**APPENDIX**

*TOPIC MODEL PYTON SCRIPT*

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import pandas as pd

import docx

import os

from tqdm import tqdm

from gensim import corpora, models, similarities

from collections import OrderedDict

import seaborn as sns

from nltk.corpus import stopwords

from string import punctuation

import re

import pyLDAvis.gensim

import matplotlib.pyplot as plt

get\_ipython().run\_line\_magic('matplotlib', 'inline')

# In[4]:

# Convert extension with libreoffice: https://libreofficehelp.com/batch-convert-writer-documents-pdf-libreoffice/

# First, we created an empty list to hold the path to all of docx files

document\_list = []

docx\_list = []

# Now, we loop through every file in the folder (and all subfolders) using os.walk(). We can alternatively use os.listdir()

# to get a list of files. It would be recommended if all files are in the same folder

# Research source url (Related Party e.g.):

# https://pesquisa.apps.tcu.gov.br/#/documento/acordao-completo/%2522partes%2520relacionadas%2522/%20/DTRELEVANCIA%20desc,%20NUMACORDAOINT%20desc/0/%20

for path, subdirs, files in os.walk(r"/home/rsa/PycharmProjects/selenium/new\_base"):

for name in files:

# For each file we find, we need to ensure it is a .docx file before adding it to our list

if os.path.splitext(os.path.join(path, name))[1] == ".docx":

document\_list.append(os.path.join(path, name))

# Now we create a loop over each file path in document\_list

for document\_path in document\_list:

document = docx.Document(document\_path) # Change the document format being loaded in each loop

docx\_list.append(document)

# In[ ]:

corpus=[]

a=[]

for i in tqdm(docx\_list):

for j in range(len(i.paragraphs)):

a = i.paragraphs[j].text

corpus.append(a)

# In[ ]:

len(corpus)

# In[5]:

# SAVE CORPUS FILE

# for possible future analysis and reproducibility

with open('corpus\_multas.txt', 'w') as file:

for listitem in corpus:

file.write('%s\n' % listitem)

# In[5]:

# LOAD CORPUS FILE

# open file and read the content in a list

corpus=[]

with open('corpus\_pr\_12outubro.txt', 'r') as filehandle:

for line in filehandle:

# remove linebreak which is the last character of the string

currentPlace = line[:-1]

# add item to the list

corpus.append(currentPlace)

# In[6]:

len(corpus)

# In[7]:

# Replacing terms according to domain knowledge and pattern established

corpus = [i.lower().replace("partes relacionadas ", "partes\_relacionadas ") for i in corpus]

corpus = [i.lower().replace("parte relacionada ", "partes\_relacionadas ") for i in corpus]

corpus = [i.lower().replace("fonte:", "fontes") for i in corpus]

corpus = [i.lower().replace("contratação", "contratações") for i in corpus]

corpus = [i.lower().replace("contrato ", "contratações") for i in corpus]

corpus = [i.lower().replace("contratos", "contratações") for i in corpus]

corpus = [i.lower().replace("contratoss", "contratações") for i in corpus]

corpus = [i.lower().replace("princípio", "princípios") for i in corpus]

corpus = [i.lower().replace("empresa ", "empresas") for i in corpus]

corpus = [i.lower().replace("acionista ", "acionistas") for i in corpus]

corpus = [i.lower().replace("valor ", "valores") for i in corpus]

corpus = [i.lower().replace("aeroporto ", "aeroportos") for i in corpus]

corpus = [i.lower().replace("mercado.", "mercado") for i in corpus]

corpus = [i.lower().replace("mercado,", "mercado") for i in corpus]

# In[7]:

# generating TXT with standard database of stopwords

stplst = list(punctuation) + stopwords.words('portuguese')

excepts = ['não', 'sem']

stplst = [i for i in stplst if i not in excepts]

pd.DataFrame(stplst).to\_csv('stopwords\_nltk.txt', index=False, header=False)

# In[8]:

# Regex to remove word up to 3 chars (if needed) --> shortword = re.compile(r'\W\*\b\w{1,3}\b')

shortword = re.compile(r'\W\*\b\w{1,1}\b')

# remove some words (after inspection) and then tokenizing

blacklist = ['\_','–',' ','Augusto','augusto','Carlos' "$",")","(",'/2015',"walton",'alencar','alves',".","(...)","[...]", 'sobre', 'sr.', 'art.', 'que', 'ser', 'ii', 'ii,', 'inciso,', 'de$', 'nº',

