**Task I – Unstructured Data Analysis**

‘Unstructured Data Japanese.xlsx’

NOTE:

1. Tokenization using Japanese tokenizer “tinysegmenter”
2. Complete different cleansing of tokens as these text are different from English one.
3. Remaining opts are same to english

To validate result, I tried Google Translate for Japanese to English and vice versa. And result were good.

Dependencies :

#! pip3 install xlrd >= 1.0.0

#! pip3 install openpyxl

#! pip3 install pattern

#!pip install JapaneseTokenizer # dependency needs to be installed mecab\_wrapper

#!pip install tinysegmenter

#! git clone <https://github.com/stopwords-iso/stopwords-ja.git>

1. Generate code for data cleansing in Python

#load module

import pandas as pd

import re

import nltk

import codecs

import string

from pattern.en import parsetree

from pattern.en import parse

from sklearn.feature\_extraction.text import CountVectorizer

#load data

data = pd.read\_excel('Unstructured Data Japanese.xls', encoding='utf-8', index\_col=0)

data.head()

data.rename(columns={'Kraft Super bowl data': 'text'}, inplace=True)

data.head(5)

# read Japanese stopword files. And put it in list.

ja\_stopwords = []

with open('stopwords-ja/raw/japanese-stopwords.txt', encoding="utf8",mode='r') as f:

stopwords\_tmp = f.readlines()

ja\_stopwords.extend(stopwords\_tmp)

with open('stopwords-ja/raw/gh-stopwords-json-ja.txt', encoding="utf8",mode='r') as f:

stopwords\_tmp = f.readlines()

ja\_stopwords.extend(stopwords\_tmp)

with open('stopwords-ja/raw/ranksnl-japanese.txt', encoding="utf8",mode='r') as f:

stopwords\_tmp = f.readlines()

ja\_stopwords.extend(stopwords\_tmp)

with open('stopwords-ja/raw/bbalet\_stopwords\_ja.txt', encoding="utf8",mode='r') as f:

stopwords\_tmp = f.readlines()

ja\_stopwords.extend(stopwords\_tmp)

ja\_stopwords = [i.strip('\n').strip('\t').strip(' ') for i in ja\_stopwords]

ja\_stopwords.extend(['「','」','：'])

print(ja\_stopwords[:10])

print(ja\_stopwords[-10:])

#string.punctuation

# japanese puntuations

ja\_punctuations = """’！ "＃$％＆\ '（）\* +、-。/ :; <=>？@" \\ "^ \_` {|}〜'!"""+string.punctuation

en\_alphabets = """QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm"""

import tinysegmenter

segmenter = tinysegmenter.TinySegmenter()

def ja\_tokenize\_and\_cleanse(text):

#replace strings between brackets and parenthesis with blank

text = re.sub(r'\(.\*?\)', '', text)

text = re.sub(r'\[.\*?\]', '', text)

# replace repeated characters with single

text = re.sub('\.+', '.', text)

# replace urls with blank

text = re.sub(r'http[s]?://(?:[a-z]|[0-9]|[$-\_@.&amp;+]|[!\*\(\),]|(?:%[0-9a-f][0-9a-f]))+', '', text)

tokens = segmenter.tokenize(u"{}".format(text))

tokens = [i.strip() for i in tokens]

tokens\_without\_stopwords = [i for i in tokens if i not in ja\_stopwords and i not in ja\_punctuations and not i.isdigit() and i not in en\_alphabets]

return tokens\_without\_stopwords

text = """.........すべての自然なカリカリクラフトピーナッツバターはalledU + 00A0U + 00BDedU + 00B8U + 008Dで亀裂のようなものです"""

print(ja\_tokenize\_and\_cleanse(text))

data['cleansed\_tokens'] = data['text'].apply(lambda x: ja\_tokenize\_and\_cleanse(x))

for rec in data[:5].iterrows():

print(rec[1]['text']," --- ",rec[1]['cleansed\_tokens']," --- ")

print()

1. Top pharses.

from collections import Counter

terms\_all = [word for wordlist in data['cleansed\_tokens'] for word in wordlist]

count\_all = Counter()

# Update the counter

count\_all.update(terms\_all)

# Print the first 5 most frequent words --> meaningful phrases

print(count\_all.most\_common(50))

#print([i[0] for i in count\_all.most\_common(250)])

#Result

‘’’

[('ウ', 144), ('Kラft', 127), ('クラフト', 98), ('ロベrtKラft', 67), ('ドン', 67), ('ビll', 67), ('スペrボwlXLイX', 67), ('メss', 66), ('Wイth', 66), ('RTムシc', 66), ('tヘ', 65), ('オ/GvHSp', 64), ('ア', 55), ('あなた', 54), ('チーズ', 52), ('ませ', 52), ('ト', 45), ('まし', 40), ('エd', 39), ('オン', 38), ('オ', 35), ('アンp', 34), ('BDedU', 31), ('fオr', 31), ('必要', 30), ('アンd', 30), ('グッデル', 25), ('レcイペs', 24), ('イン', 22), ('オオキングpゴオd', 22), ('Velveeta', 21), ('すべて', 21), ('velveeta', 21), ('エアr', 21), ('謝罪', 20), ('\u200b', 20), ('イs', 20), ('ヨウ', 19), ('BD', 19), ('たい', 18), ('イt', 18), ('持っ', 17), ('NFL', 17), ('場合', 17), ('fロン', 17), ('ヂンネr', 17), ('オウt', 16), ('tハt', 16), ('bkravitz', 15), ('多く', 15)]

’’’

#English translated

‘’’

['U','K Ra ft','Craft','Robe rt K Ra ft','Don','Bill','Super Bo wlXL Lee X','Me ss','W th' ,'RT MUSHI c','t HE','O/GvHSp','A','You','Cheese','Not','TO','Better','E d','ON'. ,'O','Ann p','BDedU','f o r','Need','Ann d','Goodell','Le c ipe s','In','Oh king p Goo d' ,'Velveeta','all','velveeta','air r','apology','\u200b','i','you','BD','tai','it',' Have','nfl','case','fron','dine r','oh t','t ha t','bkravitz','more','wa tch','video',' 'T','l ike','heese','craft cheese','love rt','oh f','you r','...','edU','make','shit',' Kraft','They','Pat Rio ts','Goode ll','Ne d','Page','Ho w','Masuka','EDU','Bobcraft','Visit' ','De l c sulfur s','idea s','i','pe ch','id d','bane g','previous','angry','just',' t hi s','hi s','sw','more','oo ds','un f eye rly','attack d','dinner','via','like',' k la ft','ha s','yun y','ne w','wa lma rt','do','middle','false','work','a','nata ',' game','・','target','print','shell','do','n y','e ps','bamboo s','video s','mac' ,'w ll','eat','method','survey','get','say']

‘’’

This section of code does term document matrix.

# most frequently occuring terms

countVectorizer = CountVectorizer()

countVector = countVectorizer.fit\_transform([' '.join(text) for text in data['cleansed\_tokens']])

print('{} Number of tweets has {} words'.format(countVector.shape[0], countVector.shape[1]))

#print(countVectorizer.get\_feature\_names())

count\_vect\_df = pd.DataFrame(countVector.toarray(), columns=countVectorizer.get\_feature\_names())

count\_vect\_df.head()

# Term-Document Matrix

term\_doc\_df = count\_vect\_df.transpose()

term\_doc\_df.head()

term\_doc\_df.index.name='Token'

# Dump as excel

term\_doc\_df[:500].to\_excel("tfidf-output-japanese.xlsx")

import openpyxl

from openpyxl.styles import Font

#wb = openpyxl.Workbook()

wb = openpyxl.load\_workbook("tfidf-output-japanese.xlsx")

sheet = wb.active

sheet.insert\_rows(0,1)

# merge cell.

sheet.merge\_cells('B1:Z1')

sheet.cell(row = 1, column = 2).value = 'S.No.(Unique Identifier of a comment.)'

# set the font style to bold

sheet.cell(row = 1, column = 2).font = Font(size = 24, bold = True)

wb.save('tfidf-output-merge-japanese.xlsx')

**Summary of Exploratory Data Analysis**

- these are general texts. still texts are quite unclean. some taken care here. still more required to be handled such as spell checker, more accurate phrase extracter using rule based parser or using NER. but langauge knowledge would have helped much.

- used Google translator to validate result. even some results have pasted to compare.

- due to lack of memory(3GB RAM) in my system, i'm dumping few samples of term-frequency records to excel. user can use same code to dump all records with having good memory system. or we can do other ways like using DASK, or writing file batch by batch.

- have mentioned comments with each code. i'm sure those are understandable. so not writing much here.

- with that 4 questions regarding to this task have been answered through code, comments, and image of results.