In [1]:

```
# scraping the pdf files to obtain the url
```

In [36]:

```
import requests
import pandas as pd
import urllib.request
import pdftotext
import promptlib #pip install prompt
import os
import glob
#conda install poppler : conda install -c conda-forge poppler
# conda install pdftotext : conda install -c conda-forge pdftotext
# conda install nltk : conda install -c anaconda nltk
from nltk import *
from nltk.corpus import *
import nltk
import sys
from sklearn.metrics.pairwise import cosine_similarity
#import the TfidfVectorizer from Scikit-Learn.
from sklearn.feature_extraction.text import TfidfVectorizer
# Import cosine_similarity
from sklearn.metrics.pairwise import linear_kernel
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [7]:

```
#get download.csv by going on https://tel.archives-ouvertes.fr/search/advanced-export/u
#then export csv
df = pd.read_csv("H:/Downloads/Datatsets/Tel/tel_docs.csv")
```

In [8]:

```
df.head()
```

Out[8]:

	halId_s	version_i	uri_s	docType_s	doiId_s	nntId_s	
0	tel-03440243	1	https://hal.univ-lorraine.fr/tel-03440243	THESE	NaN	2021LORR0152	L'impa sys d'infor hosp
1	tel-03440181	1	https://tel.archives-ouvertes.fr/tel-03440181	THESE	NaN	2021UPASG065	Algorithmes critères prédi
2	tel-03440058	1	https://tel.archives-ouvertes.fr/tel-03440058	THESE	NaN	2021NORMR027	La microfina ap microentrepre
3	tel-03439538	1	https://pastel.archives-ouvertes.fr/tel-03439538	THESE	NaN	2020IAVF0016	Importance d domesticatic la
4	tel-03439366	1	https://hal.archives-ouvertes.fr/tel-03439366	THESE	NaN	NaN	Homom Cryptograp Privacy,Cryp

5 rows × 21 columns

In [10]:

```
#convert urls to list
list_urls=df["uri_s"].values.tolist()
```

In [11]:

```
#convert ids to list
list_ids=df["halId_s"].values.tolist()
```

In [13]:

```
list_ids[:10]
```

Out[13]:

```
['tel-03440243',
 'tel-03440181',
 'tel-03440058',
 'tel-03439538',
 'tel-03439366',
 'tel-03439358',
 'tel-03439354',
 'tel-03439346',
 'tel-03439261',
 'tel-03438938']
```

list_urls

In [8]:

```
#we need 50 urls, so we just subset the first 50
```

In [7]:

```
#subset 50 ids from list
list_ids = list_ids[:50]
```

In [9]:

```
count=0
#first 50 urls
for i in list_urls[:50]:

    url = i +"/document"
    #download pdfs
    urllib.request.urlretrieve(url, "{}.pdf".format(list_ids[count]))
    print(count,end="\r")
    count +=1
```

Convert pdf to text files

In [15]:

```
extension = "pdf"
#Prompt directory in which all files were downloaded,
prompter = promptlib.Files()
#Set working directory to chosen directory
dir = prompter.dir()
os.chdir(dir)
#Get list of all pdf files in directory
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
```

In [34]:

```
#downloaded excess files so subset 50 out of the list
all_filenames=all_filenames[5:55]
```

In [41]:

```
for file in all_filenames:
    with open(file, "rb") as f:
        pdf = pdftotext.PDF(f)
    # Save all text to a txt file.
    split_string = file.split(".", 1)
    substring = split_string[0]
    with open('{}'.format(substring+".txt"), 'w',encoding="utf-8") as f:
        f.write("\n\n".join(pdf))
```

In [41]:

```
for file in all_filenames:
    with open(file, "rb") as f:
        pdf = pdftotext.PDF(f)
    # Save all text to a txt file.
    split_string = file.split(".", 1)
    substring = split_string[0]
    with open('{}'.format(substring+".txt"), 'w',encoding="utf-8") as f:
        f.write("\n\n".join(pdf))
```

LANGUAGE DETECTION

In [46]:

```
#download stopwords from ntlk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[46]:

```
True
```

In [47]:

```
stopwords.fileids()
```

Out[47]:

```
['arabic',
 'azerbaijani',
 'bengali',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'greek',
 'hungarian',
 'indonesian',
 'italian',
 'kazakh',
 'nepali',
 'norwegian',
 'portuguese',
 'romanian',
 'russian',
 'slovene',
 'spanish',
 'swedish',
 'tajik',
 'turkish']
```

In [13]:

```
# reference  : https://blog.alejandronolla.com/2013/05/15/detecting-text-language-with-

try:
    from nltk import wordpunct_tokenize
    from nltk.corpus import stopwords
except ImportError:
    print ('[!] You need to install nltk (http://nltk.org/index.html)')



#-----------------------------------------------------------------------
def _calculate_languages_ratios(text):
    """
    Calculate probability of given text to be written in several languages and
    return a dictionary that looks like {'french': 2, 'spanish': 4, 'english': 0}

    @param text: Text whose language want to be detected
    @type text: str

    @return: Dictionary with languages and unique stopwords seen in analyzed text
    @rtype: dict
    """

    languages_ratios = {}

    '''
    nltk.wordpunct_tokenize() splits all punctuations into separate tokens

    >>> wordpunct_tokenize("That's thirty minutes away. I'll be there in ten.")
    ['That', "'", 's', 'thirty', 'minutes', 'away', '.', 'I', "'", 'll', 'be', 'there',
    '''

    tokens = wordpunct_tokenize(text)
    words = [word.lower() for word in tokens]

    # Compute per language included in nltk number of unique stopwords appearing in ana
    for language in stopwords.fileids():
        stopwords_set = set(stopwords.words(language))
        words_set = set(words)
        common_elements = words_set.intersection(stopwords_set)

        languages_ratios[language] = len(common_elements) # language "score"

    return languages_ratios


#-----------------------------------------------------------------------
def detect_language(text):
    """
    Calculate probability of given text to be written in several languages and
    return the highest scored.

    It uses a stopwords based approach, counting how many unique stopwords
    are seen in analyzed text.

    @param text: Text whose language want to be detected
    @type text: str

    @return: Most scored language guessed
```

```
    @rtype: str
    """

    ratios = _calculate_languages_ratios(text)

    most_rated_language = max(ratios, key=ratios.get)

    return most_rated_language
```

In [63]:

```
# Apply to our text files
```

In [14]:

```
extension = "txt"
#Prompt directory in which all files were downloaded,
prompter = promptlib.Files()
#Set working directory to chosen directory
dir = prompter.dir()
os.chdir(dir)
#Get list of all pdf files
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
```

In [15]:

```
all_filenames[:10]
```

Out[15]:

```
['tel-03435883.txt',
 'tel-03435884.txt',
 'tel-03435885.txt',
 'tel-03435936.txt',
 'tel-03436011.txt',
 'tel-03436023.txt',
 'tel-03436024.txt',
 'tel-03436025.txt',
 'tel-03436087.txt',
 'tel-03436137.txt']
```

In [15]:

```
#detect languages in 50 theses
for file in all_filenames:
    with open(file, "r",encoding="utf-8") as f:
        #read file as text and store in variable `text`
        text = f.read()
        split_string = file.split(".", 1)
        substring = split_string[0]
        language = detect_language(text)
        print("Thesis ID :{}".format(substring) +" Language : {}".format(language))

```

```
Thesis ID :tel-03435883 Language : english
Thesis ID :tel-03435884 Language : english
Thesis ID :tel-03435885 Language : english
Thesis ID :tel-03435936 Language : english
Thesis ID :tel-03436011 Language : french
Thesis ID :tel-03436023 Language : english
Thesis ID :tel-03436024 Language : english
Thesis ID :tel-03436025 Language : french
Thesis ID :tel-03436087 Language : english
Thesis ID :tel-03436137 Language : english
Thesis ID :tel-03436157 Language : english
Thesis ID :tel-03436173 Language : english
Thesis ID :tel-03436335 Language : french
Thesis ID :tel-03436364 Language : french
Thesis ID :tel-03436368 Language : french
Thesis ID :tel-03436372 Language : french
Thesis ID :tel-03436394 Language : romanian
Thesis ID :tel-03436405 Language : french
Thesis ID :tel-03436409 Language : english
Thesis ID :tel-03436501 Language : english
Thesis ID :tel-03436527 Language : english
Thesis ID :tel-03436530 Language : english
Thesis ID :tel-03436542 Language : french
Thesis ID :tel-03436545 Language : french
Thesis ID :tel-03436548 Language : french
Thesis ID :tel-03436551 Language : french
Thesis ID :tel-03437063 Language : english
Thesis ID :tel-03437096 Language : english
Thesis ID :tel-03437282 Language : english
Thesis ID :tel-03437572 Language : english
Thesis ID :tel-03437573 Language : english
Thesis ID :tel-03437616 Language : french
Thesis ID :tel-03438100 Language : portuguese
Thesis ID :tel-03438101 Language : french
Thesis ID :tel-03438102 Language : french
Thesis ID :tel-03438103 Language : english
Thesis ID :tel-03438104 Language : english
Thesis ID :tel-03438105 Language : french
Thesis ID :tel-03438755 Language : english
Thesis ID :tel-03438811 Language : portuguese
Thesis ID :tel-03438828 Language : english
Thesis ID :tel-03438829 Language : portuguese
Thesis ID :tel-03438863 Language : english
Thesis ID :tel-03438921 Language : english
Thesis ID :tel-03438923 Language : french
Thesis ID :tel-03438925 Language : english
Thesis ID :tel-03438938 Language : french
Thesis ID :tel-03439261 Language : english
```

```
Thesis ID :tel-03439346 Language : french
Thesis ID :tel-03439354 Language : english
```

Process the data and use TF-IDF and cosine to assess similarity between the documents

In [16]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
pd.set_option("max_rows", 600)
from pathlib import Path
import glob
```

In [17]:

```
directory_path = "H:\Downloads\Datatsets\Tel\Tel_text"
text_files = glob.glob(f"{directory_path}/*.txt")

```

In [18]:

```
text_files [:10]
```

Out[18]:

```
['H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03435883.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03435884.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03435885.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03435936.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436011.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436023.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436024.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436025.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436087.txt',
 'H:\\Downloads\\Datatsets\\Tel\\Tel_text\\tel-03436137.txt']
```

In [19]:

```
text_titles = [Path(text).stem for text in text_files]

```

In [20]:

```
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop
from spacy.lang.en.stop_words import STOP_WORDS as en_stop
from spacy.lang.ro.stop_words import STOP_WORDS as ro_stop
from spacy.lang.pt.stop_words import STOP_WORDS as pt_stop

#use stopwords from spacy
final_stopwords_list = list(fr_stop) + list(en_stop) + list(ro_stop) +list(pt_stop)
# from nltk.corpus import stopwords

# final_stopwords_list = stopwords.words('english') + stopwords.words('french') + stopw
```

In [21]:

```
#instantiate vectorizer with stopwords as paramter
tfidf_vectorizer = TfidfVectorizer(input='filename', stop_words=final_stopwords_list)

```

In [22]:

```
#fit vectorizer on all text files
tfidf_vector = tfidf_vectorizer.fit_transform(text_files)

```

In [23]:

```
#Make a DataFrame out of the resulting tf-idf vector, row names as index
tfidf_df = pd.DataFrame(tfidf_vector.toarray(), index=text_titles, columns=tfidf_vector

```

In [24]:

```
#Add column for document frequency aka number of times word appears in all documents

tfidf_df.loc['00_Document Frequency'] = (tfidf_df > 0).sum()

```

In [25]:

```
# drop “OO_Document Frequency” since we were just using it for illustration purposes.
tfidf_df = tfidf_df.drop('00_Document Frequency', errors='ignore')

```

In [26]:

```
#reorganize the DataFrame so that the words are in rows rather than columns.

tfidf_df.stack().reset_index()
tfidf_df = tfidf_df.stack().reset_index()

```

In [27]:

```
#rename columns with tfidf, document, term
tfidf_df = tfidf_df.rename(columns={0:'tfidf', 'level_0': 'document','level_1': 'term',

```

In [28]:

```
#sort values such that top 5 tf-idf of every document is obtained
tfidf_df=tfidf_df.sort_values(by=['document','tfidf'], ascending=[True,False]).groupby(

```

In [29]:

```
1 tfidf_df.head(10)
```

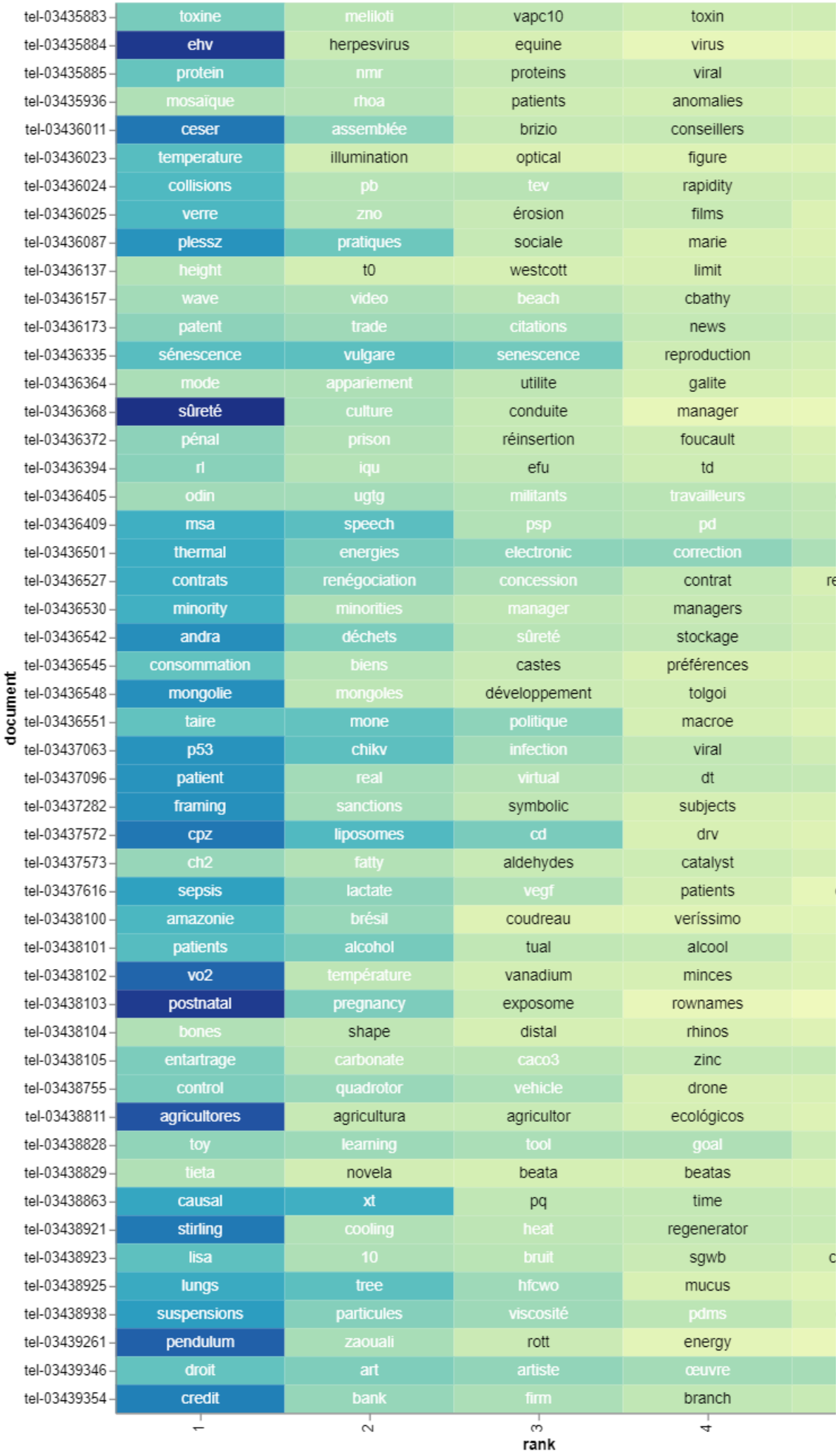
Out[29]:

	document	term	tfidf
162861	tel-03435883	toxine	0.348011
108520	tel-03435883	meliloti	0.240458
168878	tel-03435883	vapc10	0.223557
162860	tel-03435883	toxin	0.198698
168876	tel-03435883	vapc	0.196669
242882	tel-03435884	ehv	0.755768
263575	tel-03435884	herpesvirus	0.208099
247930	tel-03435884	equine	0.178468
350571	tel-03435884	virus	0.126986
264757	tel-03435884	horse	0.124838

In [30]:

```
import altair as alt
import numpy as np
term_list=[]

# adding a little randomness to break ties in term ranking
top_tfidf_plusRand = tfidf_df.copy()
top_tfidf_plusRand['tfidf'] = top_tfidf_plusRand['tfidf'] + np.random.rand(tfidf_df.sha

# base for all visualizations, with rank calculation
base = alt.Chart(top_tfidf_plusRand).encode(
    x = 'rank:O',
    y = 'document:N'
).transform_window(
    rank = "rank()",
    sort = [alt.SortField("tfidf", order="descending")],
    groupby = ["document"],
)

# heatmap specification
heatmap = base.mark_rect().encode(
    color = 'tfidf:Q'
)

# red circle over terms in above list
circle = base.mark_circle(size=100).encode(
    color = alt.condition(
        alt.FieldOneOfPredicate(field='term', oneOf=term_list),
        alt.value('red'),
        alt.value('#FFFFFF00')
    )
)

# text labels, white for darker heatmap colors
text = base.mark_text(baseline='middle').encode(
    text = 'term:N',
    color = alt.condition(alt.datum.tfidf >= 0.23, alt.value('white'), alt.value('black
)

# display the three superimposed visualizations
(heatmap + circle + text).properties(width = 600)
```

Out[30]:


document
tel-03435883 toxine meliloti vapc10 toxin
tel-03435884 ehv herpesvirus equine virus
tel-03435885 protein nmr proteins viral
tel-03435936 mosaïque rhoa patients anomalies
tel-03436011 ceser assemblée brizio conseillers
tel-03436023 temperature illumination optical figure
tel-03436024 collisions pb tev rapidity
tel-03436025 verre zno érosion films
tel-03436087 plessz pratiques sociale marie
tel-03436137 height t0 westcott limit
tel-03436157 wave video beach cbathy
tel-03436173 patent trade citations news
tel-03436335 sénescence vulgare senescence reproduction
tel-03436364 mode appariement utilite galite
tel-03436368 sûreté culture conduite manager
tel-03436372 pénal prison réinsertion foucault
tel-03436394 rl iqu efu td
tel-03436405 odin ugtg militants travailleurs
tel-03436409 msa speech psp pd
tel-03436501 thermal energies electronic correction
tel-03436527 contrats renégociation concession contrat
tel-03436530 minority minorities manager managers
tel-03436542 andra déchets sûreté stockage
tel-03436545 consommation biens castes préférences
tel-03436548 mongolie mongoles développement tolgoi
tel-03436551 taire mone politique macroe
tel-03437063 p53 chikv infection viral
tel-03437096 patient real virtual dt
tel-03437282 framing sanctions symbolic subjects
tel-03437572 cpz liposomes cd drv
tel-03437573 ch2 fatty aldehydes catalyst
tel-03437616 sepsis lactate vegf patients
tel-03438100 amazonie brésil coudreau veríssimo
tel-03438101 patients alcohol tual alcool
tel-03438102 vo2 température vanadium minces
tel-03438103 postnatal pregnancy exposome rownames
tel-03438104 bones shape distal rhinos
tel-03438105 entartrage carbonate caco3 zinc
tel-03438755 control quadrotor vehicle drone
tel-03438811 agricultores agricultura agricultor ecológicos
tel-03438828 toy learning tool goal
tel-03438829 tieta novela beata beatas
tel-03438863 causal xt pq time
tel-03438921 stirling cooling heat regenerator
tel-03438923 lisa 10 bruit sgwb
tel-03438925 lungs tree hfcwo mucus
tel-03438938 suspensions particules viscosité pdms
tel-03439261 pendulum zaouali rott energy
tel-03439346 droit art artiste œuvre
tel-03439354 credit bank firm branch
1
2
3
4
rank

N-GRAM

In [31]:

```
#method to generate n-grams:
#params:
#text-the text for which we have to generate n-grams
#ngram-number of grams to be generated from the text(1,2,3,4 etc., default value=1)
```

In [32]:

```
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop
from spacy.lang.en.stop_words import STOP_WORDS as en_stop
from spacy.lang.ro.stop_words import STOP_WORDS as ro_stop
from spacy.lang.pt.stop_words import STOP_WORDS as pt_stop

# use stopwords from spacy
final_stopwords_list = list(fr_stop) + list(en_stop) + list(ro_stop) +list(pt_stop)
# Calculate bigram by using ngram_range=(2,2)

#instance vectorizer with stopwords
tfidf_vectorizer = TfidfVectorizer(input='filename', stop_words=final_stopwords_list, r

#fit vectorizer with all the text files
tfidf_vector = tfidf_vectorizer.fit_transform(text_files)

```

In [33]:

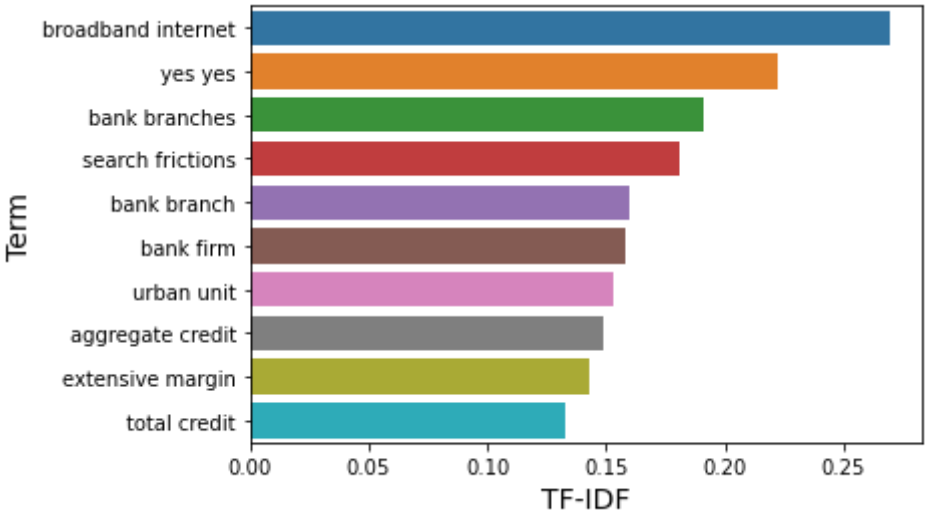
```
#create dataframe from vector
tfidf_df = pd.DataFrame(tfidf_vector.toarray(), index=text_titles, columns=tfidf_vector

tfidf_df.loc['00_Document Frequency'] = (tfidf_df > 0).sum()
# drop “OO_Document Frequency” since we were just using it for illustration purposes.
tfidf_df = tfidf_df.drop('00_Document Frequency', errors='ignore')
#sort values such that top 5 tf-idf of every document is obtained
tfidf_df = tfidf_df.stack().reset_index()


tfidf_df = tfidf_df.rename(columns={0:'tfidf', 'level_0': 'document','level_1': 'term',



```

In [34]:

```
test_df = (tfidf_df.sort_values(by=['document','tfidf'], ascending=[True,False]).groupb

```

In [37]:

```
# Construct plot
p=sns.barplot(x = "tfidf", y = "term", data = test_df)
p.set_xlabel("TF-IDF", fontsize = 14)
p.set_ylabel("Term",fontsize=14)

plt.show()
```



In [40]:

```
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop
from spacy.lang.en.stop_words import STOP_WORDS as en_stop
from spacy.lang.ro.stop_words import STOP_WORDS as ro_stop
from spacy.lang.pt.stop_words import STOP_WORDS as pt_stop


final_stopwords_list = list(fr_stop) + list(en_stop) + list(ro_stop) +list(pt_stop)

tfidf_vectorizer = TfidfVectorizer(input='filename', stop_words=final_stopwords_list, r



#testing on 5 files only

tfidf_vector = tfidf_vectorizer.fit_transform(text_files[:5])


tfidf_df = pd.DataFrame(tfidf_vector.toarray(), index=text_titles[:5], columns=tfidf_ve


tfidf_df.loc['00_Document Frequency'] = (tfidf_df > 0).sum()


tfidf_df = tfidf_df.drop('00_Document Frequency', errors='ignore')


tfidf_df.stack().reset_index()


tfidf_df = tfidf_df.stack().reset_index()


tfidf_df = tfidf_df.rename(columns={0:'tfidf', 'level_0': 'document','level_1': 'term',


tfidf_df=tfidf_df.sort_values(by=['document','tfidf'], ascending=[True,False]).groupby(

```

In [44]:

```
tfidf_df.head()
```

Out[44]:

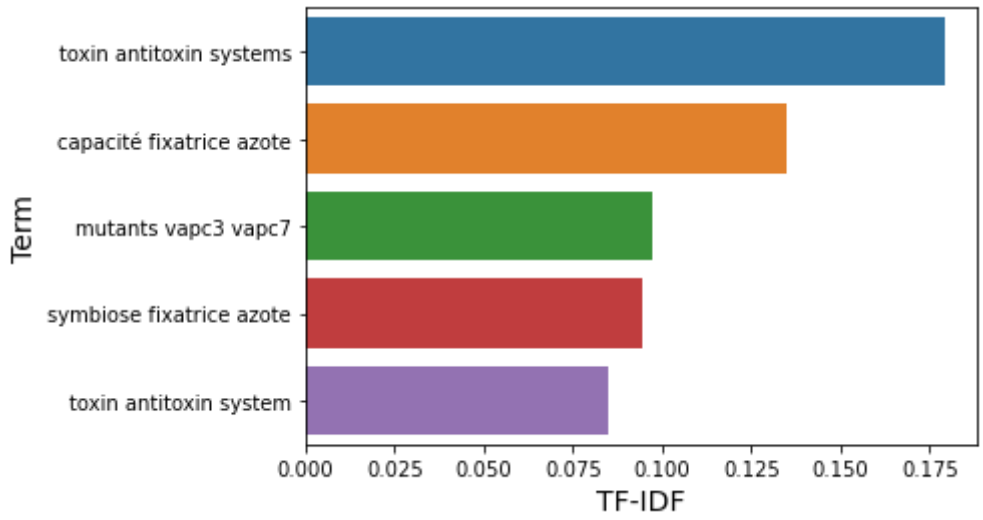
	document	term	tfidf
251726	tel-03435883	toxin antitoxin systems	0.179170
64365	tel-03435883	capacité fixatrice azote	0.135163
167348	tel-03435883	mutants vapc3 vapc7	0.097443
243732	tel-03435883	symbiose fixatrice azote	0.094300
251725	tel-03435883	toxin antitoxin system	0.084870

In [43]:

```
test_df2 = tfidf_df.head()
```

In [45]:

```
# Construct plot
p=sns.barplot(x = "tfidf", y = "term", data = test_df2)
p.set_xlabel("TF-IDF", fontsize = 14)
p.set_ylabel("Term",fontsize=14)

plt.show()
```



In [28]:

```
#https://studymachinelearning.com/cosine-similarity-text-similarity-metric/
```

check similarity between documents

In [47]:

```
#get names of the first 10 docs
docs_names=all_filenames[:10]
```

In [48]:

```
#read all the 10 text files and append them to a list
data=[]
for doc in docs_names:
    with open(doc, "r",encoding="utf-8") as f:
        #read file as text and store in variable `text`
        text = f.read()
        data.append(text)

```

In [49]:

```
#get the IDS of all the 10 docs
doc_names=[]
for doc in docs_names:

    split_string = doc.split(".", 1)
    substring = split_string[0]
    doc_names.append(substring)
```

In [50]:

```
doc_names
```

Out[50]:

```
['tel-03435883',
 'tel-03435884',
 'tel-03435885',
 'tel-03435936',
 'tel-03436011',
 'tel-03436023',
 'tel-03436024',
 'tel-03436025',
 'tel-03436087',
 'tel-03436137']
```

In [51]:

```
#convert the IDS to a dataframe
df_docnames = pd.DataFrame(doc_names)
```

In [52]:

```
df_docnames
```

Out[52]:

	0
0	tel-03435883
1	tel-03435884
2	tel-03435885
3	tel-03435936
4	tel-03436011
5	tel-03436023
6	tel-03436024
7	tel-03436025
8	tel-03436087
9	tel-03436137

In [53]:

```
#rename column o to ID
df_docnames=df_docnames.rename(columns={0:"halId_s"})
```

In [54]:

```
#merge with original dataset
df_merged=pd.merge(df_docnames,df,on="halId_s")
```

In [55]:

```
#store the names of the theses in a list
doc_names=df_merged["title_s"].tolist()
```

In [75]:

```
from sklearn.feature_extraction.text import CountVectorizer
#use Countvectorizer
count_vectorizer = CountVectorizer()
vector_matrix = count_vectorizer.fit_transform(data)
vector_matrix
```

Out[75]:

```
<10x53455 sparse matrix of type '<class 'numpy.int64'>'
        with 93787 stored elements in Compressed Sparse Row format>
```

In [76]:

```
#get names of the features
tokens = count_vectorizer.get_feature_names()

```

In [77]:

```
#convert matrix to a 2d numpy array
vector_matrix.toarray()

```

Out[77]:

```
array([[ 1,  7,  0, ...,  0,  0,  0],
       [60,  5,  0, ...,  0,  0,  0],
       [ 0, 11,  0, ...,  0,  0,  0],
       ...,
       [43,  1,  0, ...,  0,  0,  0],
       [37, 42,  0, ...,  0,  0,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=int64)
```

In [78]:

```
def create_dataframe(matrix, tokens):

    #doc_names = [f'doc_{i+1}' for i, _ in enumerate(matrix)]
    df = pd.DataFrame(data=matrix, index=doc_names, columns=tokens)
    return(df)
```

In [79]:

```
create_dataframe(vector_matrix.toarray(),tokens).head()
```

Out[79]:

	00	000	0000	0000034996	00000407	0000042954	00000457	00000569
Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	1	7	0	0	0	0	0	0
Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	60	5	0	1	0	0	0	0
Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	0	11	0	0	0	1	0	0
Caractérisation génomique des anomalies de la pigmentation cutanée en mosaïque,Genomic characterization of mosaic cutaneous pigmentary disorders	8	48	1	0	1	0	1	1

	00	000	0000	0000034996	00000407	0000042954	00000457	00000569
Le Conseil économique\, social et environnemental régional : assemblée du dialogue des intérêts organisés dans la région,The Conseil économique\, social et environnemental régional : an assembly of dialogue between organised interests in the Région	1	13	0	0	0	0	0	0

5 rows × 53455 columns

In [80]:

```
from sklearn.metrics.pairwise import cosine_similarity
#calculate the cosine similarity of matrix
cosine_similarity_matrix = cosine_similarity(vector_matrix)
```

In [81]:

```
#create dataframe from matrix with theses name
distance_matrix=create_dataframe(cosine_similarity_matrix,doc_names)

```

In [82]:

```
1 distance_matrix
```

Out[82]:

	Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	C g ar mosa chara mos
Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	1.000000	0.869968	0.436238	
Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	0.869968	1.000000	0.520814	
Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	0.436238	0.520814	1.000000	
Caractérisation génomique des anomalies de la pigmentation cutanée en mosaïque,Genomic characterization of mosaic cutaneous pigmentary disorders	0.845360	0.839129	0.624248	

Le Conseil économique\, social et environnemental régional : assemblée du dialogue des intérêts organisés dans la région,The Conseil économique\, social et environnemental régional : an assembly of dialogue between organised interests in the Région	0.826060	0.714074	0.054021
Optical Developments for Microscale Measurement and Control of Temperature in Optogenetics,Développements optiques pour la mesure et le contrôle micrométrique de la température en optogénétique	0.390573	0.459029	0.902244
Z-boson and double charm production with ALICE at the LHC,Production des bosons Z et du double charme avec ALICE auprès du LHC	0.408820	0.473688	0.893786
Caractérisation opto-mécanique du verre traité par des méthodes thermo-chimiques,Opto-mechanical characterization of glass treated by thermochemical methods	0.856415	0.745178	0.137838
La Dynamique sociale des pratiques : stratification sociale\, changement social et consommation alimentaire,The Social Dynamics of Practices : social stratification\, social change and food consumption	0.858910	0.765129	0.164582
Random surface growth models : hydrodynamic limits and fluctuations,Modèles de croissance de surfaces aléatoires : limites hydrodynamiques et fluctuations	0.425379	0.487731	0.896544

In [83]:

```
distance_matrix.to_csv(r'H:\Downloads\Datatsets\Tel\dist_m.csv',index=True)

```

In [84]:

```
distance_matrix.reset_index().to_csv(r'H:\Downloads\Datatsets\Tel\dist_m.csv',index=Tru
```

Similarity with tf-idf vectorizer instead of count vectorizer

In [85]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
#same procedure as count vectorizer but we use tfidf-vectorizer which is better
Tfidf_vect = TfidfVectorizer()
vector_matrix = Tfidf_vect.fit_transform(data)

tokens = Tfidf_vect.get_feature_names()
create_dataframe(vector_matrix.toarray(),tokens).head()
```

Out[85]:

	00	000	0000	0000034996	00000407	0000042954	00000
Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	0.000175	0.001225	0.000000	0.000000	0.000000	0.000000	0.000
Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	0.007519	0.000627	0.000000	0.000257	0.000000	0.000000	0.000
Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	0.000000	0.002206	0.000000	0.000000	0.000000	0.000411	0.000
Caractérisation génomique des anomalies de la pigmentation cutanée en mosaïque,Genomic characterization of mosaic cutaneous pigmentary disorders	0.002571	0.015426	0.000659	0.000000	0.000659	0.000000	0.000

	00	000	0000	0000034996	00000407	0000042954	00000
Le Conseil économique\, social et environnemental régional : assemblée du dialogue des intérêts organisés dans la région,The Conseil économique\, social et environnemental régional : an assembly of dialogue between organised interests in the Région	0.000089	0.001156	0.000000	0.000000	0.000000	0.000000	0.000

5 rows × 53455 columns

In [86]:

```
#get cosine similarity from applying on sparse matrix + create dataframe from matrix
cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix,doc_names)
```

Out[86]:

	Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	C g ar mosa chara mos
Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizobium-Légumineuse,The VapBC Toxin-Antitoxin systems : regulators of the nitrogen-fixing symbiosis Rhizobium-Legume	1.000000	0.615478	0.375844	
Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développement d'un outil innovant pour la mesure des anticorps neutralisants après infection ou vaccination,Phylogenic study of equid alphaherpesvirus strains isolated in France and development of an innovative assay for the measurement of neutralising antibodies after infection or vaccination	0.615478	1.000000	0.389180	
Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assemblage de la nucléocapside du virus de la rougeole	0.375844	0.389180	1.000000	
Caractérisation génomique des anomalies de la pigmentation cutanée en mosaïque,Genomic characterization of mosaic cutaneous pigmentary disorders	0.686478	0.587794	0.530602	

Le Conseil économique\, social et environnemental régional : assemblée du dialogue des intérêts organisés dans la région,The Conseil économique\, social et environnemental régional : an assembly of dialogue between organised interests in the Région	0.689506	0.511317	0.046973
Optical Developments for Microscale Measurement and Control of Temperature in Optogenetics,Développements optiques pour la mesure et le contrôle micrométrique de la température en optogénétique	0.333542	0.336816	0.813004
Z-boson and double charm production with ALICE at the LHC,Production des bosons Z et du double charme avec ALICE auprès du LHC	0.343989	0.341891	0.790952
Caractérisation opto-mécanique du verre traité par des méthodes thermo-chimiques,Opto-mechanical characterization of glass treated by thermochemical methods	0.716493	0.534202	0.122538
La Dynamique sociale des pratiques : stratification sociale\, changement social et consommation alimentaire,The Social Dynamics of Practices : social stratification\, social change and food consumption	0.727890	0.557179	0.145243
Random surface growth models : hydrodynamic limits and fluctuations,Modèles de croissance de surfaces aléatoires : limites hydrodynamiques et fluctuations	0.352988	0.346750	0.782971

In [87]:

```
#store matrix to export as csv
distance_matrix2 = create_dataframe(cosine_similarity_matrix,doc_names)
```

In [88]:

```
distance_matrix2.to_csv(r'H:\Downloads\Datatsets\Tel\dist_m2.csv',index=True)

```

Force-directed graph

MEENOWA Sarvesh

09/12/2021

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
library(qgraph)
```

```
## Warning: package 'qgraph' was built under R version 4.0.5
```

```
## cosine similarity


nba <- read_csv("H:/Downloads/Datatsets/Tel/dist_m.csv",locale = readr::locale(encoding = "utf-8"))
```

```
## New names:
## * `` -> ...1

## Rows: 10 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): index
## dbl (11): ...1, Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rownames(nba) <- nba$index
```

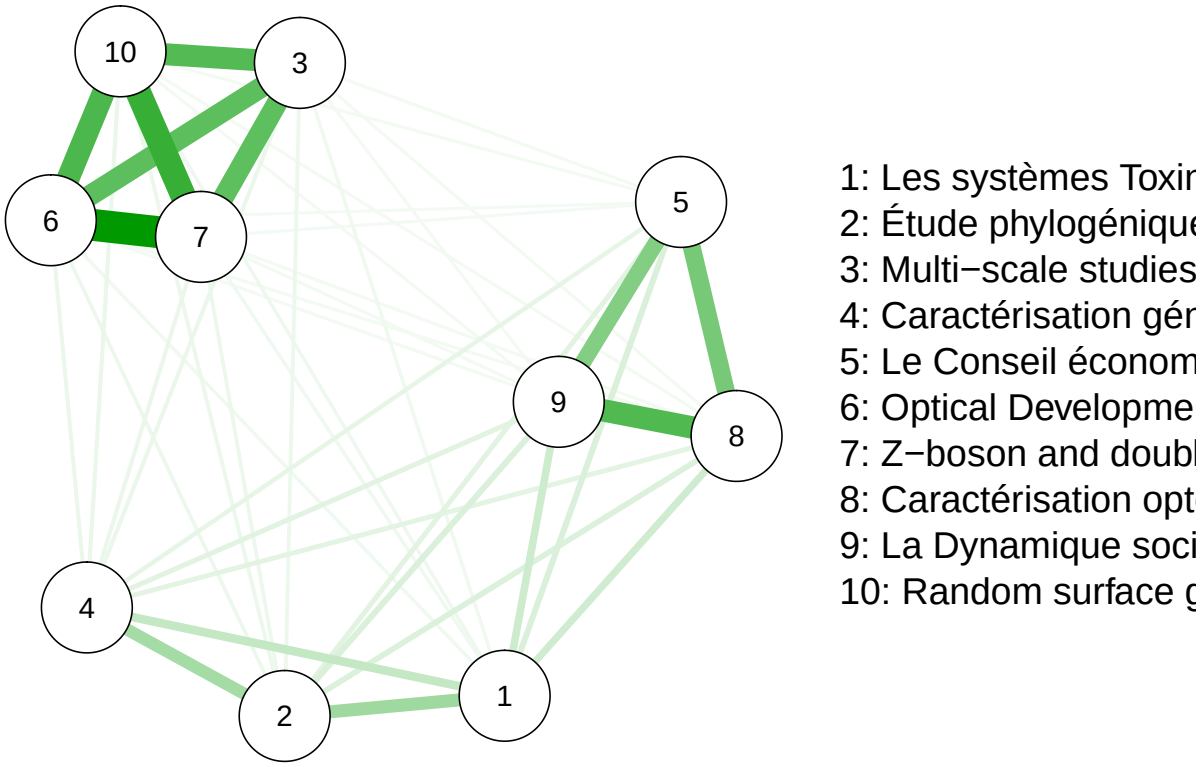
```
## Warning: Setting row names on a tibble is deprecated.
```

```
colnames(nba)
```

```
##  [1] "...1"
##  [2] "index"
##  [3] "Les systèmes Toxine-Antitoxine VapBC : des régulateurs de la symbiose fixatrice d'azote Rhizob
##  [4] "Étude phylogénique de souches d'alphaherpèsvirus isolées chez les équidés français et développe
```

```
##  [5] "Multi-scale studies of Measeles virus nucleocapsid assembly,Etudes multi-échelles de l'assembla
##  [6] "Caractérisation génomique des anomalies de la pigmentation cutanée en mosaïque,Genomic charact
##  [7] "Le Conseil économique\\, social et environnemental régional : assemblée du dialogue des intérê
##  [8] "Optical Developments for Microscale Measurement and Control of Temperature in Optogenetics,Dév
##  [9] "Z-boson and double charm production with ALICE at the LHC,Production des bosons Z et du double
## [10] "Caractérisation opto-mécanique du verre traité par des méthodes thermo-chimiques,Opto-mechanica
## [11] "La Dynamique sociale des pratiques : stratification sociale\\, changement social et consommatio
## [12] "Random surface growth models : hydrodynamic limits and fluctuations,Modèles de croissance de su
```

```
test= subset(nba, select = -c(...1,index))
#convert distance to coordinates and convert to matrix
dist_m <- as.matrix(dist(test))
dist_mi <- 1/dist_m # one over, as qgraph takes similarity matrices as input
#jpeg('voronoid_count.jpg', width=1000, height=1000, unit='px')
list_names = nba$index
qgraph(dist_mi, layout='spring', vsize=8,legend=TRUE,nodeNames=list_names,esize=20)
```



Database creation to generate MCQ exercises

In [1]:

```
import stanza
import spacy
import pandas as pd
import spacy_stanza
import os
import warnings
# import mlconjug3
import mlconjug3
warnings.filterwarnings("ignore")
```

In [2]:

```
#import excel corpus
df = pd.read_excel("7000 sentences Corpus with IDs.xlsx")
```

In [3]:

```
#get path
os.getcwd()
```

Out[3]:

```
'C:\\Users\\Administrator\\stanza_models'
```

In [4]:

```
#stanza.download(lang='en', model_dir ='./stanza_models')
nlp_en = spacy_stanza.load_pipeline("en", dir = 'C:\\Users\\Administrator\\stanza_model
```

```
2021-12-10 00:01:30 INFO: Loading these models for language: en (English):
=========================
| Processor | Package   |
-------------------------
| tokenize  | combined  |
| pos       | combined  |
| lemma     | combined  |
| depparse  | combined  |
| sentiment | sstplus   |
| ner       | ontonotes |
=========================

2021-12-10 00:01:30 INFO: Use device: cpu
2021-12-10 00:01:30 INFO: Loading: tokenize
2021-12-10 00:01:30 INFO: Loading: pos
2021-12-10 00:01:32 INFO: Loading: lemma
2021-12-10 00:01:33 INFO: Loading: depparse
2021-12-10 00:01:35 INFO: Loading: sentiment
2021-12-10 00:01:37 INFO: Loading: ner
2021-12-10 00:01:41 INFO: Done loading processors!
```

In [5]:

```
df_eng = df["English"].dropna()
```

In [13]:

```
list_df_noun = []
list_df_verb = []

for phrase in df_eng:

    list_noun = []
    list_verb = []
    doc = nlp_en(phrase)
    for token in doc:
        # Get noun and number
        if token.pos_ == 'NOUN' and len(token.morph.get("Number")) > 0:
            list_noun.append({token.text: token.morph.get("Number")[0]})

        # Get verb
        if token.pos_ == 'VERB' and len(token.morph.get("Tense")) > 0:
            prefix = ''
            if len(token.morph.get("Voice")) > 0 and token.morph.get("Voice")[0] == "Pa
                temp_token = token
                while temp_token.nbor(-1).pos_ == "AUX":
                    prefix = temp_token.nbor(-1).text + ' ' + prefix
                    temp_token = temp_token.nbor(-1)

            list_verb.append({prefix + token.text: token.morph.get("Tense")[0]})

    # Add noun
    for noun in list_noun:
        for key, value in noun.items():
            list_df_noun.append([phrase, key, value])

    # Add verb
    for verb in list_verb:
        for key, value in verb.items():
            list_df_verb.append([phrase, key, value])
```

In [27]:

```
df_noun = pd.DataFrame(list_df_noun, columns=["Phrase", "Noun", "Number"])
df_noun
```

Out[27]:

	Phrase	Noun	Number
0	The beauty of the landscape struck the travell...	beauty	Sing
1	The beauty of the landscape struck the travell...	landscape	Sing
2	The beauty of the landscape struck the travell...	travellers	Plur
3	Nobody knows the truth about this affair.	truth	Sing
4	Nobody knows the truth about this affair.	affair	Sing
...	...	...	...
5734	Computer scientists find a job quickly enough.	Computer	Sing
5735	Computer scientists find a job quickly enough.	scientists	Plur
5736	Computer scientists find a job quickly enough.	job	Sing
5737	Shoemakers rapair shoes.	Shoemakers	Plur
5738	Shoemakers rapair shoes.	shoes	Plur

5739 rows × 3 columns

In [28]:

```
#since getting verbs take time, we convert it to csv
df_noun.to_csv("df_noun.csv",index=False)
```

In [8]:

```
df_noun = pd.read_csv("df_noun.csv")
```

In [22]:

```
df_verb = pd.DataFrame(list_df_verb, columns=["Phrase", "Verb", "Tense"])
df_verb
```

Out[22]:

	Phrase	Verb	Tense
0	The beauty of the landscape struck the travell...	struck	Past
1	Nobody knows the truth about this affair.	knows	Pres
2	In a dictatorship, freedom of expression is li...	is limited	Past
3	His wickedness had no limits.	had	Past
4	His elegance impressed the assembly.	impressed	Past
...	...	...	...
2215	Teachers teach in primary schools.	teach	Pres
2216	The plumber is going to come this afternoon.	going	Pres
2217	He quit his job because his salary was too low.	quit	Past
2218	Computer scientists find a job quickly enough.	find	Pres
2219	Shoemakers rapair shoes.	rapair	Pres

2220 rows × 3 columns

In [31]:

```
# getting the tense and verb takes time, so convert it to csv
df_verb.to_csv('df_verb.csv',index=False)
```

In [10]:

```
df_verb = pd.read_csv('df_verb.csv')
```

In [8]:

```
#create function to lemmatize the verb
def get_lemma(x):
    doc=nlp_en(x)
    for token in doc:
        return token.lemma_


```

In [9]:

```
#test
get_lemma("struck")
```

Out[9]:

```
'strike'
```

In [11]:

```
#apply function to verb column to obtain lemma
df_verb["Lemma"]=df_verb["Verb"].apply(lambda x:get_lemma(x))
```

In [12]:

```
#lemma function application takes time as well, so we conver it as csv a well
df_verb.to_csv('df_verb.csv',index=False)
```

In [11]:

```
df_verb = pd.read_csv('df_verb.csv')
```

In [12]:

```
df_verb
```

Out[12]:

	Phrase	Verb	Tense	Lemma
0	The beauty of the landscape struck the travell...	struck	Past	strike
1	Nobody knows the truth about this affair.	knows	Pres	know
2	In a dictatorship, freedom of expression is li...	is limited	Past	be
3	His wickedness had no limits.	had	Past	have
4	His elegance impressed the assembly.	impressed	Past	impressed
...	...	...	...	...
2215	Teachers teach in primary schools.	teach	Pres	teach
2216	The plumber is going to come this afternoon.	going	Pres	go
2217	He quit his job because his salary was too low.	quit	Past	quit
2218	Computer scientists find a job quickly enough.	find	Pres	find
2219	Shoemakers rapair shoes.	rapair	Pres	rapair

2220 rows × 4 columns

In [14]:

```
# choose language to set as default conjugator
default_conjugator = mlconjug3.Conjugator(language='en')

```

In [15]:

```
#test instantiated function on verb "know"
test_verb = default_conjugator.conjugate("know")
all_conjugated_forms = test_verb.iterate()
#results gives a list of tuples
print(all_conjugated_forms)
```

```
[('indicative', 'indicative present', '1s', 'know'), ('indicative', 'indicat
ive present', '2s', 'know'), ('indicative', 'indicative present', '3s', 'kno
ws'), ('indicative', 'indicative present', '1p', 'know'), ('indicative', 'in
dicative present', '2p', 'know'), ('indicative', 'indicative present', '3p',
'know'), ('indicative', 'indicative past tense', '1s', 'knew'), ('indicativ
e', 'indicative past tense', '2s', 'knew'), ('indicative', 'indicative past
tense', '3s', 'knew'), ('indicative', 'indicative past tense', '1p', 'kne
w'), ('indicative', 'indicative past tense', '2p', 'knew'), ('indicative',
'indicative past tense', '3p', 'knew'), ('indicative', 'indicative present c
ontinuous', '1s', 'knowing'), ('indicative', 'indicative present continuou
s', '2s', 'knowing'), ('indicative', 'indicative present continuous', '3s',
'knowing'), ('indicative', 'indicative present continuous', '1p', 'knowin
g'), ('indicative', 'indicative present continuous', '2p', 'knowing'), ('ind
icative', 'indicative present continuous', '3p', 'knowing'), ('indicative',
'indicative present perfect', '1s', 'known'), ('indicative', 'indicative pre
sent perfect', '2s', 'known'), ('indicative', 'indicative present perfect',
'3s', 'known'), ('indicative', 'indicative present perfect', '1p', 'known'),
('indicative', 'indicative present perfect', '2p', 'known'), ('indicative',
'indicative present perfect', '3p', 'known'), ('infinitive', 'infinitive pre
sent', 'to know'), ('imperative', 'imperative present', '2s', 'know'), ('imp
erative', 'imperative present', '1p', 'know'), ('imperative', 'imperative pr
esent', '2p', 'know')]
```

In [18]:

```
#code to loop over a list of tuples and access individual elements of each tuple in the
for index, tuple in enumerate(all_conjugated_forms[:2]):
    #most tuples are of length 4 except 1 so we verify the conditions to not have index
    if len(tuple)==4:
        print(tuple[0])
        print(tuple[1])
        print(tuple[2])
        print(tuple[3])
        print("-------")
```

```
indicative
indicative present
1s
know
-------
indicative
indicative present
2s
know
-------
```

We want to conjugate the lemmatized words in a single function, so to do so we will need to return multiple values

> A clean way to do it is using pandas' and apply function, but the difference , we will need to output more than one column.

> Below is an example

In [92]:

```
#https://stackoverflow.com/questions/23586510/return-multiple-columns-from-pandas-apply
```

In [16]:

```
#define the language of the conjugator
default_conjugator = mlconjug3.Conjugator(language='en')
"""
The function takes parameter x, in normal pandas'- apply operation where only one outpu
But in this case, we don't use the lambda function , so the input is a series
To get individual values instead of series ,we extract the values from the series i.e x
We then use conjugate our lemmatize verb and we iterate over the different possibilitie
After that, we iterate over the list of tuples and access the elements of each tuple to

For example, a tuple has the format ('indicative', 'indicative present', '1s', 'know')

The length is 4:

2nd element(1st index) : verb form
3rd element (2nd index): mode:   1s = 1st person singular
                                 2s = 2nd person singular
                                 1p = 1st person plural
                                 2p = 2nd person plural

4th element : conjugated verb

Note :  we can return more types of verbs according to the needs of the project

"""
def congugate_lemma_verbs(x):
    x=x[0]
    test_verb = default_conjugator.conjugate(x)
    all_conjugated_forms = test_verb.iterate()

    for index, tuple in enumerate(all_conjugated_forms):
        if len(tuple)==4:
            if tuple[1] == "indicative present" and tuple[2] =="1s":
                conjugated_verb_1 = tuple[3]

            if tuple[1] == "indicative present" and tuple[2] =="1p":
                conjugated_verb_11 = tuple[3]

            if tuple[1] =="indicative past tense" and tuple[2]=="1s":
                conjugated_verb_2 = tuple[3]

            if tuple[1] =="indicative past tense" and tuple[2]=="1p":
                conjugated_verb_22 = tuple[3]

            if tuple[1] =="indicative present continuous" and tuple[2] =="1s":
                conjugated_verb_3 = tuple[3]

            if tuple[1] =="indicative present perfect" and tuple[2]=="1s":
                conjugated_verb_4 = tuple[3]

            if tuple[1] =="imperative present" and tuple[2] =="2s":
                conjugated_verb_5 = tuple[3]

    return conjugated_verb_1,conjugated_verb_11,conjugated_verb_2,conjugated_verb_22,cc
```

In [23]:

```
#apply function by creating 6 new columns
df_verb[["FP singular indicative present","FP plural indicative present","FP singular i
         "FP plural indicative past","FP indicative present continuous",\
         "FP indicative present perfect","SP imperative present"]]=\
df_verb[["Lemma"]].apply(congugate_lemma_verbs,axis=1, result_type="expand")
```

In [19]:

```
df_verb.head()
```

Out[19]:

	Phrase	Verb	Tense	Lemma	FP singular indicative present	FP plural indicative present	FP singular indicative past	FP plural indicative past	ind p conti
0	The beauty of the landscape struck the travell...	struck	Past	strike	strike	strike	struck	struck	
1	Nobody knows the truth about this affair.	knows	Pres	know	know	know	knew	knew	k
2	In a dictatorship, freedom of expression is li...	is limited	Past	be	am	are	was	were	
3	His wickedness had no limits.	had	Past	have	have	have	had	had	
4	His elegance impressed the assembly.	impressed	Past	impressed	impressee	impressee	impresseed	impresseed	impre