'(peça','§','(cpf','tc','maria','josé','costa','que,','quanto','(',')','75.','19.','-.','(peças-11),','silva', 'tcu', 'ainda','além', 'qualquer','parte','partes','item','tais','dias','tal',

'ano']

stoplist = stopwords.words('portuguese') + list(punctuation) + blacklist

stoplist.remove('não')

stoplist.remove('sem')

# Creating list where each entry contains the tokens of a line of the corpus

texts = [[i for i in str(shortword.sub('', str(j))).lower().split() if i not in stoplist] for j in corpus]

# In[9]:

filt\_part1 = ['partes\_relacionadas']

filt\_part2 = ['parte\_relacionada']

text\_term = [j for j in texts if all(i in j for i in filt\_part1 or filt\_part2)]

text\_total = texts

print("len text\_term:", len(text\_term))

print("len text\_total:", len(text\_total))

# In[10]:

# Geting only parts containing the required terms (about related parties)

dict\_term = corpora.Dictionary(text\_term)

corpus\_term = [dict\_term.doc2bow(i) for i in text\_term]

tfidf\_term = models.TfidfModel(corpus\_term) # step 1 -- initialize a model

corpus\_tfidf = tfidf\_term[corpus\_term] # step 2 -- use the model to transform vectors

# Geting entire database

dict\_total = corpora.Dictionary(text\_total)

corpus\_total = [dict\_total.doc2bow(i) for i in text\_total]

tfidf\_total = models.TfidfModel(corpus\_total) # step 1 -- initialize a model

corpus\_tfidf = tfidf\_total[corpus\_total] # step 2 -- use the model to transform vectors

# In[11]:

# Defining the number of topics to find in the topic modeling

total\_topics = 5

# In[12]:

# Geting only parts containing the required terms (about related parties)

lda\_term = models.LdaModel(corpus\_term, id2word=dict\_term, num\_topics=total\_topics)

corpus\_lda\_term = lda\_term[corpus\_term] # create a double wrapper over the original corpus: bow->tfidf->fold-in-lsi

# Geting entire database

lda\_total = models.LdaModel(corpus\_total, id2word=dict\_total, num\_topics=total\_topics)

corpus\_lda\_total = lda\_total[corpus\_total] # create a double wrapper over the original corpus: bow->tfidf->fold-in-lsi

# In[13]:

# Showing the first n important words in the topics found:

lda\_term.show\_topics(total\_topics)

#lda\_total.show\_topics(total\_topics)

# In[14]:

data\_lda\_term = {i: OrderedDict(lda\_term.show\_topic(i,10)) for i in range(total\_topics)}

data\_lda\_total = {i: OrderedDict(lda\_total.show\_topic(i,10)) for i in range(total\_topics)}

# In[15]:

df\_lda\_term = pd.DataFrame(data\_lda\_term).fillna(0)

df\_lda\_term = df\_lda\_term.T

print("df\_lda\_term shape:", df\_lda\_term.shape)

df\_lda\_total = pd.DataFrame(data\_lda\_total).fillna(0)

df\_lda\_total = df\_lda\_total.T

print("df\_lda\_total shape:", df\_lda\_total.shape)

# In[13]:

# Geting only parts containing the required terms (about related parties)

g=sns.clustermap(df\_lda\_term.corr(), standard\_scale=1, center=0, cmap="RdBu", metric='cosine', linewidths=.05, figsize=(15, 15))

plt.setp(g.ax\_heatmap.yaxis.get\_majorticklabels(), rotation=0)

plt.savefig('fig\_only\_parts.jpg')

plt.show()

# Geting total database

g=sns.clustermap(df\_lda\_total.corr(), standard\_scale=1, center=0, cmap="RdBu", metric='cosine', linewidths=.05, figsize=(15, 15))

plt.setp(g.ax\_heatmap.yaxis.get\_majorticklabels(), rotation=0)

plt.savefig('fig\_total.jpg')

plt.show()

# In[16]:

# Generating interdistance map of topics with pyLDAvis

panel = pyLDAvis.gensim.prepare(lda\_term, corpus\_lda\_term, dict\_term, mds='PCoA')

panel