWORX', 'PURE STORAGE'],
 ['UNIVA', 'ALTAIR ENGINEERING'],
 ['TRACFONE WIRELESS', 'VERIZON COMMUNICATIONS'],
 ['ULC ROBOTICS', 'SPX CORPORATION'],
 ['HIGHFIVE', 'DIALPAD'],
 ['SPACES', 'APPLE'],
 ['PRECO', 'SENSATA TECHNOLOGIES'],
 ['STOCKWELL', '365 RETAIL MARKETS'],
 ['ALG', 'J.D. POWER AND ASSOCIATES'],
 ['PELCO', 'MOTOROLA SOLUTIONS'],
 ['PYPE', 'AUTODESK'],
 ['DISPLAYLINK', 'SYNAPTICS'],
 ['FLO TECHNOLOGIES', 'MOEN INCORPORATED'],
 ['SILVER PEAK', 'HEWLETT PACKARD ENTERPRISE'],
 ['ELECTRIC IMP', 'TWILIO'],
 ['NET IRRIGATE', 'LINDSAY CORPORATION'],
 ['DATRIUM', 'VMWARE'],
 ['MARBLE', 'CATERPILLAR'],
 ['ZOOX', 'AMAZON'],
 ['ASTRO STUDIOS', 'PA CONSULTING GROUP']]
```

The idea now is to determine the similarity between the patents on both sides of the deals, the similarity between the purchaser's patents and those of the target. To do this, it is necessary to use the pre-established database with the similarity already calculated, or to determine the similarity between all the patents "by hand" using NLP processes. For a first approach, we will use the already calculated database, so we have to extract the good patents because the file is very heavy (20Go) and I can't put it in flash memories. For that, a simple method is to browse the JSon without putting it in the RAM (JSon stream), that increases the search time, but it works :

In [29]:

```python
infile = open('/Volumes/Samsung_T5/patent_similarity_data/raw_data/most_sim.json','r',encoding='utf-8')
```

In [30]:

```python
def similarity_patents_betw_2_firme(firms):
    acquiree = firms[0]
    acquirer = firms[1]

    assignee_id_acquiree = assignee_1[assignee_1['organization'].str.contains(acquiree, regex=True, na=False)]['assign
ee_id'].iloc[0]
    assignee_id_acquirer = assignee_1[assignee_1['organization'].str.contains(acquirer, regex=True, na=False)]['assign
ee_id'].iloc[0]

    patents_acquiree = set(assignee_2[assignee_2['assignee_id']==assignee_id_acquiree]["patent_id"])
    patents_acquirer = set(assignee_2[assignee_2['assignee_id']==assignee_id_acquirer]["patent_id"])


    similarity_acquiree_dico = {}
    similarity_acquirer_dico = {}

    for sim in infile:
        json_load = json.loads(sim)
        if json_load[0] in patents_acquiree:
            similarity_acquiree_dico[json_load[0]] = json_load[1]
        if json_load[0] in patents_acquirer:
            similarity_acquirer_dico[json_load[0]] = json_load[1]

    return(similarity_acquiree_dico,similarity_acquirer_dico)
```

result_similarity = {} for aree_arer in firm_list: sim_aree_arer = similarity_patents_betw_2_firme(aree_arer) patent_acquiree = set(sim_aree_arer[0].keys()) patent_acquirer = set(sim_aree_arer[1].keys()) for patent in patent_acquiree: print(patent) temp_dataframe = pd.DataFrame(sim_aree_arer[0][patent],columns=['patent_id','sim']) temp_dataframe = temp_dataframe[temp_dataframe['patent_id'].isin(patent_acquirer)] if len(temp_dataframe)>0: result_similarity[aree_arer[0]] = max(result_similarity.get(aree_arer[0],0),temp_dataframe['sim'].max()) else: result_similarity[aree_arer[0]] = result_similarity.get(aree_arer[0],0) print(len(temp_dataframe))

In [18]:

```python
# pickle.dump( result_similarity, open( "result_similarity_test.p", "wb" ) )
```

In [23]:

```python
with open("result_similarity_test.p", "rb") as result_sim:
    result_similarity = pickle.load(result_sim)
```

In [24]:

```python
result_similarity_final_useful = {}
for firm in result_similarity.keys():
    if result_similarity[firm]>0:
        result_similarity_final_useful[firm] = result_similarity[firm]
```

Here a dictionnary with keys corresponding to the targets and value the maximum similarity found between a patent from the target and a patent from the acquirer. I keep only the results when the similarity is above 0 (the 20Go database only contains the first 100 most similar patents for each patents, a patent is considered too remote according to the algorithm if it is not present in this list of 100 patents, and its similarity is therefore arbitrarily assigned to 0).

In [25]:

```python
result_similarity_final_useful
```

Out[25]:

```
{'NEVION': 0.33737462759017944,
 'ZENIMAX': 0.4011276662349701,
 'CRADLEPOINT': 0.3633490800857544,
 'SPACES': 0.41777321696281433,
 'PELCO': 0.3618326187133789,
 'DISPLAYLINK': 0.3821371793746948,
 'SILVER PEAK': 0.4321938157081604,
 'DATRIUM': 0.4869513511657715,
 'ZOOX': 0.4608118534088135}
```

In [83]:

```python
len(result_similarity_final_useful) / 100
```

Out[83]: 0.09

Out of 100 mergers, 31 are carried out between two companies each having at least one patent, and 9 are carried out between two companies having at least one patent sufficiently close. Out of this first sample of 100 mergers, we therefore have a model relevance for about 9% of the total mergers, which is huge. This would mean that 9% of the mergers and acquisitions notified on CrunchBase are relevant for the study. It's going to be fun.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Évaluons la pertinence du modèle sur un exemple de fusion / acquisition : DATRIUM et VMWARE.

In [31]:

```python
Dat_VMARE = similarity_patents_betw_2_firme(['DATRIUM', 'VMWARE'])
```

In [34]:

```python
datrium_id = assignee_1[assignee_1['organization'].str.contains('DATRIUM', regex=True, na=False)]['assignee_id'].iloc[
0]
datrium_id
```

Out[34]: 'org_NsPj2a4iD80LdYxRz2kT'

In [40]:

```python
patent_acquiree = Dat_VMARE[0].keys()
```

In [48]:

```python
for patent in patent_acquiree:
    print(patent)
    temp_dataframe = pd.DataFrame(Dat_VMARE[0][str(patent)],columns=['patent_id','sim'])
    temp_filter = list(temp_dataframe['patent_id'].isin(patent_acquiree))

# to remove the patents from the acquiree company, not to false the results
    temp_filter = [not boolean for boolean in temp_filter]
    temp_dataframe = temp_dataframe[temp_filter]

# print the maximum similarity between the acquiree patent and the other patents which exist in the database
    print(temp_dataframe['sim'].max())
```

```
10061706
0.4604795575141907
10089013
0.5034605860710144
10146684
0.45062124729156494
10180948
0.4766446352005005
10235044
0.5269159078598022
10359945
0.46825987100601196
9417955
0.44092288613319397
9639268
0.4494040012359619
9703490
0.4483810067176819
```

On remarque que la similarité entre entreprise A et B = max(similarity(Ai,Bj)) ou les Ai et les Bj sont tous les patents de A et de B est un peu trop fort comme classification, car il se peut qu'un patent soit très proche d'un autre patent de l'entreprise mais seulement un seul. On peut supposer que si A et B on plusieurs patents, l'idée est de faire un classement de la similarité entre entreprise en pondérant la similarité de patents : A est plus proche de C que B si A a beaucoup de patents proches de C même si B a un patent très proche de A, car B n'a seulement qu'un patent proche, alors que C en a plusieurs Il reste à trouver la fonction optimisant similarité et classement dans les prévisions de fusion entre startup et entreprise. On peut ici faire appel à un premier algorithme de régression linéaire pour déterminer une équation minimisant l'erreur dans la prévision de fusion (algorithme mettant en tête de classement des entreprises similaires de la startup l'entreprise acheteuse de la startup).

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: