In [193]:
```
import os

from pptx import Presentation #to read pptx. Note that using beautifulSoup isn't a good option
#all the .pptx have to be opened and resaved as .xml files
import PyPDF2#loads .pdf files
#to load content from docx files. The code to read docx files was borrowed from github user etienned.
try:
    from xml.etree.cElementTree import XML
except ImportError:
    from xml.etree.ElementTree import XML
import zipfile

from collections import Counter#to count files by extension type e.g. .pptx
from datetime import datetime#file create/ last modified timestamp
#import docx#this is the python-docx library needed to load .docx files. Doesn't read code within text boxes.
import re #only to help with formatting, like stripping out spaces etc. Not used for NLP!!

import pandas as pd
pd.set_option('display.max_colwidth', -1) #to prevent cell display truncation

import gc#to release unreferenced memory

import spacy
nlp = spacy.load('en_core_web_lg') #loads models for POS tagging, NER, dependency parser and 1Mn pretrained GloVe vectors
from spacy import displacy #visualizer for dependency parser
from spacy.matcher import PhraseMatcher #for exact phrase matching
matcher = PhraseMatcher(nlp.vocab)
```

In [194]:
```
temp = open('/home/bala/Documents/DEHackathonCaseStudiesMetaData/caseStudiesSampleFiles.csv', 'r')
caseStudiesSampleFiles = pd.read_csv(temp)
```

In [195]:
```
caseStudiesSampleFiles.columns
```

Out[195]:
```
Index(['174', 'Oracle CC and B Managed Services.pdf'], dtype='object')
```

In [196]:
```
fileType=[]
for file in caseStudiesSampleFiles.iloc[:,1].unique():
    #print(file)
    fileType.append(file[-4:])
print('Files types in sample:',Counter(fileType).items())
```

```
Files types in sample: dict_items([('docx', 21), ('pptx', 46), ('.pdf', 24), ('.xls', 1), ('.PDF', 1), ('xlsx', 6)])
```

In [197]:
```
#loading all pdf files in a folder
path = os.chdir('/media/sf_ForVirtualBox/KM/common')
#filtering to keep only 'pdf' files
files_pdf = [f for f in caseStudiesSampleFiles.iloc[:,1] if (f[-4:] == '.pdf' or f[-4:] == '.PDF')]
print(len(files_pdf))
```

```
25
```

In [198]:
```
#extracting the text from each pdf and creating a dataframe with text from each file in a row
fileNames = []
contentAll = []
notReadList = []
contentFile = []
for each in files_pdf:
    contentFile = []
    try:
        pdfFileObj = open(each, 'rb')
        pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
        numPagesVal = pdfReader.numPages
        for page in range(numPagesVal):
            pageObj = pdfReader.getPage(page)
            pageText = pageObj.extractText()
            contentFile.append(pageText)
        contentFileCons = '. '.join(contentFile)
        contentFileCons = contentFileCons.replace('\n', '')#for the line breaks
        contentFileCons = re.sub('\s+',' ',contentFileCons)#for the whitespaces
        contentAll.append(contentFileCons)
        fileNames.append(each)
    except Exception:
        notReadList.append(each)
        pass
```

```
PdfReadWarning: Xref table not zero-indexed. ID numbers for objects will be corrected. [pdf.py:1736]
```

In [199]:
```
print(len(fileNames))
print(len(contentAll))
print(len(notReadList))
print((notReadList))
```

```
25
25
0
[]
```

In [200]:
```
#loading all xlsx files in a folder
#filtering to keep only 'xlsx' files
files_xlsx = [f for f in caseStudiesSampleFiles.iloc[:,1] if (f[-4:] == 'xlsx' or f[-4:] == '.xls')]
print(len(files_xlsx))
```

```
7
```

In [201]:
```
#extracting the text from each xlsx and creating a dataframe with text from each file in a row
for each in files_xlsx:
    try:
        contentSheet = []
        sheets = pd.ExcelFile(each).sheet_names
        for sheet in sheets:
            readSheet = pd.read_excel(each, sheet, skiprows=1)
            readSheet = readSheet.fillna('')#takes care of the NAN due to blank or merged cells
            contentSheet.append(readSheet)
        contentFile = '. '.join(map(str,contentSheet))
        contentFile = contentFile.replace('\n', '')#for the line breaks
        contentFile = re.sub('\s+',' ',contentFile)#for the whitespaces
        contentAll.append(contentFile)
        fileNames.append(each)
    except Exception:
        notReadList.append(each)
        pass
```

In [202]:
```
print(len(fileNames))
print(len(contentAll))
print(len(notReadList))
print((notReadList))
```

```
32
32
0
[]
```

In [203]:
```
#loading all docx files in a folder
#filtering to keep only 'docx' files
files_docx = [f for f in caseStudiesSampleFiles.iloc[:,1] if (f[-4:] == 'docx' or f[-4:] == '.doc')]
print(len(files_docx))
```

```
21
```

In [204]:
```
WORD_NAMESPACE = '{http://schemas.openxmlformats.org/wordprocessingml/2006/main}'
PARA = WORD_NAMESPACE + 'p'
TEXT = WORD_NAMESPACE + 't'

def get_docx_text(path):
    """
    Take the path of a docx file as argument, return the text in unicode.
    """
    document = zipfile.ZipFile(path)
    xml_content = document.read('word/document.xml')
    document.close()
    tree = XML(xml_content)

    paragraphs = []
    for paragraph in tree.getiterator(PARA):
        texts = [node.text
                 for node in paragraph.getiterator(TEXT)
                 if node.text]
        if texts:
            paragraphs.append(''.join(texts))
            #paragraphs = paragraphs.replace('\n', '')#for the line breaks
            #paragraphs = re.sub('\s+',' ',paragraphs)#for the whitespaces

    return '\n\n'.join(paragraphs)
```

In [205]:
```
#extracting the text from each docx and creating a dataframe with text from each file in a row
for each in files_docx:
    try:
        rawText = get_docx_text(each)
        rawText = re.sub('\n','.',rawText)#for the line breaks
        rawText = re.sub('\s+',' ',rawText)#for the whitespaces
        contentAll.append(rawText)
        #contentAll = contentAll.replace('\n', '')#for the line breaks
        fileNames.append(each)

    except Exception:
        notReadList.append(each)
        pass
```

In [206]:
```
print(len(fileNames))
print(len(contentAll))
print(len(notReadList))
print((notReadList))
```

```
53
53
0
[]
```

In [207]:
```
#loading all pptx files in a folder
#filtering to keep only 'pptx' files
files_pptx = [f for f in caseStudiesSampleFiles.iloc[:,1] if (f[-4:] == 'pptx' or f[-4:] == 'PPTX')]
print(len(files_pptx))
```

```
46
```

In [208]:
```
#extracting the text from each pptx and creating a dataframe with text from each file in a row
for each in files_pptx:
    #print(each)
    try:
        contentFile = []
        prs = Presentation(each)
        for slide in prs.slides:
            contentSlide = []
            for shape in slide.shapes:
                if hasattr(shape, "text"):
                    text = shape.text.lower()
                    text = re.sub(' +', ' ', text)
                    contentSlide.append(text)
            contentSlideCons = '. '.join(contentSlide)
            contentFile.append(contentSlideCons)
        contentFileCons = ' '.join(contentFile)
        contentFileCons = contentFileCons.replace('\n', '')#for the line breaks
        contentFileCons = re.sub('\s+',' ',contentFileCons)#for the whitespaces
        contentAll.append(contentFileCons)
        fileNames.append(each)

    except Exception:
        notReadList.append(each)
        pass
```

In [209]:
```
print(len(fileNames))
print(len(contentAll))
print(len(notReadList))
print((notReadList))
```

```
99
99
0
[]
```

In [210]:
```
fileNames = pd.DataFrame(fileNames)
contentAll = pd.DataFrame(contentAll)
caseStudies = pd.concat([fileNames,contentAll], axis=1)
caseStudies.columns = ['fileNames', 'Content']
```

In [211]:
```
print('# of files that could not be loaded into Python: ', len(notReadList), '\n')
print('list of files that could not be loaded into Python:', '\n', notReadList, '\n')
print('# of files loaded into Python:', caseStudies.shape[0], '\n')
```

```
# of files that could not be loaded into Python:  0

list of files that could not be loaded into Python:
 []

# of files loaded into Python: 99
```

In [212]:
```
#adding file created time stamp
temp_1 = open('filesWCreatedTime.csv', 'r', encoding = 'latin_1')#loading the file created on Windows
filesWCreatedTime = pd.read_csv(temp_1)
caseStudiesBU = caseStudies
caseStudies = pd.merge(caseStudies, filesWCreatedTime[['fileName','createdTime']], left_on='fileNames', right_on='fileName'
)
caseStudies = caseStudies[['fileName','createdTime', 'Content']]
caseStudies.shape
```

Out[212]: (99, 3)

In [213]:
```
print(caseStudies.columns)
print(caseStudiesBU.columns)
print(caseStudies.shape)
print(caseStudiesBU.shape)
```

```
Index(['fileName', 'createdTime', 'Content'], dtype='object')
Index(['fileNames', 'Content'], dtype='object')
(99, 3)
(99, 2)
```

In [214]:
```
#breaking up the text into sentences using spaCy's tokenizer

fileNames = []
sentences = []
createdTime = []
sentencesDoc = []
#for row in range(30):
for row in range(len(caseStudies)):
    caseStudy = caseStudies.iloc[row,2]
    doc = nlp(caseStudy)
    for sent in doc.sents:
        sentence = ''.join(sent.text)
        sentences.append(sentence)
        sentenceDoc = nlp(sentence)
        sentencesDoc.append({"sentencesDoc": sentenceDoc})
        fileNames.append(caseStudies.iloc[row,0])
        createdTime.append(caseStudies.iloc[row,1])

sentencesDoc = pd.DataFrame(sentencesDoc)
sentences = pd.DataFrame(sentences)
fileNames = pd.DataFrame(fileNames)
createdTime = pd.DataFrame(createdTime)
reusableAssetsSentences = pd.concat([fileNames, createdTime, sentences, sentencesDoc], axis=1)
reusableAssetsSentences.columns = ['fileNames', 'createdTime', 'sentences', 'sentencesDoc']
```

In [216]:
```
#updated code for semantic and exact similarity. uses stored .Doc

#note: exact phrase matcher can search only for 1 to 10 word phrases

matcher = PhraseMatcher(nlp.vocab)
#testSent = 'Oracle Applications provides'
#testSent = 'purchased'
#testSent = 'purchased in bulk'
testSent = 'overall production capacity'
#testSent = 'OSP enables businesses to include supplier sourced components'
testSentNLP = nlp(testSent)

matcher.add('testSent', None, nlp.make_doc(testSent))

semanticSimilarityScores = []
exactSimilarityScores = []
fileNames = []
sentencesDoc = []
createdTime = []
sentences = []

for row in range(len(reusableAssetsSentences)):
    sentenceDoc = reusableAssetsSentences.loc[row, 'sentencesDoc']

    #word embedding vector similarity
    sentenceSimilarity = sentenceDoc.similarity(testSentNLP)
    sentenceSimilarity = round(sentenceSimilarity,2)
    semanticSimilarityScores.append(sentenceSimilarity)

    #exact phrase match
    matches = matcher(sentenceDoc)
    if matches:
        exactSimilarityScore = 1
        for match_id, start, end in matches:
            span = sentenceDoc[(start):(end+10)]
            print('File name & sentence fragment with exact match: ',
                reusableAssetsSentences.loc[row, 'fileNames'], '\n', span.text, '\n','-'*50, '\n')

    else:
        exactSimilarityScore = 0
    exactSimilarityScores.append(exactSimilarityScore)

    #meta data
    fileNames.append(reusableAssetsSentences.loc[row, 'fileNames'])
    createdTime.append(reusableAssetsSentences.loc[row, 'createdTime'])
    sentences.append(reusableAssetsSentences.loc[row, 'sentences'])
    sentencesDoc.append(reusableAssetsSentences.loc[row, 'sentencesDoc'].text)
```

```
fileNames = pd.DataFrame(fileNames)
createdTime = pd.DataFrame(createdTime)
semanticSimilarityScores = pd.DataFrame(semanticSimilarityScores)
exactSimilarityScores = pd.DataFrame(exactSimilarityScores)
sentences = pd.DataFrame(sentences)
sentencesDoc = pd.DataFrame(sentencesDoc)

filesScore = pd.concat([fileNames, semanticSimilarityScores, sentences], axis=1)
filesScore.columns = ['fileNames', 'semanticSimilarityScores', 'sentences']
bysimilarityScores = pd.merge(filesScore, filesWCreatedTime[['fileName','createdTime']], left_on='fileNames', right_on='fil
eName')
bysimilarityScores = bysimilarityScores[['fileName', 'createdTime', 'semanticSimilarityScores', 'sentences']]
bysimilarityScores.sort_values(['semanticSimilarityScores',
                                'createdTime'], ascending = False).head(5)
```

```
File name & sentence fragment with exact match:  A White Paper - Outside Processing - v1.1.pdf
 overall production capacity OSP refers to the process of entering into a contractual
 ------------------------------------------------
```

Out[216]:

	fileName	createdTime	semanticSimilarityScores	sentences
7257	OBIEE_Data Migration Approach.pptx	2018-11-27 09:42:11	0.80	manage production system performance.
5	A White Paper - Outside Processing - v1.1.pdf	2018-11-29 12:59:12	0.76	The Features of OSP include Using Specialized supplier skills in manufacturing process to help lower engineering and manufacturing cost and increase Production quality Using Supplier capacity to increase overall production capacity OSP refers to the process of entering into a contractual agreement with a Supplier to perform a certain amount of work, optionally performed with contractor material.
6443	Acelity OSP Capability Presentation FINAL.pptx	2018-11-27 09:41:02	0.76	20% improvement in the demantra monthly cycle run time due to enhanced database performance improvement in the new version5% reduction in the overall plan run delivery – production, atp and inventory optimization
3941	Resource Quality_Capability.docx	2018-11-29 12:58:33	0.75	Overall FCA rating in our annual Client Satisfaction Survey for the overall team performance...Provide transparency in individual performance levels:
6870	Cognizant - OSP - OBIA Capabilities 20150527.pptx	2018-11-27 09:41:06	0.74	improved sales force productivity & increased revenues & lower costs, due to increased market share and intelligent allocation of sales resources. .

In [ ]: