TEBS:

Python code for classification of that data into various topics:

**LDA-Topic Modelling-[Latent Dirichlet Allocation]**

**It is named as rexitComplaints in Collab Notebook**

[**https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045LDA**](https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045LDA)**:**

**LDA is outlier sensitive.,so with the unrelated documents it is not classifying them into other bucket/set.**

import numpy as np

import pandas as pd

import re

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import LatentDirichletAllocation

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

def topic\_modeling(csv\_path):

# Load the DataFrame

df = pd.read\_csv(csv\_path)

# Function to preprocess text

def preprocess\_text(text):

text = text.lower()

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

text = re.sub(r'[^\w\s]', '', text)

tokens = nltk.word\_tokenize(text)

stop\_words = set(stopwords.words('english'))

tokens = [word for word in tokens if word not in stop\_words]

lemmatizer = WordNetLemmatizer()

tokens = [lemmatizer.lemmatize(word) for word in tokens]

preprocessed\_text = ' '.join(tokens)

return preprocessed\_text

# Apply preprocessing to the Description column

df['preprocessed\_paragraphs'] = df['Description'].apply(preprocess\_text)

# TF-IDF vectorization

tfidf\_vectorizer = TfidfVectorizer(max\_df=0.95, min\_df=2, stop\_words='english')

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['preprocessed\_paragraphs'])

# Topic Modeling

lda\_model = LatentDirichletAllocation(n\_components=5, random\_state=42)

lda\_output = lda\_model.fit\_transform(tfidf\_matrix)

# Interpretation

feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()

topic\_keywords = []

for topic\_weights in lda\_model.components\_:

top\_keyword\_idxs = topic\_weights.argsort()[:-10-1:-1]

topic\_keywords.append([feature\_names[idx] for idx in top\_keyword\_idxs])

# Define topic labels

topic\_labels = {

0: "workplace and safety issues",

1: "equipment and facilities maintenance",

2: "utilities and infrastructure",

3: "outdoor and environmental conditions",

4: "personal safety equipment and transportation"

}

topic\_predictions = [topic\_labels[i] for i in lda\_output.argmax(axis=1)]

df['Predicted Topic'] = topic\_predictions

topic\_counts = df['Predicted Topic'].value\_counts()

return topic\_counts,df

# Entry point for the script when executed independently

#if \_\_name\_\_ == "\_\_main\_\_":

# Example usage of the topic\_modeling function

#result\_df = topic\_modeling("/content/generated.csv")

#print(result\_df)

**Integration of Single Python file into c#,.net:**

**This is in Visual studio code 2022-Consoleapp6**

Download pythonnet, Newtonsoft.Json; in nugetmanager.

Then properties+right click+open+Build+Platform Target(set this to 64,because the python compiler in your system is 64bit, so for compati bility)

**The output is upto json arrays**

using System;

using System.Collections.Generic;

using System.Linq;

using ConsoleApp6;

using Newtonsoft.Json;

using Python.Runtime;

namespace TopicModelingExample

{

class Program

{

static void Main(string[] args)

{

Runtime.PythonDLL = @"C:\Program Files\Python312\python312.dll";

PythonEngine.Initialize();

PythonEngine.BeginAllowThreads();

using (Py.GIL())

{

dynamic sys = Py.Import("sys");

sys.path.append(@"C:\Users\lavan\source\repos\ConsoleApp6\ConsoleApp6\PythonScripts"); // Add the path to your Python script folder

dynamic topicModelScript = Py.Import("topic\_modeling\_script");

dynamic result = topicModelScript.topic\_modeling("C:\\Users\\lavan\\source\\repos\\ConsoleApp6\\ConsoleApp6\\generated.csv");

dynamic df = result[0];

dynamic topic\_counts = result[1];

Console.WriteLine("DataFrame:");

Console.WriteLine(df);

Console.WriteLine("\nTopic Counts:");

Console.WriteLine(topic\_counts);

dynamic generateJsonString = topicModelScript.generate\_json\_string;

dynamic json\_string = generateJsonString(df, topic\_counts);

Console.WriteLine("\nJSON String:");

Console.WriteLine(json\_string);

}

}

}

}

**The python Script upto the json script:**

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import LatentDirichletAllocation

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import json

def preprocess\_text(text):

text = text.lower()

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

text = re.sub(r'[^\w\s]', '', text)

tokens = nltk.word\_tokenize(text)

stop\_words = set(stopwords.words('english'))

tokens = [word for word in tokens if word not in stop\_words]

lemmatizer = WordNetLemmatizer()

tokens = [lemmatizer.lemmatize(word) for word in tokens]

preprocessed\_text = ' '.join(tokens)

return preprocessed\_text

def topic\_modeling(csv\_path):

df = pd.read\_csv(csv\_path)

df['preprocessed\_text'] = df['Description'].apply(preprocess\_text)

tfidf\_vectorizer = TfidfVectorizer(max\_df=0.95, min\_df=2, stop\_words='english')

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['Description'])

lda\_model = LatentDirichletAllocation(n\_components=5, random\_state=42)

lda\_output = lda\_model.fit\_transform(tfidf\_matrix)

# Interpretation

feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()

topic\_keywords = []

for topic\_weights in lda\_model.components\_:

top\_keyword\_idxs = topic\_weights.argsort()[:-10-1:-1]

topic\_keywords.append([feature\_names[idx] for idx in top\_keyword\_idxs])

# Define topic labels

topic\_labels = {

0: "workplace and safety issues",

1: "equipment and facilities maintenance",

2: "utilities and infrastructure",

3: "outdoor and environmental conditions",

4: "personal safety equipment and transportation"

}

# Classification

# Assign each paragraph to the topic with the highest probability

topic\_predictions = [topic\_labels[i] for i in lda\_output.argmax(axis=1)]

# Add the predicted topics to the DataFrame

df['Predicted Topic'] = topic\_predictions

# Count the occurrences of each topic label

topic\_counts = df['Predicted Topic'].value\_counts()

# Return the DataFrame and topic counts

return df,topic\_counts

def generate\_json\_string(df, topic\_counts):

topic\_counts\_dict = topic\_counts.to\_dict()

result\_dict = {}

for index, row in df.iterrows():

count = row['Predicted Topic']

nmd = row['NMD']

description = row['Description']

immediate\_cause = row['Immediate Cause Corrective Actions']

reported\_by = row['Reporter Company']

one\_maestro = row['One Maestro']

row\_dict = {

'NMD': nmd,

'Description': description,

'Immediate Cause Corrective Actions': immediate\_cause,

'Reporter Company': reported\_by,

'One Maestro': one\_maestro

}

if count not in result\_dict:

result\_dict[count] = {'count': topic\_counts\_dict[count], 'descriptions': []}

result\_dict[count]['descriptions'].append(row\_dict)

json\_string = json.dumps(result\_dict, indent=4)

return json\_string

# Entry point for the script when executed independently

if \_\_name\_\_ == "\_\_main\_\_":

df, topic\_counts, \_ = topic\_modelling("/content/generated.csv")

json\_string = generate\_json\_string(df, topic\_counts)

print(json\_string)

print("DataFrame:")

print(df)

print("\nTopic Counts:")

print(topic\_counts)

**NMF:[Non negative matrix-Factorisation]:**

<https://medium.com/blend360/topic-modelling-a-comparison-between-lda-nmf-bertopic-and-top2vec-part-i-3c16372d51f0#:~:text=More%20traditional%20algorithms%2C%20such%20as,)%20over%20raw%2Dword%20frequencies>.

NMF-[NMF collab Notebook].

**NMF is working well with the unrelated\_data as well**

[The topic names were assigned based on the most frequently occurring words by their TF-IDF weights for a particular topic. Interestingly, the study concluded that NMF’s results were more aligned with human judgment, ultimately outperforming LDA. However, as we know, topic extraction using LDA and NMF primarily relies on hyperparameters.]

Python script using NMF:

import pandas as pd

import numpy as np

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import NMF

def preprocess\_text(text):

    text = text.lower()

    text = re.sub(r'\d+', '', text)

    text = re.sub(r'[^\w\s]', '', text)

    tokens = nltk.word\_tokenize(text)

    stop\_words = set(stopwords.words('english'))

    tokens = [word for word in tokens if word not in stop\_words]

    lemmatizer = WordNetLemmatizer()

    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    preprocessed\_text = ' '.join(tokens)

    return preprocessed\_text

def topic\_modeling(csv\_path):

    df = pd.read\_csv(csv\_path)

    df['preprocessed\_text'] = df['Description'].apply(preprocess\_text)

    vectorizer = TfidfVectorizer(max\_df=0.95, min\_df=2, stop\_words='english')

    tfidf\_matrix = vectorizer.fit\_transform(df['preprocessed\_text'])

    nmf\_model = NMF(n\_components=6, random\_state=42)

    nmf\_output = nmf\_model.fit\_transform(tfidf\_matrix)

    feature\_names = vectorizer.get\_feature\_names\_out()

    top\_keywords = []

    for topic\_weights in nmf\_model.components\_:

        top\_keyword\_idxs = topic\_weights.argsort()[:-10-1:-1]  # Top 10 keywords per topic

        top\_keywords.append([feature\_names[idx] for idx in top\_keyword\_idxs])

    # Define topic labels

    topic\_labels = {

        0: "Workplace and Safety Issues",

        1: " Utilities and Infrastructure",

        2: "Personal Safety Equipment and Transportation",

        3: "Test/Trial Data (Uncertain)",

        4: "Outdoor and Environmental Conditions",

        5:"Equipment and Facilities Maintenance"

    }

    for i, keywords in enumerate(top\_keywords):

      print(f"Topic {i+1} keywords: {', '.join(keywords)}")

    topic\_predictions = [topic\_labels[i] for i in nmf\_output.argmax(axis=1)]

    df['topic'] = topic\_predictions

    topic\_counts = df['topic'].value\_counts()

    return df,topic\_counts

if \_\_name\_\_ == "\_\_main\_\_":

    df, topic\_counts= topic\_modeling("/content/generated.csv")

    #json\_string = generate\_json\_string(df, topic\_counts)

    #print(json\_string)

    print(df)

    print(topic\_counts)

The topic names were assigned based on the most frequently occurring words by their TF-IDF weights for a particular topic. Interestingly, the study concluded that NMF’s results were more aligned with human judgment, ultimately outperforming LDA. However, as we know, topic extraction using LDA and NMF primarily relies on hyperparameters.

The labels assigned from: Based on the provided text, here are the keywords for each topic:

**Topic 0: Workplace and Safety Issues**

Keywords: rusted, damaged, missing, observed, noncompliance, leak, hazard, improper, disconnected, tripped, exposed, broken, hazardous, rusting, loose, disconnected, blocked, improper operation, improper connection, damaged condition, injury, leakage, unhygienic, unattended, obstacle, potential danger, fallen, protruding, risk, fallen object, unsafe, slipping hazard, barricaded, obstructed, trip hazard, protruding rebar, inadequate, obstructing, lacking, repetitive, nonavailability, accumulation, rusty, injury risk, lack of maintenance

**Topic 1: Utilities and Infrastructure**

Keywords: leak, water, drainage, accumulation, overflow, rusted, damaged, cleaning, purifier, drainage water, accumulated, drain, valve, spillage, low water level, leaking, valve handle broken, drainage system, water level, rainwater, purifier filter, water tank, purification, maintenance, overflow pipe, leaking continuously, continuous leak, rusting, algae, water tank observed, water leak, drainage pipe, water line, water leakage, sewerage, leakage observed, accumulated water, drainage valve, overflowing, water pump, water purifier, water accumulation, rusting observed, drainage line, water pooling, water leakage observed, waterlogged, drain valve, overflowed, water hose, leakage, water drain, water line damage, water pipe, water line rusting, water discharge, water overflow, water supply, water leaking, waterlogged area, rusted condition, water leaking continuously, waterlogged condition, drainage leakage, water tank overflow, drainage pipe line, water drain valve, drainage water accumulated, drainage water overflow, drainage water accumulation, water tank observed lot algae, water hose kept, drainage system good

**Topic 2: Equipment and Facilities Maintenance**

Keywords: light, working, damaged, malfunctioning, sensor, door, tube light, ventilation fan, electrical, barrier, access system, maintenance, scaffold, worker, street light, access system gate, tube light working, malfunctioning temperature gauge, ventilation fan damage, light bullet, light working gate, flap barrier, access system gate b, worker found, tubelight working, sensor working, camera found, worker observed, street light jb, access system, light working, camera wire, sensor working, tubelight working, light fitting, flap barrier display, unused battery unit, manual hand siren, barricading, emergency door, broken, operational, scaffolding, improper tool, repair, malfunction, working condition, equipment malfunction, broken window handle, door sensor, broken tile, broken ladder, faulty equipment, damaged barrier, broken equipment, broken light, broken door, broken fan, broken switch, broken sensor, broken machine, broken lock, broken barricade, broken tool, broken switchboard, broken gate, broken pump, broken camera, broken signage, broken ventilation, broken system, broken barrier, broken fixture, broken window, broken component, broken machinery, broken infrastructure, broken facility, broken safety equipment, broken furniture, broken barrier, broken fitting, broken machinery, broken part, broken gadget, broken mechanism, broken appliance, broken instrument, broken device

**Topic 3: Test/Trial Data (Uncertain)**

Keywords: test, trial, trial nmd, trial test, trial found, test nmd, test found, test bath, soil test, preparation spt, found test, deviation, observed test, observed trial, observed trial nmd, soil test worker, trial test nmd, deviation small, deviation cafeteria, deviation lab, deviation observed, deviation cafeteria rd, deviation soil, deviation worker, deviation material, deviation lab temperature, deviation small plant, deviation trial, deviation found, deviation maintenance, deviation equipment, deviation discovered, deviation cafeteria rd floor, deviation noted, deviation observed test, deviation observed trial, deviation observed trial nmd, deviation observed test nmd, deviation observed soil, deviation observed worker, deviation discovered trial, deviation cafeteria rd, deviation discovered soil, deviation discovered worker, deviation discovered maintenance, deviation discovered equipment, deviation discovered test, deviation discovered trial, deviation discovered worker, deviation discovered material, deviation discovered lab, deviation discovered maintenance, deviation discovered equipment, deviation discovered cafeteria

**Topic 4: Outdoor and Environmental Conditions**

Keywords: vegetation, grass, overgrown, debris, soil, waste, garbage, scrap, metal, trash, rubbish, tree, plant, foliage, leaf, pathway, obstruction, barricade, barricading, protruding, protrusion, obstacle, hazard, fallen object, fallen, slipping hazard, unsafe, dangerous, risky, hazardous, potential danger, injury risk, injury hazard, slippery, blocked, obstructed, blocked pathway, overgrown vegetation, grass cutting, waste disposal, garbage disposal, waste management, environmental hazard, environmental concern, debris accumulation, debris removal, rubbish removal, vegetation removal, foliage removal, tree removal, plant removal, grass removal, grass cutting, pathway obstruction, safety hazard, unsafe conditions, dangerous conditions, slippery conditions, hazardous conditions, environmental damage, environmental issues, outdoor hazards, outdoor safety, outdoor risks, environmental risks, environmental dangers, outdoor dangers, outdoor risks, outdoor obstacles

**Topic 5: Personal Safety Equipment and Transportation**

Keywords: helmet, safety shoe, gloves, PPE, safety gear, safety equipment, safety violation, noncompliance, safety compliance, safety standards, safety precautions, safety practice, safety measures, safety hazard, safety concern, safety issue, safety protocol, safety guideline, safety violation, safety rule, safety regulation, safety inspection, safety breach, safety requirement, safety neglect, safety lapse, safety infraction, safety breach, safety deviation, safety violation, safety non-compliance, safety infringement, safety negligence, safety oversight, safety breach, safety lapse, safety negligence, safety concern, safety risk, safety hazard, safety issue, safety violation, safety negligence, safety breach, safety oversight, safety deviation, safety

**Integration of NMF.py into c#,.net:**

* **This is in Visual studio code 2022-NMF**
* Download pythonnet, Newtonsoft.Json; in nugetmanager.
* Please place the generated.csv,python script(NMF.py) in the location of NMF program in C#)
* Please set the path carefully
* Then properties+right click+open+Build+Platform Target(set this to 64,because the python compiler in your system is 64bit, so for compati bility)
* Then run the file.



